CIS 575. Introduction to Algorithm Analysis
Material for January 17, 2024

Sorting and Selection: Specifications

1 Sorting

As argued in the *Cormen* textbook (p.158), sorting may be considered *the most fundamental problem* in the study of algorithms, for several reasons that include:

- real-world applications almost always make use of sorting, to present the results and/or internally to facilitate processing;
- there exists a variety of sorting algorithms (many of which we shall see in this course) which illustrate a number of design principles
- some of these algorithms are "as good as we can get" in that the time they take to sort n elements is proportional to n times the logarithm of n which one can prove is a lower bound for (comparison-based) sorting (for many other problems, there is a gap between the best known algorithm and the best known theoretical lower bound).

2 Specification of Sorting

We shall assume that input comes as an array $A[1..n]$ where each $A[i]$ is a record with a key (in examples, we shall just show the key and ignore the other record elements). Keys will typically be integers, but could come from any set with a total order (that is, given two different keys, exactly one of them will be less than the other). We shall write $A[i] < A[j]$ ($A[i] \leq A[j]$) if the key of $A[i]$ is less than (less than or equal to) the key of $A[j]$.

Assuming that our goal is to put the records with the largest keys at the end, we must require that after sorting, A are *non-decreasing*. (The case where our goal is to put the records with the largest keys at the beginning, and accordingly must require A to be non-increasing, is fully symmetric and thus does not merit separate treatment; everything developed in the following can still be applied, *mutatis mutandis*.) One may be tempted to even demand that A is *strictly* increasing, but that would not allow A to contain two records with the same key.

Since all natural languages can be quite ambiguous, and English probably more than most, there is a great risk that the parties may disagree on the meaning of a contract written in English. Therefore, it may be advantageous to express key parts of the contract in (predicate) logic which is unambigious. For the property of being non-descreasing, this can be done in several ways:

$$\forall i \forall j \ (1 \leq i < j \leq n \Rightarrow A[i] \leq A[j])$$

is perhaps the most natural but we may allow i to equal j:

$$\forall i \forall j \ (1 \leq i \leq j \leq n \Rightarrow A[i] \leq A[j])$$

and we may even do with only one quantified variable (then being careful abound the bound):

$$\forall i \ (1 \leq i < n \Rightarrow A[i] \leq A[i+1])$$

Are we done? No, since still a contractor may circumvent our intentions, and given say $A = [5, 8, 3, 7]$ return $A = [1, 2, 3, 4]$ which surely is non-decreasing but not at all what we had in mind! We must thus add the demand that A is a permutation of the original records which we shall denote A_0 (using the common convention of a subscript zero to denote the original value). Then one way to express our extra demand is: there exists a bijection f on $\{1 \ldots n\}$ such that

$$\forall i \in 1..n : A[i] = A_0[f(i)]$$

3 The Selection Problem

While not as important as the sorting problem, the selection problem is still quite relevant in many contexts. The task is:

> Given an array $A[1..n]$, and $k \in 1 \ldots n$, find the $k'th$ smallest element of A.

For example, if $A = [7, 5, 7, 9]$, then clearly 5 is the (1st) smallest element, and 7 is the 2nd smallest element. But which, 7 or 9, is the 3rd smallest element? The answer is that we want it to be 7 (and thus 9 to be the 4th smallest element). To justify that 7 is both the 2nd and 3rd smallest element, we observe that less than 2 array elements are < 7, but at least 3 array elements are ≤ 7.

In general, we claim that x is the k'th smallest element of A if and only if

$$\begin{array}{rcl} |\{i \mid A[i] < x\}| & < & k \\ |\{i \mid A[i] \leq x\}| & \geq & k \end{array}$$