Dati e Algoritmi 2 – Ingegneria Informatica

Compito, 20/3/2006 (Durata: 3h)

Nome, Cognome, Matricola: _____

Corso di studio: _____ ☐ Triennale ☐ Specialistica

Prima Parte: domande a risposta unica

Si forniscano negli appositi spazi **solo le risposte** alle seguenti domande:

1. Utilizzando le proprietà strutturali dell'albero associato alla seguente ricorrenza:
$$T(n) = \begin{cases} 0 & n = 1 \\ T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + 3 & n > 1, \end{cases}$$
si determini il valore $T(37)$.

$T(37) = 108$

(**Solution strategy:**) The recurrence yields a tree with the same structure as the tree associated to the recursive MAX algorithm seen in class, that is, a full binary tree with exactly n leaves. This tree has $n-1$ internal nodes, each contributing work 3. Leaves contribute 0 work. As a consequence, we have $T(n) = 3(n-1)$, hence $T(37) = 3 \cdot 36 = 108$.

2. Sia data l'istanza $i = (147, 7)$ del problema della moltiplicazione di interi, con gli operandi rappresentati su $n = 8$ bit. Le tre istanze generate al primo livello di ricorsione dall'algoritmo di Karatsuba, espresse in decimale, sono:

$i_1 = (12, 7), i_2 = (9, 0), i_3 = (3, 7)$

(**Solution strategy:**) On 8 bits, we have $A = (147)_{10} = (10010011)_2$ and $B = (7)_{10} = (00000111)_2$, hence $A_1 = 9, A_0 = 3, B_1 = 0, B_0 = 7$. Then, instance $(A_1 + A_0, B_1 + B_0)$ is $(12, 7)$, instance (A_1, B_1) is $(9, 0)$ and instance (A_0, B_0) is $(3, 7)$.

3. Sia $\boldsymbol{a} = (2, 0, 0, 0, 5, 0, 0, 0)$ e sia $\boldsymbol{b} = C(\boldsymbol{a}) \times \boldsymbol{a}$. Si calcoli $DFT_8(\boldsymbol{b})$.

$DFT_8(\boldsymbol{b}) = (49, 9, 49, 9, 49, 9, 49, 9)$

(**Solution strategy:**) We have $\boldsymbol{b} = C(\boldsymbol{a}) \times \boldsymbol{a} = \boldsymbol{a} \circledast \boldsymbol{a}$. By the cyclic convolution theorem, it follows that $DFT_8(\boldsymbol{b}) = DFT_8(\boldsymbol{a}) \odot DFT_8(\boldsymbol{a})$, where $\odot$ denotes componentwise product. Since $\boldsymbol{a}$ is an $(8, 4)$-sparse vector, its DFT is obtained by concatenating $DFT_2(2, 5) = (7, -3)$ four times. Hence $DFT_8(\boldsymbol{a}) = (7, -3, 7, -3, 7, -3, 7, -3)$, $DFT_8(\boldsymbol{b}) = (49, 9, 49, 9, 49, 9, 49, 9)$.

4. Per $n > 1$, sia $T(n)$ il numero esatto di esecuzioni del comando $a \leftarrow a + 2$ nel seguente frammento codice:

$$\begin{array}{l}\textbf{for } i \leftarrow 1 \textbf{ to } n-1 \textbf{ do} \\ \quad \textbf{for } j \leftarrow n-i+1 \textbf{ to } n \textbf{ do} \\ \quad\quad a \leftarrow a+2\end{array}$$

Si calcoli $T(24)$.

$T(24) = 276$

(**Solution strategy:**) For even values of n, we have:

$$\begin{aligned} T(n) &= \sum_{i=1}^{n-1} \sum_{j=n-i+1}^{n} 1 \\ &= \sum_{i=1}^{n-1} i \\ &= n(n-1)/2 \end{aligned}$$

Hence $T(24) = 24 \cdot 23/2 = 276$.

5. Dato un file sull'alfabeto $C = \{$'A', 'B', 'C', 'D', 'E', 'F'$\}$ con frequenze 'A':9%, 'B':10%, 'C':15%, 'D':13%, 'E':14%, 'F':39%, si determini la codeword e('A') ottenuta con la codifica di Huffman.

e('A') = 1110

(**Solution strategy:**) The answer follows from executing the algorithm for the given frequencies. We obtain: e('A') = 1110, e('B') = 1111, e('C') = 110, e('D') = 100, e('E') = 101, e('F') = 0 .

Dati e Algoritmi 2 – Ingegneria Informatica

Compito, 20/3/2006 (Durata: 3h)

Nome, Cognome, Matricola: _____

Corso di studio: _____ ☐ Triennale ☐ Specialistica

Prima Parte: domande a risposta unica

Si forniscano negli appositi spazi **solo le risposte** alle seguenti domande:

1. Utilizzando le proprietà strutturali dell'albero associato alla seguente ricorrenza:
$$T(n) = \begin{cases} 0 & n = 1 \\ T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + 3 & n > 1, \end{cases}$$
si determini il valore $T(35)$.

$$\boxed{T(35) = 102}$$

(**Solution strategy:**) The recurrence yields a tree with the same structure as the tree associated to the recursive MAX algorithm seen in class, that is, a full binary tree with exactly n leaves. This tree has $n-1$ internal nodes, each contributing work 3. Leaves contribute 0 work. As a consequence, we have $T(n) = 3(n-1)$, hence $T(35) = 3 \cdot 34 = 102$.

2. Sia data l'istanza $i = (149, 11)$ del problema della moltiplicazione di interi, con gli operandi rappresentati su $n = 8$ bit. Le tre istanze generate al primo livello di ricorsione dall'algoritmo di Karatsuba, espresse in decimale, sono:

$$\boxed{i_1 = (14, 11), i_2 = (9, 0), i_3 = (5, 11)}$$

(**Solution strategy:**) On 8 bits, we have $A = (149)_{10} = (10010101)_2$ and $B = (11)_{10} = (00001011)_2$, hence $A_1 = 9, A_0 = 5, B_1 = 0, B_0 = 11$. Then, instance $(A_1 + A_0, B_1 + B_0)$ is $(14, 11)$, instance (A_1, B_1) is $(9, 0)$ and instance (A_0, B_0) is $(5, 11)$.

3. Sia $\boldsymbol{a} = (3, 0, 0, 0, 6, 0, 0, 0)$ e sia $\boldsymbol{b} = C(\boldsymbol{a}) \times \boldsymbol{a}$. Si calcoli $DFT_8(\boldsymbol{b})$.

$$\boxed{DFT_8(\boldsymbol{b}) = (81, 9, 81, 9, 81, 9, 81, 9)}$$

(**Solution strategy:**) We have $\boldsymbol{b} = C(\boldsymbol{a}) \times \boldsymbol{a} = \boldsymbol{a} \circledast \boldsymbol{a}$. By the cyclic convolution theorem, it follows that $DFT_8(\boldsymbol{b}) = DFT_8(\boldsymbol{a}) \odot DFT_8(\boldsymbol{a})$, where $\odot$ denotes componentwise product. Since $\boldsymbol{a}$ is an $(8, 4)$-sparse vector, its DFT is obtained by concatenating $DFT_2(3, 6) = (9, -3)$ four times. Hence $DFT_8(\boldsymbol{a}) = (9, -3, 9, -3, 9, -3, 9, -3)$, $DFT_8(\boldsymbol{b}) = (81, 9, 81, 9, 81, 9, 81, 9)$.

4. Per $n > 1$, sia $T(n)$ il numero esatto di esecuzioni del comando $a \leftarrow a + 2$ nel seguente frammento codice:

$$\begin{array}{l}\textbf{for } i \leftarrow 1 \textbf{ to } n-1 \textbf{ do} \\ \quad \textbf{for } j \leftarrow n-i+1 \textbf{ to } n \textbf{ do} \\ \quad\quad a \leftarrow a+2\end{array}$$

Si calcoli $T(26)$.

$$\boxed{T(26) = 325}$$

(**Solution strategy:**) For even values of n, we have:

$$\begin{aligned} T(n) &= \sum_{i=1}^{n-1} \sum_{j=n-i+1}^{n} 1 \\ &= \sum_{i=1}^{n-1} i \\ &= n(n-1)/2 \end{aligned}$$

Hence $T(26) = 26 \cdot 25/2 = 325$.

5. Dato un file sull'alfabeto $C = \{$'A', 'B', 'C', 'D', 'E', 'F'$\}$ con frequenze 'A':9%, 'B':10%, 'C':15%, 'D':13%, 'E':14%, 'F':39%, si determini la codeword e('B') ottenuta con la codifica di Huffman.

$$\boxed{e(\text{'B'}) = 1111}$$

(**Solution strategy:**) The answer follows from executing the algorithm for the given frequencies. We obtain: e('A') = 1110, e('B') = 1111, e('C') = 110, e('D') = 100, e('E') = 101, e('F') = 0 .

Dati e Algoritmi 2 – Ingegneria Informatica

Compito, 20/3/2006 (Durata: 3h)

Nome, Cognome, Matricola: _____

Corso di studio: _____ ☐ Triennale ☐ Specialistica

Prima Parte: domande a risposta unica

Si forniscano negli appositi spazi **solo le risposte** alle seguenti domande:

1. Utilizzando le proprietà strutturali dell'albero associato alla seguente ricorrenza:
$$T(n) = \begin{cases} 0 & n = 1 \\ T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + 3 & n > 1, \end{cases}$$
si determini il valore $T(39)$.

$T(39) = 114$

(**Solution strategy:**) The recurrence yields a tree with the same structure as the tree associated to the recursive MAX algorithm seen in class, that is, a full binary tree with exactly n leaves. This tree has $n-1$ internal nodes, each contributing work 3. Leaves contribute 0 work. As a consequence, we have $T(n) = 3(n-1)$, hence $T(39) = 3 \cdot 38 = 114$.

2. Sia data l'istanza $i = (151, 13)$ del problema della moltiplicazione di interi, con gli operandi rappresentati su $n = 8$ bit. Le tre istanze generate al primo livello di ricorsione dall'algoritmo di Karatsuba, espresse in decimale, sono:

$i_1 = (16, 13), i_2 = (9, 0), i_3 = (7, 13)$

(**Solution strategy:**) On 8 bits, we have $A = (151)_{10} = (10010111)_2$ and $B = (13)_{10} = (00001101)_2$, hence $A_1 = 9, A_0 = 7, B_1 = 0, B_0 = 13$. Then, instance $(A_1 + A_0, B_1 + B_0)$ is $(16, 13)$, instance (A_1, B_1) is $(9, 0)$ and instance (A_0, B_0) is $(7, 13)$.

3. Sia $\boldsymbol{a} = (4, 0, 0, 0, 7, 0, 0, 0)$ e sia $\boldsymbol{b} = C(\boldsymbol{a}) \times \boldsymbol{a}$. Si calcoli $DFT_8(\boldsymbol{b})$.

$DFT_8(\boldsymbol{b}) = (121, 9, 121, 9, 121, 9, 121, 9)$

(**Solution strategy:**) We have $\boldsymbol{b} = C(\boldsymbol{a}) \times \boldsymbol{a} = \boldsymbol{a} \circledast \boldsymbol{a}$. By the cyclic convolution theorem, it follows that $DFT_8(\boldsymbol{b}) = DFT_8(\boldsymbol{a}) \odot DFT_8(\boldsymbol{a})$, where $\odot$ denotes componentwise product. Since $\boldsymbol{a}$ is an $(8, 4)$-sparse vector, its DFT is obtained by concatenating $DFT_2(4, 7) = (11, -3)$ four times. Hence $DFT_8(\boldsymbol{a}) = (11, -3, 11, -3, 11, -3, 11, -3)$, $DFT_8(\boldsymbol{b}) = (121, 9, 121, 9, 121, 9, 121, 9)$.

4. Per $n > 1$, sia $T(n)$ il numero esatto di esecuzioni del comando $a \leftarrow a + 2$ nel seguente frammento codice:

$$\begin{array}{l}\textbf{for } i \leftarrow 1 \textbf{ to } n-1 \textbf{ do} \\ \quad \textbf{for } j \leftarrow n-i+1 \textbf{ to } n \textbf{ do} \\ \quad\quad a \leftarrow a+2\end{array}$$

Si calcoli $T(22)$.

$$T(22) = 231$$

(**Solution strategy:**) For even values of n, we have:

$$\begin{aligned} T(n) &= \sum_{i=1}^{n-1} \sum_{j=n-i+1}^{n} 1 \\ &= \sum_{i=1}^{n-1} i \\ &= n(n-1)/2 \end{aligned}$$

Hence $T(22) = 22 \cdot 21/2 = 231$.

5. Dato un file sull'alfabeto $C = \{$'A', 'B', 'C', 'D', 'E', 'F'$\}$ con frequenze 'A':9%, 'B':10%, 'C':15%, 'D':13%, 'E':14%, 'F':39%, si determini la codeword e('C') ottenuta con la codifica di Huffman.

$$e\text{('C')} = 110$$

(**Solution strategy:**) e answer follows from executing the algorithm for the given frequencies. We obtain: e('A') = 1110, e('B') = 1111, e('C') = 110, e('D') = 100, e('E') = 101, e('F') = 0 .

Dati e Algoritmi 2 – Ingegneria Informatica

Compito, 20/3/2006 (Durata: 3h)

Nome, Cognome, Matricola: _____

Corso di studio: _____ ☐ Triennale ☐ Specialistica

Prima Parte: domande a risposta unica

Si forniscano negli appositi spazi **solo le risposte** alle seguenti domande:

1. Utilizzando le proprietà strutturali dell'albero associato alla seguente ricorrenza:
$$T(n) = \begin{cases} 0 & n = 1 \\ T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + 3 & n > 1, \end{cases}$$
si determini il valore $T(41)$.

$$\boxed{T(41) = 120}$$

(**Solution strategy:**) The recurrence yields a tree with the same structure as the tree associated to the recursive MAX algorithm seen in class, that is, a full binary tree with exactly n leaves. This tree has $n-1$ internal nodes, each contributing work 3. Leaves contribute 0 work. As a consequence, we have $T(n) = 3(n-1)$, hence $T(41) = 3 \cdot 40 = 120$.

2. Sia data l'istanza $i = (152, 14)$ del problema della moltiplicazione di interi, con gli operandi rappresentati su $n = 8$ bit. Le tre istanze generate al primo livello di ricorsione dall'algoritmo di Karatsuba, espresse in decimale, sono:

$$\boxed{i_1 = (17, 14), i_2 = (9, 0), i_3 = (8, 14)}$$

(**Solution strategy:**) On 8 bits, we have $A = (152)_{10} = (10011000)_2$ and $B = (14)_{10} = (00001110)_2$, hence $A_1 = 9, A_0 = 8, B_1 = 0, B_0 = 14$. Then, instance $(A_1 + A_0, B_1 + B_0)$ is $(17, 14)$, instance (A_1, B_1) is $(9, 0)$ and instance (A_0, B_0) is $(8, 14)$.

3. Sia $\boldsymbol{a} = (5, 0, 0, 0, 8, 0, 0, 0)$ e sia $\boldsymbol{b} = C(\boldsymbol{a}) \times \boldsymbol{a}$. Si calcoli $DFT_8(\boldsymbol{b})$.

$$\boxed{DFT_8(\boldsymbol{b}) = (169, 9, 169, 9, 169, 9, 169, 9)}$$

(**Solution strategy:**) We have $\boldsymbol{b} = C(\boldsymbol{a}) \times \boldsymbol{a} = \boldsymbol{a} \circledast \boldsymbol{a}$. By the cyclic convolution theorem, it follows that $DFT_8(\boldsymbol{b}) = DFT_8(\boldsymbol{a}) \odot DFT_8(\boldsymbol{a})$, where $\odot$ denotes componentwise product. Since $\boldsymbol{a}$ is an $(8, 4)$-sparse vector, its DFT is obtained by concatenating $DFT_2(5, 8) = (13, -3)$ four times. Hence $DFT_8(\boldsymbol{a}) = (13, -3, 13, -3, 13, -3, 13, -3)$, $DFT_8(\boldsymbol{b}) = (169, 9, 169, 9, 169, 9, 169, 9)$.

4. Per $n > 1$, sia $T(n)$ il numero esatto di esecuzioni del comando $a \leftarrow a + 2$ nel seguente frammento codice:

$$\begin{array}{l}\textbf{for } i \leftarrow 1 \textbf{ to } n-1 \textbf{ do} \\ \quad \textbf{for } j \leftarrow n-i+1 \textbf{ to } n \textbf{ do} \\ \quad\quad a \leftarrow a+2\end{array}$$

Si calcoli $T(20)$.

$$\boxed{T(20) = 190}$$

(**Solution strategy:**) For even values of n, we have:

$$\begin{aligned} T(n) &= \sum_{i=1}^{n-1} \sum_{j=n-i+1}^{n} 1 \\ &= \sum_{i=1}^{n-1} i \\ &= n(n-1)/2 \end{aligned}$$

Hence $T(20) = 20 \cdot 19/2 = 190$.

5. Dato un file sull'alfabeto $C = \{$'A', 'B', 'C', 'D', 'E', 'F'$\}$ con frequenze 'A':9%, 'B':10%, 'C':15%, 'D':13%, 'E':14%, 'F':39%, si determini la codeword e('D') ottenuta con la codifica di Huffman.

$$\boxed{e\text{('D')} = 100}$$

(**Solution strategy:**) The answer follows from executing the algorithm for the given frequencies. We obtain: e('A') = 1110, e('B') = 1111, e('C') = 110, e('D') = 100, e('E') = 101, e('F') = 0 .

Seconda Parte: risoluzione di problemi

Esercizio 1 [15 punti] Si consideri la seguente relazione di ricorrenza, definita per valori arbitrari del parametro n:

$$T(n) = \begin{cases} 18, & 1 \leq n \leq 5 \\ T\left(\left\lfloor \frac{n}{4} \right\rfloor\right) + 2T\left(\left\lceil \frac{n}{8} \right\rceil\right) + 3n, & n > 5 \end{cases}$$

Utilizzando l'induzione parametrica, si determini una costante $c > 0$ per cui $T(n) \leq cn$ per $n \geq 1$.

Answer: Let us use parametric induction to determine a set of constraints on constant c. We will obtain the final value by selecting the minimum value satisfying all constraints. For the base $1 \leq n \leq 5$, we obtain:

$$T(n) = 18 \leq 18n \stackrel{?}{\boxed{\leq}} cn,$$

which is verified by any constant $c \geq 18$. Consider now $n > 5$ and assume that the inductive hypothesis holds for all values of the parameter less than n. We obtain:

$$\begin{array}{rcl} T(n) & \leq & c\left\lfloor \frac{n}{4} \right\rfloor + 2c\left\lceil \frac{n}{8} \right\rceil + 3n \\ & \leq & c\frac{n}{4} + 2c\left(\frac{n}{8} + 1\right) + 3n \\ & = & c\frac{n}{2} + 2c + 3n \\ & \stackrel{?}{\boxed{\leq}} & cn \end{array}$$

which can be rewritten as follows:

$$c\frac{n}{2} - 2c \stackrel{?}{\boxed{\geq}} 3n.$$

Observe that the above constraint must hold *simultaneously* for all values of the parameter n greater than 5. Consider the two lines of equation $y = 3n$ and $y_c = c\frac{n}{2} - 2c$. The first line goes through the origin, while the second yields the negative value $-2c$ for $n = 0$. In order for the second line to be above the first for all values of $n > 5$, it then suffices that the two lines intersect at $n = 6$, whence we obtain $c = 18$. Therefore any value of $c \geq 18$ allows us to carry out the induction.

The final value of constant c is obtained by intersecting the two constraints.

We have $c = \max\{18, 18\} = 18$. □

Esercizio 2 [15 punti] Si ricorda che una stringa $Z = \langle z_1, z_2, \ldots, z_m \rangle$ è *palindroma* se $z_{1+h} = z_{m-h}$, per $0 \leq h \leq m-1$. Si consideri il problema di determinare, data in ingresso una stringa $X = \langle x_1, x_2, \ldots, x_n \rangle$, la massima lunghezza di una sottosequenza palindroma di X. Ad esempio, per $X = \langle S, O, B, S \rangle$, la massima lunghezza di una sua sottosequenza palindroma è 3, realizzata dalla sottosequenza $Z = \langle S, O, S \rangle$. Si realizzi un algoritmo di programmazione dinamica che risolve il problema **impostando il procedimento secondo i seguenti punti**:

- **Punto 1.** Si determini un adeguato spazio dei sottoproblemi;
- **Punto 2.** Si enunci e si provi una proprietà di sottostruttura ottima sullo spazio dei sottoproblemi
- **Punto 3.** Si determini la ricorrenza associata alla proprietà, si fornisca il codice **iterativo** per la risoluzione bottom-up della ricorrenza e se ne analizzi la complessità in spazio e tempo. Per ottenere punteggio pieno, il codice deve usare spazio $O(n^2)$ e eseguire $O(n^2)$ confronti tra caratteri della stringa.

Answer:

Point 1 A palindrome subsequence $Z = \langle z_1, z_2, \ldots, z_m \rangle$ of a string $X = \langle x_1, x_2, \ldots, x_n \rangle$ recursively contains the palindrome substring $Z_{2..m-1}$ which is surely a palindrome subsequence of $X_{2..n-1}$. Therefore it is natural to select the set $\mathcal{S}_X = \{X_{i..j} : 1 \leq i \leq j \leq n\}$ of all nonempty substrings of X as a subproblem space. Observe that $|\mathcal{S}_X| = \Theta(n^2)$. Given that we are interested in deriving an $O(n^2)$ algorithm, we can afford to solve all subproblems in $\mathcal{S}_X$ as long as we perform only a constant number of character comparisons on each of them.

Point 2 We will prove the following optimal substructure property. Let $Z = \langle z_1, z_2, \ldots, z_m \rangle$ be a Longest Palindrome Subsequence (LPS) of $X_{i..j}$.

1. If $i = j$, then $Z = \langle x_i \rangle = X_{i..j}$;
2. If $j = i + 1$ and $x_i = x_j$, then $Z = \langle x_i, x_j \rangle = X_{i..j}$;
3. If $j = i + 1$ and $x_i \neq x_j$, then either $Z = \langle x_i \rangle$ or $Z = \langle x_j \rangle$;
4. If $j > i + 1$ and $x_i = x_j$, then $z_1 = z_m = x_i$ and $Z_{2..m-1}$ is a LPS of $X_{i+1..j-1}$;
5. If $j > i + 1$ and $x_i \neq x_j$, then either Z is a LPS of $X_{i+1..j}$ or a LPS of $X_{i..j-1}$.

Let us prove the above stated property. The three base cases (Points 1, 2, and 3) relative to substrings of length 1 and 2 are trivial. Consider now the case $j > i + 1$ (i.e., substrings of length at least 3), and let $Z = \langle x_{i_1}, x_{i_2}, \ldots x_{i_m} \rangle$, for

$i \leq i_1 < i_2 \ldots < i_m \leq j$. For Point 4, let $x_i = x_j$ and assume that $z_1, z_m \neq x_i$. Then $i_1 > i$ and $i_m < j$, hence the string $Z' = \langle x_i, Z, x_j \rangle$ is still a palindrome subsequence of $X_{i..j}$ (realized by the sequence of indices $i'_1 = i$, $i'_{j+1} = i_j$ for $1 \leq j \leq m$, and $i'_{m+2} = j$) which is longer than Z, which contradicts the fact that Z is a LPS of $X_{i..j}$. We can then assume that $i_1 = i$ and $i_m = j$. But then $Z_{2..m-1}$ is a palindrome subsequence of $X_{i+1..j-1}$ and it must be a LPS, or otherwise there would exist a longer LPS than Z for $X_{i..j}$. For Point 5, let $x_i \neq x_j$. Since Z is a palindrome, we have $z_1 = z_m$ hence either $i_1 > i$ or $i_m < j$. In the first case Z is a palindrome subsequence of $X_{i+1..j}$, while in the second case Z is a palindrome subsequence of $X_{i..j-1}$. In both cases, Z must be the respective LPS or otherwise one could obtain a longer LPS than Z for $X_{i..j}$.

Point 3 Let $p(i,j) = |LPS(X_{i..j})|$. The optimal substructure property proved in Point 2 yields the following recurrence on $p(i,j)$, for $1 \leq i \leq j \leq n$:

$$p(i,j) = \begin{cases} 1 & i = j, \\ 2 & j = i+1 \text{ and } x_i = x_j, \\ 1 & j = i+1 \text{ and } x_i \neq x_j, \\ 2 + p(i+1, j-1) & j > i+1 \text{ and } x_i = x_j, \\ \max\{p(i+1,j), p(i,j-1)\} & j > i+1 \text{ and } x_i \neq x_j. \end{cases}$$

Let $\ell = j - i + 1$ be the length of the substring $X_{i..j}$. Observe that a bottom-up computation of our solution $p(1,n)$ requires computing the subproblems for increasing values of ℓ. Indeed, the base cases concern the values $\ell = 1$ and $\ell = 2$, while for $\ell \geq 3$, a subproblem $X_{i..j}$ of size ℓ requires the solution to subproblems $X_{i+1..j-1}$, $X_{i+1..j}$, and $X_{i..j-1}$ whose sizes are $\ell - 2$, $\ell - 1$ and $\ell - 1$, respectively. Therefore a diagonal scan of the upper triangle of a table $P[i,j]$ suffices to correctly compute all relevant values. The code follows.

```
COMPUTE_P(X)
n ← length(X)
for i ← 1 to n − 1 do
    P[i, i] ← 1
    if (x_i = x_{i+1})
        then P[i, i + 1] ← 2
        else P[i, i + 1] ← 1
P[n, n] ← 1
{ initialize the first two diagonals with the base cases ℓ = 1 and ℓ = 2}
```

```
for ℓ ← 3 to n do
    for i ← 1 to n − ℓ + 1 do
        j ← i + ℓ − 1
        if (x_i = x_j)
            then P[i, j] ← P[i + 1, j − 1] + 2
            else P[i, j] ← MAX(P[i + 1, j], P[i, j − 1])
return P[1, n]
```

The correctness of the above algorithm follows from the previous discussion. The algorithm makes use of $\Theta(n^2)$ space (the $n \times n$ table P) and performs exactly one character comparison for each substring of length $\ell \geq 2$. Its running time is therefore

$$T(n) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} 1 = \frac{n(n-1)}{2} = \Theta(n^2).$$

Finally, observe that the space requirement can be lowered to $3n$ by only retaining, at iteration $\ell \geq 3$, the values on diagonals of order $\ell - 1$ and $\ell - 2$, which are the only values needed at that point. At position i, a vector relative to a given diagonal of order ℓ will store the optimal cost for subproblem $X_{i..i+\ell-1}$, for $1 \leq i \leq n - \ell + 1$, while the other values are irrelevant. The code for this space-saving strategy follows:

```
COMPUTE_P(X)
n ← length(X)
for i ← 1 to n − 1 do
    P1[i] ← 1
    if (x_i = x_{i+1})
        then P2[i] ← 2
        else P2[i] ← 1
P1[n] ← 1
{ initialize P1 and P2 with the first two diagonals.}
for ℓ ← 3 to n do
    for i ← 1 to n − ℓ + 1 do
        j ← i + ℓ − 1
        if (x_i = x_j)
            then P3[i] ← P1[i + 1] + 2
            else P3[i] ← MAX(P2[i + 1], P2[i])
    P1 ← P2; P2 ← P3
return P3[1]
```

□

Dati e Algoritmi 2 – Ingegneria Informatica
Compito, 3/4/2006 (Durata: 3h)

Nome, Cognome, Matricola: _____

Corso di studio: _____ ☐ Triennale ☐ Specialistica

Prima Parte: domande a risposta unica

Si forniscano negli appositi spazi le risposte alle seguenti domande:

1. Sia $\Pi = \mathcal{I} \times \mathcal{S}$ il problema di determinare il **numero di valori pari** in una stringa **finita** e di lunghezza **arbitraria** di interi. Si definiscano $\mathcal{I}$ e $\mathcal{S}$ **coerentemente** con la data specifica algebrica di Π.

   $$\mathcal{I} = \boldsymbol{Z}^\star,\ \mathcal{S} = \boldsymbol{N},\ \langle z_1, \ldots, z_n \rangle \Pi k \Leftrightarrow k = |\{i : \exists j \in \boldsymbol{Z} : z_i = 2j, 1 \leq i \leq n\}|$$

   (**Solution strategy:**) The instance is a (possibly empty) string Z of integers. The solution is simply a natural number, representing the number of even components of Z.

2. Per $n > 0$, si consideri una ricorrenza $T(n)$ il cui albero delle chiamate è **binario e pieno** (cioè, ogni nodo interno ha due figli) **con** $2n$ **foglie**, ciascuna associata a lavoro 2 e con ogni nodo interno che contribuisce lavoro 3. Si determini il valore $T(37)$.

   $$T(37) = 367$$

   (**Solution strategy:**) A full binary binary tree with m leaves has $m - 1$ internal nodes. Therefore, $T(n) = 2 \cdot (2n) + 3 \cdot (2n - 1)$. Hence $T(37) = 367$.

3. Si determini la parte reale (Re) e la parte immaginaria (Im) di ω_8^{124}.

   $$\mathrm{Re}(\omega_8^{124}) = -1 \quad \mathrm{Im}(\omega_8^{124}) = 0$$

   (**Solution strategy:**) From the well-known properties of the roots of unity, we have that $\omega_8^{124} = \omega_8^{124 \bmod 8} = \omega_8^4 = \omega_2 = -1$. It follows that $\mathrm{Re}(\omega_8^{124}) = -1$ and $\mathrm{Im}(\omega_8^{124}) = 0$.

4. Sia $X = \langle x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8 \rangle = \langle 1, 3, 5, 7, 8, 6, 2, 4 \rangle$. Si determini $W = \sum_{i=1}^{8} \ell(i)$, dove $\ell(i) = |MIS(i)|$ è la lunghezza di una *Monotonically Increasing Subsequence* di X con primo carattere uguale a x_i, $1 \leq i \leq 8$.

   $$W = 19$$

(**Solution strategy:**) The vector whose i-th value is $\ell(i)$ is $(5, 4, 3, 2, 1, 1, 2, 1)$. Hence $W = 19$.

5. Si consideri un insieme di 6 attività a_i, $1 \leq i \leq 6$, caratterizzate dai seguenti vettori $\boldsymbol{s}$ e $\boldsymbol{f}$ di tempi di inizio e fine:

$$\boldsymbol{s} = (1, 5, 2, 3, 7, 11) \text{ e } \boldsymbol{f} = (3, 7, 9, 10, 11, 12)$$

L'insieme S delle attività compatibili selezionate da GREEDY-ACTIVITY-SELECTOR($\boldsymbol{s}, \boldsymbol{f}$) è:

$$S = \{a_1, a_2, a_5, a_6\}$$

(**Solution strategy:**) The answer follows by executing the algorithm on the given start and end vectors.

Dati e Algoritmi 2 – Ingegneria Informatica
Compito, 3/4/2006 (Durata: 3h)

Nome, Cognome, Matricola: _____

Corso di studio: _____ ☐ Triennale ☐ Specialistica

Prima Parte: domande a risposta unica

Si forniscano negli appositi spazi le risposte alle seguenti domande:

1. Sia $\Pi = \mathcal{I} \times \mathcal{S}$ il problema di determinare il **numero di zeri** in una stringa **finita** e di lunghezza **arbitraria** di **bit**. Si definiscano $\mathcal{I}$ e $\mathcal{S}$ **coerentemente** con la data specifica algebrica di Π.

   $\mathcal{I} = \{0,1\}^\star$, $\mathcal{S} = \boldsymbol{N}$, $\langle b_1, \ldots, b_n \rangle \Pi k \Leftrightarrow k = |\{i : b_i = 0, 1 \leq i \leq n\}|$

   (**Solution strategy:**) The instance is a (possibly empty) string B of bits. The solution is simply a natural number, representing the number of occurrences of 0 in B.

2. Per $n > 0$, si consideri una ricorrenza $T(n)$ il cui albero delle chiamate è **binario e pieno** (cioè, ogni nodo interno ha due figli) **con** $2n$ **foglie**, ciascuna associata a lavoro 2 e con ogni nodo interno che contribuisce lavoro 3. Si determini il valore $T(39)$.

   $T(39) = 387$

   (**Solution strategy:**) A full binary binary tree with m leaves has $m-1$ internal nodes. Therefore, $T(n) = 2 \cdot (2n) + 3 \cdot (2n-1)$. Hence $T(39) = 387$.

3. Si determini la parte reale (Re) e la parte immaginaria (Im) di ω_8^{122}.

   $\mathrm{Re}(\omega_8^{122}) = 0 \quad \mathrm{Im}(\omega_8^{122}) = 1$

   (**Solution strategy:**) From the well-known properties of the roots of unity, we have that $\omega_8^{122} = \omega_8^{122 \bmod 8} = \omega_8^2 = \omega_4 = i$. It follows that $\mathrm{Re}(\omega_8^{124}) = 0$ and $\mathrm{Im}(\omega_8^{124}) = 1$.

4. Sia $X = \langle x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8 \rangle = \langle 1, 3, 5, 8, 7, 6, 2, 4 \rangle$. Si determini $W = \sum_{i=1}^{8} \ell(i)$, dove $\ell(i) = |MIS(i)|$ è la lunghezza di una *Monotonically Increasing Subsequence* di X con primo carattere uguale a x_i, $1 \leq i \leq 8$.

   $W = 15$

(**Solution strategy:**) The vector whose i-th value is $\ell(i)$ is $(4, 3, 2, 1, 1, 1, 2, 1)$. Hence $W = 15$.

5. Si consideri un insieme di 6 attività a_i, $1 \leq i \leq 6$, caratterizzate dai seguenti vettori $\boldsymbol{s}$ e $\boldsymbol{f}$ di tempi di inizio e fine:

$$\boldsymbol{s} = (1, 5, 2, 3, 7, 10) \text{ e } \boldsymbol{f} = (3, 7, 9, 10, 11, 12)$$

L'insieme S delle attività compatibili selezionate da GREEDY-ACTIVITY-SELECTOR($\boldsymbol{s}, \boldsymbol{f}$) è:

$$S = \{a_1, a_2, a_5\}$$

(**Solution strategy:**) The answer follows by executing the algorithm on the given start and end vectors.

Seconda Parte: risoluzione di problemi

Esercizio 1 [15 punti]

1. **Punto 1** Sia $n \geq 3$. Dato $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1}) \in \boldsymbol{C}^n$, si definisca il vettore complesso $\omega(\boldsymbol{x}) = (x_0 \cdot \omega_n^0, x_1 \cdot \omega_n^2, \ldots, x_j \cdot \omega_n^{2j}, \ldots, x_{n-1} \cdot \omega_n^{2(n-1)})$, dove ω_n denota la radice principale n-sima dell'unità. Determinare la relazione esistente tra le componenti del vettore $\boldsymbol{X} = DFT_n(\boldsymbol{x})$ e le componenti del vettore $\boldsymbol{Y} = DFT_n(\omega(\boldsymbol{x}))$.

2. **Punto 2** Si fornisca e si analizzi lo pseudocodice di un algoritmo che, dato in ingresso il vettore trasformato $\boldsymbol{X} = DFT_n(\boldsymbol{x})$, con $n \geq 3$, restituisca in uscita il vettore $\boldsymbol{Y} = DFT_n(\omega(\boldsymbol{x}))$ eseguendo un numero **lineare** di passi elementari.

Answer: Point 1 Recall that $\boldsymbol{Y} = DFT_n(\omega(\boldsymbol{x}))$ and $\boldsymbol{X} = DFT_n(\boldsymbol{x})$. We have:

$$\begin{aligned} Y_i &= [DFT_n(\omega(\boldsymbol{x})]_i = [F_n \times \omega(\boldsymbol{x})]_i \\ &= \sum_{j=0}^{n-1} \omega_n^{ij}(x_j \omega_n^{2j}) = \sum_{j=0}^{n-1} \omega_n^{(i+2)j} x_j \\ &= \sum_{j=0}^{n-1} \omega_n^{[(i+2) \bmod n]j} x_j \\ &= [DFT_n(\boldsymbol{x})]_{(i+2) \bmod n} = X_{(i+2) \bmod n}. \end{aligned}$$

Therefore, vector $\boldsymbol{Y}$ is simply a double cyclic left shift of vector $\boldsymbol{X}$

Point 2 From the relation established in Point 1, it follows that the algorithm required must simply perform a double cyclic left shift of its input vector. The code follows:

```
SHIFT_TRANSFORM(X)
n ← length(X)
for i ← 0 to n − 3 do
    Y_i ← X_{i+2}
Y_{n−2} ← X_0
Y_{n−1} ← X_1
return Y
```

The correctness of the algorithm follows from Point 1, and its running time is clearly linear in the length of its input. □

Esercizio 2 [15 punti]

1. **Punto 1** Sia $n > 0$. Si consideri la seguente ricorrenza $M(i, j)$, definita su tutte le coppie (i, j) con $1 \leq i \leq j \leq n$:

$$M(i,j) = \begin{cases} 1 & i = j, \\ 2 & j = i + 1 \\ M(i+1, j-1) \cdot M(i+1, j) \cdot M(i, j-1) & j > i + 1. \end{cases}$$

   Si fornisca una coppia di procedure INIT_M(n) e REC_M(i, j) per il calcolo **memoizzato** di $M(1, n)$.

2. **Punto 2** Si calcoli **il numero esatto** $T(n)$ di moltiplicazioni tra interi eseguite per calcolo di $M(1, n)$.

Answer:

Point 1 Recall that INIT_M(n) must return directly on base cases, otherwise it must initialize a look-up table with base and default values and then invoke REC_M($1, n$), which computes recursively all nonbase values, storing them in the table to avoid repeated calls. Observe that 0 is a suitable default value, since $M(i, j) > 0$ for $1 \leq i \leq j \leq n$. The code follows.

```
INIT_M(n)
if n = 1 then return 1
    {M(1,1) = 1, base case}
if n = 2 then return 2
    {M(1,2) = 2, base case}
for i ← 1 to n − 1 do
    M[i,i] ← 1
    M[i,i+1] ← 2
M[n,n] ← 1
for i ← 1 to n − 2 do
    forj ← i + 2 to n do
        M[i,j] ← 0
return REC_M(1,n)
```

```
REC_M(i,j)
if (M[i,j] = 0) then
    M[i,j] ← REC_M(i+1,j−1)∗
             ∗REC_M(i+1,j) ∗ REC_M(i,j−1)
return M[i,j]
```

Point 2 First, observe that INIT_M(n) does not perform any multiplications. Furthermore, the recursion tree associated to REC_M($1, n$) has exactly one internal node for each pair of indices (i, j), with $1 \leq i \leq j + 2 \leq n$, since these are all the subproblems that will be recursively solved to obtain $M(1, n)$. Each internal

node contributes work 2, while all leaves contribute work 0. Therefore:

$$\begin{aligned} T(n) &= \sum_{i=1}^{n-2} \sum_{j=i+2}^{n} 2 \\ &= 2 \sum_{i=1}^{n-2} n - i - 1 \\ &= 2 \sum_{k=1}^{n-2} k \\ &= (n-2)(n-1) \end{aligned}$$

□

Dati e Algoritmi 2 – Ingegneria Informatica

Compito, 3/7/2006 (Durata: 3h)

Nome, Cognome, Matricola: _____

Corso di studio: _____ ☐ Triennale ☐ Specialistica

Prima Parte: domande a risposta unica

Si forniscano negli appositi spazi le risposte alle seguenti domande:

1. Si consideri il seguente algoritmo ricorsivo REC(n), definito per valori positivi dell'input intero n:

   REC(n)
   if $n = 1$ **then return** 1
   else return $3 \cdot \text{REC}(\lceil n/2 \rceil) + \text{REC}(\lceil n/3 \rceil)$

   Si calcoli il numero di foglie F dell'albero delle chiamate associato all'esecuzione di REC(11).

   $\boxed{F = 9}$

   (**Solution strategy:**) REC(11) invokes REC(6) and REC(4). In turn, REC(6) invokes REC(3) and REC(2), while REC(4) invokes REC(2) twice. Finally, REC(3) invokes REC(2) and one instance of REC(1), a leaf. The four instances of REC(2) generate 8 leaves in total. Therefore, the total number of leaves is 9.

2. Utilizzando l'induzione parametrica, si individui la minima costante $c > 0$ per cui risulti $T(n) \leq cn^2$, per $n \geq 1$, dove

$$T(n) = \begin{cases} 7, & 1 \leq n \leq 8 \\ 12T(\lfloor n/8 \rfloor) + T(\lfloor n/4 \rfloor) + 9n^2, & n > 8 \end{cases}$$

   $\boxed{c = 12}$

   (**Solution strategy:**) From the (parametric) induction base, we obtain $c \geq 7$. When going from inductive hypothesis to thesis, we get:

$$\begin{aligned} T(n) &\leq 12c(n^2/64) + c(n^2/16) + 9n^2 \\ &= cn^2/4 + 9n^2 + \boxed{\leq} cn^2 \\ &\Leftrightarrow 3c/4 \geq 9 \\ &\Leftrightarrow c \geq 12. \end{aligned}$$

   Hence $c = \max\{7, 12\} = 12$.

3. Sia $\boldsymbol{x} = (10, 5, 5, 5, 0, 5, 5, 5)$ e sia $\boldsymbol{y} = DFT_8(\boldsymbol{x})$. Allora:

$$\boldsymbol{y} = (40, 10, 0, 10, 0, 10, 0, 10)$$

(**Solution strategy:**) Using Cooley-Tukey with $n = 2 \times 4$, the first column transform involves vector $(10, 0)$, yielding the new column $(10, 10)$, while the remaining three column transforms involve vector $(5, 5)$, yielding new columns $(10, 0)$. Multiplication by the twiddle-factors is empty. Then the transform for the first row yields new row $(40, 0, 0, 0)$, while the transform of the second row yields new row $(10, 10, 10, 10)$. By reading in column-major order, we obtain $y = (40, 10, 0, 10, 0, 10, 0, 10)$.

4. Si consideri il seguente frammento di codice :

```
sum ← 0
for i ← 1 to n do
    for j ← i − 1 to i + 2 do
        for k ← j to 2j − 1 do
            sum ← sum + 1
```

Si calcoli il valore finale di sum quando $n = 10$.

$$sum|_{n=10} = 240$$

(**Solution strategy:**) We have

$$\begin{aligned}
sum &= \sum_{i=1}^{n} \sum_{j=i-1}^{i+2} \sum_{k=j}^{2j-1} 1 \\
&= \sum_{i=1}^{n} \sum_{j=i-1}^{i+2} j \\
&= \sum_{i=1}^{n} (i - 1 + i + i + 1 + i + 2) \\
&= \sum_{i=1}^{n} (4i + 2) \\
&= 2n(n + 1) + 2n \\
&= 2n^2 + 4n
\end{aligned}$$

Therefore, when $n = 10$, we have $sum = 240$

5. Sia $X = \langle \mathrm{a}, \mathrm{c}, \mathrm{c}, \mathrm{a} \rangle$ e $Y = \langle \mathrm{a}, \mathrm{c}, \mathrm{a} \rangle$. Si definisca $W = \sum_{i=0}^{4} \sum_{j=0}^{3} \ell(i, j)$, dove $\ell(i, j) = |\mathrm{LCS}(X_i, Y_j)|$, per $0 \leq i \leq 4$ e $0 \leq j \leq 3$. Determinare W.

$$W = 19$$

(**Solution strategy:**) In order to obtain the values of all the $\ell(i,j)$'s, we run the LCS algorithm to obtain the final matrix $L[i,j] = \ell(i,j)$. We have

$$L = \begin{pmatrix} & - & \mathrm{a} & \mathrm{c} & \mathrm{a} \\ - & 0 & 0 & 0 & 0 \\ \mathrm{a} & 0 & 1 & 1 & 1 \\ \mathrm{c} & 0 & 1 & 2 & 2 \\ \mathrm{c} & 0 & 1 & 2 & 2 \\ \mathrm{a} & 0 & 1 & 2 & 3 \end{pmatrix}, \text{ whence } W = 19.$$

Seconda Parte: risoluzione di problemi

Si forniscano soluzioni esaurienti e rigorose ai due problemi seguenti. Risposte immotivate e prive di prova di correttezza non saranno considerate. Gli algoritmi vanno codificati utilizzando lo **pseudocodice** usato in classe.

Esercizio 1 [15 punti]

1. **[10 punti]** Dato un arbtrario vettore $\boldsymbol{a} \in \boldsymbol{C}^n$, si dimostri che la colonna di indice 1 della matrice di Fourier, F_n^1, (ovvero, la colonna di elemento i-simo $[F_n^1]_i = \omega_n^i$) è un *autovettore* della matrice circolante $C(\boldsymbol{a})$ associato all'*autovalore* $A_{n-1} = (DFT_n(\boldsymbol{a}))_{n-1}$. In altre parole, si dimostri che il prodotto matrice-vettore $C(\boldsymbol{a}) \times F_n^1$ è uguale a $A_{n-1}F_n^1$.
   (*Suggerimento:* Si studi il prodotto $C(\boldsymbol{a}) \times F_n^1$ applicando il teorema di convoluzione ciclica e ricordando la struttura delle matrici F_n e F_n^{-1}.)

2. **[5 punti]** Si fornisca un algoritmo che, dato in ingresso $\boldsymbol{a}$, ritorni l'autovalore A_{n-1} definito al Punto 1. Per avere punteggio pieno, l'algoritmo deve compiere un numero di operazioni tra scalari complessi **lineare** in n.

Answer:

Point 1 By well known properties of circulant matrices, we know that

$$C(\boldsymbol{a})F_n^1 = \boldsymbol{a} \circledast F_n^1,$$

where $\circledast$ denotes cyclic convolution. Let $\boldsymbol{A} = DFT_n(\boldsymbol{a})$ and $\boldsymbol{G} = DFT_n(F_n^1)$. By the cyclic convolution theorem, we have that

$$\boldsymbol{a} \circledast F_n^1 = DFT_n^{-1}(\boldsymbol{A} \odot \boldsymbol{G}),$$

where $\odot$ denotes component-wise product. Let us first determine the components of $\boldsymbol{G}$. For $0 \leq i < n$, recall that $(F_n^1)_i = \omega_n^i$. Therefore, by the definition of Discrete Fourier Transform we have that,

$$G_i = \sum_{j=0}^{n-1} (F_n^1)_j \omega_n^{ij} = \sum_{j=0}^{n-1} \omega_n^{(i+1)j}.$$

Observe that by the summation lemma, $G_i = 0$ for $0 \leq i < n-1$, while $G_{n-1} = n$. Therefore $\boldsymbol{A} \odot \boldsymbol{G}$ is a vector $\boldsymbol{B}$ with (possibly) a single nonnull component, that is, $B_i = 0$, for $0 \leq i < n-1$ and $B_{n-1} = nA_{n-1}$. We have:

$$DFT_n^{-1}(\boldsymbol{B}) = F_n^{-1}\boldsymbol{B} = nA_{n-1}(F_n^{-1})^{n-1},$$

where $(F_n^{-1})^{n-1}$ denotes the last column of the inverse of the Fourier matrix, F_n^{-1}. Recall that the i-th component of $(F_n^{-1})^{n-1}$ is $(1/n)\omega_n^{-i(n-1)} = (1/n)\omega_n^i = (1/n)(F_n^1)_i$, hence

$$DFT_n^{-1}(\boldsymbol{B}) = nA_{n-1}(1/n)F_n^1 = A_{n-1}F_n^1.$$

We have shown that $C(\boldsymbol{a}) \times F_n^1 = A_{n-1}F_n^1$, hence F_n^1 is an eigenvector of $C(\boldsymbol{a})$ associated with eigenvalue $A_{n-1} = (DFT_n(\boldsymbol{a}))_{n-1}$.

Point 2 We have that

$$A_{n-1} = (F_n\boldsymbol{a})_{n-1} = \sum_{j=0}^{n-1} a_j\omega_n^{(n-1)j}.$$

Rather than invoking the FFT algorithm to compute *all components* of $DFT_n(\boldsymbol{a})$ (which would cost $O(n \log n)$) we only compute A_{n-1} using the definition as follows.

```
COMPUTE(a)
n ← length(a)
r ← e^{i2π(n−1)/n} {r = ω_n^{n−1}}
rj ← 1 {rj will be set to consecutive powers of r during the iterations}
A1 ← 0 {A1 accumulates the various contributions to A_{n−1}}
for j ← 0 to n − 1 do
    A1 ← A1 + a_j ∗ rj
    rj ← rj ∗ r
return A1
```

The correctness of the above algorithm follows directly from the definition of Discrete Fourier Transform and its complexity is clearly linear in n. □

Esercizio 2 [15 punti]

1. **Punto 1 [10 punti]** Sia $n > 0$. Si consideri la seguente ricorrenza $M(n)$:

$$M(n) = \begin{cases} 1 & n = 1, \\ \sum_{i=1}^{n-1} M(i) \cdot M(n-i) & n > 1. \end{cases}$$

   Si fornisca una coppia di procedure INIT_M(n) e REC_M(k) per il calcolo **memoizzato** di $M(n)$.

2. **Punto 2 [5 punti]** Si calcoli **il numero esatto** $T(n)$ di **moltiplicazioni** tra interi eseguite per calcolo di $M(n)$.

Answer:

Point 1 Recall that INIT_M(n) must return directly on base cases, otherwise it must initialize a look-up table with base and default values and then invoke REC_M(n), which computes recursively all nonbase values, storing them in the table to avoid repeated calls. Observe that 0 is a suitable default value, since $M(n) > 0$ for $n \geq 1$. The code follows.

```
INIT_M(n)                          REC_M(k)
if n = 1 then return 1             if (M[k] = 0) then
    {M(1) = 1, base case}              sum ← 0
M[1] ← 1                               for i ← 1 to k − 1 do
for i ← 2 to n do                          sum ← sum + REC_M(i) · REC_M(k − i)
    M[i] ← 0                           M[k] ← sum
return REC_M(n)                    return M[k]
```

Point 2 First, observe that INIT_M(n) does not perform any multiplications. Furthermore, the recursion tree associated to REC_M(n) has exactly one internal node for each index k, with $2 \leq k \leq n$, since these are all the subproblems that will be recursively solved to obtain $M(n)$. An internal node associated to instance k contributes work $k - 1$, since REC_M performs one multiplication per iteration of the **for** loop. It follows that the total work is

$$\begin{aligned} T(n) &= \sum_{k=2}^{n} (k-1) \\ &= \sum_{i=1}^{n-1} i \\ &= (n-1)n/2 \end{aligned}$$

□

Dati e Algoritmi 2 – Ingegneria Informatica
Compito, 28/8/2006 (Durata: 3h)

Prima Parte: domande a risposta unica

Si forniscano negli appositi spazi le risposte alle seguenti domande:

1. Sia $\Pi = \mathcal{I} \times \mathcal{S}$ il problema del calcolo della potenza intera (a esponente positivo, negativo o nullo) di un numero naturale. Si definiscano $\mathcal{I}$ e $\mathcal{S}$.

   $\boxed{\mathcal{I} = \boldsymbol{N} \times \boldsymbol{Z} - \{(0,0)\},\ \mathcal{S} = \boldsymbol{Q}^+ \cup \{0\}}$

   (**Solution strategy:**) The general instance is made of a natural number and an integer. Since by definition natural numbers contain 0, we must exclude the undefined case 0^0. The solution is clearly a nonnegative rational number (to encompass negative powers).

2. Sia n una potenza di due. Utilizzando l'induzione parametrica, si individui la minima costante $c > 0$ per cui risulti $T(n) \leq cn \log_2 n$, per $n \geq 1$, dove

   $$T(n) = \begin{cases} 0, & n = 1, 2, 4, 8 \\ T(n/2) + 4T(n/8) + 14n, & n > 8 \end{cases}$$

   $\boxed{c = 7}$

   (**Solution strategy:**) From the (parametric) induction base, we obtain $c \geq 0$. When going from inductive hypothesis to thesis, we get:

   $$\begin{aligned} T(n) &\leq c(n/2)(\log_2(n/2)) + 4c(n/8)(\log_2(n/8)) + 14n \\ &= c(n/2)(\log_2 n - \log_2 2) + c(n/2)(\log_2 n - \log_2 8) + 14n \\ &= cn \log_2 n - 2cn + 14n \boxed{\leq} cn \log_2 n \Leftrightarrow c \geq 7. \end{aligned}$$

   Hence $c = \max\{0, 7\} = 7$.

3. Sia $\boldsymbol{x} = (1, 7, 1, 0, 1, 0, 1, 0)$. Si elenchino in *ordine row-major* gli elementi della matrice $X \in {}_4\boldsymbol{C}_2$ risultante dopo l'applicazione del Passo 2 (trasformazione delle colonne) dell'algoritmo di Cooley-Tukey con $p = 4$ e $q = 2$.

   $\boxed{\text{row-major}(X) = (4, 7, 0, 7, 0, 7, 0, 7)}$

(**Solution strategy:**) After folding $\boldsymbol{x}$ into the 4×2 matrix $X = \begin{pmatrix} 1 & 7 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \end{pmatrix}$, Step 2 of Cooley-Tukey requires transforming the two 4×1 columns of X separately. Recalling that the DFT of an vector with n components equal to $a \in \boldsymbol{C}$ is a vector whose only nonzero component is its first, with value na; and that, vice versa, the DFT of the latter vector is a vector with all components equal to its first, the new matrix is $X = \begin{pmatrix} 4 & 7 \\ 0 & 7 \\ 0 & 7 \\ 0 & 7 \end{pmatrix}$.

4. Sia $T(n)$ il numero esatto di esecuzioni del comando $a \leftarrow a \star 2$ nel seguente codice:

$$\begin{array}{l} \textbf{for } i \leftarrow 1 \textbf{ to } n-1 \textbf{ do} \\ \quad \textbf{for } j \leftarrow i \textbf{ to } i+1 \textbf{ do} \\ \quad\quad \textbf{for } k \leftarrow j+2 \textbf{ to } 2j+1 \textbf{ do} \\ \quad\quad\quad a \leftarrow a \star 2 \end{array}$$

Si calcoli $T(21)$.

$\boxed{T(21) = 440}$

(**Solution strategy:**) We have

$$\begin{aligned} T(n) &= \sum_{i=1}^{n-1} \sum_{j=i}^{i+1} \sum_{k=j+2}^{2j+1} 1 \\ &= \sum_{i=1}^{n-1} \sum_{j=i}^{i+1} j \\ &= \sum_{i=1}^{n-1} (i + (i+1)) \\ &= \sum_{i=1}^{n-1} (2i+1) \\ &= (n-1)n + (n-1) \\ &= (n-1)(n+1) \end{aligned}$$

Therefore $T(21) = 20 \cdot 22 = 440$

5. Dato un file sull'alfabeto $\Sigma = \{$'A', 'B', 'C', 'D', 'E', 'F'$\}$ con frequenze 'A':9%, 'B':11%, 'C':15%, 'D':25%, 'E':40%, si determinino le 5 codeword ottenute con la codifica di Huffman.

$$e(\text{'A'}) = 1110, e(\text{'B'}) = 1111, e(\text{'C'}) = 110, e(\text{'D'}) = 10, e(\text{'E'}) = 0$$

(**Solution strategy:**) The answer follows from executing the algorithm for the given frequencies.

Seconda Parte: risoluzione di problemi

Esercizio 1 [15 punti]

1. **[10 punti]** Dato un arbtrario vettore $\boldsymbol{a} \in \boldsymbol{C}^n$, si dimostri che per $0 \leq j \leq n-1$, la colonna di indice j della matrice di Fourier, F_n^j, ovvero, la colonna di elemento i-simo $[F_n^j]_i = \omega_n^{ij}$, è un *autovettore* della matrice circolante $C(\boldsymbol{a})$ associato all'*autovalore* $A_{(n-j) \bmod n} = (DFT_n(\boldsymbol{a}))_{(n-j) \bmod n}$. In altre parole, si dimostri che per $0 \leq j \leq n-1$, il prodotto matrice-vettore $C(\boldsymbol{a}) \times F_n^j$ è uguale a $A_{(n-j) \bmod n} F_n^j$.

2. **[5 punti]** Si fornisca un algoritmo di complessità $O(n \log n)$ che, dato in ingresso $\boldsymbol{a}$, ritorni il vettore $\boldsymbol{b}$, dove $b_j = A_{(n-j) \bmod n}$ è l'autovalore associato al autovettore F_n^j, per $0 \leq j \leq n-1$.

Answer:

Point 1 We recall that by well known properties of circulant matrices, we have:

$$C(\boldsymbol{a})F_n^j = \boldsymbol{a} \circledast F_n^j,$$

where $\circledast$ denotes cyclic convolution. We will now show that $DFT_n(C(\boldsymbol{a})F_n^j) = DFT_n(A_{(n-j) \bmod n} F_n^j)$, and the claim will follow.

Consider first $DFT_n(C(\boldsymbol{a})F_n^j)$. Let $\boldsymbol{A} = DFT_n(\boldsymbol{a})$ and $\boldsymbol{G}^j = DFT_n(F_n^j)$. By the cyclic convolution theorem, we have that

$$DFT_n(C(\boldsymbol{a})F_n^j) = DFT_n(\boldsymbol{a} \circledast F_n^j) = \boldsymbol{A} \odot \boldsymbol{G}^j,$$

where $\odot$ denotes component-wise product. Let us first determine the components of $\boldsymbol{G}^j$. For $0 \leq k < n$, recall that $(F_n^j)_k = \omega_n^{kj}$. Therefore, by the definition of Discrete Fourier Transform we have that,

$$G_i^j = \sum_{k=0}^{n-1} (F_n^j)_k \omega_n^{ik} = \sum_{k=0}^{n-1} \omega_n^{(i+j)k}.$$

Observe that by the summation lemma, $G^j_i = 0$ for $i \neq (n-j) \bmod n$, while $G^j_{(n-j) \bmod n} = n$. Therefore $\boldsymbol{A} \odot \boldsymbol{G}^j$ is a vector $\boldsymbol{B}^j$ with (possibly) a single nonnull component, that is, $B^j_i = 0$, for $i \neq (n-j) \bmod n$ and $B^j_{(n-j) \bmod n} = nA_{(n-j) \bmod n}$.

Consider now $DFT_n(A_{(n-j) \bmod n} F^j_n)$. We have:

$$\begin{aligned} DFT_n(A_{(n-j) \bmod n} F^j_n) &= A_{(n-j) \bmod n} DFT_n(F^j_n) \\ &= A_{(n-j) \bmod n} \boldsymbol{G}^j \\ &= \boldsymbol{B}^j. \end{aligned}$$

Point 2 It is sufficient to observe that vector $\boldsymbol{b}$ can be obtained as $\boldsymbol{A}^{\mathbf{rev}}$, where $A = DFT_n(\boldsymbol{a})$. The algorithm follows.

```
COMPUTE(b)
n ← length(a)
A ← FFT(a)
b_0 ← A_0
for i ← 1 to n − 1 do
    b_i ← A_{n−i}
return b
```

The correctness of the above algorithm is immediate and its complexity is clearly $\Theta(n \log n)$. □

Esercizio 2 [15 punti]

1. **Punto 1 [10 punti]** Si consideri la quantità $c(i, j)$, definita per ricorrenza su coppie di indici (i, j), con $1 \leq i, j \leq n$, come segue:

$$c(i,j) = \begin{cases} j, & i = 1 \text{ e } 1 \leq j \leq n, \\ i, & 1 < i \leq n \text{ e } j = n, \\ c(i-1,j) \cdot c(i,j+1), & 1 < i \leq n \text{ e } 1 \leq j < n. \end{cases}$$

   Si voglia calcolare la quantità $c(n, 1)$.

   Si fornisca una coppia di procedure INIT_C(n) e REC_C(i, j) per il calcolo **memoizzato** di $c(n, 1)$.

2. **Punto 2 [5 punti]** Si calcoli **il numero esatto** $T(n)$ di **prodotti** tra interi eseguite dall'algoritmo sviluppato al Punto 1.

Answer:

Point 1 Recall that INIT_C(n) must return directly on base cases, otherwise it must initialize a look-up table with base and default values and then invoke

REC_C$(n, 1)$, which computes recursively all nonbase values, storing them in the table to avoid repeated calls. Observe that 0 is a suitable default value, since $c(i, j) > 0$ for $1 \leq i, j \leq n$. The code follows.

```
INIT_C(n)
if n = 1 then return 1
    {c(1,1) = 1, base case}
C[1,1] ← 1
for k ← 2 to n do
    C[1,k] ← C[k,n] ← k
for i ← 2 to n do
    for j ← 1 to n − 1 do
        C[i,j] ← 0
return REC_C(n,1)
```

```
REC_C(i,j)
if (C[i,j] = 0) then
    C[i,j] ← REC_C(i − 1,j) · REC_C(i,j + 1)
return C[i,j]
```

Point 2 First, observe that INIT_C(n) does not perform any integer products. Furthermore, the recursion tree associated to REC_C$(n, 1)$ has exactly one internal node for each index pair (i, j), with $2 \leq i \leq n$ and $1 \leq j \leq n - 1$, since these are all the subproblems that will be recursively solved to obtain $c(n, 1)$. Each internal node contributes unit work. It follows that the total work is $T(n) = (n - 1)^2$. □

v4

Dati e Algoritmi 2 – Ingegneria Informatica
Compito, 19/3/2007 (Durata: 3h)

Nome, Cognome, Matricola: _____

Corso di studio: _____ ☐ Triennale ☐ Specialistica

Prima Parte: domande a risposta unica

Si forniscano negli appositi spazi **solo le risposte** alle seguenti domande:

1. Utilizzando il Master Theorem, si determini l'ordine di grandezza associato alla ricorrenza $T(n) = 2T(n/2) + \sqrt{n}\log^2 n$

   $T(n) = \Theta(n)$

   (**Solution strategy:**) In the above recurrence, we have $a = 2$ and $b = 2$, hence $\log_b a = 1$. For $\epsilon = 1/3$, we have $\sqrt{n}\log^2 n = O(n^{1-\epsilon}) = O(\sqrt{n} \cdot n^{1/6})$, hence we are in the first case of the Master Theorem. Therefore $T(n) = \Theta(n^{\log_b a}) = \Theta(n)$, and the answer follows.

2. Si consideri la seguente doppia sommatoria:

$$S(n) = \sum_{i=4}^{n-3} \sum_{j=1}^{n-2-i} 1.$$

   Si valuti $S(31)$.

   $S(31) = 325$

   (**Solution strategy:**) We have:

$$\begin{aligned} S(n) &= \sum_{i=4}^{n-3} \sum_{j=1}^{n-2-i} 1 \\ &= \sum_{i=4}^{n-3} (n-2-i) \\ &= \sum_{k=1}^{n-6} k \\ &= (n-6)(n-5)/2 \end{aligned}$$

   For $n = 31$, we obtain $S(31) = 25 \cdot 26/2 = 325$.

3. Sia $\boldsymbol{a} = (-2, 0, 0, 0, -3, 0, 0, 0)$ e sia $\boldsymbol{b} = \boldsymbol{a} \circledast \boldsymbol{a} \circledast \boldsymbol{a}$. Si calcoli $DFT_8(\boldsymbol{b})$.

   $DFT_8(\boldsymbol{b}) = (-125, 1, -125, 1, -125, 1, -125, 1)$

(**Solution strategy:**) By applying the cyclic convolution theorem twice, it follows that $DFT_8(\boldsymbol{b}) = DFT_8(\boldsymbol{a}) \odot DFT_8(\boldsymbol{a}) \odot DFT_8(\boldsymbol{a})$, where $\odot$ denotes componentwise product. Since $\boldsymbol{a}$ is an $(8,4)$-sparse vector, its DFT_8 is obtained by concatenating $DFT_2(-2,-3) = (-5,1)$ four times. Hence

$$\begin{aligned} DFT_8(\boldsymbol{a}) &= (-5,1,-5,1,-5,1,-5,1), \text{ and} \\ DFT_8(\boldsymbol{b}) &= (-125,1,-125,1,-125,1,-125,1) \end{aligned}$$

4. Si consideri la seguente matrice di pesi W relativa a un grafo diretto con quattro nodi:

$$W = \begin{bmatrix} 0 & 1 & 7 & 6 \\ 4 & 0 & 2 & \infty \\ 1 & \infty & 0 & 4 \\ 5 & \infty & \infty & 0 \end{bmatrix}$$

Se determinini costo$(\pi^{(2)}_{4,3})$ secondo la strategia APSP di Floyd-Warshall.

costo$(\pi^{(2)}_{4,3}) = 8$

(**Solution strategy:**) The only simple paths in the graph between nodes 4 and 3 that may only use 1 and/or 2 as an intermediate node are either $\langle 4,1,3\rangle$ of cost 12, or $\langle 4,1,2,3\rangle$ of cost 8, and the answer follows.

5. Dato un file sull'alfabeto $C = \{$'A', 'B', 'C', 'D', 'E', 'F'$\}$ con frequenze f('A') = 0.14, f('B') = 0.15, f('C') = 0.16, f('D') = 0.17, f('E') = 0.18, f('F') = 0.2, si determini il costo ottimo $B(T^\star)$ relativo all'albero associato alla codifica di Huffman.

$B(T^\star) = 2.62$

(**Solution strategy:**) The answer follows from executing the algorithm for the given frequencies. We obtain: $d_{T^\star}$('A') = $d_{T^\star}$('B') = $d_{T^\star}$('C') = $d_{T^\star}$('D') = 3, $d_{T^\star}$('E') = $d_{T^\star}$('F') = 2. Hence $B(T^\star) = 3 \cdot 0.62 + 2 \cdot 0.38 = 2.62$.

Seconda Parte: risoluzione di problemi

Esercizio 1 [15 punti]

- **Punto 1 [7 punti]** Sia $n > 0$. Si dimostri rigorosamente che per ogni vettore $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1}) \in \boldsymbol{C}^n$ si ha:

$$(F_n)^2\boldsymbol{x} = n \cdot \boldsymbol{x}^R,$$

  dove $(F_n)^2 = F_n \times F_n$ denota il quadrato della matrice di Fourier di ordine n e $\boldsymbol{x}^R = (x_0, x_{n-1}, x_{n-2} \ldots, x_1)$ denota il *reverse* del vettore $\boldsymbol{x}$.

- **Punto 2. [8 punti]** Utilizzando la relazione provata al Punto 1, si fornisca lo pseudocodice e si analizzi un algoritmo *divide-and-conquer* KFT$(\boldsymbol{x}, k)$ che, dati in ingresso un vettore complesso $\boldsymbol{x} \in \boldsymbol{C}^n$, con n potenza di due e k un generico intero positivo o nullo, restituisca $\boldsymbol{y} = (F_n)^k\boldsymbol{x}$ eseguendo $T(n, k) = O(n(k + \log n))$ operazioni aritmetiche tra scalari complessi. (*Suggerimento:* si osservi che per $k \geq 2$ vale $(F_n)^k\boldsymbol{x} = (F_n)^2((F_n)^{k-2}\boldsymbol{x})$ ...)

Answer: Point 1 In order to prove the stated relation, let $\boldsymbol{y} = F_n\boldsymbol{x}$. Then we have:

$$(F_n)^2\boldsymbol{x} = n\boldsymbol{x}^R \Leftrightarrow F_n\boldsymbol{y} = n\boldsymbol{x}^R \Leftrightarrow \boldsymbol{x}^R = \frac{1}{n}F_n\boldsymbol{y} \Leftrightarrow \boldsymbol{x} = \frac{1}{n}(F_n\boldsymbol{y})^R \Leftrightarrow \boldsymbol{x} = (F_n)^{-1}\boldsymbol{y},$$

where the (trivially true) last relation follows from the reduction of DFT_n^{-1} to DFT_n discussed in class.

Point 2 Let $\boldsymbol{x} \in \boldsymbol{C}^n$ and $k \geq 0$. In order to derive a divide-and-conquer algorithm, let us first discuss the base cases. If $k = 0$, we have $(F_n)^0 = I_n$, hence we return $\boldsymbol{x}$. If $k = 1$, we have $(F_n)^1 = F_n$, hence we return $F_n\boldsymbol{x}$ by calling FFT$(\boldsymbol{x})$. When $k \geq 2$, we observe that $(F_n)^k\boldsymbol{x} = (F_n)^2\boldsymbol{z}$, where $\boldsymbol{z} = (F_n)^{k-2}\boldsymbol{x}$. We can then obtain $\boldsymbol{z}$ recursively and then obtain $(F_n)^k\boldsymbol{x}$ by applying the relation proved in Point 1, by first reversing $\boldsymbol{z}$ and then multiplying each of its components by n. The pseudocode follows.

KFT$(\boldsymbol{x}, k)$
$n \leftarrow$ length$(\boldsymbol{x})$
if $(k = 0)$ **then return** $\boldsymbol{x}$
if $(k = 1)$ **then return** FFT$(\boldsymbol{x})$
$\boldsymbol{z} \leftarrow$ KFT$(\boldsymbol{x}, k - 2)$
$y_0 \leftarrow n \cdot z_0$
for $i \leftarrow 1$ **to** $n - 1$ **do** $y_i \leftarrow n \cdot z_{n-i}$
return $\boldsymbol{y}$

The above algorithm makes use of the routine FFT developed in class, and its correctness follows from the previous discussion. The recurrence relation on the number of arithmetic operations between complex scalars performed by KFT$(\boldsymbol{x}, k)$ is:

$$T(n,k) = \begin{cases} 0 & k = 0, \\ cn \log n & k = 1 \\ T(n, k-2) + n & k > 1, \end{cases}$$

for some constant c (determined by the call to FFT). The above relation unfolds to $T(n,k) = T(n, k-2i) + ni$, for $1 \leq i \leq \lfloor k/2 \rfloor$. For even k, we then obtain $T(n,k) = nk/2 + T(n,0) = nk/2$, while for odd k (the worst case) we get $T(n,k) = n\lfloor k/2 \rfloor + T(n,1) = n\lfloor k/2 \rfloor + cn \log n$. Therefore $T(n,k) = O(n(k + \log n))$.

Indeed, this is not the best algorithm for computing $(F_n)^k \boldsymbol{x}$, since it is not difficult to prove (reasoning along similar lines) that the following closed formula holds. Let $\boldsymbol{y} = F_n \boldsymbol{x}$. Then:

$$(F_n)^k \boldsymbol{x} = \begin{cases} n^{k/2} \cdot \boldsymbol{x} & \text{if } k \bmod 4 = 0, \\ n^{(k-1)/2} \cdot \boldsymbol{y} & \text{if } k \bmod 4 = 1, \\ n^{k/2} \cdot \boldsymbol{x}^R & \text{if } k \bmod 4 = 2, \\ n^{(k-1)/2} \cdot \boldsymbol{y}^R & \text{if } k \bmod 4 = 3. \end{cases}$$

Implementing the above formula, in the worst case (k mod 4 an odd number), we would only need to compute $n^{(k-1)/2}$ in $O(\log k)$ time, transform $\boldsymbol{x}$ in $O(n \log n)$ time and perform n additional scalar multiplications, for a total of $O(n \log n + \log k)$ time. □

Esercizio 2 [15 punti] Data una stringa $X = \langle x_1, x_2, \ldots, x_n \rangle$, si consideri la seguente ricorrenza $\ell(i,j)$, definita per $1 \leq i \leq j \leq n$:

$$\ell(i,j) = \begin{cases} 1 & i = j, \\ 2 & i = j - 1 \\ 2 + \ell(i+1, j-1) & (i < j-1) \wedge (x_i = x_j) \\ \sum_{k=i}^{j-1} (\ell(i,k) + \ell(k+1,j)) & (i < j-1) \wedge (x_i \neq x_j). \end{cases}$$

- **Punto 1 [10 punti]** Si fornisca lo pseudocodice della coppia di procedure INIT_L(X) e REC_L(X,i,j) che, data in ingresso una stringa X di lunghezza n, restituiscano in uscita $\ell(1,n)$ utilizzando la memoizzazione.

- **Punto 2 [5 punti]** Si analizzi l'albero associato alla chiamata REC_L$(X,1,n)$ e si determini la complessità **al caso migliore** $T_B(n)$, supponendo che le uniche operazioni di costo unitario e non nullo siano i confronti tra caratteri.

Answer:

Point 1 Recall that INIT_L(X) must return directly on base cases ($n = 1$ or $n = 2$ in this case), otherwise it must initialize a look-up table with base and default values and then invoke REC_L($1, n$), which computes recursively all non-base values, storing them in the table to avoid repeated calls. Observe that 0 is a suitable default value, since $\ell(i, j) > 0$ for $1 \leq i \leq j \leq n$. The code follows.

```
INIT_L(X, n)
if n = 1 then return 1
    {ℓ(1,1) = 1, base case}
if n = 2 then return 2
    {ℓ(1,2) = 2, base case}
for i ← 1 to n − 1 do
    L[i, i] ← 1
    L[i, i + 1] ← 2
L[n, n] ← 1
for i ← 1 to n − 2 do
    forj ← i + 2 to n do
        L[i, j] ← 0
return REC_L(X, 1, n)
```

```
REC_L(X, i, j)
if (L[i, j] = 0)
    then
        if (x_i = x_j)
            then
            L[i, j] ← 2 + REC_L(X, i + 1, j − 1)
            else
            for k ← i to j − 1 do
                L[i, j] ←L[i, j]+
                        REC_L(X, i, k)+
                        REC_L(X, k + 1, j)
return L[i, j]
```

Point 2 First, observe that INIT_L(X) does not perform any character comparisons. Let us now analyze the recursion tree associated to the call REC_L($X, 1, n$). The best case is clearly the one where all characters are the same, since the recursion tree in that case is unary and its internal nodes correspond to the calls with indices $(1, n), (2, n-1), \ldots (k, n-k+1)$, each associated to a cost of 1 (one comparison). There is one such call for every $1 \leq k \leq n - k - 1$ (nonbase cases), whence $1 \leq k \leq (n-1)/2$, for a running time $T_B(n) \leq \lfloor n/2 \rfloor = O(n)$.

For completeness, let us also consider the worst case. If $x_i \neq x_j$ for $1 \leq i < j \leq n$ all subproblems are eventually solved. In this case, thanks to memoization, there is exactly one internal node for each call with indices (i, j), with $1 \leq i < j - 1$, each associated to a cost of 1 (again, one comparison only), for a total time

$$T_W(n) = \sum_{i=1}^{n-2} \sum_{j=i+2}^{n} 1 = \sum_{i=1}^{n-2} n - 1 - i = \sum_{k=1}^{n-2} k = \frac{(n-2)(n-1)}{2} = \Theta(n^2).$$

Finally, note that if we considered additions as operations also requiring unit cost, the running time in the best case would stay linear in n, while the running time in the worst case would become cubic in n. □

v2

Dati e Algoritmi 2 – Ingegneria Informatica
Compito, 2/4/2007 (Durata: 3h)

Nome, Cognome, Matricola: _____

Corso di studio: _____ ☐ Triennale ☐ Specialistica

Prima Parte: domande a risposta unica

Si forniscano negli appositi spazi **solo le risposte** alle seguenti domande:

1. Sia $\Pi = \mathcal{I} \times \mathcal{S}$ il problema di determinare il **prodotto modulo** 2 di due interi. Si definiscano $\mathcal{I}$ e $\mathcal{S}$ e si completi la specifica algebrica di Π.

   $$\mathcal{I} = \boldsymbol{Z} \times \boldsymbol{Z},\ \mathcal{S} = \{0,1\},\ i = (x,y)\Pi s \Leftrightarrow s = (xy) \bmod 2$$

   (**Solution strategy:**) The set of instances is clearly any pair of integers, hence $\mathcal{I} = \boldsymbol{Z} \times \boldsymbol{Z}$. Possible solutions are either 0, 1, hence $\mathcal{S} = \{0,1\}$. The algebraic specification of the problem must relate each pair (x,y) to the product of its components modulo 2.

2. Sia n una potenza di due. Utilizzando l'induzione parametrica, si individui la minima costante $c > 0$ per cui risulti $T(n) \leq cn \log_2 n$, per $n \geq 1$, dove

   $$T(n) = \begin{cases} 0, & n = 1,2,4 \\ T(n/2) + 4T(n/8) + 28n, & n \geq 8 \end{cases}$$

   $$c = 14$$

   (**Solution strategy:**) From the (parametric) induction base, we obtain $c \geq 0$. When going from inductive hypothesis to thesis, we get:

   $$\begin{aligned} T(n) &\leq c(n/2)(\log_2(n/2)) + 4c(n/8)(\log_2(n/8)) + 28n \\ &= c(n/2)(\log_2 n - \log_2 2 + \log_2 n - \log_2 8) + 28n \\ &= cn \log_2 n - 2cn + 28n \boxed{\leq} cn \log_2 n \Leftrightarrow c \geq 14. \end{aligned}$$

   Hence $c = \max\{0, 14\} = 14$.

3. Si determini la parte reale (Re) e la parte immaginaria (Im) di ω_{16}^{68}.

   $$\mathrm{Re}(\omega_{16}^{68}) = 0 \quad \mathrm{Im}(\omega_{16}^{136}) = 1$$

(**Solution strategy:**) From the cancellation lemma, we have that $\omega_{16}^{68} = \omega_{4\cdot 4}^{4\cdot 17} = \omega_4^{17} = \omega_4 = i$. It follows that $\mathrm{Re}(\omega_{16}^{68}) = 0$ and $\mathrm{Im}(\omega_{16}^{68}) = 1$.

4. Sia $X = \langle 1, 3, 4, 2, 8, 5, 7, 6 \rangle$. Si fornisca il vettore di 8 componenti $\boldsymbol{\ell}$, dove $\ell[i]$ rappresenta la lunghezza di una LIS che ha come ultimo carattere x_i, per $1 \leq i \leq 8$.

$$\boldsymbol{\ell} = (1, 2, 3, 2, 4, 4, 5, 5)$$

(**Solution strategy:**) The vector is obtained by manually running the LIS algorithm.

5. Si consideri un insieme di 6 attività a_i, $1 \leq i \leq 6$, caratterizzate dai seguenti vettori $\boldsymbol{s}$ e $\boldsymbol{f}$ di tempi di inizio e fine:

$$\boldsymbol{s} = (2, 1, 4, 6, 3, 7) \text{ e } \boldsymbol{f} = (4, 5, 7, 8, 9, 10)$$

L'insieme S delle attività compatibili selezionate da GREEDY-ACTIVITY-SELECTOR($\boldsymbol{s}, \boldsymbol{f}$) è:

$$S = \{a_1, a_3, a_6\}$$

(**Solution strategy:**) The answer follows by executing the algorithm on the given start and end vectors.

Seconda Parte: risoluzione di problemi

Esercizio 1 [15 punti] Sia n una potenza di due. Dato $\boldsymbol{b} \in \boldsymbol{C}^n$, sia $\boldsymbol{b}^{(k)}$ il vettore ottenuto applicando a $\boldsymbol{b}$ uno shift ciclico destro di k posizioni, cioè $\boldsymbol{b}^{(0)} = \boldsymbol{b}$, e $\boldsymbol{b}^{(k)} = (b_{n-k}, \ldots, b_{n-1}, b_0, b_1, \ldots, b_{n-k-1})$, per $1 \leq k \leq n-1$. Si fornisca lo pseudocodice e si analizzi un algoritmo MAX_K($\boldsymbol{a}, \boldsymbol{b}$) che, dati in ingresso due vettori $\boldsymbol{a}, \boldsymbol{b} \in \boldsymbol{C}^n$ restituisca in uscita il massimo prodotto interno di $\boldsymbol{a}$ per uno shift di $\boldsymbol{b}$, ovvero il valore

$$\max_{0 \leq k < n} \left\{ \langle \boldsymbol{a}, \boldsymbol{b}^{(k)} \rangle = \sum_{j=0}^{n-1} \boldsymbol{a}_j \boldsymbol{b}_j^{(k)}. \right\}$$

L'algoritmo deve eseguire $O(n \log n)$ operazioni tra scalari complessi.
(*Suggerimento:* Si riconduca il problema al calcolo di un'opportuna convoluzione.)

Answer: Recall that

$$\boldsymbol{x} \circledast \boldsymbol{y} = C(\boldsymbol{y}) \times \boldsymbol{x},$$

where $C(\boldsymbol{y})$ is the circulant matrix whose first column is $\boldsymbol{y}$. Since the columns of $C(\boldsymbol{y})$ are all the right cyclic shifts of $\boldsymbol{y}$, it follows that the rows of $C(\boldsymbol{y})$ are all the right cyclic shifts of $\boldsymbol{y}^R$, where $y_i^R = y_{(n-i) \bmod n}$, for $0 \leq i < n$, that is, $[C(\boldsymbol{y})]_k = (\boldsymbol{y}^R)^{(k)}$. By the definition of matrix-by-vector product, it then follows that k-th component of $\boldsymbol{x} \circledast \boldsymbol{y}$ is the inner product $\langle \boldsymbol{x}, (\boldsymbol{y}^R)^{(k)} \rangle$.

Since $(\boldsymbol{b}^R)^R = \boldsymbol{b}$, by applying the above observation to vectors $\boldsymbol{a}$ and $\boldsymbol{b}^R$, we obtain that the k-th component of $\boldsymbol{a} \circledast \boldsymbol{b}^R$ is equal to $\langle \boldsymbol{a}, \boldsymbol{b}^{(k)} \rangle$, the quantity that we have to maximize over all k, with $0 \leq k < n$. The pseudocode follows.

```
MAX_K(a, b)
n ← length(a)
y_0 ← b_0
for i ← 1 to n − 1 do y_i ← b_{n−i}
A ← FFT(a); Y ← FFT(y)
for i ← 0 to n − 1 do Z_i ← A_i · Y_i
z ← INV_FFT(Z)
max ← z_o
for i ← 1 to n − 1 do
    if max < z_i then max ← z_i
return max
```

The correctness of the above algorithm follows from the previous discussion. Its running time is dominated by the two calls to FFT and the one call to INV_FFT, which require $O(n \log n)$ scalar operations. □

Esercizio 2 [15 punti] Per $n \geq 2$ si consideri una successione di alfabeti $C_n = \{c_1, c_2, \ldots, c_n\}$ e, per ogni alfabeto, un file $F_n \in C_n^\star$ con frequenze $f(c_i) = 1/2^i, 1 \leq i \leq n-1$ e $f(c_n) = 1/2^{n-1}$.

- **Punto 1 [8 punti]** Si determinino le codeword associate ai caratteri del generico alfabeto C_n dalla codifica di Huffman, supponendo che, durante l'algoritmo, nel caso di estrazione di due alberi x e y di pari frequenza minima dalla coda di priorità Q, l'albero estratto di profondità maggiore diventi il sottoalbero sinistro del nuovo albero ottenuto dal merge di x e y.

- **Punto 1 [7 punti]** Sia T^n l'albero ottimo per F_n, e sia $b(n) = B(T^n)$. Si determini una ricorrenza su $b(n)$ e se ne fornisca una soluzione.
  (*Suggerimento:* i primi $n-2$ livelli di T_n coincidono con T_{n-1}...)

Answer:

Point 1 When run on F_n, HUFFMAN will initially merge the leaves associated to c_n and c_{n-1}, yielding a tree H_1 of depth 1 and frequency $1/2^{n-1}+1/2^{n-1} = 1/2^{n-2}$. Observe that this is also the frequency of c_{n-2}. Therefore, the next merging step will merge H_1 and the leaf associated to c_{n-2}, yielding a new tree H_2 of depth 2 and frequency $1/2^{n-3}$. In order to iterate this process, let H_0 be the tree containing a single node, associated to character c_n. Then, for $1 < i \leq n-1$, the i-th merging step is between H_{i-1} and the leaf associated to c_{n-i}, which both have frequency $1/2^{n-i}$, yielding a tree T_i of frequency $1/2^{n-i-1}$. Also, observe that H_{i-1} is always the left subtree of H_i. At the end of the algorithm, in the optimal tree $H_{n-1} = T^n$ there is *exactly* one leaf (always a right child) at each level i (starting from the root), with $1 \leq i \leq n-2$: such leaf is associated with c_i. Finally, at level $n-1$ there are the two leaves associated with c_{n-1} and c_n. As a consequence, the codeword for c_i is $0^{i-1}1$, for $1 \leq i \leq n-2$, while the codewords for the least frequent characters c_{n-1} and c_n are $0^{n-2}1$ and 0^{n-1} (the assignment of these last two codewords depends on which character between c_{n-1} and c_n is associated with the leaf which becomes the left child in H_1).

Point 2 First, we note that T^2 is a complete binary tree with two leafs, hence $b(2) = B(T^2) = 2 \cdot 1/2 = 1$. For $n > 2$, as suggested, we observe that T^n coincides with T^{n-1} for the first $n-2$ levels. The only difference between the two trees is that the left leaf of frequency $1/2^{n-2}$ on level $n-2$ of T^{n-1} becomes an internal node in T^n attached to two leaves each of frequency $1/2^{n-1}$. In order to obtain $b(n) = b(T^n)$ from $b(n-1) = B(T^{n-1})$ we must then subtract the contribution of that leaf (now an internal node in T^n) and add the contributions of the two new

leaves. Therefore:

$$b(n) = b(n-1) - (n-2) \cdot \frac{1}{2^{n-2}} + 2 \cdot (n-1) \cdot \frac{1}{2^{n-1}} = b(n-1) + \frac{1}{2^{n-2}}.$$

By unfolding the above recurrence, we obtain

$$b(n) = b(2) + \sum_{i=1}^{n-2} \frac{1}{2^i} = \sum_{i=0}^{n-2} \frac{1}{2^i} = 2\left(1 - \frac{1}{2^{n-1}}\right) = 2 - \frac{1}{2^{n-2}}.$$

□

v1

Dati e Algoritmi 2 – Ingegneria Informatica
Compito, 19/3/2008 (Durata: 3h)

Nome, Cognome, Matricola: _____

Corso di studio: _____ ☐ Triennale ☐ Specialistica

Prima Parte: domande a risposta unica

Si forniscano negli appositi spazi **solo le risposte** alle seguenti domande:

1. Utilizzando il Master Theorem, si determini l'ordine di grandezza associato alla ricorrenza $T(n) = 125 \cdot T(n/5) + n^2 \log^7 n$

   $T(n) = \Theta(n^3)$

   (**Solution strategy:**) In the above recurrence, we have $a = 125$ and $b = 5$, hence $\log_b a = 3$. For $\epsilon = 1/2$, we have $n^2 \log^7 n = O(n^{3-\epsilon}) = O(n^2 n^{1/2})$, hence we are in the first case of the Master Theorem. Therefore $T(n) = \Theta(n^{\log_b a}) = \Theta(n^3)$, and the answer follows.

2. Sia $\boldsymbol{x} = (5, 3, 0, 3, 0, 3, 0, 3)$. Si elenchino in **ordine column-major** gli elementi della matrice $X \in {}_4\boldsymbol{C}_2$ risultante dopo l'applicazione del Passo 2 (trasformazione delle colonne) dell'algoritmo di Cooley-Tukey con $p = 4$ e $q = 2$.

   $\text{column-major}(X) = (5, 5, 5, 5, 12, 0, 0, 0)$

   (**Solution strategy:**) After folding $\boldsymbol{x}$ into the 4×2 matrix $X = \begin{pmatrix} 5 & 3 \\ 0 & 3 \\ 0 & 3 \\ 0 & 3 \end{pmatrix}$, Step 2 of Cooley-Tukey requires transforming the two 4×1 columns of X separately. From the two closed-form transforms: $DFT_4(\ (y, 0, 0, 0)\ ) = (y, y, y, y)$ and $DFT_4((x, x, x, x)) = (4x, 0, 0, 0)$, we obtain the following new matrix: $X = \begin{pmatrix} 5 & 12 \\ 5 & 0 \\ 5 & 0 \\ 5 & 0 \end{pmatrix}$, whence column-major$(X) = (5, 5, 5, 5, 12, 0, 0, 0)$.

3. Si consideri la seguente doppia sommatoria:

$$S(n) = \sum_{i=3}^{n-2} \sum_{j=1}^{n-1-i} 1.$$

   Si valuti $S(13)$.

$S(13) = 45$

(**Solution strategy:**) We have:

$$\begin{aligned} S(n) &= \sum_{i=3}^{n-2} \sum_{j=1}^{n-1-i} 1 \\ &= \sum_{i=3}^{n-2} (n-1-i) \\ &= \sum_{k=1}^{n-4} k \\ &= (n-4)(n-3)/2 \end{aligned}$$

For $n = 13$, we obtain $S(13) = 9 \cdot 10/2 = 45$.

4. Sia $X = \langle \mathrm{a}, \mathrm{l}, \mathrm{a} \rangle$ e $Y = \langle \mathrm{a}, \mathrm{l}, \mathrm{g}, \mathrm{a} \rangle$. Si definisca $W = \sum_{i=0}^{3} \sum_{j=0}^{4} \ell(i,j)$, dove per $0 \leq i \leq 3$ e $0 \leq j \leq 4$, $\ell(i,j)$ denota la lunghezza di una LCS dei prefissi X_i e Y_j. Determinare W.

$W = 19$

(**Solution strategy:**) In order to obtain the values of all the $\ell(i,j)$'s, we run the LCS algorithm to obtain the final matrix $L[i,j] = \ell(i,j)$. We have

$$L = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 2 & 2 \\ 0 & 1 & 2 & 2 & 3 \end{pmatrix}, \text{ whence } W = 19.$$

5. Si consideri un insieme di 6 attività a_i, $1 \leq i \leq 6$, caratterizzate dai seguenti vettori $\boldsymbol{s}$ e $\boldsymbol{f}$ di tempi di inizio e fine:

$$\boldsymbol{s} = (2, 3, 4, 2, 1, 9) \text{ e } \boldsymbol{f} = (4, 7, 8, 8, 10, 11)$$

L'insieme S delle attività compatibili selezionate da GREEDY-ACTIVITY-SELECTOR($\boldsymbol{s}, \boldsymbol{f}$) è:

$S = \{a_1, a_3, a_6\}$

(**Solution strategy:**) The answer follows by executing the algorithm on the given start and end vectors.

v2

Dati e Algoritmi 2 – Ingegneria Informatica
Compito, 19/3/2008 (Durata: 3h)

Nome, Cognome, Matricola: _____

Corso di studio: _____ ☐ Triennale ☐ Specialistica

Prima Parte: domande a risposta unica

Si forniscano negli appositi spazi **solo le risposte** alle seguenti domande:

1. Utilizzando il Master Theorem, si determini l'ordine di grandezza associato alla ricorrenza $T(n) = 256 \cdot T(n/4) + n^{7/2} \log^5 n$

   $T(n) = \Theta(n^4)$

   (**Solution strategy:**) In the above recurrence, we have $a = 256$ and $b = 4$, hence $\log_b a = 4$. For $\epsilon = 1/4$, we have $n^{7/2} \log^5 n = O(n^{4-\epsilon}) = O(n^{7/2} n^{1/4})$, hence we are in the first case of the Master Theorem. Therefore $T(n) = \Theta(n^{\log_b a}) = \Theta(n^4)$, and the answer follows.

2. Sia $\boldsymbol{x} = (2, 7, 2, 0, 2, 0, 2, 0)$. Si elenchino in **ordine row-major** gli elementi della matrice $X \in {}_4\boldsymbol{C}_2$ risultante dopo l'applicazione del Passo 2 (trasformazione delle colonne) dell'algoritmo di Cooley-Tukey con $p = 4$ e $q = 2$.

   $\text{row-major}(X) = (8, 7, 0, 7, 0, 7, 0, 7)$

   (**Solution strategy:**) After folding $\boldsymbol{x}$ into the 4×2 matrix $X = \begin{pmatrix} 2 & 7 \\ 2 & 0 \\ 2 & 0 \\ 2 & 0 \end{pmatrix}$, Step 2 of Cooley-Tukey requires transforming the two 4×1 columns of X separately. From the two closed-form transforms: $DFT_4(\ (y, 0, 0, 0)\ ) = (y, y, y, y)$ and $DFT_4((x, x, x, x)) = (4x, 0, 0, 0)$, we obtain the following new matrix: $X = \begin{pmatrix} 8 & 7 \\ 0 & 7 \\ 0 & 7 \\ 0 & 7 \end{pmatrix}$, whence $\text{row-major}(X) = (8, 7, 0, 7, 0, 7, 0, 7)$.

3. Si consideri la seguente doppia sommatoria:

$$S(n) = \sum_{i=4}^{n} \sum_{j=1}^{n-i+1} 1.$$

   Si valuti $S(10)$.

$S(10) = 28$

(**Solution strategy:**) We have:

$$\begin{aligned} S(n) &= \sum_{i=4}^{n} \sum_{j=1}^{n-i+1} 1 \\ &= \sum_{i=4}^{n} (n-i+1) \\ &= \sum_{k=1}^{n-3} k \\ &= (n-3)(n-2)/2 \end{aligned}$$

For $n = 10$, we obtain $S(10) = 7 \cdot 8/2 = 28$.

4. Sia $X = \langle \mathrm{e}, \mathrm{m}, \mathrm{o} \rangle$ e $Y = \langle \mathrm{e}, \mathrm{l}, \mathrm{m}, \mathrm{o} \rangle$. Si definisca $W = \sum_{i=0}^{3} \sum_{j=0}^{4} \ell(i,j)$, dove per $0 \leq i \leq 3$ e $0 \leq j \leq 4$, $\ell(i,j)$ denota la lunghezza di una LCS dei prefissi X_i e Y_j. Determinare W.

$W = 17$

(**Solution strategy:**) In order to obtain the values of all the $\ell(i,j)$'s, we run the LCS algorithm to obtain the final matrix $L[i,j] = \ell(i,j)$. We have

$$L = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 2 & 2 \\ 0 & 1 & 1 & 2 & 3 \end{pmatrix}, \text{ whence } W = 17.$$

5. Si consideri un insieme di 6 attività a_i, $1 \leq i \leq 6$, caratterizzate dai seguenti vettori $\boldsymbol{s}$ e $\boldsymbol{f}$ di tempi di inizio e fine:

$$\boldsymbol{s} = (3, 4, 6, 8, 9, 10) \text{ e } \boldsymbol{f} = (9, 9, 10, 12, 13, 14)$$

L'insieme S delle attività compatibili selezionate da GREEDY-ACTIVITY-SELECTOR($\boldsymbol{s}, \boldsymbol{f}$) è:

$S = \{a_1, a_5\}$

(**Solution strategy:**) The answer follows by executing the algorithm on the given start and end vectors.

v3

Dati e Algoritmi 2 – Ingegneria Informatica
Compito, 19/3/2008 (Durata: 3h)

Nome, Cognome, Matricola: _____

Corso di studio: _____ ☐ Triennale ☐ Specialistica

Prima Parte: domande a risposta unica

Si forniscano negli appositi spazi **solo le risposte** alle seguenti domande:

1. Utilizzando il Master Theorem, si determini l'ordine di grandezza associato alla ricorrenza $T(n) = 9 \cdot T(n/3) + n \log^3 n$

   $T(n) = \Theta(n^2)$

   (**Solution strategy:**) In the above recurrence, we have $a = 9$ and $b = 3$, hence $\log_b a = 2$. For $\epsilon = 1/2$, we have $n \log^3 n = O(n^{2-\epsilon}) = O(nn^{1/2})$, hence we are in the first case of the Master Theorem. Therefore $T(n) = \Theta(n^{\log_b a}) = \Theta(n^2)$, and the answer follows.

2. Sia $\boldsymbol{x} = (8, 9, 8, 9, 6, 7, 6, 7)$. Si elenchino in **ordine column-major** gli elementi della matrice $X \in {}_2\boldsymbol{C}_4$ risultante dopo l'applicazione del Passo 2 (trasformazione delle colonne) dell'algoritmo di Cooley-Tukey con $p = 2$ e $q = 4$.

   column-major$(X) = (14, 2, 16, 2, 14, 2, 16, 2)$

   (**Solution strategy:**) After folding $\boldsymbol{x}$ into the 2×4 matrix $X = \begin{pmatrix} 8 & 9 & 8 & 9 \\ 6 & 7 & 6 & 7 \end{pmatrix}$, Step 2 of Cooley-Tukey requires transforming the four 2×1 columns of X separately. From the closed-form transform: $DFT_2(\,(x, y)\,) = (x + y, x - y)$, we obtain the following new matrix: $X = \begin{pmatrix} 14 & 16 & 14 & 16 \\ 2 & 2 & 2 & 2 \end{pmatrix}$, whence column-major$(X) = (14, 2, 16, 2, 14, 2, 16, 2)$.

3. Si consideri la seguente doppia sommatoria:

$$S(n) = \sum_{i=1}^{n+1} \sum_{j=1}^{n-i+2} 1.$$

   Si valuti $S(12)$.

   $S(12) = 91$

(**Solution strategy:**) We have:

$$\begin{aligned} S(n) &= \sum_{i=1}^{n+1} \sum_{j=1}^{n-i+2} 1 \\ &= \sum_{i=1}^{n+1} (n-i+2) \\ &= \sum_{k=1}^{n+1} k \\ &= (n+1)(n+2)/2 \end{aligned}$$

For $n = 12$, we obtain $S(12) = 13 \cdot 14/2 = 91$.

4. Sia $X = \langle \mathrm{a}, \mathrm{r}, \mathrm{m}, \mathrm{o} \rangle$ e $Y = \langle \mathrm{o}, \mathrm{r}, \mathrm{o} \rangle$. Si definisca $W = \sum_{i=0}^{4} \sum_{j=0}^{3} \ell(i,j)$, dove per $0 \leq i \leq 4$ e $0 \leq j \leq 3$, $\ell(i,j)$ denota la lunghezza di una LCS dei prefissi X_i e Y_j. Determinare W.

$W = 8$

(**Solution strategy:**) In order to obtain the values of all the $\ell(i,j)$'s, we run the LCS algorithm to obtain the final matrix $L[i,j] = \ell(i,j)$. We have

$$L = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \end{pmatrix}, \text{ whence } W = 8.$$

5. Si consideri un insieme di 6 attività a_i, $1 \leq i \leq 6$, caratterizzate dai seguenti vettori $\boldsymbol{s}$ e $\boldsymbol{f}$ di tempi di inizio e fine:

$$\boldsymbol{s} = (1, 5, 4, 6, 8, 10) \text{ e } \boldsymbol{f} = (6, 7, 8, 9, 9, 11)$$

L'insieme S delle attività compatibili selezionate da GREEDY-ACTIVITY-SELECTOR($\boldsymbol{s}, \boldsymbol{f}$) è:

$S = \{a_1, a_4, a_6\}$

(**Solution strategy:**) The answer follows by executing the algorithm on the given start and end vectors.

Dati e Algoritmi 2 – Ingegneria Informatica
Compito, 19/3/2008 (Durata: 3h)

Nome, Cognome, Matricola: _____

Corso di studio: _____ ☐ Triennale ☐ Specialistica

Prima Parte: domande a risposta unica

Si forniscano negli appositi spazi **solo le risposte** alle seguenti domande:

1. Utilizzando il Master Theorem, si determini l'ordine di grandezza associato alla ricorrenza $T(n) = 32 \cdot T(n/2) + n^{13/3} \log^4 n$

   $T(n) = \Theta(n^5)$

   (**Solution strategy:**) In the above recurrence, we have $a = 32$ and $b = 2$, hence $\log_b a = 5$. For $\epsilon = 1/3$, we have $n^{13/3} \log^4 n = O(n^{5-\epsilon}) = O(n^{13/3} n^{1/3})$, hence we are in the first case of the Master Theorem. Therefore $T(n) = \Theta(n^{\log_b a}) = \Theta(n^5)$, and the answer follows.

2. Sia $\boldsymbol{x} = (3, 3, 2, 2, 2, 2, 3, 3)$. Si elenchino in **ordine row-major** gli elementi della matrice $X \in {}_2\boldsymbol{C}_4$ risultante dopo l'applicazione del Passo 2 (trasformazione delle colonne) dell'algoritmo di Cooley-Tukey con $p = 2$ e $q = 4$.

   $\text{row-major}(X) = (5, 5, 5, 5, 1, 1, -1, -1)$

   (**Solution strategy:**) After folding $\boldsymbol{x}$ into the 2×4 matrix $X = \begin{pmatrix} 3 & 3 & 2 & 2 \\ 2 & 2 & 3 & 3 \end{pmatrix}$, Step 2 of Cooley-Tukey requires transforming the four 2×1 columns of X separately. From the closed-form transform: $DFT_2(\,(x, y)\,) = (x + y, x - y)$, we obtain the following new matrix: $X = \begin{pmatrix} 5 & 5 & 5 & 5 \\ 1 & 1 & -1 & -1 \end{pmatrix}$, whence $\text{row-major}(X) = (5, 5, 5, 5, 1, 1, -1, -1)$.

3. Si consideri la seguente doppia sommatoria:

$$S(n) = \sum_{i=2}^{n-4} \sum_{j=1}^{n-i-3} 1.$$

   Si valuti $S(16)$.

   $S(16) = 66$

(**Solution strategy:**) We have:

$$\begin{aligned} S(n) &= \sum_{i=2}^{n-4} \sum_{j=1}^{n-i-3} 1 \\ &= \sum_{i=2}^{n-4} (n-i-3) \\ &= \sum_{k=1}^{n-5} k \\ &= (n-5)(n-4)/2 \end{aligned}$$

For $n = 16$, we obtain $S(16) = 11 \cdot 12/2 = 66$.

4. Sia $X = \langle \mathrm{o}, \mathrm{s}, \mathrm{t}, \mathrm{e} \rangle$ e $Y = \langle \mathrm{e}, \mathrm{t}, \mathrm{e} \rangle$. Si definisca $W = \sum_{i=0}^{4} \sum_{j=0}^{3} \ell(i,j)$, dove per $0 \leq i \leq 4$ e $0 \leq j \leq 3$, $\ell(i,j)$ denota la lunghezza di una LCS dei prefissi X_i e Y_j. Determinare W.

$W = 6$

(**Solution strategy:**) In order to obtain the values of all the $\ell(i,j)$'s, we run the LCS algorithm to obtain the final matrix $L[i,j] = \ell(i,j)$. We have

$$L = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \end{pmatrix}, \text{ whence } W = 6.$$

5. Si consideri un insieme di 6 attività a_i, $1 \leq i \leq 6$, caratterizzate dai seguenti vettori $\boldsymbol{s}$ e $\boldsymbol{f}$ di tempi di inizio e fine:

$$\boldsymbol{s} = (2,1,2,3,4,7) \text{ e } \boldsymbol{f} = (4,4,6,11,12,13)$$

L'insieme S delle attività compatibili selezionate da GREEDY-ACTIVITY-SELECTOR($\boldsymbol{s}, \boldsymbol{f}$) è:

$S = \{a_1, a_5\}$

(**Solution strategy:**) The answer follows by executing the algorithm on the given start and end vectors.

Seconda Parte: risoluzione di problemi

Esercizio 1 [15 punti]

- **Punto 1 [9 punti]** Sia $n > 0$ una potenza di due. Si fornisca lo pseudocodice e si analizzi la correttezza e la complessità (come numero di operazioni tra scalari complessi) di un algoritmo CYCL_CUBIC($\boldsymbol{a}$) che, dato in ingresso un vettore $\boldsymbol{a} \in \boldsymbol{C}^n$, restituisca in uscita un vettore $\boldsymbol{x} \in \boldsymbol{C}^n$ tale che $(\boldsymbol{x} \circledast \boldsymbol{x}) \circledast \boldsymbol{x} = \boldsymbol{a}$. Nel codice, si supponga di avere a disposizione la routine CUBIC(z) che, dato un valore $z \in \boldsymbol{C}$ ritorna in tempo unitario le sue tre radici cubiche $\{z_1, z_2, z_3\}$ (si noti che per $z = 0$, $z_1 = z_2 = z_3 = 0$).
- **Punto 2 [6 punti]** Si fornisca lo pseudocodice e si analizzi un algoritmo NUM_CYCL_CUBIC($\boldsymbol{a}$) che restituisca $\left|\left\{\boldsymbol{x} \in \boldsymbol{C}^n : (\boldsymbol{x} \circledast \boldsymbol{x}) \circledast \boldsymbol{x} = \boldsymbol{a}\right\}\right|$.

Answer: Point 1 Let $\boldsymbol{x} \in \boldsymbol{C}^n$ be any vector such that $(\boldsymbol{x} \circledast \boldsymbol{x}) \circledast \boldsymbol{x} = \boldsymbol{a}$. Let $\boldsymbol{X} = DFT_n(\boldsymbol{x})$ and $\boldsymbol{A} = DFT_n(\boldsymbol{a})$. By applying the cyclic convolution theorem twice, we have:

$$\begin{aligned}
&(\boldsymbol{x} \circledast \boldsymbol{x}) \circledast \boldsymbol{x} = \boldsymbol{a} \\
&\quad \Leftrightarrow \quad DFT_n^{-1}\left(DFT_n(\boldsymbol{x} \circledast \boldsymbol{x}) \odot DFT_n(\boldsymbol{x})\right) = \boldsymbol{a} \\
&\quad \Leftrightarrow \quad DFT_n(\boldsymbol{x} \circledast \boldsymbol{x}) \odot \boldsymbol{X} = \boldsymbol{A} \\
&\quad \Leftrightarrow \quad \boldsymbol{X} \odot \boldsymbol{X} \odot \boldsymbol{X} = \boldsymbol{A},
\end{aligned}$$

where $\odot$ denotes componentwise product. Therefore, for $0 \leq i < n$, it must be $(X_i)^3 = A_i$, which is satisfied by assigning any cubic root of A_i to X_i. The output vector $\boldsymbol{x}$ can then be obtained as $DFT_n^{-1}(\boldsymbol{X})$. The code follows.

```
CYCL_CUBIC(a)
n ← length(a)
A ← FFT(a)
for i ← 0 to n − 1 do
    {z1, z2, z3} ← CUBIC(Ai)
    Xi ← z1
return INV_FFT(X)
```

The correctness of CYCL_CUBIC follows from the previous discussion, and its running time $T_{\mathrm{CC}}(n)$ is clearly dominated by the two calls to FFT and INV_FFT on vectors of size n. Hence $T_{\mathrm{CC}}(n) = \Theta(n \log n)$.

Point 2 Since each nonzero complex has three distinct cubic roots, we observe from Point 1 that for each component $0 \leq i < n$ such that $A_i \neq 0$, we have three

distinct choices for X_i, while the choice is fixed ($X_i = 0$) for $A_i = 0$. Let k be the number of nonzero components of $\boldsymbol{A} = DFT_n(\boldsymbol{a})$. Since DFT_n is a bijection, any of the 3^k possible combinations yields a distinct vector $\boldsymbol{x} = DFT_n^{-1}(\boldsymbol{X})$ satisfying the constraint $(\boldsymbol{x} \circledast \boldsymbol{x}) \circledast \boldsymbol{x} = \boldsymbol{a}$. The pseudocode follows.

```
NUM_CYCL_CUBIC(a)
n ← length(a)
A ← FFT(a)
s ← 1
for i ← 0 to n − 1 do
    if (A_i ≠ 0) then s ← 3 · s
return s
```

The correctness of NUM_CYCL_CUBIC follows from the previous discussion, and from the fact that at the end of iteration i, with $0 \leq i < n$, $s = 3^h$, where $h = |\{j : (0 \leq j \leq i) \wedge (A_j \neq 0)\}|$. Its running time $T_{\text{NCC}}(n)$ is clearly dominated by the one call to FFT on a vector of size n. Hence $T_{\text{NCC}}(n) = \Theta(n \log n)$. □

Esercizio 2 [15 punti]

- **Punto 1 [10 punti]** Data una stringa di numeri complessi $A = \langle a_1, a_2, \ldots, a_n \rangle$, si consideri la seguente ricorrenza $z(i, j)$ definita per ogni coppia di valori (i, j), con $1 \leq i, j \leq n$:

$$z(i,j) = \begin{cases} a_j, & i = 1, 1 \leq j \leq n, \\ a_{n+1-i}, & j = n, 1 < i \leq n, \\ (z(i-1,j) \cdot z(i,j+1)) - z(i-1,j+1), & \text{altrimenti.} \end{cases}$$

  Si fornisca lo pseudocodice e si provi la correttezza di un algoritmo iterativo **bottom-up** COMPUTE_Z(A) che, data in ingresso la stringa A restituisca in uscita il valore $z(n, 1)$.

- **Punto 2 [5 punti]** Si valuti il **numero esatto** $T_{\text{CZ}}(n)$ di operazioni tra complessi eseguite dall'algoritmo sviluppato al Punto 1.

Answer: Point 1 Given the dependencies between the indices in the recurrence, one of the correct scans is a **reverse column-major** one, where we compute entries by decreasing colum index and, within the same column, by increasing row index. In order to prove that such a scan is correct, it is sufficient to observe that for $n - 1 \geq j \geq 1$ and $2 \leq i \leq n$ it guarantees that the values $z(i-1, j)$ (same column index, lower row index), $z(i, j+1)$ (higher column, same row), and $z(i-1, j+1)$ (higher column, lower row), are computed prior to compute $z(i, j)$, which needs these values. The code follows.

```
COMPUTE_Z(A)
n ← length(A)
for i ← 1 to n do
    z[1, i] ← a_i; z[i, n] ← a_{n+1−i}
for j ← n − 1 downto 1 do
    for i ← 2 to n do
        z[i, j] ← (z[i − 1, j] · z[i, j + 1]) − z[i − 1, j + 1]
return z[n, 1]
```

Observe that a **reverse diagonal** scan (scanning by diagonals parallel to the main diagonal, starting from the one containing only $z[1, n]$ and going backwards up to the one containing only $z[n, 1]$) would also yield a correct algorithm.

Point 2 Each iteration of the double nested loop of Algorithm COMPUTE_Z performs two arithmetic operations between complexes, therefore the running time of the algorithm is

$$\begin{aligned} T_{\mathrm{CZ}}(n) &= \sum_{j=1}^{n-1} \sum_{i=2}^{n} 2 \\ &= \sum_{j=1}^{n-1} 2(n-1) \\ &= 2(n-1)^2. \end{aligned}$$

We could have reached the same conclusion by observing that the algorithm performs two operations for each of the entries of a square matrix of size $(n-1) \times (n-1)$. □

v1

Dati e Algoritmi 2 – Ingegneria Informatica
Compito, 31/3/2008 (Durata: 3h)

Nome, Cognome, Matricola: _____

Corso di studio: _____ ☐ Triennale ☐ Specialistica

Prima Parte: domande a risposta unica

Si forniscano negli appositi spazi **solo le risposte** alle seguenti domande:

1. Sia $\Pi = \mathcal{I} \times \mathcal{S}$ il problema della risoluzione di equazioni di primo grado $ax + b = 0$, con a e b coefficienti reali **negativi** (ovvero, $a, b \in \boldsymbol{R}^-$). I **più piccoli** insiemi $\mathcal{I}$ e $\mathcal{S}$ che caratterizzano istanze e soluzioni di Π sono:

   $\boxed{\mathcal{I} = \boldsymbol{R}^- \times \boldsymbol{R}^-, \mathcal{S} = \boldsymbol{R}^-}$

   (**Solution strategy:**) The generic instance is **any** pair of negative reals (a, b), which are the coefficients of the equation to be solved. The solution to such equation is $x = -b/a$, which can be an arbitrary negative real. Therefore, $\mathcal{I} = \boldsymbol{R}^- \times \boldsymbol{R}^-$, and $\mathcal{S} = \boldsymbol{R}^-$.

2. Utilizzando l'induzione parametrica, si individui la **minima** costante $c > 0$ per cui risulti essere $T(n) \leq cn$, per $n \geq 1$, dove

   $$T(n) = \begin{cases} 3 & 1 \leq n \leq 7 \\ T(\lfloor n/6 \rfloor) + 2T(\lfloor n/4 \rfloor) + 18n & n \geq 8 \end{cases}$$

   $\boxed{c = 54}$

   (**Solution strategy:**) From the (parametric) induction base, we obtain $c \geq 3$. When going from inductive hypothesis to thesis, we get:

   $$T(n) \leq c(n/6) + 2c(n/4) + 18n \boxed{\leq} cn \Leftrightarrow cn/3 \geq 18n \Leftrightarrow c \geq 54$$

   Hence $c \geq \max\{3, 54\} = 54$.

3. Sia $\boldsymbol{x} = (5, -1, -1, -1, -7, -1, -1, -1)$ e sia $\boldsymbol{y} = DFT_8(\boldsymbol{x})$. Allora:

   $\boxed{\boldsymbol{y} = (-8, 12, 0, 12, 0, 12, 0, 12)}$

(**Solution strategy:**) Using Cooley-Tukey with $n = 2 \times 4$, the first column transform involves vector $(5, -7)$, yielding the new column $(-2, 12)$, while the remaining three column transforms are of vector $(-1, -1)$, yielding new columns $(-2, 0)$. Multiplication by the twiddle-factors is empty. Then the transform for the first row yields new row $(-8, 0, 0, 0)$, while the transform of the second row yields new row $(12, 12, 12, 12)$. By reading in column-major order, we obtain $y = (-8, 12, 0, 12, 0, 12, 0, 12)$.

4. Si consideri la seguente doppia sommatoria, definita per valori **dispari** di n:

$$S(n) = \sum_{i=1}^{(n-1)/2} \sum_{j=i+2}^{n+1-i} 2$$

Si valuti $S(17)$.

$$\boxed{S(17) = 128}$$

(**Solution strategy:**) We have:

$$\begin{aligned} S(n) &= \sum_{i=1}^{(n-1)/2} \sum_{j=i+2}^{n+1-i} 2 \\ &= 2 \sum_{i=1}^{(n-1)/2} (n - 2i) \\ &= 2n(n-1)/2 - 2 \cdot 2[(n-1)/2 \cdot (n+1)/2]/2 \\ &= n(n-1) - (n-1)(n+1)/2 \\ &= (n-1)^2/2 \end{aligned}$$

For $n = 17$, we obtain $S(17) = 16^2/2 = 256/2 = 128$.

5. Dato un file sull'alfabeto $\Sigma = \{\text{'A'}, \text{'B'}, \text{'C'}, \text{'D'}, \text{'E'}\}$ con frequenze 'A':0.09, 'B':0.11, 'C':0.15, 'D':0.25, 'E':0.40, si determinino le 5 codeword ottenute con la codifica di Huffman vista in classe.

$$\boxed{e(\text{'A'}) = 1110, e(\text{'B'}) = 1111, e(\text{'C'}) = 110, e(\text{'D'}) = 10, e(\text{'E'}) = 0}$$

(**Solution strategy:**) The answer follows from executing the algorithm for the given frequencies.

v2

Dati e Algoritmi 2 – Ingegneria Informatica
Compito, 31/3/2008 (Durata: 3h)

Nome, Cognome, Matricola: _____

Corso di studio: _____ ☐ Triennale ☐ Specialistica

Prima Parte: domande a risposta unica

Si forniscano negli appositi spazi **solo le risposte** alle seguenti domande:

1. Sia $\Pi = \mathcal{I} \times \mathcal{S}$ il problema della risoluzione di equazioni di primo grado $ax + b = 0$, con a e b coefficienti reali **positivi** (ovvero, $a, b \in \boldsymbol{R}^+$). I **più piccoli** insiemi $\mathcal{I}$ e $\mathcal{S}$ che caratterizzano istanze e soluzioni di Π sono:

   $\boxed{\mathcal{I} = \boldsymbol{R}^+ \times \boldsymbol{R}^+, \mathcal{S} = \boldsymbol{R}^-}$

   (**Solution strategy:**) The generic instance is a **any** pair of positive reals (a, b), which are the coefficients of the equation to be solved. The solution to such equation is $x = -b/a$, which can be an arbitrary negative real. Therefore, $\mathcal{I} = \boldsymbol{R}^+ \times \boldsymbol{R}^+$, and $\mathcal{S} = \boldsymbol{R}^-$.

2. Utilizzando l'induzione parametrica, si individui la **minima** costante $c > 0$ per cui risulti essere $T(n) \leq cn$, per $n \geq 1$, dove

   $$T(n) = \begin{cases} 3 & 1 \leq n \leq 7 \\ T(\lfloor n/6 \rfloor) + 2T(\lfloor n/4 \rfloor) + 14n & n \geq 8 \end{cases}$$

   $\boxed{c = 42}$

   (**Solution strategy:**) From the (parametric) induction base, we obtain $c \geq 3$. When going from inductive hypothesis to thesis, we get:

   $$T(n) \leq c(n/6) + 2c(n/4) + 14n \boxed{\leq} cn \Leftrightarrow cn/3 \geq 14n \Leftrightarrow c \geq 42$$

   Hence $c \geq \max\{3, 42\} = 42$.

3. Sia $\boldsymbol{x} = (-5, -2, -2, -2, 1, -2, -2, -2)$ e sia $\boldsymbol{y} = DFT_8(\boldsymbol{x})$. Allora:

   $\boxed{\boldsymbol{y} = (-16, -6, 0, -6, 0, -6, 0, -6)}$

(**Solution strategy:**) Using Cooley-Tukey with $n = 2 \times 4$, the first column transform involves vector $(-5, 1)$, yielding the new column $(-4, -6)$, while the remaining three column transforms are of vector $(-2, -2)$, yielding new columns $(-4, 0)$. Multiplication by the twiddle-factors is empty. Then the transform for the first row yields new row $(-16, 0, 0, 0)$, while the transform of the second row yields new row $(-6, -6, -6, -6)$. By reading in column-major order, we obtain $y = (-16, -6, 0, -6, 0, -6, 0, -6)$.

4. Si consideri la seguente doppia sommatoria, definita per valori **dispari** di n:

$$S(n) = \sum_{i=1}^{(n-1)/2} \sum_{j=i+2}^{n+1-i} 2$$

Si valuti $S(13)$.

$$\boxed{S(13) = 72}$$

(**Solution strategy:**) We have:

$$\begin{aligned} S(n) &= \sum_{i=1}^{(n-1)/2} \sum_{j=i+2}^{n+1-i} 2 \\ &= 2 \sum_{i=1}^{(n-1)/2} (n - 2i) \\ &= 2n(n-1)/2 - 2 \cdot 2[(n-1)/2 \cdot (n+1)/2]/2 \\ &= n(n-1) - (n-1)(n+1)/2 \\ &= (n-1)^2/2 \end{aligned}$$

For $n = 13$, we obtain $S(13) = 12^2/2 = 144/2 = 72$.

5. Dato un file sull'alfabeto $\Sigma = \{\text{'A'}, \text{'B'}, \text{'C'}, \text{'D'}, \text{'E'}\}$ con frequenze 'A':0.12, 'B':0.11, 'C':0.53, 'D':0.10, 'E':0.14, si determinino le 5 codeword ottenute con la codifica di Huffman vista in classe.

$$\boxed{e(\text{'A'}) = 010, e(\text{'B'}) = 001, e(\text{'C'}) = 1, e(\text{'D'}) = 000, e(\text{'E'}) = 011}$$

(**Solution strategy:**) The answer follows from executing the algorithm for the given frequencies.

Seconda Parte: risoluzione di problemi

Esercizio 1 [15 punti] Sia $n > 0$. Un vettore $\boldsymbol{x} \in \boldsymbol{C}^n$ si dice **antipodo** se $\boldsymbol{x}^R = \boldsymbol{x}$ (si ricorda che $x_i^R = x_{(n-i) \bmod n}$, $0 \leq i < n$).

- **Punto 1 [10 punti]** Si dimostri che la trasformata $\boldsymbol{y} = DFT_n(\boldsymbol{x})$ di un vettore antipodo $\boldsymbol{x}$ è anch'essa un vettore antipodo (ovvero, $\boldsymbol{y}^R = \boldsymbol{y}$).
- **Punto 2 [5 punti]** Si dimostri che per un vettore antipodo $\boldsymbol{x}$ vale: $DFT_n^{-1}(\boldsymbol{x}) = (1/n)DFT_n(\boldsymbol{x})$

Answer: Point 1 Let $\boldsymbol{x}$ be an antipode, and let $\boldsymbol{y} = DFT_n(\boldsymbol{x})$. We have to show that $\boldsymbol{y}^R = \boldsymbol{y}$, that is, for $0 \leq i < n$, $y_i = y_{(n-i) \bmod n}$. Recall that by the definition of DFT_n we have:

$$y_i = \sum_{j=0}^{n-1} x_j \omega_n^{ij}.$$

We transform the index in the above summation as follows. Let $s = (n-j) \bmod n$. When j ranges between 0 and $n-1$, so does s. Moreover, the expression can be easily inverted to yield $j = (n-s) \bmod n$, since $s = 0$ yields $j = 0 = (n-0) \bmod n$ and otherwise $0 < s = n - j < n$, hence $j = n - s = (n-s) \bmod n$. Therefore we obtain:

$$\begin{aligned}
y_i &= \sum_{s=0}^{n-1} x_{(n-s) \bmod n} \omega_n^{i[(n-s) \bmod n]} \\
&= \sum_{s=0}^{n-1} x_s \omega_n^{ni-is} = \sum_{s=0}^{n-1} x_s \omega_n^{-is} \\
&= \text{(since } \boldsymbol{x} \text{ is an antipode and the integer powers of } \omega_n \text{ are periodic)} \\
&= \sum_{s=0}^{n-1} x_s \omega_n^{(n-i)s} = \sum_{s=0}^{n-1} x_s \omega_n^{[(n-i) \bmod n]s} \\
&= y_{(n-i) \bmod n},
\end{aligned}$$

which proves that $\boldsymbol{y} = DFT_n(\boldsymbol{x})$ is itself an antipode.

Point 2 By the relation between direct and inverse DFT and the result in Point 1 we have

$$\begin{aligned}
DFT_n^{-1}(\boldsymbol{x}) &= (1/n)[DFT_n(\boldsymbol{x})]^R \\
&= (1/n)[DFT_n(\boldsymbol{x})],
\end{aligned}$$

which proves the statement. □

Esercizio 2 [15 punti] Sia $n > 0$. Dato un vettore di numeri reali positivi $\boldsymbol{a} = (a_0, a_1, \ldots, a_{n-1})$, si consideri la seguente ricorrenza $f(i,j)$ definita per ogni coppia di valori (i,j), con $1 \leq i \leq j \leq n$:

$$f(i,j) = \begin{cases} a_{j-i} & j - i = 0, 1 \\ f(i, j-1) + f(i+1, j-1) + a_{j-i} & \text{altrimenti.} \end{cases}$$

- **Punto 1 [10 punti]** Si fornisca lo pseudocodice di una coppia di routine INIT_F($\boldsymbol{a}$) e REC_F($\boldsymbol{a}, i, j$) per il calcolo memoizzato di $f(1,n)$.
- **Punto 2 [5 punti]** Si determini il **numero esatto** di addizioni tra reali svolte dall'algoritmo memoizzato per il calcolo di $f(1,n)$ quando n è un numero **dispari**.

Answer: Point 1 We first discuss a straightforward memoized implementation, upon which we will improve considerably later. Let us make use of a two-dimensional table F, with the idea that whenever $f(i,j)$ is computed for the first time, with $1 \leq i \leq j \leq n$, its value will be stored in $F[i,j]$ for eventual subsequent uses (observe that entries $F[i,j]$ with $i > j$ will never be set). INIT_F($\boldsymbol{a}$) must return values for the base cases, initialize the table, and finally call REC_F($\boldsymbol{a}, 1, n$). The latter call will recursively compute the required value $f(1,n)$. Since $f(i,j)$ is a positive real for all values of i and j, with $1 \leq i \leq j \leq n$, the value 0 can be used as a default value to indicate that a certain value $f(i,j)$ is yet to be computed. The code follows.

```
INIT_F(a)                                  REC_F(a, i, j)
n ← length(a)                              if F[i, j] = 0 then
if n ≤ 2 then return a_{n-1}                  a ← REC_F(a, i, j − 1)
for i = 1 to n − 1 do                         b ← REC_F(a, i + 1, j − 1)
    F[i, i] ← a_0; F[i, i + 1] ← a_1          F[i, j] ← a + b + a_{j−i}
F[n, n] ← a_0                              return F[i, j]
for i = 2 to n − 2 do
    for j = i + 2 to n do F[i, j] ← 0
return REC_F(a, 1, n)
```

Point 2 Observe that for odd $n > 2$, when computing $f(1,n)$, all and only nonbase instances (i,j) used are those with $i + 2 \leq j \leq n + 1 - i$, since all other nonbase instances can never be reached from $(1,n)$ by incrementing the first index by one and either keeping the second index or decrementing it by one. The maximum value of i for which a nonbase instance is used can thus be obtained by setting $i + 2 = n + 1 - i$, which yields $i = (n-1)/2$ (a positive integer for odd $n > 1$). For each such instance (i,j), REC_F executes exactly two additions between reals

in the *only* call REC_F($\boldsymbol{a}, i, j$) which finds $F[i, j] = 0$. Since INIT_F($\boldsymbol{a}$) does not perform any addition, the exact number of additions performed is

$$T(n) = \sum_{i=1}^{(n-1)/2} \sum_{j=i+2}^{n+1-i} 2.$$

Therefore, by the derivation made in Exercise 4 of Part 1, we obtain $T(n) = (n-1)^2/2$ for odd $n > 2$ and 0 for $n = 1$.

A linear-time algorithm A much better algorithm for computing $f(1, n)$ can be obtained by observing that for each ℓ, $0 \leq \ell < n$, and for any two values i, i', with $1 \leq i, i' \leq n - \ell$, we have that $f(i, i + \ell) = f(i', i' + \ell)$. In other words, all values of f on pairs (i, j) residing on the same diagonal $j - i = \ell$ are the same. (The proof is a simple induction on ℓ, and is left to the reader). Therefore, it suffices to compute a single value $d(\ell)$ for each such diagonal, which will obey the following recurrence:

$$d(\ell) = \begin{cases} a_\ell & \ell = 0, 1 \\ d(\ell - 1) + d(\ell - 2) + a_\ell & \text{otherwise.} \end{cases}$$

The above recurrence is immediately obtained from the recurrence on f by observing that whenever (i, j) is such that $(j - i) = \ell$, then $((j - 1) - i) = \ell - 1$ and $((j - 1) - (i + 1)) = \ell - 2$. The memoized algorithm follows.

```
INIT_F2(a)
n ← length(a)
if n ≤ 2 then return a_{n-1}
d[0] ← a_0; d[1] ← a_1
for ℓ ← 2 to n − 1 do
    d[ℓ] ← 0
return REC_F2(a, n − 1)
```

```
REC_F2(a, ℓ)
if d[ℓ] = 0 then
   a ← REC_F2(a, ℓ − 1)
   b ← REC_F2(a, ℓ − 2)
   d[ℓ] ← a + b + a_ℓ
return d[ℓ]
```

The correctness of the above algorithm follows from the previous considerations. As for its running time, oberve that for $n > 2$ the recursion tree associated to the call REC_F2($\boldsymbol{a}, n - 1$) has one internal node for each instance $(\boldsymbol{a}, \ell)$, with $2 \leq \ell \leq n - 1$, each contributing two additions, for a total of $2(n - 2)$ additions. □

v1

Dati e Algoritmi 2 – Ingegneria Informatica
Compito, 18/6/2008 (Durata: 3h)

Nome, Cognome, Matricola: _____

Corso di studio: _____ ☐ Triennale ☐ Specialistica

Prima Parte: domande a risposta unica

Si forniscano negli appositi spazi **solo le risposte** alle seguenti domande:

1. Sia $\Pi = \mathcal{I} \times \mathcal{S}$ il problema del calcolo della convoluzione lineare di due vettori complessi, uno di lunghezza 7 e uno di lunghezza 4. Gli insiemi $\mathcal{I}$ e $\mathcal{S}$ che caratterizzano istanze e soluzioni di Π sono:

   $\boxed{\mathcal{I} = \boldsymbol{C}^7 \times \boldsymbol{C}^4, \mathcal{S} = \boldsymbol{C}^{10}}$

   (**Solution strategy:**) The generic instance is a pair of complex vectors with 7 and 4 components, respectively. The resulting convolution is a a complex vector with 10 components.

2. Utilizzando l'induzione parametrica, si individui la **minima** costante $c > 0$ per cui risulti essere $T(n) \leq cn$, per $n \geq 1$, dove

   $$T(n) = \begin{cases} 1 & 1 \leq n \leq 7 \\ T(\lfloor n/2 \rfloor) + 3T(\lfloor n/8 \rfloor) + 2n & n \geq 8 \end{cases}$$

   $\boxed{c = 16}$

   (**Solution strategy:**) From the (parametric) induction base, we obtain $c \geq 1$. When going from inductive hypothesis to thesis, we get:

   $$T(n) \leq c(n/2) + 3c(n/8) + 2n \boxed{\leq} cn \Leftrightarrow cn/8 \geq 2n \Leftrightarrow c \geq 16$$

   Hence $c \geq \max\{1, 16\} = 16$.

3. Sia $\boldsymbol{x} = (5, -3, -3, -3, -11, -3, -3, -3)$ e sia $\boldsymbol{z} = DFT_8^{-1}(\boldsymbol{x})$. Allora:

   $\boxed{\boldsymbol{z} = (-3, 2, 0, 2, 0, 2, 0, 2)}$

(**Solution strategy:**) Let us first compute $\boldsymbol{y} = DFT_8(\boldsymbol{x})$. Using Cooley-Tukey with $n = 2 \times 4$, the first column transform involves vector $(5, -11)$, yielding the new column $(-6, 16)$, while the remaining three column transforms are of vector $(-3, -3)$, yielding new columns $(-6, 0)$. Multiplication by the twiddle-factors is empty. Then the transform for the first row yields new row $(-24, 0, 0, 0)$, while the transform of the second row yields new row $(16, 16, 16, 16)$. By reading in column-major order, we obtain $\boldsymbol{y} = (-24, 16, 0, 16, 0, 16, 0, 16)$. We now can obtain $\boldsymbol{z}$ as $(1/8)\boldsymbol{y^R}$, hence $\boldsymbol{z} = (-3, 2, 0, 2, 0, 2, 0, 2)$.

4. Si consideri il seguente frammento di codice :

```
sum ← 0
for i ← 2 to n − 1 do
    for j ← i to i + 3 do
        for k ← j + 1 to 2j − 1 do
            sum ← sum + 1
```

Si calcoli il valore finale di sum quando $n = 10$.

$$sum|_{n=10} = 192$$

(**Solution strategy:**) We have

$$\begin{aligned}
sum &= \sum_{i=2}^{n-1} \sum_{j=i}^{i+3} \sum_{k=j+1}^{2j-1} 1 \\
&= \sum_{i=2}^{n-1} \sum_{j=i}^{i+3} j - 1 \\
&= \sum_{i=2}^{n-1} (i - 1 + i + i + 1 + i + 2) \\
&= \sum_{i=2}^{n-1} (4i + 2) \\
&= 2(n-1)n - 4 + 2(n-2) \\
&= 2n^2 - 8
\end{aligned}$$

Therefore, when $n = 10$, we have $sum = 192$

5. Sia $X = \langle x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8 \rangle = \langle 1, 3, 5, 7, 8, 6, 2, 4 \rangle$. Si determini $L = \sum_{i=1}^{8} \ell(i)$, dove $\ell(i) = |LMIS(i)|$ è la lunghezza di una *Longest Monotonically Increasing Subsequence* di X con primo carattere uguale a x_i, $1 \leq i \leq 8$.

$$L = 19$$

(**Solution strategy:**) The vector whose i-th value is $\ell(i)$ is $(5, 4, 3, 2, 1, 1, 2, 1)$. Hence $L = 19$.

v2

Dati e Algoritmi 2 – Ingegneria Informatica
Compito, 18/6/2008 (Durata: 3h)

Nome, Cognome, Matricola: _____

Corso di studio: _____ ☐ Triennale ☐ Specialistica

Prima Parte: domande a risposta unica

Si forniscano negli appositi spazi **solo le risposte** alle seguenti domande:

1. Sia $\Pi = \mathcal{I} \times \mathcal{S}$ il problema del calcolo della convoluzione lineare di due vettori complessi, uno di lunghezza 5 e uno di lunghezza 9. Gli insiemi $\mathcal{I}$ e $\mathcal{S}$ che caratterizzano istanze e soluzioni di Π sono:

   $$\boxed{\mathcal{I} = \boldsymbol{C}^5 \times \boldsymbol{C}^9, \mathcal{S} = \boldsymbol{C}^{13}}$$

   (**Solution strategy:**) The generic instance is a pair of complex vectors with 5 and 9 components, respectively. The resulting convolution is a a complex vector with 13 components.

2. Utilizzando l'induzione parametrica, si individui la **minima** costante $c > 0$ per cui risulti essere $T(n) \leq cn$, per $n \geq 1$, dove

   $$T(n) = \begin{cases} 1 & 1 \leq n \leq 7 \\ T(\lfloor n/2 \rfloor) + 3T(\lfloor n/8 \rfloor) + 4n & n \geq 8 \end{cases}$$

   $$\boxed{c = 32}$$

   (**Solution strategy:**) From the (parametric) induction base, we obtain $c \geq 1$. When going from inductive hypothesis to thesis, we get:

   $$T(n) \leq c(n/2) + 3c(n/8) + 4n \boxed{\leq} cn \Leftrightarrow cn/8 \geq 4n \Leftrightarrow c \geq 32$$

   Hence $c \geq \max\{1, 32\} = 32$.

3. Sia $\boldsymbol{x} = (4, -8, -8, -8, -20, -8, -8, -8)$ e sia $\boldsymbol{z} = DFT_8^{-1}(\boldsymbol{x})$. Allora:

   $$\boxed{\boldsymbol{z} = (-8, 3, 0, 3, 0, 3, 0, 3)}$$

(**Solution strategy:**) Let us first compute $\boldsymbol{y} = DFT_8(\boldsymbol{x})$. Using Cooley-Tukey with $n = 2 \times 4$, the first column transform involves vector $(4, -20)$, yielding the new column $(-16, 24)$, while the remaining three column transforms are of vector $(-8, -8)$, yielding new columns $(-16, 0)$. Multiplication by the twiddle-factors is empty. Then the transform for the first row yields new row $(-64, 0, 0, 0)$, while the transform of the second row yields new row $(24, 24, 24, 24)$. By reading in column-major order, we obtain $\boldsymbol{y} = (-64, 24, 0, 24, 0, 24, 0, 24)$. We now can obtain $\boldsymbol{z}$ as $(1/8)\boldsymbol{y}^{\boldsymbol{R}}$, hence $\boldsymbol{z} = (-8, 3, 0, 3, 0, 3, 0, 3)$.

4. Si consideri il seguente frammento di codice :

```
sum ← 0
for i ← 2 to n − 1 do
    for j ← i to i + 3 do
        for k ← j + 1 to 2j − 1 do
            sum ← sum + 1
```

Si calcoli il valore finale di sum quando $n = 11$.

$$sum|_{n=11} = 234$$

(**Solution strategy:**) We have

$$\begin{aligned} sum &= \sum_{i=2}^{n-1} \sum_{j=i}^{i+3} \sum_{k=j+1}^{2j-1} 1 \\ &= \sum_{i=2}^{n-1} \sum_{j=i}^{i+3} j - 1 \\ &= \sum_{i=2}^{n-1} (i - 1 + i + i + 1 + i + 2) \\ &= \sum_{i=2}^{n-1} (4i + 2) \\ &= 2(n-1)n - 4 + 2(n-2) \\ &= 2n^2 - 8 \end{aligned}$$

Therefore, when $n = 11$, we have $sum = 234$

5. Sia $X = \langle x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8 \rangle = \langle 3, 1, 5, 7, 8, 6, 2, 4 \rangle$. Si determini $L = \sum_{i=1}^{8} \ell(i)$, dove $\ell(i) = |LMIS(i)|$ è la lunghezza di una *Longest Monotonically Increasing Subsequence* di X con primo carattere uguale a x_i, $1 \leq i \leq 8$.

$$L = 18$$

(**Solution strategy:**) The vector whose i-th value is $\ell(i)$ is $(4, 4, 3, 2, 1, 1, 2, 1)$. Hence $L = 18$.

Seconda Parte: risoluzione di problemi

Esercizio 1 [15 punti] Siano k e n potenze di due, con $1 \leq k \leq n$. Un vettore $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1})$ si dice (n,k)-*padded* se $x_i = 0$ per $k \leq i \leq n-1$.

- **Punto 1 [9 punti]** Si fornisca lo pseudocodice e si provi la correttezza di un algoritmo PADDED_FFT$(\boldsymbol{x}, k)$, ottenuto modificando l'algoritmo ricorsivo FFT visto in classe, per il caso di vettori (n,k)-padded. (*Suggerimento*: la ricorsione deve essere in funzione del parametro k, con caso di base $k = 1$.)
- **Punto 2 [6 punti]** Si imposti e si risolva la ricorrenza sul numero di operazioni aritmetiche tra scalari complessi $T_{\mathrm{P}}(n,k)$ effettuate dall'algoritmo sviluppato al Punto 1.

Answer:

Point 1 We first observe that when $k = 1$, an (n,k)-padded vector $\boldsymbol{x}$ has only its first component (possibly) different from zero. In this case, we can compute $DFT_n(\boldsymbol{x})$ directly as $F_n \cdot \boldsymbol{x} = (x_0, x_0, \ldots, x_0)$. When $k > 1$, let $\boldsymbol{x}^{[0]} = (x_0, x_2, \ldots, x_{n-2})$ and $\boldsymbol{x}^{[1]} = (x_1, x_3, \ldots, x_{n-1})$ be the two $n/2$-vectors containing, respectively, the even-indexed and odd-indexed components of $\boldsymbol{x}$. The straightforward but crucial property upon which we will base our recursive strategy is the following: if $\boldsymbol{x}$ is (n,k)-padded, then both $\boldsymbol{x}^{[0]}$ and $\boldsymbol{x}^{[1]}$ are $(n/2, k/2)$-padded. Recall that the FFT algorithm computes $DFT_n(\boldsymbol{x})$ by recursively computing $DFT_{n/2}(\boldsymbol{x}^{[0]})$ and $DFT_{n/2}(\boldsymbol{x}^{[1]})$ and then performing another $\Theta(n)$ additional operations. When invoked on an (n,k)-padded vector, such recursive calls are needed only if $k > 1$. Otherwise, $DFT_n(\boldsymbol{x})$ is obtained directly. The algorithm follows.

PADDED_FFT$(\boldsymbol{x}, k)$
$n \leftarrow \mathrm{length}(\boldsymbol{x})$
if $k = 1$
  then for $j \leftarrow 0$ **to** $n-1$ **do** $y_j \leftarrow x_0$
       return $\boldsymbol{y}$
$\boldsymbol{x}^{[0]} \leftarrow (x_0, x_2, \ldots, x_{n-2})$
$\boldsymbol{x}^{[1]} \leftarrow (x_1, x_3, \ldots, x_{n-1})$
$\boldsymbol{y}^{[0]} \leftarrow$ PADDED_FFT$(\boldsymbol{x}^{[0]}, k/2)$
$\boldsymbol{y}^{[1]} \leftarrow$ PADDED_FFT$(\boldsymbol{x}^{[1]}, k/2)$
$om1 \leftarrow e^{2\pi i/n}$ $omj \leftarrow 1$
for $j \leftarrow 0$ **to** $n/2 - 1$ **do**
  $y_j \leftarrow y_j^{[0]} + omj \cdot y_j^{[1]}$
  $y_{j+n/2} \leftarrow y_j^{[0]} - omj \cdot y_j^{[1]}$
  $omj \leftarrow omj * om1$
return $\boldsymbol{y}$

The correctness of the above algorithm follows from the correctness of the FFT algorithm and the observations made above.

Point 2 We can write the following recurrence in n and k:

$$T_{\mathrm{P}}(n,k) = \begin{cases} 0, & k = 1, \\ 2 \cdot T_{\mathrm{P}}\left(\frac{n}{2}, \frac{k}{2}\right) + c \cdot n, & 1 < k \leq n \end{cases}$$

where c is a fixed positive constant. By unfolding, we obtain:

$$\begin{aligned} T_{\mathrm{P}}(n,k) &= 2T_{\mathrm{P}}\left(\frac{n}{2}, \frac{k}{2}\right) + cn \\ &= 2^2 T_{\mathrm{P}}\left(\frac{n}{2^2}, \frac{k}{2^2}\right) + cn + cn \\ &\vdots \\ &= 2^i T_{\mathrm{P}}\left(\frac{n}{2^i}, \frac{k}{2^i}\right) + c \cdot i \cdot n \\ &= 2^{\log k} T_{\mathrm{P}}\left(\frac{n}{2^{\log k}}, \frac{k}{2^{\log k}}\right) + cn \log k \\ &= kT_{\mathrm{P}}\left(\frac{n}{k}, 1\right) + cn \log k \\ &= cn \log k. \end{aligned}$$

Therefore, $T_{\mathrm{P}}(n) = \Theta(n \log k)$. □

Esercizio 2 [15 punti] Sia $n > 0$. Dato un insieme di numeri reali positivi e distinti $S = \{a_1, a_2, \ldots, a_n\}$, con $0 < a_i < a_j < 1$ per $1 \leq i < j \leq n$, un *(2,1)-boxing* di S è una partizione $\mathcal{P} = \{S_1, S_2, \ldots, S_k\}$ di S in k sottoinsiemi (cioè, $\bigcup_{j=1}^{k} S_j = S$ e $S_r \cap S_t = \emptyset, 1 \leq r \neq t \leq k$) che soddisfa inoltre i seguenti vincoli:

$$|S_j| \leq 2 \quad \text{e} \quad \sum_{a \in S_j} a \leq 1, \ 1 \leq j \leq k.$$

In altre parole, ogni sottoinsieme contiene al più due valori la cui somma è al più uno. Dato S, si voglia determinare un (2,1)-boxing che minimizzi il numero di sottoinsiemi della partizione.

- **Punto 1 [7 punti]** Si fornisca lo pseudocodice di un algoritmo greedy che restituisce un (2,1)-boxing ottimo in tempo lineare. (*Suggerimento*: si creino i sottoinsiemi in modo opportuno basandosi sulla sequenza ordinata).
- **Punto 2 [5 punti]** Si provi la proprietà di scelta greedy per l'algoritmo sviluppato al Punto 1.

- **Punto 3 [3 punti]** Si provi la proprietà di sottostruttura ottima per l'algoritmo sviluppato al Punto 1.

Answer: **Point 1** The greedy strategy works as follows: we try to pair the smallest (a_1) and the largest (a_n) element in S (only if $n > 1$). In case their sum is at most one, we create $S_1 = \{a_1, a_n\}$. In all other cases, we create $S_1 = \{a_n\}$, since no other element, if any, could possibly be paired with a_n legally. We then proceed on the residual problem. The algorithm follows.

```
2-1_BOXING(S)
n ← |S|
⋆ Let S = {a_1, a_2, ..., a_n} ⋆
P ← ∅; first ← 1; last ← n
while (first ≤ last)
    if ((first < last) and (a_first + a_last ≤ 1))
      then P ← P ∪ { {a_first, a_last} }; first ← first + 1;
      else  P ← P ∪ { {a_last} }
    last ← last − 1
return P
```

The above algorithm scans each element once, hence it requires linear time in n.

Point 2 Observe that the greedy choice is $\{a_1, a_n\}$ if $n > 1$ and $a_1 + a_n \leq 1$, and $\{a_n\}$ otherwise. We have to prove that there is always an optimal solution containing the greedy choice. The proof is trivial for $n = 1$, or for $n > 1$ and $a_1 + a_n > 1$, since in these cases any feasible solution must contain the subset $\{a_n\}$. Therefore let us assume that $n > 1$ and that the greedy choice is $\{a_1, a_n\}$. Consider an arbitrary optimal solution, and assume that a_1 and a_n are not paired together. Then, there exist two subsets S_1 and S_2, with $|S_1|, |S_2| \leq 2$ and $S_1 \cap S_2 = \emptyset$, such that $a_1 \in S_1$ and $a_n \in S_2$. We can substitute these two sets with $S_1' = \{a_1, a_n\}$ (this is the greedy choice, hence it is a legal subset), and $S_2' = S_1 \cup S_2 - \{a_1, a_n\}$. Indeed, $|S_2'| \leq 2 + 2 - 2 = 2$, and whenever $|S_2'| = 2$, then $S_2' = \{a_s, a_t\}$ with $a_s \in S_1$ and $a_t \in S_2$, but then $a_s < a_n$ (since $s < n$) and $a_t \leq 1 - a_n$ (since a_t was legally paired with a_n in S_2), hence $a_s + a_t < 1$. Therefore the new solution which includes these two new subsets is feasible and still optimal.

Point 3 Any optimal solution $\mathcal{P}^\star$ which contains the greedy choice, also contains a (2,1)-boxing $\mathcal{P}'$ of the residual set of values (either $\{a_1, \ldots, a_{n-1}\}$ or $\{a_2, \ldots, a_{n-1}\}$). If $\mathcal{P}'$ were not optimal, we could improve on the optimal solution for the original problem by considering a (2,1)-boxing obtained by adding the greedy choice to the optimal (2,1)-boxing $\mathcal{P}'^\star$ for the residual problem, a contradiction. □

Dati e Algoritmi 2 – CdL Magistrale in Ingegneria Informatica
Compito, 13/2/2009 (Durata: 3h)

Nome, Cognome, Matricola: _____

Corso di studio: _____

Prima Parte: domande teoriche

Si forniscano risposte il più possibile **rigorose e succinte** ai seguenti quesiti riguardanti argomenti trattati nel corso. Risposte approssimative, prolisse o in cattivo italiano saranno **fortemente penalizzate**. Ai fini del superamento dell'esame, è necessario conseguire una valutazione sufficiente alla prima parte.

T1 Si fornisca lo pseudocodice dell'algoritmo ricorsivo efficiente R_POWER(z, k) per il calcolo della k-sima potenza di un numero complesso z e se ne valuti la complessità asintotica. Non è richiesta la prova di correttezza.

Solution: The algorithm is the following:

```
R_POWER(z, k)
if (k = 0) then return 1
w ← R_POWER(z, ⌊k/2⌋)
if EVEN(k)
    then return w · w
    else return z · w · w
```

In the above code, procedure EVEN(k) returns **true** iff k is even. The algorithm can be easily proved correct by induction (proof not required). As for its running time, the associated recurrence $T(n) = T(n/2) + \Theta(1)$ obeys to Case 2 of the Master Theorem, since $n^{\log_2 1} = n^0 = \Theta(1) = w(n)$. Therefore, $T(n) = \Theta(\log n)$.

T2 Si enunci e si provi il lemma di somma per le radici n-esime dell'unità nel campo complesso.

Solution: The summation lemma states that for $n \geq 1$ and $i \geq 0$, the following equality holds:

$$\sum_{j=0}^{n-1} (w_n^i)^j = \begin{cases} n & \text{if } i \text{ is a multiple of } n, \\ 0 & \text{otherwise.} \end{cases}$$

Consider the first case. If i is a multiple of n, then it is $i = dn$ for some integer d, hence for $0 \leq j < n$, $(w_n^i)^j = (w_n^{dn})^j = (w_n^n)^{dj} = 1$, and the thesis follows. Otherwise, since $\omega_n^i \neq 1$, we can use the closed formula for the geometric series to get:

$$\sum_{j=0}^{n-1} (w_n^i)^j = \frac{(\omega_n^i)^n - 1}{w_n^i - 1} = \frac{(\omega_n^n)^i - 1}{w_n^i - 1} = \frac{1-1}{w_n^i - 1} = 0.$$

T3 Dimostrare che:
$\langle G = (V, E), k \rangle \in$ CLIQUE $\Leftrightarrow \langle G^c = (V, E^c), |V| - k \rangle \in$ VERTEX COVER.

Solution:

$\langle G = (V, E), k \rangle \in$ CLIQUE

$\Leftrightarrow \quad \exists V' \subseteq V, |V'| = k : \forall u, v \in V, u \neq v : [(u \in V') \wedge (v \in V')] \Rightarrow \{u, v\} \in E$

$\Leftrightarrow \quad \exists V' \subseteq V, |V'| = k : \forall u, v \in V, u \neq v : \{u, v\} \notin E \Rightarrow [(u \notin V') \vee (v \notin V')]$

$\Leftrightarrow \quad \exists V'' \subseteq V, |V''| = |V| - k : \forall u, v \in V, u \neq v : \{u, v\} \in E^c \Rightarrow [(u \in V'') \vee (v \in V'')]$

$\Leftrightarrow \langle G^c = (V, E^c), |V| - k \rangle \in$ VERTEX COVER

The above derivation is obtained by introducing the new quantified variable $V'' = V - V'$, and by observing that $|V''| = |V| - |V'|$, $\{u, v\} \notin E \Leftrightarrow \{u, v\} \in E^c$ and $(u \notin V') \Leftrightarrow (u \in V'')$:

Seconda Parte: risoluzione di problemi

Si forniscano soluzioni esaurienti e rigorose ai tre problemi seguenti. Gli algoritmi vanno codificati utilizzando lo **pseudocodice** usato in classe. **Attenzione:** Risposte immotivate e prive di prova di correttezza non saranno considerate.

Esercizio 1 [11 punti] Siano dati un algoritmo ricorsivo A_R e uno iterativo A_I per un certo problema $\Pi = \mathcal{I} \times \mathcal{S}$, dove la taglia di ogni istanza $i \in \mathcal{I}$ è una potenza di due. Si supponga che la complessità di A_I sia $T_I(n) = (1/4)n^2 \log_2 n$, mentre quella di A_R sia regolata dalla ricorrenza

$$T_R(n) = \begin{cases} 0 & n = 1 \\ 4T_R\left(\frac{n}{2}\right) + 12n & n > 1 \end{cases}$$

Si determini il valore ottimale n_0 tale che, invocando A_I per la risoluzione di istanze di taglia $n \leq n_0$ e utilizzando la parte ricorsiva di A_R per istanze di taglia maggiore, si ottiene il migliore algoritmo ibrido A_H tra A_I e A_R.

Answer: The generic recurrence to be solved for the hybrid algorithm is the following:

$$T_H(n, n_0) = \begin{cases} (1/4)n^2 \log_2 n & n \leq n_0 \\ 4T_H\left(\frac{n}{2}\right) + 12n & n > n_0 \end{cases}$$

Let $f(n) = n/2$. Since $f^{(i)}(n) = n/2^i$, we have

$$n/2^i > n_0 \Leftrightarrow i < \log_2(n/n_0)$$

whence $f^\star(n, n_0) = \log_2(n/n_0) - 1$, since both n and n_0 are powers of two. By applying the general formula we obtain, for $n \geq n_0$:

$$\begin{aligned} T_H(n, n_0) &= \sum_{\ell=0}^{\log_2(n/n_0)-1} 4^\ell(12n/2^\ell) + 4^{\log_2(n/n_0)}(1/4)n_0^2 \log_2 n_0 \\ &= 12n(2^{\log_2(n/n_0)} - 1) + (1/4)n^2 \log_2 n_0 \\ &= 12n(n/n_0 - 1) + (1/4)n^2 \log_2 n_0 \end{aligned}$$

In order to to find the best base value for n_0, let us study the sign of the partial derivative of $T_H(n, n_0)$ with respect to n_0:

$$\frac{\delta T(n, n_0)}{\delta n_0} = -12n^2/n_0^2 + (1/(4 \ln 2))n^2/n_0 \geq 0 \Leftrightarrow n_0 \geq 48 \ln 2 \simeq 33.27$$

Therefore, the best candidate values for n_0 are either 32 or 64. For $n_0 = 32$, the constant of the leading term of n^2 becomes $12/32 + 5/4 = 52/32 = 13/8 = 1.625$, while for $n_0 = 64$ we obtain $12/64 + 6/4 = 3/16 + 3/2 = 27/16 = 1.6875$. Therefore, the best choice turns out to be $n_0 = 32$, which yields $T_H(n, 32) = (13/8)n^2 - 12n$. Observe that the purely recursive algorithm A_R features a rather larger constant for the leading term, since $T_R(n) = T_H(n, 1) = 12n^2 - 12n$, a running time which is (asymptotically) about 7 times larger than the one attained by the best hybrid algorithm. □

Esercizio 2 [11 punti] Sia n una potenza di due. Dati $\boldsymbol{a}, \boldsymbol{b} \in \boldsymbol{C}^n$, la matrice circolante $C(\boldsymbol{b})$ si dice **multipla** della matrice circolante $C(\boldsymbol{a})$ se esite un vettore $\boldsymbol{x} \in \boldsymbol{C}^n$ tale che $C(\boldsymbol{b}) = C(\boldsymbol{a}) \cdot C(\boldsymbol{x})$. Si progetti, si dia lo pseudocodice e si analizzi un algoritmo IS_MULTIPLE($\boldsymbol{a}, \boldsymbol{b}$) che restituisce 1 se $C(\boldsymbol{b})$ è multipla di $C(\boldsymbol{a})$, e 0 altrimenti. Per avere punteggio pieno, IS_MULTIPLE($\boldsymbol{a}, \boldsymbol{b}$) deve eseguire $O(n \log n)$ operazioni tra scalari complessi.

Answer: Recalling that $C(\boldsymbol{a}) \cdot C(\boldsymbol{x}) = C(\boldsymbol{a} \circledast \boldsymbol{x})$, we have that $C(\boldsymbol{b})$ is multiple of $C(\boldsymbol{a})$ if and only if there exists a vector $\boldsymbol{x} \in \boldsymbol{C}^n$ such that

$$\boldsymbol{a} \circledast \boldsymbol{x} = \boldsymbol{b}.$$

In order to study the above vector equation, let $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{X}$ denote the Discrete Fourier Transforms of the respective lower-case vectors. By applying the Cyclic Convolution Theorem to $\boldsymbol{a} \circledast \boldsymbol{x}$ and transforming both sides of the equation with DFT_n, we obtain the equivalent vector equation:

$$\boldsymbol{A} \odot \boldsymbol{X} = \boldsymbol{B},$$

where $\odot$ denotes component-wise multiplication. Observe that the above equation reduces to the following simple system of n scalar equations:

$$A_i \cdot X_i = B_i, \text{ for } 0 \leq i < n.$$

This system has at least one solution (in fact, either one or infinite) if and only if $(A_i = 0) \Rightarrow (B_i = 0)$, for $0 \leq i < n$. This is the (multiple) condition to test to ensure that $C(\boldsymbol{b})$ is multiple of $C(\boldsymbol{a})$. The code of the algorithm follows.

```
IS_MULTIPLE(a, b)
A ← FFT(a)
B ← FFT(b)
for i ← 0 to n − 1 do
    if [(A_i = 0) ∧ (B_i ≠ 0)]
        then return 0
return 1
```

The correctness of the above algorithm follows from the previous discussion. Its running time $T_{\text{IM}}(n)$ is dominated by the two calls to the FFT algorithm on vectors of size n, hence $T_{\text{IM}}(n) = \Theta(n \log n)$. □

Esercizio 3 [11 punti] Si consideri il seguente problema decisionale:

2-OUTPUT BC-SAT:
ISTANZA: $\langle C(x_1, x_2, \ldots x_n), t\rangle$: C circuito booleano a due output y_1 e y_2, $t \in \{0, 1\}$
DOMANDA: Esiste un assegnamento di valori di verità $\boldsymbol{b} \in \{0, 1\}^n$ sotto il quale $y_1 \vee y_2 = t$?

Si dimostri che 2-OUTPUT BC-SAT $\in NPH$.

Answer: The reduction is clearly from BC-SAT. Given an instance of BC-SAT $\langle C(x_1, x_2, \ldots x_n)\rangle$, let y be its (only) output. Consider now the two-output circuit $\langle C'(x_1, x_2, \ldots x_n)\rangle$ simply obtained from C by adding a NOT gate, fed by input x_1, and an AND gate, fed by the output of the NOT gate and again by input x_1. Let $y_1 = y$ and y_2 be the output of the new AND gate (observe that $y_2 = 0$ under any truth assignment $\boldsymbol{b} \in \{0, 1\}^n$). We set $f(\langle C(x_1, x_2, \ldots x_n)\rangle = \langle C'(x_1, x_2, \ldots x_n), 1\rangle$ and observe that f is clearly polynomial-time (in fact, linear-time) computable. By observing that $y_1 \vee y_2 = 1$ if and only if $y_1 = 1$ (since y_2 is always 0), we have that $x = \langle C(x_1, x_2, \ldots x_n)\rangle \in$ BC_SAT $\Leftrightarrow f(x) = \langle C'(x_1, x_2, \ldots x_n), 1\rangle \in$ 2-OUTPUT BC-SAT. □

Dati e Algoritmi 2 – CdL Magistrale in Ingegneria Informatica
Compito, 28/1/2009 (Durata: 3h)

Nome, Cognome, Matricola: _____

Corso di studio: _____

Prima Parte: domande teoriche

Si forniscano risposte il più possibile **rigorose e succinte** ai seguenti quesiti riguardanti argomenti trattati nel corso. Risposte approssimative, prolisse o in cattivo italiano saranno **fortemente penalizzate**. Ai fini del superamento dell'esame, è necessario conseguire una valutazione sufficiente alla prima parte.

T1 Data una generica contrazione $f(n)$, si dia la definizione formale della relativa funzione $f^\star(n, n_0)$, per $n > n_0$. Si determini inoltre $f^\star(n, 2)$ quando $f(n) = \sqrt{n}$ e $n = 2^{2^i}$, per $i > 0$.

Solution: Define $f^{(0)}(n) = n$, and, for $k > 0$, $f^{(k)}(n) = f(f^{(k-1)}(n))$. Then

$$f^\star(n, n_0) = \max\{k \geq 0 : f^{(k)}(n) > n_0\}.$$

When $f(n) = \sqrt{n} = n^{1/2}$, clearly we have $f^{(k)}(n) = n^{1/2^k}$, for $k \geq 0$. Therefore

$$f^{(k)}(n) > 2 \Leftrightarrow n^{1/2^k} > 2 \Leftrightarrow (\log_2 n)/2^k > 1 \Leftrightarrow 2^k < \log_2 n \Leftrightarrow k < \log_2 \log_2 n$$

Observe that $\log_2 \log_2 n$ is an integer when $n = 2^{2^i}$, hence $f^\star(n, 2) = \log_2 \log_2 n - 1$.

T2 Si enunci e si provi l'identità polinomiale su cui si basa la strategia ricorsiva dell'algoritmo FFT.

Solution: Let n be a power of two, and let $A(x) = \sum_{i=0}^{n-1} a_i x^i$ be a polynomial of degree-bound n. Also, let $A^{[0]}(x) = \sum_{j=0}^{n/2-1} a_{2j} x^j$ and $A^{[1]}(x) = \sum_{j=0}^{n/2-1} a_{2j+1} x^j$. Then, we have

$$A(x) = A^{[0]}(x^2) + xA^{[1]}(x^2).$$

The proof follows by observing that

$$A^{[0]}(x^2)+xA^{[1]}(x^2) = \sum_{j=0}^{n/2-1} a_{2j}x^{2j}+\sum_{j=0}^{n/2-1} a_{2j+1}x^{2j+1} = \sum_{i=0,\, i \text{ even}}^{n-2} a_i x^i+\sum_{i=1,\, i \text{ odd}}^{n-1} a_i x^i$$

and the two summations on the right hand side contain all the terms of $\sum_{i=0}^{n-1} a_i x^i = A(x)$, partitioned between even-indexed and odd-indexed terms.

T3 Dati $L', L \in \{0,1\}^\star$, si provi che $(L' <_P L) \wedge (L \in P) \Rightarrow (L' \in P)$.

Solution: From the hypotheses, there is a polynomial-time computable function f such that $x \in L' \Leftrightarrow f(x) \in L$. Moreover, there exists a polynomial-time decision algorithm A_L for L. Let A_f be the algorithm which computes f. Then, the following algorithm:

$A_{L'}(x)$
    return $A_L(A_f(x))$

is clearly polynomial, since it is obtained by composing two polynomial-time algorithms. Also,

$$x \in L' \Leftrightarrow f(x) \in L \Leftrightarrow A_L(A_f(x)) = A_{L'}(x) = 1,$$

hence $A_{L'}$ is a polynomial-time decision algorithm for L'. Therefore $L' \in P$.

Seconda Parte: risoluzione di problemi

Si forniscano soluzioni esaurienti e rigorose ai tre problemi seguenti. Gli algoritmi vanno codificati utilizzando lo **pseudocodice** usato in classe. **Attenzione:** Risposte immotivate e prive di prova di correttezza non saranno considerate.

Esercizio 1 [8 punti] Utilizzando la formula generale, si risolva la seguente ricorrenza, definita per valori del parametro $n = 2^{2^i}$, per $i \geq 0$.

$$T(n) = \begin{cases} 0 & n = 2 \\ 8T(\sqrt{n}) + \log_2 n & n > 2 \end{cases}$$

Answer: Thanks to exercise **T1**, we have already proved that, when $f(n) = \sqrt{n}$, we have $f^{(\ell)}(n) = n^{1/2^\ell}$ and $f^\star(n, 2) = \log_2 \log_2 n - 1$. We can apply the general formula by observing that its second term is zero, since $T(n_0) = T(2) = 0$, and, for what concerns the first term, we have $s(n) = 8$, hence $\Pi_{j=0}^{\ell-1} s(f^{(j)}(n)) = 8^\ell$, and $w(f^{(\ell)}(n)) = \log_2 n^{1/2^\ell} = (\log_2 n)/2^\ell$. Therefore

$$T(n) = \sum_{\ell=0}^{\log_2 \log_2 n - 1} 8^\ell \frac{\log_2 n}{2^\ell} = (\log_2 n) \sum_{\ell=0}^{\log_2 \log_2 n - 1} 4^\ell = (\log_2 n) \frac{4^{\log_2 \log_2 n} - 1}{3}$$

Observe that $4^{\log_2 \log_2 n} = 2^{2 \log_2 \log_2 n} = 2^{\log_2 (\log_2^2 n)} = \log_2^2 n$, therefore

$$T(n) = \frac{(\log_2 n)(\log_2^2 n - 1)}{3} = \Theta(\log^3 n).$$

□

Esercizio 2 [12 punti] Una stringa $Z = \langle z_1, z_2, \ldots, z_k \rangle$ si dice *palindroma* se $z_{1+h} = z_{k-h}$, per $0 \leq h < k$. Data una stringa non vuota $X = \langle x_1, x_2, \ldots, x_n \rangle$, il problema LONGEST_PALINDROME_SUBSEQUENCE (LPS) richiede di determinare la lunghezza massima di una **sottosequenza** palindroma di X.

- **Punto 1 [7 punti]** Si fornisca una proprietà di sottostruttura ottima per LPS sullo spazio di sottoproblemi $\{X_{i..j} : 1 \leq i \leq j \leq n\}$ (l'insieme delle **sottostringhe** non vuote di X) e si derivi l'associata ricorrenza sui costi.
- **Punto 2 [5 punti]** Si scriva e si analizzi lo pseudocodice per il calcolo **memoizzato** della lunghezza di una LPS di X, utilizzando il modello di costo in cui solo il confronto tra due caratteri ha costo unitario e non nullo.

Answer:

Point 1 Let $Z = \langle z_1, z_2, \ldots, z_k \rangle$ be an LPS of $X_{i..j}$. The base cases are for $1 \leq i = j \leq n$, where $Z = \langle x_i \rangle$, and $j = i + 1$, for $1 \leq i < n$, where $Z = \langle x_i, x_j \rangle$ if $x_i = x_j$ or Z is either $\langle x_i \rangle$ or $\langle x_j \rangle$ if $x_i \neq x_j$. For $j > i + 1$, we have:

1. If $x_i = x_j$, then $z_1 = z_k = x_i = x_j$ and $Z_{2..k-1}$ is an LPS of $X_{i+1..j-1}$
2. If $x_i \neq x_j$, then Z is either an LPS of $X_{i+1..j}$ or an LPS of $X_{i..j-1}$.

The proof for the base cases is trivial. For $j > i + 1$ and $x_i = x_j$, if $z_1 = z_k \neq x_i$, then the string $\langle x_i, Z, x_i \rangle$ would still be a palindrome subsequence of $X_{i..j}$ longer than the LPS Z, yielding a contradiction. Also, $Z_{2..k-1}$ is still palindrome and is a subsequence of $X_{i+1..j-1}$. If it were not an LPS of $X_{i+1..j-1}$., then there would exist a longer palindrome subsequence than Z for $X_{i..j}$, again, a contradiction. For what concerns the case $x_i \neq x_j$ observe that it must either be $z_1 = z_k \neq x_i$ or $z_1 = z_k \neq x_j$. In the first case, Z is also a subsequence of $X_{i+1..j}$, and of $X_{i..j-1}$ in the second case. In both cases it must be an LPS of these respective substrings or otherwise Z would not be an LPS for $X_{i..j}$. Let $c(i, j) = |Z|$. The above optimal substructure property yields the following recurrence, defined for $1 \leq i \leq j \leq n$:

$$c(i,j) = \begin{cases} 1 & i = j \\ 2 & x_i = x_j, j = i+1 \\ 1 & x_i \neq x_j, j = i+1 \\ 2 + c(i+1, j-1) & x_i = x_j, j > i+1 \\ \max\{c(i+1,j), c(i,j-1)\} & \text{otherwise.} \end{cases}$$

Point 2 The pair of routines for the memoized computation of $c(1, n)$ are given in the following:

```
INIT_LPS(X)
n ← length(X)
if n = 1 then return 1
if n = 2
    then if (x_1 = x_2)
        then return 2
        else return 1
c[n, n] ← 1
for i ← 1 to n − 1 do
    c[i, i] ← 1;
    if (x_i = x_{i+1})
        then c[i, i + 1] ← 2
        else c[i, i + 1] ← 1
    for j ← i + 2 to n do
        c[i, j] ← 0
return REC_LPS(X, 1, n)
```

INIT_LPS solves base cases directly ($n = 1, 2$), initializes a table with the base cases and the default value 0 (never used in the recurrence), and finally calls REC_LPS, which performs the actual computation:

```
REC_LPS(X, i, j)
if (c[i, j] = 0) then
    if (x_i = x_j)
        then c[i, j] ← 2 + REC_LPS(X, i + 1, j − 1)
        else c[i, j] ← MAX(REC_LPS(X, i + 1, j), REC_LPS(X, i, j − 1))
return c[i, j]
```

In the above program, MAX denotes the simple routine which returns the maximum value between its two operands. The correctness of the above code follows from the correctness of the recurrence derived in Point 1. As for its running time, observe that the combined execution of INIT_LPS and REC_LPS performs, in the worst case, a number of character comparisons equal to the number of substrings of X of size at least 2. This number is exactly $\binom{n}{2} = \frac{n(n-1)}{2} = \Theta(n^2)$. Observe however that in the best case (when X is itself a palindrome) the memoized algorithm only performs a linear number of comparisons. □

Esercizio 3 [12 punti] Una formula Φ si definisce **diretta** se non contiene negazioni. Si consideri il seguente problema decisionale:

K DIRECT-2-SAT (K-D2S):
ISTANZA: $\langle \Phi(x_1, x_2, \ldots x_n) = C_1 \wedge C_2 \wedge \ldots \wedge C_m, k \rangle$,
Φ formula 2-CNF diretta e $1 \leq k \leq n$.
DOMANDA: Esiste un assegnamento di valori di verità $\boldsymbol{b} \in \{0,1\}^n$ con **esattamente** k **valori a 1**, che soddisfa Φ?

A titolo di esempio, la formula $\Phi(x_1, x_2, x_3) = (x_1 \vee x_3) \wedge (x_2 \vee x_3)$ è 2-CNF diretta e la corrispettiva istanza $\langle \Phi, 1 \rangle$ è positiva (si consideri $\boldsymbol{b} = (0, 0, 1)$).
Si dimostri che K-D2S $\in NPH$. (*Suggerimento:* Si riduca da VERTEX COVER).

Answer: Following the hint, we reduce from VERTEX_COVER as follows. Let $\langle G = (V, E), k \rangle$ be a generic instance of VERTEX_COVER, and let

$$f(\langle G = (V, E), k \rangle) = \langle \Phi_G(x_1, x_2, \ldots, x_{|V|}) = C_1 \wedge C_2 \wedge \ldots \wedge C_{|E|}, k \rangle$$

where $C_j = (x_r \vee x_s) \Leftrightarrow e_j = \{v_r, v_s\}$. In other words, the formula is defined on $|V|$ variables, corresponding to the nodes of G, and there is one clause for each edge, containing the two variables corresponding to the endpoints of the edge. The above reduction is clearly polynomial (in fact, linear) in $|\langle G = (V, E), k \rangle|$.

In order to prove that f is indeed a reduction, we must show that

$$\langle G = (V, E), k \rangle \in \text{VERTEX\_COVER} \Leftrightarrow f(\langle G = (V, E), k \rangle) \in \text{K-D2S}.$$

($\Rightarrow$) Given a vertex cover $V' = \{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$ of size k, set $b_{i_j} = 1$ for $1 \leq j \leq k$ and all other values to zero. The resulting truth assignment $\boldsymbol{b}$ has exactly k values set to 1 and satisfies Φ_G, since each clause (which corresponds to an edge) must contain a variable corresponding to a vertex in V'.

($\Leftarrow$) Starting from a satisfying assignment $\boldsymbol{b}$ with exactly k values set to 1, the set V' of vertices corresponding to the indices $b_{i_j} = 1$, for $1 \leq j \leq k$ has size k and the property that each edge is incident on at least one such vertex, or otherwise the clause corresponding to that edge would not be true under the assignment.

□

Dati e Algoritmi 2 – CdL Magistrale in Ingegneria Informatica
Compito, 15/9/2009

Nome, Cognome, Matricola: _____

Corso di studio: _____

Prima Parte: domande teoriche

Si forniscano risposte il più possibile **rigorose e succinte** ai seguenti quesiti riguardanti argomenti trattati nel corso. Risposte approssimative, prolisse o in cattivo italiano saranno **fortemente penalizzate**. Ai fini del superamento dell'esame, è necessario conseguire una valutazione sufficiente alla prima parte.

T1 Si enunci e si provi la proprietà di scelta Greedy relativa all'algoritmo di codifica di Huffman.

Solution: The greedy choice property related to Huffman's algorithm is stated as follows:

Let $x, y \in \Sigma$ be the two characters of the alphabet with lowest frequency. Then, there exists an optimal prefix tree in which the leaves associated with x and y are sibling.

Proof. Without loss of generality, assume that $f[x] \leq f[y]$. Let $\hat{T}$ be an (arbitrary) optimal prefix tree, and let $a, b \in \Sigma$, with $f[a] \leq f[b]$, be two characters such that their leaves have maximum depth in $\hat{T}$. Observe that $f[x] \leq f[a]$. By swapping the leaves associated to x and a, we obtain a new tree $T^\star$, such that

$$\begin{aligned} & B(\hat{T}) - B(T^\star) \\ &= f[a]d_{\hat{T}}[a] + f[x]d_{\hat{T}}[x] - f[a]d_{T^\star}[a] - f[x]d_{T^\star}[x] \\ &= f[a]d_{\hat{T}}[a] + f[x]d_{\hat{T}}[x] - f[a]d_{\hat{T}}[x] - f[x]d_{\hat{T}}[a] \\ &= f[a](d_{\hat{T}}[a] - d_{\hat{T}}[x]) - f[x](d_{\hat{T}}[a] - d_{\hat{T}}[x]) \\ &= (f[a] - f[x])(d_{\hat{T}}[a] - d_{\hat{T}}[x]) \\ &\geq 0, \end{aligned}$$

whence $B(T^\star) = B(\hat{T})$, therefore $T^\star$ is still optimal. The proof follows by observing that $f[y] \leq f[b]$ and repeating the above argument, swapping their respective leaves.

T2 Si enunci e si provi il lemma di cancellazione per le radici n-esime dell'unità nel campo complesso.

Solution: The cancellation lemma states that for any integers $n, d \geq 1$ and $k \geq 0$,

$$\omega_{dn}^{dk} = \omega_n^k$$

Proof.

$$\begin{aligned} \omega_{dn}^{dk} &= \left(e^{2\pi i/dn}\right)^{dk} \\ &= \left(e^{2\pi i/n}\right)^{k} \\ &= \omega_n^k. \end{aligned}$$

T3 Dimostrare che:
$\langle G = (V, E), k\rangle \in$ VERTEX COVER $\Leftrightarrow \langle G^c = (V, E^c), |V| - k\rangle \in$ CLIQUE.

Solution:

$\langle G = (V, E), k\rangle \in$ VERTEX COVER

$\Leftrightarrow \quad \exists V' \subseteq V, |V'| = k : \forall u, v \in V, u \neq v : \{u, v\} \in E \Rightarrow [(u \in V') \vee (v \in V')]$

$\Leftrightarrow \quad \exists V'' \subseteq V, |V''| = |V| - k : \forall u, v \in V, u \neq v : \{u, v\} \notin E^c \Rightarrow [(u \notin V'') \vee (v \notin V'')]$

$\Leftrightarrow \quad \exists V'' \subseteq V, |V''| = |V| - k : \forall u, v \in V, u \neq v : [(u \in V'') \wedge (v \in V'')] \Rightarrow \{u, v\} \in E^c$

$\langle G = (V, E), |V| - k\rangle \in$ CLIQUE

The above derivation is obtained by introducing the new quantified variable $V'' = V - V'$, and by observing that $|V''| = |V| - |V'|$, $\{u, v\} \in E \Leftrightarrow \{u, v\} \notin E^c$ and $(u \in V') \Leftrightarrow (u \notin V'')$:

Seconda Parte: risoluzione di problemi

Si forniscano soluzioni esaurienti e rigorose ai tre problemi seguenti. Gli algoritmi vanno codificati utilizzando lo **pseudocodice** usato in classe. **Attenzione:** Risposte immotivate e prive di prova di correttezza non saranno considerate.

Esercizio 1 [10 punti] Si consideri la seguente ricorrenza, definita per tutti i valori del parametro intero $n > 0$.

$$T(n) = \begin{cases} 18 & 0 < n < 20 \\ T\left(\left\lceil \frac{3n}{10} + 3 \right\rceil\right) + T\left(\left\lfloor \frac{2n}{5} - 1 \right\rfloor\right) + 3n & n \geq 20 \end{cases}$$

Utilizzando l'induzione parametrica, si determini una opportuna costante $c > 0$ tale che $T(n) \leq cn$ per $n > 0$.

Answer: Parametric induction lets us collect constraints on feasible values $c > 0$ by "simulating" an inductive argument. For the base cases $0 < n < 20$, observe that it must be $18 = T(n) \leq cn$, whence $c \geq 18/n$. The strictest constraint is the one for $n = 1$, which yields $c \geq 18$. Assume now that the statement is true for $T(k)$, with $0 < k < n$ and $n \geq 20$. We have:

$$\begin{aligned} T(n) &= T(\lceil (3n/10) + 3 \rceil) + T(\lfloor (2n/5) - 1 \rfloor) + 3n \\ &\leq c\lceil (3n/10) + 3 \rceil + c\lfloor (2n/5) - 1 \rfloor + 3n \\ &\leq c(3n/10 + 4) + c(2n/5 - 1) + 3n \\ &\leq c(7/10)n + 3c + 3n. \end{aligned}$$

In order to obtain the inductive thesis, it suffices to pick a single value $c > 0$ such that $c(7/10)n + 3c + 3n \leq cn$, for all values $n \geq 20$, or, equivalently, $c(3/10)n - 3c \geq 3n$. Consider the family of straight lines $y_c = c(3/10)x - 3c$ and the straight line $y = 3x$. For the inequality to be true for all values $n \geq 20$, it suffices to pick any value c corresponding to a straight line y_c in the family which intersects $y = 3x$ at an abscyssa $x_{\text{in}} \leq 20$. This yields the simple inequality $3c \geq 60$, whence $c \geq 20$.

In conclusion, the whole inductive argument carries through by picking $c = \max\{18, 20\} = 20$, whence $T(n) \leq 20n$, for $n > 0$. □

Esercizio 2 [12 punti] Dato un array $A[1..n]$ di numeri interi *arbitrari* (sia positivi che negativi), si consideri il problema di determinare il valore

$$m^\star = \max\left\{ \sum_{k=i}^{j} A[k] : 1 \leq i \leq j \leq n \right\}.$$

Descrivere e analizzare algoritmo di programmazione dinamica FIND_MAX(A) che ritorni il valore $m^\star$ in tempo $O(n)$.

Suggerimento: Dimostrare che la soluzione è ottenibile calcolando, *per ogni* i, $1 \leq i \leq n$, il valore massimo $m(i)$ ottenibile sommando gli elementi dell'array in un qualche intervallo di indici che abbia i come estremo inferiore. La programmazione dinamica va applicata al calcolo di tali valori $m(i)$...

Answer: For $1 \leq i \leq n$, define

$$m(i) = \max \left\{ \sum_{k=i}^{j} A[k] : i \leq j \leq n \right\}.$$

Clearly, $m(i)$ represents the largest sum obtainable by summing elements of A with indices starting from i. Define now

$$\hat{m} = \max \{ m(i) : 1 \leq i \leq n \}.$$

We have that $\hat{m} \leq m^\star$, since the set of values $\{m(i) : 1 \leq i \leq n\}$ is contained in the set $\left\{ \sum_{k=i}^{j} A[k] : 1 \leq i \leq j \leq n \right\}$. But it is also $m^\star \leq \hat{m}$, since for each $1 \leq i \leq j \leq n$, we have $\sum_{k=i}^{j} A[k] \leq m(i)$. Therefore we have that $m^\star = \hat{m}$.

The values $m(i)$ can be computed according to the following, simple recurrence:

$$m(i) = \begin{cases} A[n] & i = n. \\ \max\{A[i], A[i] + m(i+1)\} & \text{otherwise.} \end{cases}$$

Let us show that the recurrence is valid using (backward) induction. The statement clearly holds for $i = n$, since $[n, n]$ is the only interval of indices starting from n. Assume the statement holds down to $i+1$. Consider now the largest sum obtainable with indices starting from i. Such sum is either $A[i]$ or, otherwise, it must be $A[i] + m[i+1]$, since it can be neither smaller ($A[i] + m[i+1]$ is a sum of indices starting from i) or larger (or otherwise $m[i+i]$ would not be maximum). A simple, linear algorithm to solve the problem follows. In the algorithm, subroutine MAX(a, b) returns $\max\{a, b\}$.

```
FIND_MAX(A)
n ← length(A)
m[n] ← m̂ ← A[n]
for i ← n − 1 downto 1 do
    m[i] ← MAX(A[i], A[i] + m[i + 1])
    m̂ ← MAX(m̂, m[i])
return m̂
```

□

Esercizio 3 [10 punti] Si consideri il seguente problema decisionale:

PARTITION:

INSTANCE: $\langle S \rangle$, $S \subseteq \boldsymbol{N}$, insieme finito

QUESTION: Esistono $S_1, S_2 \subseteq S$, con $S_1 \cup S_2 = S$ e $S_1 \cap S_2 = \emptyset$ tali che

$$\sum_{s_1 \in S_1} s_1 = \sum_{s_2 \in S_2} s_2 \ ?$$

Si dimostri che PARTITION $<_P$ SUBSET SUM.

Answer: Recall that an instance of SUBSET SUM is $\langle S, t \rangle$, with $S \subset \boldsymbol{N}$ and $t \in \boldsymbol{N}$. Given an instance $\langle S \rangle$ of PARTITION, we set $a = \sum_{s \in S} s$ and define our reduction function as follows:q

$$f\left(\langle S \rangle\right) = \begin{cases} \langle \emptyset, 1 \rangle & \text{if } a \text{ is odd,} \\ \langle S, a/2 \rangle & \text{otherwise.} \end{cases}$$

Function f is clearly computable in polynomial time. Let us now show that f indeed reduces PARTITION to SUBSET SUM.

Let $x = \langle S \rangle \in$ PARTITION. Then, there exists a partition of S into subsets S_1 and S_2 such that $\sum_{s_1 \in S_1} s_1 = \sum_{s_2 \in S_2}$. Let $b = \sum_{s_1 \in S_1} s_1$. Then $a = \sum_{s \in S} s = 2b$ is an even number and $f(x) = \langle S, a/2 \rangle \in$ SUBSET SUM, since there exists a subset of S (either S_1 or S_2) whose elements sum to $a/2$.

Viceversa, let $f(x) \in$ SUBSET SUM. Then, it must be $f(x) = \langle S, a/2 \rangle$ necessarily (since $f(x)$ is a positive instance). Also, there is a subset $S_1 \subset S$ such that $\sum_{s_1 \in S_1} s_1 = a/2$, whence $\sum_{s_2 \in S - S_1} s_2 = a - a/2 = a/2$, which implies that $x = \langle S \rangle \in$ PARTITION. □

Dati e Algoritmi 2 – CdL Magistrale in Ingegneria Informatica

Compito, 1/2/2010 (Durata: 3h)

Nome, Cognome, Matricola: _____

Corso di studio: _____

Prima Parte: domande teoriche

Si forniscano risposte il più possibile **rigorose e succinte** ai seguenti quesiti riguardanti argomenti trattati nel corso. Risposte approssimative, prolisse o in cattivo italiano saranno **fortemente penalizzate**. Ai fini del superamento dell'esame, è necessario conseguire una valutazione sufficiente alla prima parte.

T1 Data una generica contrazione $f(n)$, si dia la definizione formale della relativa funzione $f^\star(n, n_0)$, per $n > n_0$. Si determini inoltre $f^\star(n, 16)$ quando $f(n) = \sqrt{n}$ e $n = 2^{2^i}$, per $i > 0$.

Solution: Define $f^{(0)}(n) = n$, and, for $k > 0$, $f^{(k)}(n) = f(f^{(k-1)}(n))$. Then

$$f^\star(n, n_0) = \max\{k \geq 0 : f^{(k)}(n) > n_0\}.$$

When $f(n) = \sqrt{n} = n^{1/2}$, clearly we have $f^{(k)}(n) = n^{1/2^k}$, for $k \geq 0$. Therefore

$$f^{(k)}(n) > 16 \Leftrightarrow n^{1/2^k} > 16 \Leftrightarrow (\log_2 n)/2^k > 4 \Leftrightarrow 2^{k+2} < \log_2 n \Leftrightarrow k < \log_2 \log_2 n - 2$$

Observe that $\log_2 \log_2 n - 2$ is an integer when $n = 2^{2^i}$, hence $f^\star(n, 16) = \log_2 \log_2 n - 3$.

T2 Si enunci e si provi il lemma di somma per le radici n-sime dell'unità in campo complesso.

Solution: The summation lemma states that for $n > 0$ and any integer k, we have:

$$\sum_{i=0}^{n-1} \left(\omega_n^k\right)^i = \begin{cases} n, & \text{if } k \bmod n = 0, \\ 0, & \text{otherwise.} \end{cases}$$

To prove the above identity, let us first consider the case $k \bmod n = 0$. Then $\omega_n^k = \omega_n^{k \bmod n} = \omega_n^0 = 1$, hence $\left(\omega_n^k\right)^i = 1$ for any $0 \leq i \leq n-1$, and the sum evaluates to n. When $k \bmod n \neq 0$, we have instead that $\omega_n^k \neq 1$, hence the

formula for the truncated geometric series applies, yielding

$$\sum_{i=0}^{n-1} \left(\omega_n^k\right)^i = \frac{\left(\omega_n^k\right)^n - 1}{\omega_n^k - 1} = \frac{\left(\omega_n^n\right)^k - 1}{\omega_n^k - 1} = \frac{1^k - 1}{\omega_n^k - 1} = 0.$$

T3 Dati $L_1, L_2 \in \{0,1\}^\star$, si provi che $(L_1 <_P L_2) \wedge (L_2 \in P) \Rightarrow (L_1 \in P)$.

Solution: Since $(L_1 <_P L_2)$, there exists a function $f : \{0,1\}^\star \to \{0,1\}^\star$, which is computed by Algorithm $A_f(x)$ in time $T_{A_f}(|x|) = O(|x|^{k_1})$, such that $x \in L_1 \Leftrightarrow f(x) \in L_2$. Also, since $L_2 \in P$, there exists an algorithm $A_{L_2}(x)$, which decides L_2 in time $T_{A_{L_2}}(|x|) = O(|x|^{k_2})$. By composing the two algorithms we obtain:

$A_{L_1}(x)$
return $A_{L_2}(A_f(x))$

Algorithm A_{L_1} runs in time $T_{A_{L_1}}(|x|) = O(|x|^{k_1} + |x|^{k_1 k_2}) = O(|x|^{k_1 \max\{1,k_2\}})$ hence it is still polynomial. Moreover

$$x \in L_{A_{L_1}} \Leftrightarrow A_{L_2}(A_f(x)) = 1 \Leftrightarrow A_f(x) \in L_2 \Leftrightarrow f(x) \in L_2 \Leftrightarrow x \in L_1,$$

therefore $L_{A_{L_1}} = L_1$, hence A_{L_1} is a polynomial-time decision algorithm for L_1, and the thesis follows.

Seconda Parte: risoluzione di problemi

Si forniscano soluzioni esaurienti e rigorose ai tre problemi seguenti. Gli algoritmi vanno codificati utilizzando lo **pseudocodice** usato in classe. **Attenzione:** Risposte immotivate e prive di prova di correttezza non saranno considerate.

Esercizio 1 [10 punti] Utilizzando l'induzione parametrica, si determini una costante c per cui la soluzione della seguente ricorrenza, definita per valori del parametro $n \geq 2$,

$$T(n) = \begin{cases} 0 & n = 2, 3 \\ \lfloor\sqrt{n}\rfloor T\left(\lfloor\sqrt{n}\rfloor\right) + 14n & n > 3 \end{cases}$$

sia tale che $T(n) \leq cn \log_2 \log_2 n$ per ogni $n \geq 2$.

Answer: Let us apply parametric induction to collect constraints on feasible values of c. For the base, it must be $0 \leq c \log \log n$ for $n = 2, 3$, which yields $c \geq 0$. Assume now that the inductive hypothesis holds for values of the parameter less than $n > 3$. We obtain:

$$\begin{aligned} T(n) &= \lfloor\sqrt{n}\rfloor T\left(\lfloor\sqrt{n}\rfloor\right) + 14n \\ &\leq \lfloor\sqrt{n}\rfloor c \lfloor\sqrt{n}\rfloor \log_2 \log_2 \lfloor\sqrt{n}\rfloor + 14n \\ &\leq cn \log_2 \log_2 \sqrt{n} + 14n \\ &= cn \log_2 \left(\frac{\log_2 n}{2}\right) + 14n \\ &= cn \log_2 \log_2 n - cn + 14n \overset{?}{\boxed{\leq}} cn \log_2 \log_2 n \\ &\Leftrightarrow c \geq 14. \end{aligned}$$

From the above derivation it follows that it suffices to choose $c = \max\{0, 14\} = 14$ to guarantee that $T(n) \leq cn \log_2 \log_2 n$ for $n \geq 2$. □

Esercizio 2 [11 punti]

- **Punto 1 [8 punti]** Si dimostri che il vettore $\boldsymbol{F}_n^0 = (1, 1, \ldots, 1)$ è un *autovettore* di una qualsiasi matrice circolante $C(\boldsymbol{a})$, con $\boldsymbol{a} \in \mathbb{C}^n$, ovvero, che esiste un (autovalore) $\lambda \in \mathbb{C}$ per cui si ha: $C(\boldsymbol{a})\boldsymbol{F}_n^0 = \lambda \boldsymbol{F}_n^0$.
- **Punto 1 [3 punti]** Si progetti e si analizzi un algoritmo efficiente EIGENVALUE($\boldsymbol{a}$) che, dato in ingresso il vettore $\boldsymbol{a}$, ritorni l'autovalore λ associato all'autovettore $\boldsymbol{F}_n^0$ della matrice $C(\boldsymbol{a})$.

Answer:

Part 1 Recall that $C(\boldsymbol{a})\boldsymbol{F}_n^0 = \boldsymbol{a} \circledast \boldsymbol{F}_n^0$, and, by the cyclic convolution theorem, the latter convolution can be computed as $DFT_n^{-1}(DFT_n(\boldsymbol{a}) \odot DFT_n(\boldsymbol{F}_n^0))$. Let us now consider $\boldsymbol{y} = DFT_n(\boldsymbol{F}_n^0)) = DFT_n((1, 1, \ldots, 1))$. By the definition of DFT_n, we have that

$$y_i = \sum_{j=0}^{n-1} \omega_n^{ij} = \begin{cases} n & i = 0 \\ 0 & i > 0 \end{cases} .$$

Let $A_0 = [DFT_n(\boldsymbol{a})]_0$. It follows that $DFT_n(\boldsymbol{a}) \odot DFT_n(\boldsymbol{F}_n^0) = DFT_n(\boldsymbol{a}) \odot \boldsymbol{y} = (nA_0, 0, \ldots, 0)$, whose inverse transform is the vector $F_n^{-1}(nA_0, 0, \ldots, 0) = nA_0[F_n^{-1}]^0 = (A_0, A_0, \ldots, A_0)$. Observe that this vector can be seen as $\lambda \boldsymbol{F}_n^0$, by setting $\lambda = A_0 = [DFT_n(\boldsymbol{a})]_0$, hence the proof follows.

Part 2 The algorithm EIGENVALUE($\boldsymbol{a}$) must return $A_0 = [DFT_n(\boldsymbol{a})]_0$. Rather than invoking the FFT algorithm, we simply compute the first component of the transform directly by observing that

$$[DFT_n(\boldsymbol{a})]_0 = [F_n \boldsymbol{a}]_0 = \sum_{j=0}^{n-1} a_j \omega_n^{0 \cdot j} = \sum_{j=0}^{n-1} a_j .$$

The algorithm follows.

```
EIGENVALUE(a)
n ← a.len
A0 ← a_0
for i ← 1 to n − 1 do
    A0 ← A0 + a_i
return A0
```

The correctness of algorithm EIGENVALUE($\boldsymbol{a}$) follows from the proof in Part 1 and from the observation that the first component of the Discrete Fourier Transform of any vector is simply the sum of its components. As for its running time, in terms of the number of arithmetic operations between complex scalars, it is clearly $T_E(n) = n - 1 = O(n)$. □

Esercizio 3 [11 punti] Una formula Φ si definisce **diretta** se non contiene negazioni. Si consideri il seguente problema decisionale:

K DIRECT-2-SAT (K-D2S):

ISTANZA: $\langle \Phi(x_1, x_2, \ldots x_n) = C_1 \wedge C_2 \wedge \ldots \wedge C_m, k \rangle$,
Φ formula 2-CNF diretta e $1 \leq k \leq n$.

DOMANDA: Esiste un assegnamento di valori di verità $\boldsymbol{b} \in \{0, 1\}^n$ con **esattamente** k **valori a 1**, che soddisfa Φ?

A titolo di esempio, la formula $\Phi(x_1, x_2, x_3) = (x_1 \vee x_3) \wedge (x_2 \vee x_3)$ è 2-CNF diretta e la corrispettiva istanza $\langle \Phi, 1 \rangle$ è positiva (si consideri $\boldsymbol{b} = (0, 0, 1)$).
Si dimostri che K-D2S $\in NPH$. (*Suggerimento:* si riduca da VERTEX-COVER).

Answer: Following the hint, we reduce from VERTEX-COVER as follows. Let $\langle G = (V, E), k \rangle$ be a generic instance of VERTEX-COVER, and let

$$f(\langle G = (V, E), k \rangle) = \langle \Phi_G(x_1, x_2, \ldots, x_{|V|}) = C_1 \wedge C_2 \wedge \ldots \wedge C_{|E|}, k \rangle$$

where $C_j = (x_r \vee x_s) \Leftrightarrow e_j = \{v_r, v_s\}$. In other words, the formula is defined on $|V|$ variables, corresponding to the nodes of G, and there is one clause for each edge, containing the two variables corresponding to the endpoints of the edge. The above reduction is clearly polynomial (in fact, linear) in $|\langle G = (V, E), k \rangle|$.

In order to prove that f is indeed a reduction, we must show that

$$\langle G = (V, E), k \rangle \in \text{VERTEX-COVER} \Leftrightarrow f(\langle G = (V, E), k \rangle) \in \text{K-D2S}.$$

($\Rightarrow$) Given a vertex cover $V' = \{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$ of size k, set $b_{i_j} = 1$ for $1 \leq j \leq k$ and all other values to zero. The resulting truth assignment $\boldsymbol{b}$ has exactly k values set to 1 and satisfies Φ_G, since each clause (which corresponds to an edge) must contain a variable corresponding to a vertex in V'.

($\Leftarrow$) Starting from a satisfying assignment $\boldsymbol{b}$ with exactly k values set to 1, the set V' of vertices corresponding to the indices $b_{i_j} = 1$, for $1 \leq j \leq k$ has size k and the property that each edge is incident on at least one such vertex, or otherwise the clause corresponding to that edge would not be true under the assignment.

□

Dati e Algoritmi 2 – CdL Magistrale in Ingegneria Informatica
Fondamenti di Informatica II – CdL in Ingegneria Informatica (V. O.)

Compito, 15/2/2010 (Durata: 3h)

Nome, Cognome, Matricola: _____

Corso di studio: _____

Prima Parte: domande teoriche

Si forniscano risposte il più possibile **rigorose e succinte** ai seguenti quesiti riguardanti argomenti trattati nel corso. Risposte approssimative, prolisse o in cattivo italiano saranno **fortemente penalizzate**. Ai fini del superamento dell'esame, è necessario conseguire una valutazione sufficiente alla prima parte.

T1 Data la ricorrenza

$$T(n) = \begin{cases} 0 & n \leq n_0, \\ s(n)T(f(n)) + w(n) & n > n_0, \end{cases}$$

si forniscano dominio e codominio delle funzioni s e f e si scriva la formula chiusa, descrivendo **sinteticamente** le relazioni tra le varie componenti della formula e le informazioni contentute nell'albero della ricorrenza.

Solution: We have that $s, f : \boldsymbol{N} \to \boldsymbol{N}$. Moreover, they are nondecreasing, and f is a contraction, that is, $f(n) < n$ for $n > 0$. Since $T(n) = 0$ for $n \leq n_0$ the closed formula reduces to

$$T(n) = \sum_{\ell=0}^{f^\star(n,n_0)} \left(\prod_{j=0}^{\ell-1} s\left(f^{(j)}(n)\right) \right) w\left(f^{(\ell)}(n)\right).$$

In the above formula, the summation accounts for the contributions of the internal levels of the recursion tree, going from the root ($\ell = 0$) to the level immediately preceeding the level of the leaves ($\ell = f^\star(n, n_0)$). Within the summation, the product corresponds to the number of nodes at level ℓ (obtained by multiplying the outdegrees of the nodes on the preceeding levels), while the term $w\left(f^{(\ell)}(n)\right)$ is the work associated to each node on level ℓ.

We observe that the formula holds also for general nonnegative real-valued

values of $s(n)$, but the interpretation of its components on the recursion tree is clearly no longer valid.

T2 Si dimostri la seguente proposizione: una qualsiasi parentesizzazione ottima per la catena di matrici $A_{i..j} = \langle A_i, A_{i+1}, \ldots, A_j \rangle$, con $1 \leq i < j \leq n$, contiene al suo interno due parentesizzazioni ottime di sottocatene $A_{i..k}$ e $A_{k+1..j}$, per un qualche k, $i \leq k < j$.

Solution: Let $T^{\star}_{i,j}$ be an optimal parenthesization (represented as a full binary tree) of $A_{i..j}$. Since $i < j$, the root of $T^{\star}_{i,j}$ is an internal node, hence it has two subtrees T_1 and T_2. Let T_1 contain the leaves $A_i, \ldots, A_k$ and T_2 contain the leaves $A_{k+1}, \ldots, A_j$, for some k, $i \leq k < j$. We claim that T_1 is optimal for $A_{i..k}$ and T_2 is optimal for $A_{k+1..j}$. Suppose that this were not the case, e.g., T_1 is not optimal, and let $T^{\star}_{i,k}$ be an optimal tree for $A_{i..k}$. Then, we could obtain a tree of cost lower than $T^{\star}_{i,j}$ for $A_{i..j}$ by simply substituting T_1 with $T^{\star}_{i,k}$ in $T^{\star}_{i,j}$, a contradiction.

T3 Dando per nota la proposizione

$$\forall L_1, L_2 \in \{0,1\}^{\star} : (L_1 <_P L_2) \wedge (L_2 \in P) \Rightarrow (L_1 \in P),$$

si dimostri formalmente che

$$(P \cap NPC \neq \emptyset) \Rightarrow (P = NP)$$

Solution: Since it is already known that $P \subseteq NP$, it suffices to show that under the hypothesis $(P \cap NPC \neq \emptyset)$ we also have $NP \subseteq P$. To this purpose, let $\bar{L} \in P \cap NPC$, which exists since we are assuming that $P \cap NPC \neq \emptyset$. Since $\bar{L} \in NP$, we have that $\forall L \in NP : (L <_P \bar{L})$. Moreover, $(\bar{L} \in P)$. Hence, using the first proposition, $\forall L \in NP : L \in P$, and the thesis follows.

Seconda Parte: risoluzione di problemi

Si forniscano soluzioni esaurienti e rigorose ai tre problemi seguenti. Gli algoritmi vanno codificati utilizzando lo **pseudocodice** usato in classe. **Attenzione:** Risposte immotivate e prive di prova di correttezza non saranno considerate.

Esercizio 1 [11 punti] Utilizzando la formula generale vista in classe, si determini la formula chiusa della seguente ricorrenza per valori del parametro $n = 3^{2^k}, k \geq 0$, e se ne verifichi la correttezza per induzione:

$$T(n) = \begin{cases} 0 & n = 3, \\ n^{\frac{1}{3}}\, T\left(n^{\frac{1}{2}}\right) + n^{\frac{2}{3}} & n > 3. \end{cases}$$

(*Suggerimento:* Si faccia attenzione alle basi dei logaritmi.)

Answer: Observe that the above recurrence cannot yield an integer function, since $\sqrt[3]{n}$ is not an integer for $n = 3^{2^k}, k \geq 0$. However, the general formula is robust with respect to noninteger values of $s(n)$, hence we can still apply it with $s(n) = \sqrt[3]{n}$, $f(n) = \sqrt{n}$, and $w(n) = n^{\frac{2}{3}}$. Also, set $T_0 = 0$ and $n_0 = 3$. Since $f^{(j)}(n) = n^{\frac{1}{2^j}}$, we have

$$n^{\frac{1}{2^j}} > 3 \Leftrightarrow j < \log_2 \log_3 n.$$

Since the latter double logarithm is an integer for $n = 3^{2^k}$, we obtain $f^\star(n, 3) = \log_2 \log_3 n - 1$. Now,

$$\begin{aligned} \prod_{j=0}^{\ell-1} s\left(f^{(j)}(n)\right) &= \prod_{j=0}^{\ell-1} n^{\frac{1}{2^j}\frac{1}{3}} \\ &= n^{\frac{1}{3}\sum_{j=0}^{\ell-1}\frac{1}{2^j}} \\ &= n^{\frac{2}{3}\left(1-\frac{1}{2^\ell}\right)}. \end{aligned}$$

Observe that $T_0 = 0$, hence the closed formula has a single term:

$$\begin{aligned} T(n) &= \sum_{\ell=0}^{\log_2 \log_3 n - 1} n^{\frac{2}{3}\left(1-\frac{1}{2^\ell}\right)} w\left(f^{(l)}(n)\right) \\ &= \sum_{\ell=0}^{\log_2 \log_3 n - 1} n^{\frac{2}{3}\left(1-\frac{1}{2^\ell}\right)} n^{\frac{1}{2^\ell}\frac{2}{3}} \\ &= \sum_{\ell=0}^{\log_2 \log_3 n - 1} n^{\frac{2}{3}} \\ &= n^{\frac{2}{3}} \log_2 \log_3 n. \end{aligned}$$

The closed formula can be easily verified by induction on $k \geq 0$, when $n = 3^{2^k}$. Indeed, for $k = 0$, we have $n = 3$ and $\log_2 \log_3 3 = 0$ whence $T(n) = 0$. Let the hypothesis hold for values less than k, with $k > 0$, and recall that $n = 3^{2^k}$. We have:

$$\begin{aligned} T(n) &= \sqrt[3]{n} T\left(\sqrt{n}\right) + n^{\frac{2}{3}} \\ &= n^{\frac{1}{3}} n^{\frac{1}{2}\frac{2}{3}} \log_2 \log_3 n^{\frac{1}{2}} + n^{\frac{2}{3}} \\ &= n^{\frac{2}{3}} (\log_2 \log_3 n - 1) + n^{\frac{2}{3}} \\ &= n^{\frac{2}{3}} \log_2 \log_3 n, \end{aligned}$$

and the thesis follows. □

Esercizio 2 [11 punti] Sia $n > 1$ una potenza di due. Dati $a, b \in \boldsymbol{C}$, un vettore $x \in \boldsymbol{C}^n$ si dice un (n, a, b)-*scalino* se $x_0 = x_1 = \ldots = x_{n/2-1} = a$ e $x_{n/2} = x_{n/2+1} = \ldots = x_{n-1} = b$. Si fornisca lo pseudocodice e si analizzi la correttezza e la complessità di un algoritmo STEP_DFT($\boldsymbol{x}$) che, dato in ingresso un vettore (n, a, b)-*scalino*, ritorni $\boldsymbol{y} = DFT_n(\boldsymbol{x})$ eseguendo un numero al più lineare di operazioni aritmetiche (somme, sottrazioni, moltiplicazioni e divisioni) tra scalari complessi.

Answer: For $n > 1$ a power of two, let $\boldsymbol{x} \in \boldsymbol{C}^n$ be an (n, a, b)-step, and let $\boldsymbol{y} = DFT_n(\boldsymbol{x})$. We obtain a closed formula on the components of $\boldsymbol{y}$ as follows. Observe that

$$\begin{aligned} y_i &= \sum_{j=0}^{n-1} x_j \omega_n^{ij} \\ &= a \sum_{j=0}^{n/2-1} \omega_n^{ij} + b\omega_n^{in/2} \sum_{j=0}^{n/2-1} \omega_n^{ij} \\ &= \left(a + b\omega_n^{in/2}\right) \sum_{j=0}^{n/2-1} \omega_n^{ij}. \end{aligned}$$

Let us first consider the case $i = 0$. Then we obtain $y_0 = (n/2)(a + b)$. For $0 < i < n$, we have that $\omega_n^i \neq 1$, hence we can apply the closed formula for the (truncated) power series to obtain:

$$\begin{aligned} y_i &= \left(a + b\omega_n^{in/2}\right) \frac{1 - \omega_n^{in/2}}{1 - \omega_n^i} \\ &= \left(a + b(-1)^i\right) \frac{(1 - (-1)^i)}{1 - \omega_n^i} \end{aligned}$$

where the last equation holds since, by the cancellation lemma, $\omega_n^{in/2} = \omega_2^i = (-1)^i$. Since $(-1)^i = 1$ if i is even, and $(-1)^i = -1$ if i is odd, we conclude that for $0 \leq i < n$:

$$y_i = \begin{cases} (n/2)(a+b) & i = 0, \\ 0 & i > 0, i \text{ even}, \\ (a-b)\frac{2}{1-\omega_n^i} & i \text{ odd}. \end{cases}$$

The algorithm follows.

```
STEP_DFT(x)
n ← x.length
a ← x_0; b ← x_{n-1}
y_0 ← (n/2)(a + b)
ONI ← ω_n
y_1 ← 2 · (a − b)/(1 − ONI)
for j ← 1 to n/2 − 1 do
    y_{2j} ← 0
    ONI ← ONI · ω_n · ω_n
    y_{2j+1} ← 2 · (a − b)/(1 − ONI)
return y
```

The correctness of the algorithm follows from the closed formula obtained above, and its running time is clearly linear in n. As an aside, observe that the assumption on n being a power of two is too restrictive, since in fact the formula (and the algorithm) holds for any *even value* of n.

We could have also modified the standard FFT algorithm by observing that for $n > 2$ a power of two, the even and odd indexed components of an (n, a, b)-step are both an $(n/2, a, b)$-step, hence a single recursive call suffices. Care must be taken to stop the recurrence at $n = 2$ (returning vector $(a + b, a - b)$), since a $(2, a, b)$-step constitutes a base case which cannot be further decomposed. This yields an algorithm whose running time is $T(n) = T(n/2) + O(n)$, whence $T(n) = O(n)$ by the Master Theorem. This approach, however, is less general than the previous one, since it only applies to values of $n > 1$ which are powers of two. □

Esercizio 3 [11 punti] Sia $n > 0$ e sia $\boldsymbol{a} \in \boldsymbol{R}^n$ un vettore di n reali. Per $1 \leq i \leq j \leq n$, si definisca la seguente ricorrenza:

$$m(i,j) = \begin{cases} a_i & i = j, \\ m(i, j-1) - m(i+1, j) + a_i - a_j & i < j. \end{cases}$$

Si progetti e si analizzi un algoritmo *iterativo* COMPUTE($\boldsymbol{a}$) che, dato in ingresso il vettore $\boldsymbol{a}$, ritorni $m(1, n)$ eseguendo $O(n^2)$ operazioni tra reali e utilizzando solo spazio lineare in n.

Answer: Observe that for $i < j$, $m(i,j)$ depends on values $m(i,j-1)$ and $m(i+1,j)$, and both these values are found on the diagonal of equation $\ell' = j - i$, at the i-th and $(i+1)$-th position of the diagonal, respectively, while $m(i,j)$ itself is situated at the i-th position on the "next" diagonal of equation $\ell = j - i + 1$. Therefore, if we perform a diagonal-wise scan, it will suffice to maintain a single vector L containing, at the beginning of the ℓ-th iteration, the values computed for the previous diagonal ℓ'. Vector L is updated as we compute the values on diagonal ℓ. Observe that each iteration updates one element less than the previous iteration. The value to be returned will be found in $L[1]$ for $\ell = n$. The algorithm follows

```
COMPUTE(a)
n ← a.length
for i ← 1 to n do L[i] ← a_i
for ℓ ← 2 to n do
    for i ← 1 to n − ℓ + 1 do
    j ← i + ℓ − 1
    L[i] ← L[i] − L[i + 1] + a_i − a_j
return L[1]
```

The algorithm implements a correct diagonal scan, and uses linear space. As for its running time, the number of operations between reals is

$$\begin{aligned} T(n) &= \sum_{\ell=2}^{n} \sum_{i=1}^{n-\ell+1} 3 \\ &= 3 \sum_{\ell=2}^{n} n - \ell + 1 \\ &= 3 \sum_{\ell=1}^{n-1} 1 \\ &= \frac{3(n-1)n}{2} \\ &= O(n^2). \end{aligned}$$

□

Dati e Algoritmi 2 – CdL Magistrale in Ingegneria Informatica

Compito, 2/9/2010 (Durata: 3h)

Nome, Cognome, Matricola: _____

Corso di studio: _____

Prima Parte: domande teoriche

Si forniscano risposte il più possibile **rigorose e succinte** ai seguenti quesiti riguardanti argomenti trattati nel corso. Risposte approssimative, prolisse o in cattivo italiano saranno **fortemente penalizzate**. Ai fini del superamento dell'esame, è necessario conseguire una valutazione **pienamente sufficiente** alla prima parte.

T1 Si spieghi brevemente perché al generico livello ℓ dell'albero delle chiamate dell'algoritmo di Karatsuba possono trovarsi sottoistanze di taglia diversa e si fornisca un semplice esempio di tale fenomeno per input di taglia $n = 4$ e $\ell = 1$.

T2 Si enunci formalmente, senza provarla, la proprietà di sottostruttura ottima su cui si basa l'algoritmo di programmazione dinamica per la moltiplicazione di matrici a catena.

T3 Si provi:

$$\langle G = (V, E), k\rangle \in \text{CLIQUE} \Leftrightarrow \langle G^c = (V, E^c), k\rangle \in \text{INDEPENDENT SET}$$

Answers

T1 The phenomenon is due to the fact that, in general, the subinstances generated by the Karatsuba strategy on an instance of size n may have size $\lfloor n/2 \rfloor$, $\lceil n/2 \rceil$, and $\lceil n/2 \rceil + 1$, respectively, hence different sizes are to be expected at the same level of the recursion tree. For $n = 4$, consider input $(1010, 1111)$. Then the three subinstances at level 1 are $(10, 11)$, $(10, 11)$, and $(100, 110)$ of size 2, 2, and 3, respectively.

T2 Let $T^\star_{i..j}$ be the tree corresponding to the optimal parenthesization for computing $A_{i..j}$. Then, $T^\star_{i..j}$ consists solely of the root if $i = j$. For $i < j$, there exists a value $k^\star$, with $i \leq k^\star < j$, such that $T^\star_{i..j}$ contains the optimal trees $T^\star_{i..k^\star}$ and $T^\star_{k^\star+1..j}$ as its left and right subtree, respectively. Moreover,

$$k^\star = \text{argmin}\left\{\text{cost}\left(T^\star_{i..k}\right) + \text{cost}\left(T^\star_{k+1..j}\right) + p_{i-1}p_k p_j : i \leq k < j\right\}$$

T3

$$\begin{aligned}
&\langle G = (V, E), k\rangle \in \text{CLIQUE} \\
&\quad \Leftrightarrow \quad \exists V' \subseteq V, |V'| = k : \forall u, v \in V, u \neq v : u, v \in V' \Rightarrow \{u, v\} \in E \\
&\quad \Leftrightarrow \quad \exists V' \subseteq V, |V'| = k : \forall u, v \in V, u \neq v : u, v \in V' \Rightarrow \{u, v\} \notin E^c \\
&\langle G^c = (V, E^c), k\rangle \in \text{INDEPENDENT SET}
\end{aligned}$$

Seconda Parte: risoluzione di problemi

Si forniscano soluzioni esaurienti e rigorose ai tre problemi seguenti. Gli algoritmi vanno codificati utilizzando lo **pseudocodice** usato in classe. **Attenzione:** Risposte immotivate e prive di prova di correttezza non saranno considerate.

Esercizio 1 [10 punti] Sia n una potenza di due. Per un dato problema computazionale Π, si supponga di avere un algoritmo iterativo A_1, la cui complessità su istanze di taglia n sia $T_1(n) = n^2$ e un algoritmo ricorsivo A_2, la cui complessità sia regolata dalla seguente ricorrenza:

$$T(n) = \begin{cases} 1 & n = 1, \\ 2T(n/2) + 7n & n > 1. \end{cases}$$

1. Analizzando l'albero delle chiamate, si determini la complessità $T_{n_0}(n)$ del generico algoritmo ibrido A_H che utilizza la strategia ricorsiva di A_2 per taglie dell'istanza $n > n_0$ e invoca A_1 per taglie $n \leq n_0$, dove n_0 è una potenza di due.

2. Si determini il valore $\bar{n}_0$ che fornisce il miglior algoritmo ibrido.

Answer:

Point 1 Let i denote a generic instance of the problem, and let $|i|$ denote its size, assumed to be a power of two. The pseudocode of the hybrid algorithm is the following:

$A_H(i)$
if $|i| \leq n_0$
  then return $A_1(i)$
$\star\star$ divide, recurse, and conquer part of $A_2(i)$ where recursive calls to A_2 are replaced by recursive calls to A_H $\star\star$

The correctness of A_H is clearly implied by the correctness of A_1 and A_2, while its running time is regulated by the following recurrence parametric in n_0:

$$T_{n_0}(n) = \begin{cases} n^2 & n \leq n_0, \\ 2T_{n_0}(n/2) + 7n & n > n_0. \end{cases}$$

Using the general formula seen in class with $s(n) = 2$, $f(n) = n/2$, $w(n) = 7n$, $f^{(\ell)}(n) = n/2^\ell$ (whence $f^\star(n, n_0) = \log(n/n_0) - 1$) we obtain, for $n \geq n_0$:

$$\begin{aligned} T_{n_0}(n) &= \left(\sum_{\ell=0}^{\log(n/n_0)-1} 2^\ell \cdot 7(n/2^\ell) \right) + 2^{\log(n/n_0)} \cdot n_0^2 \\ &= 7n \log(n/n_0) + n \cdot n_0 \\ &= 7n \log n + n(n_0 - 7 \log n_0) \end{aligned}$$

Point 2 In order to achieve the best running time, we study the sign of the partial derivative of $T_{n_0}(n)$ w.r.t. n_0:

$$\begin{aligned} \frac{\partial T_{n_0}(n)}{\partial n_0} &\geq 0 \\ &\Leftrightarrow n \left(1 - \frac{7 \log e}{n_0}\right) \geq 0 \\ &\Leftrightarrow n_0 \geq 7 \log e \simeq 10.1 \end{aligned}$$

Therefore, the best choiche for n_0 is either 8 or 16. By plugging in these two values we obtain $T_8(n) = 7n \log n - 13n$, and $T_{16}(n) = 7n \log n - 12n$, hence we conclude that the best choice is $n_0 = 8$. □

Esercizio 2 [10 punti] Si consideri un file $\mathcal{F}$ definito sull'alfabeto $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$, con relative frequenze $f[\mathtt{a}], f[\mathtt{b}], f[\mathtt{c}]$. Si determini, se esiste, un opportuno assegnamento di valori alle tre frequenze che conduca alle seguenti codifiche di Huffman, oppure si dimostri formalmente che tali codifiche non sono mai ottenibili:

1. $e(\mathtt{a}) = \mathtt{0}, e(\mathtt{b}) = \mathtt{10}, e(\mathtt{c}) = \mathtt{11}$;
2. $e(\mathtt{a}) = \mathtt{0}, e(\mathtt{b}) = \mathtt{1}, e(\mathtt{c}) = \mathtt{00}$;
3. $e(\mathtt{a}) = \mathtt{10}, e(\mathtt{b}) = \mathtt{01}, e(\mathtt{c}) = \mathtt{00}$.

Answer:

1. The set of codewords is indeed a valid prefix-free code, which will be the output of the Huffman coding algorithm whenever (a) $f[\mathtt{b}] < f[\mathtt{c}] < f[\mathtt{a}]$, which implies that the nodes corresponding to `b` and `c` will be merged as the greedy choice, with the former node becoming the left leaf of the resulting subtree; and (b) $f[\mathtt{a}] < f[\mathtt{b}] + f[\mathtt{c}]$, which implies that the node corresponding to `a` will become the left child of the root of the final tree. A set of frequencies satisfying constraints (a) and (b) are $f[\mathtt{a}] = 0.4, f[\mathtt{b}] = 0.25, f[\mathtt{c}] = 0.35$.
2. Codeword $e(\mathtt{a}) = \mathtt{0}$ is a prefix of codeword $e(\mathtt{c}) = \mathtt{00}$. Therefore, the resulting code is not prefix-free, hence it will never be output by the Huffman coding algorithm

3. The tree corresponding to this prefix-free code is not a full tree, since the right child of the root (an internal node) has only the left child, hence such a tree cannot be optimal. Therefore, the resulting code will never be output by the Huffman coding algorithm.

□

Esercizio 3 [11 punti] Si ricordi che, dato un grafo non orientato $G = (V, E)$, un insieme indipendente $S \subseteq V$ è un sottoinsieme di nodi tale che:

$$\forall u, v \in V, u \neq v : (u \in S) \wedge (v \in S) \Rightarrow \{u, v\} \notin E.$$

Si consideri ora il seguente problema decisionale:

DOUBLE INDEPENDENT SET (DIS):
ISTANZA: $\langle G = (V, E), k\rangle$, G grafo non orientato, $k \leq \lfloor |V|/2 \rfloor$.
DOMANDA: Esistono due insiemi independenti $S_1, S_2 \subseteq V$ tali che $|S_1| = |S_2| = k$ e $S_1 \cap S_2 = \emptyset$?

Si dimostri che DIS $\in NPH$.

Answer: Clearly, it makes sense to reduce from INDEPENDENT SET (IS). Let $\langle G = (V, E), k\rangle$ be an instance of IS, with $V = \{v_1, v_2, \ldots v_{|V|}\}$. The first idea that springs to mind is to reduce IS to DIS, obtaining $f(\langle G = (V, E), k\rangle) = \langle G' = (V', E'), k\rangle$ by collating two distinct copies of G. However, we must pay attention to preserve the maximum size of independent sets in G, or the reduction could not possibly work. To see this, consider a distinct copy of G, call it $G^1 = (V^1, E^1)$, with $V^1 = \{v_1^1, v_2^1, \ldots, v_{|V|}^1\}$, $V \cap V^1 = \emptyset$, and, for $1 \leq i < j \leq |V|$, $\{v_i^1, v_j^1\} \in E^1 \Leftrightarrow \{v_i, v_j\} \in E$. If we set $G' = (V \cup V_1, E \cup E_1)$, then the above reduction cannot be valid, since G' has larger independent sets than G. (For instance, consider a graph G of two nodes connected by an edge, which clearly has only independent sets of size 1, while G' has two disjoint independent sets of size 2!)

We solve the above problem by making sure that no independent set in G' may contain both nodes in V and V^1, by fully connecting the two copies, i.e., adding an edge between each pair v_i and v_j^1, for $1 \leq i, j \leq |V|$. Hence, we set $V' = V \cup V_1$ and

$$E' = \{\{v_i^1, v_j^1\} : \{v_i, v_j\} \in E\} \cup \{\{v_i, v_j^1\} : 1 \leq i, j \leq |V|\}.$$

An easy consequence of the above definition is every independent set of G is an independent set of G' (in fact, two distinct copies of such an independent set exist in G') but, most importantly, an independent set in G' always corresponds to an independent set of G of the same size.

We now show that $f(\langle G = (V, E), k\rangle) = \langle G' = (V', E'), k\rangle$ is a valid polynomial-time computable reduction from IS to DIS. Clearly, f is computable in polynomial (in fact, at most quadratic) time. Assume now that $\langle G = (V, E), k\rangle \in$ IS. Then there is an independent set in G of size k, and, clearly there are two disjoint copies

of such an independent set, one in G and one in G^1, hence $\langle G' = (V', E'), k\rangle \in \text{DIS}$. Vice versa, if G' contains two disjoint independent sets of size k, this implies that there is an independent set in G of the same size, by the correspondence argued above. Thus, we have shown that

$$\langle G = (V, E), k\rangle \in \text{IS} \Leftrightarrow f(\langle G = (V, E), k\rangle) = \langle G' = (V', E'), k\rangle \in \text{DIS}$$

whence $\text{DIS} \in NPH$.

□