Chapter 1

Formal Specification of Computational Problems

Problem

A *computational problem* Π is a *relation* between a set $\mathcal{I}$ (the set of *instances*) and a set $\mathcal{S}$ (the set of *solutions*). Algebraically, we have

$$\Pi \subseteq \mathcal{I} \times \mathcal{S}.$$

As an additional constraint, we require that for any instance $i \in \mathcal{I}$ there is *at least* one solution $s \in \mathcal{S}$ such that $(i, s) \in \Pi$. We say that s is a *solution to instance* i of problem Π.

Note: For a single $i \in \mathcal{I}$ there could be two *distinct* solutions $s_1, s_2 \in \mathcal{S}$ such that $(i, s_1) \in \Pi$ and $(i, s_2) \in \Pi$. In general, there can be several solutions to the same problem instance.

Examples

Integer Sum Let Z denote the set of all integers. Then

$$\begin{aligned} \mathcal{I} &= Z \times Z; \\ \mathcal{S} &= Z; \\ \Pi &\subseteq \mathcal{I} \times \mathcal{S} = (Z \times Z) \times Z \\ &= \{((a, b), c) : a, b, c \in Z \text{ and } c = a + b\}. \end{aligned}$$

Graph Reachability A *directed graph* is a pair $G = (V, E)$, with $V \subseteq Z^+$ and $E \subseteq V \times V$. V is the set of *nodes* of G, while E is the set of *edges*. A *path* in G is a sequence $\pi = \langle v_1, v_2, \ldots, v_k \rangle$, $k \geq 1$, with $v_i \in V$, $1 \leq i \leq k$ and $(v_j, v_{j+1}) \in E$, $1 \leq j < k$. Our problem can be defined as follows:

$$\begin{aligned}
\mathcal{I} &= \left\{ \langle G = (V, E), u, v \rangle : V \subseteq Z^+, E \subseteq V \times V \text{ and } u, v \in V \right\}; \\
\mathcal{S} &= \left\{ \langle v_1, v_2, \ldots, v_k \rangle : k \geq 1, v_i \in Z^+, 1 \leq i \leq k \right\} \cup \{\epsilon\}; \\
\Pi &= \{ (\langle G, u, v \rangle, \pi) : \pi = \langle v_1, v_2, \ldots, v_k \rangle \text{ is a path in } G \text{ with } v_1 = u \text{ and } v_k = v \} \\
&\quad \cup \{ (\langle G, u, v \rangle, \epsilon) : \text{ there is no path } \pi \text{ in } G \text{ from } u \text{ to } v \}.
\end{aligned}$$

Size of a problem instance

The *size* of an instance is a *reasonable* measure of "how large" the instance is. The concept of size can not be made completely formal and depends on the particular problem being studied. For example, for *Integer Sum*, we can use the number of bits of the binary representation of the two integers as the size of the instance; for *Graph Reachability*, the most natural measure for the size of an instance is $|V| + |E|$; for *Sorting* it is natural to take the number of items to be sorted as the size of the instance.

Algorithm

An *algorithm* is a well-defined, *deterministic* computational procedure that transforms a given *input* into a *unique output* through a finite sequence of *basic steps*. Therefore, an algorithm computes a *function* whose domain is the set of inputs and whose values are the ouputs. An algorithm can be specified once we agree on a *computational model*, that is, an abstraction of a computing device characterized by a rigorously defined set of basic steps. A popular model of computation is the *Random Access Machine* (RAM), an abstraction of a traditional, sequential computer and its instruction set.

Each basic step of the computational model can be associated with a *cost*. The *running time* of an algorithm on a particular input is the global cost of the basic steps executed by the algorithm on that input. To ease the analysis of the running time of an algorithm for a particular problem, it is customary to identify a subset of "crucial" basic steps, which are given unit cost, while the remaining basic steps are neglected by assigning them zero cost. Particular care must be exercised in selecting the "crucial" steps, in particular, selection must encompass all those steps whose number is roughly equal to the total number of steps

executed. When in doubt, it is advisable to assign unit cost to *all* steps. As an example, the running time of a sorting algorithm is often evaluated by assigning unit cost uniquely to comparison steps.

We say that an algorithm $\mathcal{A}$ *solves* a problem $\Pi \subseteq \mathcal{I} \times \mathcal{S}$ if and only if $\mathcal{A}$ computes a function $f_{\mathcal{A}}$ satisfying the following properties:

(a) $domain(f_{\mathcal{A}}) \supseteq \mathcal{I}$;

(b) $\forall i \in \mathcal{I} : (i, f_{\mathcal{A}}(i)) \in \Pi$.

Note: An algorithm associates a *single* solution to any problem instance, even when multiple solutions exist.

Example

Consider the following "toy" problem:

$$\begin{aligned} \mathcal{I} &= \{1, 2, 3\}; \\ \mathcal{S} &= \{a, b, c, d\}; \\ \Pi &= \{(1, a), (1, b), (2, c), (3, d)\}. \end{aligned}$$

The following is an algorithm for Π.

$\mathcal{A}_\Pi(x)$
  if $x = 1$ **then return** a
  if $x = 2$ **then return** c
  if $x = 3$ **then return** d
  else return f

Algorithm $\mathcal{A}_\Pi$ satisfies both Properties (a) and (b), therefore $\mathcal{A}_\Pi$ solves Π. Note that $\mathcal{A}_\Pi$ does something more, since it returns f for any value of x different from 1, 2, or 3. Therefore $\mathcal{A}_\Pi$ also solves $\Pi' = \mathcal{I}' \times \mathcal{S}'$, with $\mathcal{I}' = \{i : i \geq 4\}$ and $\mathcal{S}' = \{f\}$. This shows that a single algorithm may solve more than one problem. In contrast, there may exist many algorithms solving the same problem.

Exercise 1.1 We say that two algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ are *functionally distinct* if the functions $f_{\mathcal{A}_1}$ and $f_{\mathcal{A}_2}$, respectively computed by the two algorithms, differ on at least one input $x \in \mathcal{I}$.

(a) How many functionally distinct algorithms may exist for the *Integer Sum* problem seen in class?

(b) How many for the *Graph Reachability* problem seen in class?

Please, justify your answers.

Answer:

(a) No two functionally distinct algorithms may exist for *Integer Sum*, since there is a unique solution for any instance.

(b) For *Graph Reachability*, there are infinitely many mutually distinct algorithms, since there are infinitely many instances that admit more than one solution.

□

Exercise 1.2 Let U be a finite set. Given two arbitrary subsets of U, $A, B \subseteq U$, consider the problem of returning an element $u \in A \cap B$, if $A \cap B \neq \emptyset$, or returning ϵ if $A \cap B = \emptyset$. Cast this as a *computational problem* by specifying

(a) the set of instances $\mathcal{I}$;

(b) the set of solutions $\mathcal{S}$;

(c) the appropriate relation Π.

Answer:

(a) Let $\mathcal{F}(U)$ be the family of all subsets of U. Then $\mathcal{I} = \mathcal{F}(U) \times \mathcal{F}(U)$.

(b) $\mathcal{S} = U \cup \{\epsilon\}$ (note that we are assuming that $\epsilon \notin U$.)

(c) Given $(A, B) \in \mathcal{I}$ and $y \in \mathcal{S}$ we have:

$$(A, B)\ \Pi\ y \iff (y \in A \cap B) \text{ or } \left[(A \cap B = \emptyset) \text{ and } (y = \epsilon)\right].$$

□

Exercise 1.3 Intuitively, the merging problem consists in combining two sorted sequences $(x_1, x_2, \ldots, x_m)$ and $(x_{m+1}, x_{m+2}, \ldots, x_n)$ into one sorted sequence $(y_1, y_2, \ldots, y_n)$.

(a) Formally specify the sets $\mathcal{I}$ and $\mathcal{S}$ and the relation $\Pi \subseteq \mathcal{I} \times \mathcal{S}$ for the merging problem.

(b) Give a reasonable measure for the size of an instance $i \in \mathcal{I}$.

Answer:

(a) Let U be a totally ordered universe set, and let SS be the set of *Sorted Sequences* of elements of U, i.e.

$$SS = \{(a_1, a_2, \ldots, a_k) \in U^k : a_1 \leq a_2 \leq \ldots \leq a_k, k \in Z^+\}.$$

Then, $\mathcal{I} = SS \times SS$ and $\mathcal{S} = SS$. Problem $\Pi \in \mathcal{I} \times \mathcal{S}$ is specified as:

$$(((x_1, x_2, \ldots, x_m), (x_{m+1}, x_{m+2}, \ldots, x_n)), (y_1, y_2, \ldots, y_n)) \in \Pi$$

iff the two *multisets* (i.e., sets with possibly repeated elements)

$$\{x_1, x_2, \ldots, x_m, x_{m+1}, x_{m+2}, \ldots, x_n\} \text{ and } \{y_1, y_2, \ldots, y_n\}$$

are equal.

(b) Given $((x_1, x_2, \ldots, x_m), (x_{m+1}, x_{m+2}, \ldots, x_n)) \in \mathcal{I}$ as input, n is the most natural measure of the input size.

□

Exercise 1.4 Give a formal characterization of the problem of sorting an arbitrary sequence of integers.

Exercise 1.5 Give a formal characterization of the following problem. Given a sequence of integers $(x_1, x_2, \ldots, x_n)$, determine whether there exist indices i, j, with $1 \leq i \neq j \leq n$, such that $x_i = x_j$.