Algorithm

Chapter 1

Foundations

1.1 Computational Problems and Algorithms

Definition 1.1. A **computational problem** is a relation

$$P \subseteq X \times Y,$$

where X is called the set of **instances** and Y is called the sets of **solutions**.

Example. Let X be the set of nonempty sequence of distinct integers and let Y be the set of positive integers. Then we can define $P \subseteq X \times Y$ such that

$$((a_1, a_2, \ldots, a_n), i) \in P$$

if and only if $1 \leq i \leq n$ and $a_i \geq a_j$ for all $j \in \{1, \ldots, n\}$. We can write this problem in the following form.

Problem 1.A (Champion Problem).

- Input: A sequence of n distinct integers $A[1], A[2], \ldots, A[n]$.
- Output: The index $i \in \{1, \ldots, n\}$ such that $A[i] \geq A[j]$ for all $j \in \{1, \ldots, n\}$.

Definition 1.2. We will assume the **random-access machine (RAM)** model of computation as our implementaion technology for most of this note. In this model, we have an infinite sequence of w-bit words, and we assume $w = \lceil c \lg n \rceil$ for some constant $c \geq 1$, where n is the input size. We can perform some basic operations on these words, including

- arithmetic operations (e.g., addition, subtraction, multiplication, division),
- data movement operations (e.g., load, store, copy), and
- control operations (e.g., branch, subroutine call, return).

Definition 1.3. Given a computational model, an **algorithm** is defined as a finite sequence of basic operations that transforms a given input into a unique output.

- We say that an algorithm **solves** a computational problem $P \subseteq X \times Y$ if it transforms every instance $x \in X$ into a solution $y \in Y$ such that $(x, y) \in P$.

- The **running time** of an algorithm on a specific input is defined as the number of basic operations performed.

Example. We can find the index of maximum of a sequence by the following algorithm Max-Index.

Max-Index(A)
1 $n \leftarrow A.size$
2 $i \leftarrow 1$
3 **for** $j \leftarrow 2$ **to** n
4   **if** $A[i] < A[j]$
5         $i \leftarrow j$
6 **return** i

Note that we will describe algorithm is pseudocode so that implementaion details can be hidden.

Theorem 1.4. The algorithm Max-Index solves the champion problem using the least number of comparisons.

Proof. We show that Max-Index solves the champion problem as follows. We focus on the loop invariant that at the start of each iteration of the **for** loop of lines 3 – 5, $A[i]$ is the maximum of the subarray $A[1\,..\,j-1]$. The loop invariant is obviously true when entering the loop, and it remains true between iterations since $A[i] \geq A[j]$ must hold at the end of each iteration due to lines 4 and 5. Thus, when the **for** loop terminates, $A[i]$ should be the maximum of $A[1\,..\,n]$.

Furthremore, the algorithm Max-Index uses $n-1$ comparisons, and it is optimal with respect to the number of comparisons performed, since at least $n-1$ comparisons are necessary to determine the maximum. $\square$