Exercises for Lecture 3

March 7, 2023

Exercise 1 (Cormen 6.4-2). Argue the correctness of Heapsort using the following loop invariant:

At the start of each iteration of the for loop of lines 2–5 of the pseudocode, the subarray $A[1, \ldots, i]$ is a max-heap containing the i smallest elements of $A[1, \ldots, n]$, and the subarray $A[i+1, \ldots, n]$ contains the $n-i$ largest elements of $A[1, \ldots, n]$, sorted.

Exercise 2 (Part of Cormen's Problem 6-2). A d-ary heap is like a binary heap, but (with one possible exception) non-leaf nodes have d children instead of 2 children.

1. How would you represent a d-ary heap in an array? In other words: describe at which index of the array you can find the parent of the node representing the element at index i, and at which index you can find its k-th child.

2. What is the height of a d-ary heap of n elements in terms of n and d?

Exercise 3 (Cormen 7.2-2). What is the running time of Quicksort when all elements of array A have the same value?

Exercise 4 (Cormen 7.4-5). We can improve the running time of quicksort in practice by taking advantage of the fast running time of insertion sort when its input is *nearly* sorted. Upon calling quicksort on a subarray with fewer than k elements, let it simply return without sorting the subarray. After the top-level call to quicksort returns, run insertion sort on the entire array to finish the sorting process. Argue that this sorting algorithm runs in $O(nk + n\log(n/k))$ *expected* time. How should we pick k, both in theory and in practice?