Homework 0

Note: E0 225 is a *second* course in Design and Analysis of Algorithms. The following homework is for students to assess their preparedness for taking E0 225. This homework will *not* be graded. If you are not able to solve majority of the problems in this homework (say, at least eight problems), then you are likely to find the course very challenging.

1. You are given two sorted arrays of size m and n. Give an $O(\log m + \log n)$ time algorithm for computing the kth smallest element in the union of the two arrays.

2. Define a *sequence* $[i, j]$ as the list of consecutive integers $i, i+1, \ldots, j-1, j$; and the value of a sequence $[i, j]$ is defined as $\sum_{k=i}^{j} k$. Given an integer n, in $O(n)$ time report *all* the sequences whose value is n. For example, if $n = 9$, then output $[4, 5]$ and $[2, 4]$. (*Note: Pay special attention in proving the correctness of your algorithm. Specifically, why does the "window" of interest only move forward?)*

3. Prove that $\sum_{j=0}^{h} j2^{h-j} = O(n)$, where $h = \log n$. This shows up during the analysis of the linear-time algorithm for constructing a heap on n elements.

4. Solve the following recurrence relations and give a big-Oh bound for each of them. Avoid using Master theorem. If possible, first try to make an intelligent "guess" of the answer, and then proceed to solve it.

- $T(n) = T(\sqrt{n}) + 1$.
- $T(n) = 2T(n/2) + O\left(\frac{n}{\log n}\right)$.

5. The following is a *randomized* algorithm for finding the median of a set S consisting of n elements: Pick an element uniformly at random from S. Call this the *pivot* element, say v. Now split the elements of S into three categories: elements smaller than v, equal to v (there can be duplicates), and elements greater than v. Call them S_L, S_v, and S_R, respectively. The claim is that, to find the median, it is enough to *recurse* on exactly one of the three sets defined above. Do you see why? If so, which element will you be searching for in the recursive call? Informally, can you show why this algorithm will take $O(n)$ time on "average"?

6 (Dasgupta et al.) Show that any array of integers $x_1, x_2, \ldots, x_n$ can be sorted in $O(n + M)$ time, where

$$M = \max_i x_i - \min_i x_i.$$

For small M, this is linear time: why doesn't the $\Omega(n \log n)$ lower bound apply in this case?

7. *Prove* that if an undirected graph with n vertices has k connected components, then it has at least $n-k$ edges.

8 (Dasgupta et al.) Consider an undirected graph $G = (V, E)$ with nonnegative edge weights $w_e \geq 0$. Suppose that you have computed a minimum spanning tree of G, and that you have also computed shortest paths to all nodes from a particular node $s \in V$. Now suppose each edge weight is increased by 1: the new weights are $w'_e = w_e + 1$.

1. Does the minimum spanning tree change? Give an example where it changes or prove it cannot change.

2. Do the shortest paths change? Give an example where it changes or prove it cannot change.

9. Give an $O(nt)$ time dynamic programming algorithm for the following problem. You are given a list of n positive integers $a_1, a_2, \ldots, a_n$, and a target positive integer t. Does their exist some subset of the a_i's which add up to t? Of course, you can use each a_i at most once.

10. Suppose you are told that the largest clique in a graph can be computed in polynomial time. If so, can you use this fact to compute the largest independent set in a graph also in polynomial time? If yes, then prove it formally; otherwise, show an example where it will fail.