Chapter 6

Linear Time Sorts and Lower Bounds in the Comparison Model.

6.1 Heapsort (David's group).

We will start by introducing the heap-sort algorithm and providing a proof of its correctness.

1 $A \leftarrow$ Build-Heap(A)
2 **for** $i \in [n]$ **do**
3   swap $A_1 \leftrightarrow A_{n-i+1}$
4   heapsize(A) $\leftarrow n - i$
5   heapify(A, 1)
6 **end**
7 return A

Algorithm 1: Heap-sort(A)

Correctness. We are going to prove the following statement.

Claim 6.1.1. *At the end of the ith iteration, $A_{n-i+1}, A_{n-i+2}, ..A_n$ are the i largest elements of A placed in order, and $A_1, A_2, ..A_{n-i}$ is a maximum heap.*

Proof. By induction.

1. Base. A_n is set in line (3) to be the root of the heap, and therefore is the maximum of A. After line (4), A_1 is the parent of the heap's roots as line (4) excludes A_n from the heap (So the heap inequality holds for any $j \in [2, (n-1)/2]$). Therefore, by the correctness of heapify, we get that after line (5), $A_1, A_2, ..A_{n-1}$ is a heap.

2. Assumption. Assume the correctness of the claim for any $i' < i$.

3. Step. Consider the ith iteration. By the induction assumption, A_1 is a root of the heap $A_1, A_2, ..A_{n-i+1}$ and therefore is their maximum. So after the swapping in line (3), we get that A_{n-i+1} is the element which is greater than

$n - i$ elements in A. By using the induction assumption again, we know that it is also less than $A_{n-i+2}, A_{n-i+3}, ..A_n$, so after line (3) and by the fact that $A_{n-i+2}, A_{n-i+3}, ..A_n$ are the $i-1$ largest elements placed in order, we have that $A_{n-i+1}, A_{n-i+2}, A_{n-i+3}, ..A_n$ are the i largest elements placed in order.

By repeating the same arguments in the base case, we can conclude, based on the correctness of heapify, that after line (5), A_1 is either the root of a heap or the heap inequality did not hold for some $i \in [2, (n-i)/2]$. In the latter case, this would contradict the induction assumption (since before line (3), $A_1..A_{n-i+1}$ were heaps).

$\square$