Dati e Algoritmi 2 – CdL Magistrale in Ingegneria Informatica

Compito, 1/2/2011 (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

Compito, 23/2/2011 (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 ingiustificate, approssimative, prolisse o in cattivo italiano saranno **fortemente penalizzate**. Ai fini del superamento dell'esame, è necessario conseguire una valutazione sufficiente alla prima parte, che richiede che almeno due esercizi su tre siano svolti in modo sufficiente.

T1 Si dimostri che $[F_n^{-1}]_{ij} = (1/n)\omega_n^{-ij}$.

Solution: It suffices to show that $F_n F_n^{-1} = F_n^{-1} F_n = I_n$, where I_n denotes the identity matrix. We have

$$\begin{aligned}[F_n F_n^{-1}]_{ij} &= (1/n)\sum_{k=0}^{n-1} \omega_n^{ik}\omega_n^{-kj} \\ &= (1/n)\sum_{k=0}^{n-1} \left(\omega_n^{i-j}\right)^k\end{aligned}$$

Observe that when $0 \leq i, j \leq n-1$, we have that $-(n-1) \leq i-j \leq (n-1)$, hence $i-j$ is a multiple of n if and only if $i-j=0$, that is, $i=j$. Thus, by applying the summation lemma, we obtain that for $0 \leq i, j < n$:

$$[F_n F_n^{-1}]_{ij} = \begin{cases} (1/n)\cdot n = 1, & i = j, \\ 0, & i \neq j \end{cases} = [I_n]_{ij}$$

The same argument can be applied to prove that $F_n^{-1} F_n = I_n$, and the statement follows.

T2 Si elenchino le parole di codice risultanti dalla codifica di Huffman per l'alfabeto $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}\}$ quando le frequenze dei singoli caratteri sono le seguenti: $f(\mathtt{a}) = 0.08, f(\mathtt{b}) = 0.23, f(\mathtt{c}) = 0.01, f(\mathtt{d}) = 0.48, f(\mathtt{e}) = 0.05, f(\mathtt{f}) = 0.15$. Nella costruzione dell'albero, si ricordi di etichettare con 0 l'arco che connette un nodo interno con il suo sottoalbero di frequenza cumulativa minore.

Solution: The codwords can be obtained by running the algorithm implementing the greedy strategy which returns an optimal prefix code for the given alphabet and frequencies. They are: $e(\texttt{a}) = \texttt{1101}$, $e(\texttt{b}) = \texttt{10}$, $e(\texttt{c}) = \texttt{11000}$, $e(\texttt{d}) = \texttt{0}$, $e(\texttt{e}) = \texttt{11001}$, $e(\texttt{f}) = \texttt{111}$.

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:

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

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 valori del parametro $n = 3^{3^i}, i \geq 0$:

$$T(n) = \begin{cases} 0 & n = 3, \\ n^{4/3} T\left(n^{1/3}\right) + n^2 & n > 3. \end{cases}$$

Punto 1 Si determini una formula chiusa per la funzione $T(n)$.

Punto 2 Si verifichi la correttezza dell'espressione analitica ottenuta applicando l'induzione.

Answer:

Point 1 Let us apply the closed formula studied in class, taking into account that $s(n) = n^{4/3}$, $f(n) = n^{1/3}$, $n_0 = 3$, (whence $f^{(j)}(n) = n^{1/3^j}$ for $j \geq 0$, and $f^\star(n, n_0) = f^\star(n, 3) = \log_3 \log_3 n - 1$), $w(n) = n^2$, and, finally, $T_0 = 0$. Observe that the contribution of the leaves of the recursion tree to the global work is null, hence it suffices to evaluate the contribution of the internal nodes. For $0 \leq \ell \leq \log_3 \log_3 n - 1$, the work contributed by each node of level ℓ is $w\left(f^\ell(n)\right) = n^{2/3^\ell}$, and the number of such nodes is:

$$\begin{aligned} \prod_{j=0}^{\ell-1} s\left(f^{(j)}(n)\right) &= \prod_{j=0}^{\ell-1} \left(n^{1/3^j}\right)^{4/3} \\ &= n^{(4/3)\sum_{j=0}^{\ell-1} 1/3^j} \\ &= n^{2(1-1/3^\ell)} \end{aligned}$$

Therefore,

$$\begin{aligned} T(n) &= \sum_{\ell=0}^{\log_3 \log_3 n - 1} n^{2(1-1/3^\ell)} n^{2/3^\ell} \\ &= \sum_{\ell=0}^{\log_3 \log_3 n - 1} n^2 \\ &= n^2 \log_3 \log_3 n. \end{aligned}$$

Point 2 Since $\log_3 \log_3 3 = \log_3 1 = 0$, the closed formula is correct for the base case $n = 3$. Assume now that hypothesis holds for double powers of 3 which are less than n. Then we have:

$$\begin{aligned} T(n) &= n^{4/3} T\left(n^{1/3}\right) + n^2 \\ &= n^{4/3} n^{2/3} \log_3 \log_3 n^{1/3} + n^2 \\ &= n^2 \log_3 \left((1/3) \log_3 n\right) + n^2 \\ &= n^2 (\log_3 \log_3 n - 1) + n^2 \\ &= n^2 \log_3 \log_3 n \end{aligned}$$

and the thesis follows. □

Esercizio 2 [11 punti] Dati n programmi $P_1, P_2, \ldots, P_n$, si supponga che il tempo di esecuzione di P_i su una macchina $\mathcal{M}$ sia t_i, con $1 \leq i \leq n$ e, per semplicità, si assuma che i t_i siano tutti valori distinti. Un *ordine di esecuzione* degli n programmi su $\mathcal{M}$ è una permutazione $\pi : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$, che prescrive che i programmi vengano eseguiti uno dopo l'altro su $\mathcal{M}$ nella sequenza $\langle P_{\pi(1)}, P_{\pi(2)}, \ldots, P_{\pi(n)} \rangle$. Dato un ordine di esecuzione π, il tempo di attesa dell' i-simo programma eseguito, per $i \geq 2$, è la quantità $\tau_i = \sum_{j=1}^{i-1} t_{\pi(j)}$ e l'attesa cumulativa di π è l'espressione $A_\pi = \sum_{i=2}^{n} \tau_i$. Si vuole determinare l'ordine di esecuzione $\pi^\star$ associato alla minima attesa cumulativa $A_{\pi^\star}$.

Si dimostri la seguente proprietà di scelta greedy: per un *qualsiasi* ordine di esecuzione ottimo $\pi^\star$ si ha che $\pi^\star(1) = i_{\min} = \text{argmin}\{t_i : 1 \leq i \leq n\}$.

Answer: For a given execution order π, observe that

$$A_\pi = \sum_{i=2}^{n} \tau_i = \sum_{i=2}^{n} \sum_{j=1}^{i-1} t_{\pi(j)}.$$

In the above double summation, the term $t_{\pi(1)}$ (the running time of the first job executed in the order) appears $n - 1$ times, the term $t_{\pi(2)}$ appears $n - 2$ times, and so forth. In general, for $1 \leq k \leq n$, the term $t_{\pi(k)}$ appears $n - k$ times. Hence, we can re-write A_π as follows:

$$A_\pi = \sum_{k=1}^{n} (n - k) t_{\pi(k)}.$$

Let $\pi^\star$ be an optimal execution order and let $i_{\min} = \text{argmin}\{t_i : 1 \leq i \leq n\}$. Assume now, for the sake of contradiction, that $\pi^\star(1) \neq i_{\min}$. Therefore, $\pi^\star(1) =$

i_1, with $t_{i_1} > t_{i_{\min}}$ and there is a value $h > 1$ such that $\pi^\star(h) = i_{\min}$. We can then obtain a new execution order $\hat{\pi}$ as follows:

$$\hat{\pi}(k) = \begin{cases} i_{\min} & k = 1 \\ i_1 & k = h \\ \pi^\star(k) & \text{otherwise} \end{cases}$$

In other words, in $\hat{\pi}$ we swap the order of execution of $P_{i_{\min}}$ (which now becomes the first job to be executed) and P_{i_1} (which becomes the h-th job to be executed). Then we have:

$$\begin{aligned} A_{\pi^\star} - A_{\hat{\pi}} &= \sum_{k=1}^{n}(n-k)t_{\pi^\star}(k) - \sum_{k=1}^{n}(n-k)t_{\hat{\pi}(k)} \\ &= (n-1)t_{i_1} + (n-h)t_{i_{\min}} - (n-1)t_{i_{\min}} - (n-h)t_{i_1} \\ &= (n-1)(t_{i_1} - t_{i_{\min}}) - (n-h)(t_{i_1} - t_{i_{\min}}) \\ &= (h-1)(t_{i_1} - t_{i_{\min}}) \\ &> 0, \end{aligned}$$

since $h > 1$ and $t_{i_1} > t_{i_{\min}}$. This is a contradiction, since we were assuming that $\pi^\star$ was an optimal execution order, which should be associated with the minimum cumulative wait time $A_{\pi^\star}$. □

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

7-3-CNF-SAT (7-3-CNF-SAT):

ISTANZA: $\langle \Phi(x_1, x_2, \ldots x_n) = C_1 \wedge C_2 \wedge \ldots \wedge C_m \rangle$,
Φ formula 3-CNF

DOMANDA: Esistono almeno **sette** assegnamenti distinti di valori di verità $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_7 \in \{0,1\}^n$ che soddisfano Φ?

Si dimostri che 7-3-CNF-SAT è NP-Hard.

Answer: We reduce from 3-CNF-SAT as follows. Let $\langle \Phi(x_1, x_2, \ldots x_n) = C_1 \wedge C_2 \wedge \ldots \wedge C_m \rangle$ be an instance of 3-CNF-SAT. Our reduction function f is defined as follows:

$$\begin{aligned} &f\left(\langle \Phi(x_1, x_2, \ldots x_n) = C_1 \wedge C_2 \wedge \ldots \wedge C_m \rangle\right) \\ &\quad = \langle \Phi'(x_1, x_2, \ldots x_n, x_{n+1}, x_{n+2}, x_{n+3}) = C_1 \wedge C_2 \wedge \ldots \wedge C_m \wedge C_{m+1} \rangle \end{aligned}$$

where $C_{m+1} = (x_{n+1} \vee x_{n+2} \vee x_{n+3})$. Clearly, f is computable in polynomial (in fact, linear) time.

Assume that $\langle \Phi(x_1, x_2, \ldots x_n) \rangle \in$ 3-CNF-SAT. Then, there exists a truth assignment $\boldsymbol{b} \in \{0,1\}^n$ to its n variables such that $\Phi(\boldsymbol{b}) = 1$. Since the clause C_{m+1} is satisfied by any of the seven assignments to three variables which contain at least one component set to 1, $\boldsymbol{t}_i \in \{0,1\}^3$, $1 \leq i \leq 7$, we have that $\Phi'(\boldsymbol{b}|\boldsymbol{t}_i) = 1$, for $1 \leq i \leq 7$, hence $f\left(\langle \Phi(x_1, x_2, \ldots x_n) \rangle\right) \in$ 7-3-CNF-SAT. Vice versa, assume that $\langle \Phi'(x_1, x_2, \ldots x_n, x_{n+1}, x_{n+2}, x_{n+3}) \rangle = f\left(\langle \Phi(x_1, x_2, \ldots x_n) \rangle\right) \in$ 7-3-CNF-SAT, and observe that any of the 7 truth assignments $\boldsymbol{c} \in \{0,1\}^{n+3}$ satisfying Φ' is such that $\boldsymbol{c}_{1\ldots n} \in \{0,1\}^n$ must satisfy Φ, which is a factor of Φ'. Therefore, $\langle \Phi(x_1, x_2, \ldots x_n) \rangle \in$ 3-CNF-SAT.

□

Dati e Algoritmi 2 – CdL Magistrale in Ingegneria Informatica

Compito, 4/7/2011 (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 ingiustificate, approssimative, prolisse o in cattivo italiano saranno **fortemente penalizzate**. Ai fini del superamento dell'esame, è necessario conseguire una valutazione sufficiente alla prima parte, che richiede che almeno due esercizi su tre siano svolti in modo sufficiente.

T1 Si enunci la condizione di regolarità associata all Caso 3 del Master Theorem e si determini una costante $c < 1$ che soddisfi la suddetta condizione per la ricorrenza $T(n) = T(n/2) + n^3$.

Solution: The regularity condition is a property that $w(n)$ must exhibit so that Case 3 of the Master Theorem holds. Let $T(n) = aT(n/b) + w(n)$ be the recurrence under consideration. The condition states that there exists a constant $c < 1$ such that, for every value of n, it is $a \cdot w(n/b) \leq c \cdot w(n)$. For the given recurrence, we have $a = 1$, $b = 2$, and $w(n) = n^3$, whence it must be $(n/2)^3 \leq c \cdot n^3$. Clearly, $c = 1/8$ satisfies the condition for all values of n.

T2 Si provi che

$$DFT_n^{-1}(\boldsymbol{a}) = (1/n)DFT_n^R(\boldsymbol{a}).$$

Solution: We have:

$$\begin{aligned}
\left[DFT_n^{-1}(\boldsymbol{a})\right]_i &= (1/n)\sum_{j=0}^{n-1} a_j \omega_n^{-ij} \\
&= (1/n)\sum_{j=0}^{n-1} a_j \omega_n^{-ij+nj} \\
&= (1/n)\sum_{j=0}^{n-1} a_j \omega_n^{(n-i)j} \\
&= (1/n)\sum_{j=0}^{n-1} a_j \omega_n^{[(n-i) \bmod n]j}
\end{aligned}$$

$$
\begin{aligned}
&= (1/n)[DFT_n(\boldsymbol{a})]_{(n-i) \bmod n} \\
&= (1/n)[DFT_n^R(\boldsymbol{a})]_i
\end{aligned}
$$

T3 Sa $G = (V, E)$ un grafo non orientato e sia $k \in \{1, 2, \ldots, |V|\}$. Si provi che

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

Solution:

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

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

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

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

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]** Sia n una potenza di quattro. Per un dato problema computazionale Π, si supponga di avere un algoritmo iterativo A_1 la cui complessità su istanze di taglia n sia $T_1(n) = n^4$, e un algoritmo ricorsivo A_2 la cui complessità sia regolata dalla seguente ricorrenza:

$$T(n) = \begin{cases} 1 & n = 1, \\ 4T(n/4) + 90n^3 & n > 1. \end{cases}$$

1. **[6 punti]** Si determini la complessità $T(n, n_0)$ 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 quattro.

2. **[5 punti]** Si determini il valore $\bar{n}_0$ che fornisce il miglior algoritmo ibrido per $n > \bar{n}_0$.

Answer: Point 1 The recurrence associated to the generic hybrid algorithm is:

$$T(n, n_0) = \begin{cases} n^4 & n \leq n_0, \\ 4T(n/4, n_0) + 90n^3 & n > n_0 \end{cases}$$

For $n \geq n_0$, the recursion tree associated to the above recurrence has $\log_4(n/n_0)$ levels. On level ℓ, $0 \leq \ell < \log_4(n/n_0) - 1$, there are 4^ℓ nodes, each referring to an instance of size $n/4^\ell$, hence contributing work $90(n/4^\ell)^3$, while on level $\log_4(n/n_0)$ there are $4^{\log_4(n/n_0)} = n/n_0$ nodes each contributing work ${n_0}^4$. The total work is then

$$\begin{aligned} T(n, n_0) &= \sum_{\ell=0}^{\log_4(n/n_0)-1} 4^\ell \cdot 90 \left(\frac{n}{4^\ell}\right)^3 + \left(\frac{n}{n_0}\right) {n_0}^4 \\ &= 90n^3 \left(1 - \frac{1}{16^{\log_4(n/n_0)}}\right) \frac{16}{15} + nn_0^3 \\ &= 96n^3 \left(1 - \left(\frac{n_0}{n}\right)^2\right) + nn_0^3 \\ &= 96n^3 - 96nn_0^2 + nn_0^3. \end{aligned}$$

Point 2 Let us now take the partial derivative of $T_(n, n_0)$ with respect to n_0:

$$\frac{\partial T(n, n_0)}{\partial n_0} = 3nn_0(n_0 - 64),$$

which is nonnegative when either $n_0 \leq 0$ or $n_0 \geq 64$. Therefore, the optimal choice $\bar{n}_0$ is 64, a power of four, which yields the best function $T(n, 64) = 96n^3 - 132072n$ (compare it against $T(n, 1) = 96n^3 - 95n$).

□

Esercizio 2 [11 punti] La seguente ricorrenza definisce i *numeri di Stirling del secondo tipo* $S_2(n, k)$ per $0 \leq k \leq n$:

$$S_2(n, k) = \begin{cases} 1 & n = k \\ 0 & n > 0 \text{ e } k = 0 \\ kS_2(n-1, k) + S_2(n-1, k-1) & 0 < k < n. \end{cases}$$

- **[7 punti]** Si fornisca lo pseudocodice di un algoritmo memoizzato per il calcolo di $S_2(n, k)$, avendo l'accortezza di non inizializzare elementi della tabella che non sono usati nel calcolo.
- **[4 punti]** Si analizzi la **complessità esatta** $T_{S_2}(n, k)$ dell'algoritmo proposto supponendo che ogni prodotto tra interi abbia costo unitario e tutte le altre operazioni abbiano costo nullo.

Answer:

Point 1 The memoized code consists of a pair of subroutines, INIT_S2(n, k) and REC_S2(n, k). Recall that INIT_S2(n, k) must return directly on base cases, otherwise it must initialize an $(n+1) \times (k+1)$ look-up table with base and default values and then invoke REC_S2(n, k), which computes recursively all nonbase values, storing them in the table to avoid repeated calls. Observe that -1 is a suitable default value, since $S_2(n, k)$ is nonnegative for all values of n and k, with $0 \leq k \leq n$. The code follows.

```
INIT_S2(n, k)
if n = k then return 1
if k = 0 then return 0
    { base cases}
for i ← 0 to k do
    S[i, i] ← 1
for i = 1 to n − k do
    S[i, 0] ← 0
for j ← 1 to k do
    for i ← j + 1 to n − k + j do
        {we do not initialize useless table entries}
        S[i, j] ← −1
return REC_S2(n, k)
```

```
REC_S2(i, j)
if (S[i, j] = −1) then
    S[i, j] ← jREC_S2(i − 1, j)+
              +REC_S2(i − 1, j − 1)
return S[i, j]
```

Point 2 First, observe that INIT_S2(n) does not perform any multiplication. Furthermore, the recursion tree associated to REC_S2($1, n$) has exactly one internal node of unit cost for each pair of indices (i, j), for which an entry was initialized with -1 in INIT_S2. As a consequence, for $0 \leq k \leq n$ we have:

$$T_{S_2}(n,k) = \sum_{j=1}^{k} \sum_{i=j+1}^{n-k+j} 1 = \sum_{j=1}^{k} (n-k) = k(n-k).$$

□

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

ALTERNATE SAT (ASAT):

ISTANZA: $\langle \Phi(x_1, x_2, \ldots x_n), \Psi(x_1, x_2, \ldots x_n) \rangle$, Φ, Ψ formule booleane.

DOMANDA: Esiste un assegnamento di verità che soddisfa una sola tra Φ e Ψ?

Dimostrare che ASAT è NP-hard.

Answer: We exhibit a very natural and simple reduction from FORMULA SAT (SAT). Given a SAT instance $\langle \Phi(x_1, x_2, \ldots, x_n) \rangle$, our reduction function f is:

$$f\left(\langle \Phi(x_1, x_2, \ldots, x_n) \rangle\right) =< \Phi(x_1, x_2, \ldots, x_n), \Psi(x_1, x_2, \ldots, x_n) = (x_1 \wedge (\neg x_1)) \rangle.$$

Observe that Ψ is a contradiction, that is, its value is 0 under any truth assignment. Function f is clearly polynomial (indeeed, linear)-time computable.

Assume that $\langle \Phi(x_1, x_2, \ldots, x_n) \rangle \in \text{SAT}$. Then, there exists a truth assignment $\boldsymbol{b} \in \{0,1\}^n$ such that $\Phi(\boldsymbol{b}) = 1$. Since $\Psi(\boldsymbol{b}) = 0$, we have that $\langle \Phi(x_1, x_2, \ldots, x_n), \Psi(x_1, x_2, \ldots, x_n) \rangle = f(\langle \Phi(x_1, x_2, \ldots, x_n) \rangle \in \text{ASAT}$.

Assume that $f\left(\langle \Phi(x_1, x_2, \ldots, x_n) \rangle\right) =< \Phi(x_1, x_2, \ldots, x_n), \Psi(x_1, x_2, \ldots, x_n) \rangle \in$ ASAT. Then there must exist a truth assignment $\boldsymbol{b} \in \{0,1\}^n$ which satisfies only one between Φ and Ψ. Since Ψ is a contradiction, $\boldsymbol{b}$ must satisfy Φ, hence $\langle \Phi(x_1, x_2, \ldots, x_n) \rangle \in \text{SAT}$.

□

Dati e Algoritmi 2 – CdL Magistrale in Ingegneria Informatica

Compito, 19/6/2012 (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 ingiustificate, approssimative, prolisse o in cattivo italiano saranno **fortemente penalizzate**. Ai fini del superamento dell'esame, è necessario conseguire una valutazione sufficiente alla prima parte, che richiede che almeno due esercizi su tre siano svolti in modo sufficiente.

T1 Data l'istanza (A, B), con A e B interi di $n > 3$ bit ciascuno, si descrivano le tre sottoistanze generate dalla fase di *Divide* dell'algoritmo di Karatsuba KMULT($A.B$) e si determini la taglia di ciascuna di esse, al caso peggiore.

Solution: Let $A_0 \equiv a_{\lfloor n/2 \rfloor - 1} \ldots a_0$, $A_1 \equiv a_{n-1} \ldots a_{\lfloor n/2 \rfloor}$, and define B_0 and B_1 similarly. Let also $U = (A_0 + A_1)$ and $W = (B_0 + B_1)$. The three subinstances are $(A_0, B_0), (A_1, B_1),$ and(U, W), whose respective sizes are, in the worst case, $\lfloor n/2 \rfloor$, $\lceil n/2 \rceil$, and $\lceil n/2 \rceil + 1$.

T2 Si enunci e si provi la relazione tra un polinomio di grado limitato da n (pari) e due sottopolinomi di grado limitato da $n/2$ su cui è basata la fase di *Conquer* dell'algoritmo Fast Fourier Trasform (FFT).

Solution: Let $A(x) \equiv (a_0, \ldots, a_{n-1})$ be a polynomial of degree-bound n, and let $A^{[0]}(x) = (a_0, \ldots, a_{2i}, \ldots, a_{n-2})$ and $A^{[1]}(x) = (a_1, \ldots, a_{2i+1}, \ldots, a_{n-1})$ be the two subpolynomials of degree-bound $n/2$ obtained by separating the coefficients of monomials of even and odd degree of $A(x)$, respectively. Then, the following identity holds for all values of $x \in \boldsymbol{C}$:

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

To prove the above identity, observe that $A(x) = \sum_{i=0}^{n-1} a_i x^i$, $A^{[0]}(x) = \sum_{j=0}^{n/2-1} a_{2j} x^j$, and $A^{[1]}(x) = \sum_{j=0}^{n/2-1} a_{2j+1} x^j$. Therefore

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

since the two summations of $n/2$ terms account for all and only terms of even and odd index of the last summation.

T3 Data una istanza $\langle \Phi(x_1, x_2, \ldots, x_n) = C_1 \wedge C_2 \wedge \cdots \wedge C_m \rangle$ di 3-CNF-SAT si descriva l'istanza $f(\langle \Phi \rangle) = \langle G_\Phi = (V_\Phi, E_\Phi), k_\Phi \rangle$ di CLIQUE alla base della riduzione vista in classe 3-CNF-SAT $<_P$ CLIQUE.

Solution: Let $C_i = y_i^1 \vee y_i^2 \vee y_i^3$, where each y_i^j is a literal over $\{x_1, \ldots, x_n\}$, for $1 \leq i \leq m$ and $1 \leq j \leq 3$. For each occurrence of a literal y_i^j, let us define a node v_i^j. We have that $V_\Phi = \{v_i^j : 1 \leq i \leq m, 1 \leq j \leq 3\}$, $E_\Phi = \{\{v_{i_1}^{j_1}, v_{i_2}^{j_2}\} : (i_1 \neq i_2) \wedge (y_{i_1}^{j_1} \neq \neg y_{i_2}^{j_2}), 1 \leq i_1, i_2 \leq m, 1 \leq j_1, j_2 \leq 3\}$, and $k_\Phi = m$.

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] Si consideri la seguente equazione di ricorrenza per valori arbitrari positivi del parametro intero n:

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

Si dimostri **rigorosamente** che $T(n) = O(n)$.

Answer: Using parametric induction, we determine a constant $c > 0$ such that $T(n) \leq cn$, for all positive integer values of n. For $1 \leq n \leq 3$, observe that $T(n) = 1$, whence $c \geq 1$ (the most stringent constraint coming from the case $n = 1$, which requires that $1 \leq c \cdot 1$.) For $n > 3$ we have:

$$\begin{aligned} T(n) &= T(\lfloor n/2 \rfloor) + T(\lfloor n/4 \rfloor) + 2n \\ &\leq c\lfloor n/2 \rfloor + c\lfloor n/4 \rfloor + 2n \\ &\leq c(3/4)n + 2n \stackrel{?}{\leq} cn \\ &\Leftrightarrow c \geq 8 \end{aligned}$$

Therefore, it suffices to choose $c = \max\{1, 8\} = 8$, whence $T(n) \leq cn$ for all positive integer values of n. □

Esercizio 2 [11 punti] Sia dato un insieme $X = \{x_1, x_2, \ldots x_n\}$ di n numeri reali distinti, con $0 < x_1 < x_2 < \ldots < x_n < 1$. Si vuole determinare un insieme di massima cardinalità $\mathcal{C}$ di coppie di elementi di X tale che nessun elemento compaia più di una volta in più di una coppia e che la somma dei due elementi in ogni coppia sia minore di 1. Ad esempio, per $X = \{0.05, 0.25, 0.45, 0.6, 0.95\}$, una soluzione ottima è data da $\mathcal{C} = \{(0.05, 0.6), (0.25, 0.45)\}$.

- **[6 punti]** Si fornisca lo pseudocodice di un algoritmo greedy PAIRS(X) che, dato in ingresso l'insieme $X = \{x_1, x_2, \ldots x_n\}$ ordinato per valori crescenti, restituisce l'insieme $\mathcal{C}$ effettuando $O(n)$ confonti tra numeri reali.
- **[5 punti]** Si analizzi la correttezza di PAIRS(X) provando le proprietà di scelta greedy e di sottostruttura ottima.

Answer:

Point 1 In order to discard the least number of elements of X, our greedy choice pairs the first (smallest) element x_1 with the largest element x_ℓ, with $\ell > 1$, such that the sum $x_1 + x_\ell$ is less than one (if any exists) (observe that for $\ell < n$, the elements $x_{\ell+1}, \ldots, x_n$ cannot appear in any feasible solution.) Then, the residual subproblem (which may be empty) is $\{x_2, \ldots x_{\ell-1}\}$. The pseudocode follows.

```
PAIRS(X)
⋆X = {x_1, x_2, ..., x_n}⋆
C ← ∅
min ← 1
ℓ ← n
while (min < ℓ) do
    if (x_min + x_ℓ < 1)
        then
            C ← C ∪ {(x_min, x_ℓ)}
            min ← min + 1
    ℓ ← ℓ − 1
return C
```

Since variable ℓ is decremented at each iteration, the number of comparisons $T_P(n)$ performed by PAIRS on a set of n reals is at most n. Hence $T_P(n) \in O(n)$.

Point 2 First observe that if the optimal solution is empty, then the algorithm correctly returns $\mathcal{C} = \emptyset$, since clearly no pair of elements will pass the test ($x_{min} + x_\ell < 1$). Let us assume then that the optimal solution is nonempty, and let $g = \max\{\ell : x_1 + x_\ell < 1, 2 \leq \ell \leq n\}$ (g is clearly well defined when the optmal solution is nonempty). Our greedy choice is (x_1, x_g). Let us prove that there is an optimal solution $\mathcal{C}^\star$ which contains (x_1, x_g). Let $\hat{\mathcal{C}}$ be a generic optimal solution

that does not contain the greedy choice. We have to distinguish a number of cases. If $\hat{\mathcal{C}}$ contains a pair (x_1, x_h), with $h \neq g$, but no pair (x_k, x_g), we simply replace (x_1, x_h) with (x_1, x_g). The case where $\hat{\mathcal{C}}$ contains a pair (x_k, x_g), with $k \neq 1$, but no pair (x_1, x_h) is treated in a similar way. Finally, we are left with the case where $\hat{\mathcal{C}}$ contains pairs (x_1, x_h) and (x_k, x_g), with $h \neq g$, and $k \neq 1$. Then, we simply replace these two pairs with the pairs (x_1, x_g) and (x_h, x_k). Indeed, we have that $x_1 + x_g < 1$ by construction, and $x_h + x_k < x_k + x_g < 1$, since $h < g$ (by the definition of g, since $x_1 + x_h < 1$). In all cases, we obtain the desired optimal solution $\mathcal{C}^\star$ which contains the greedy choice.

Let now $\hat{\mathcal{C}} = \{(x_1, x_g)\} \cup \mathcal{C}'$ be an optimal solution containing the greedy choice. Then $\mathcal{C}'$ must be an optimal solution for the residual problem $\{x_2, \ldots, x_{g-1}\}$, since any larger solution $\mathcal{C}''$ would yield a feasible solution $\{(x_1, x_g)\} \cup \mathcal{C}''$ larger than $\hat{\mathcal{C}}$ for the original problem. □

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

DIFFERENT FORMULAE (DF):

ISTANZA: $\langle \Phi(x_1, x_2, \ldots, x_n), \Psi(x_1, x_2, \ldots, x_n) \rangle$, Φ, Ψ formule booleane

DOMANDA: Esiste un assegnamento di valori di verità $\boldsymbol{b} \in \{0,1\}^n$ per cui $\Phi(\boldsymbol{b}) \neq \Psi(\boldsymbol{b})$?

Dimostrare che DF è NP-hard.

Answer: We reduce from SAT. The simple intuition behind our reduction is that a formula Φ is satisfiable if and only if it differs from a formula which is a contradiction, that is, it is always false *on all truth assignments* $\boldsymbol{b} \in \{0,1\}^n$. Let $K(x_1, \ldots, x_n) = (x_1 \wedge (\neg x_1))$, and observe that K is clearly a contradiction, that is, $K(\boldsymbol{b}) = 0$ for all $\boldsymbol{b} \in \{0,1\}^n$. Let $\langle \Phi(x_1, x_2, \ldots, x_n) \rangle$ be an instance of SAT. Our reduction function is the following:

$$f\left(\langle \Phi(x_1, x_2, \ldots, x_n) \rangle\right) = \langle \Phi(x_1, x_2, \ldots, x_n), K(x_1, x_2, \ldots, x_n) \rangle.$$

Clearly, f is computable in polynomial (in fact, linear) time. We prove that f is indeed a reduction from SAT to DF using the following sequence of equivalences:

$$\begin{aligned}
x = \langle \Phi(x_1, x_2, \ldots, x_n) \rangle \in\ & \text{SAT} \\
\Leftrightarrow\ & \exists \boldsymbol{b} \in \{0,1\}^n : \Phi(\boldsymbol{b}) = 1 \\
\Leftrightarrow\ & \exists \boldsymbol{b} \in \{0,1\}^n : \Phi(\boldsymbol{b}) \neq 0 \\
\Leftrightarrow\ & \exists \boldsymbol{b} \in \{0,1\}^n : \Phi(\boldsymbol{b}) \neq K(\boldsymbol{b}) \\
\Leftrightarrow\ & f(x) = \langle \Phi(x_1, x_2, \ldots, x_n), K(x_1, x_2, \ldots, x_n) \rangle \in \text{DF}
\end{aligned}$$

□