Due in class: Feb 2.

In general, Homework problems are to be done by yourself (unless specified) with no help from another person. If you use any sources other than the textbooks for the class, you are to cite them. In any case, the writeup should be your own and not simply taken from another source.

1. You are given $n$ nuts and $n$ matching bolts in a jumbled heap. In each operation you are allowed to take one nut and one bolt and to check if they fit. You can conclude that either you have found the matching nut for the bolt, or the nut is too big or too small. (You cannot compare two nuts or two bolts directly.) Design a “good” algorithm to find the matching pairs of nuts and bolts. How many operations does your algorithm perform?

2. You are given $n$ numbers in arbitrary order and an integer $k$. Design an efficient algorithm that finds the $k$ largest numbers. The running time of your algorithm should be MUCH better than $O(n \log n)$ (i.e. you cannot just sort all the numbers and choose the top $k$).

3. In a directed graph, a get-stuck vertex is one that has in-degree $|V| - 1$ and out-degree 0. Assume that the adjacency matrix representation is used. Design an $O(|V|)$ algorithm to determine if a given graph has a sink. (Yes, this problem can be solved without even looking at the entire input matrix.) Write a proof of correctness for your algorithm.

4. Assume that we have a network (a connected undirected graph) in which each edge $e_i$ has an associated bandwidth $b_i$. If we have a path $P$, from $s$ to $v$, then the capacity of the path is defined to be the minimum bandwidth of all the edges that belong to the path $P$. We define $capacity(s, v) = \max_{P(s, v)} capacity(P)$. (Essentially, $capacity(s, v)$ is equal to the maximum capacity path from $s$ to $v$.) Give an efficient algorithm to compute $capacity(s, v)$, for each vertex $v$; where $s$ is some fixed source vertex. Show that your algorithm is “correct”, and analyze its running time.

(Design something that is no more than $O(|V|^2)$, and with the right data structures takes $O(|E| \log |V|)$ time.)

5. You are given a directed acyclic graph $G = (V, E)$, and a subset of vertices, $S \subseteq V$. Design an efficient algorithm that computes the number of simple paths in $G$ that visit all the vertices in $S$ and start and end at a vertex in $S$. (Hint: there is an algorithm that runs in $O(|S| \log |S| + |V| + |E|$)).