CS 624: Analysis of Algorithms

Assignment 2

Due: February 28, 2026

Topics: Heaps, Sorting

1. Argue the correctness of HEAPSORT using the following loop invariant:

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

2. At which levels in a max-heap might the k^{th} largest element reside, for $2 \leq k \leq \lfloor n/2 \rfloor$, assuming all elements are distinct?

3. The procedure `Max-Heap-Delete(A,i)` deletes the item in node i from heap A. Give an implementation of `Max-Heap-Delete` that runs in $O(\log n)$ for an n-element Max-Heap. Assume that the heap elements are mapped into indices, so you have access to the i^{th} node. Note that the problem asks you to give an algorithm that runs in $O(\log n)$ time. So you not only have to give the algorithm, you also have to show that it really does run in $O(\log n)$ time.

4. Suppose you start with a rectangular array of numbers. Perform the following operations:

   - First sort each row (smallest to largest).
   - Then sort each column (smallest to largest).

   Show that after sorting the columns, each row is still sorted. (Hint: Prove by contradiction)

5. Exercise 3.2 in the Lecture 3 handout (on page 7 of the handout). Don't be sloppy here! I'm looking for a precise explanation.

6. Exercise 6.1 in the Lecture 3 handout (on page 13 of the handout).

7. Exercise 3.1 from the Lecture 4 handout (page 7).

8. Based on exercises 8.2-1 - 8.2-3 in the 4th edition (slightly edited).

   (a) Show the run of counting-Sort on the array $A = [6, 0, 2, 0, 1, 3, 4, 6, 1, 3, 2]$. No need to draw out every stage. Just show the array C after line 5, C after line 8 and the first three stages of the final sorting (similar to figure 8.2)

   (b) Prove that Counting-Sort is stable. In other words, equal elements appear in the sorted array in the same order as they were in the original input – if we have two equal elements $A[i]$ and $A[j]$ such that $i < j$, $A[i]$ will appear before $A[j]$ in the sorted array.

(c) Show that if we rewrite the for loop header in line 11 of the Counting-Sort pseudo code as `for i=1 to n` (going from the start to the end), the algorithm still works properly, but it is not stable. Then rewrite the pseudocode for counting sort so that elements with the same value are written into the output array in order of increasing index and the algorithm is stable.

9. Problem 8-5 (a-d only) (page 207 in 3rd edition, 221 in 4th edition). Here it is: Suppose that, instead of sorting an array, we just require that the elements increase on average. More precisely, we call an n-element array A k-sorted if, for all $i = 1, 2, \ldots, n-k$, the following holds:

$$\frac{\sum_{j=i}^{i+k-1} A[j]}{k} \leq \frac{\sum_{j=i+1}^{i+k} A[j]}{k}$$

(a) What does it mean for an array to be 1-sorted?

(b) Give a permutation of the numbers 1,2,..,10 that is 2-sorted, but not sorted.

(c) Prove that an n-element array is k-sorted if and only if $A[i] \leq A[i+k]$ for all $i = 1, 2, \ldots, n-k$

(d) Give an algorithm that k-sorts an n-element array in $O(n \log(n/k))$time.