CS 180, Fall 2015 Homework 2

The following homework is due on Wednesday, Oct 14th at the beginning of lecture.

When submitting your homework, please include your name at the top of each page. If you submit multiple pages, please staple them together. Late submissions are not accepted.

1. For a directed graph, the *underlying undirected graph* is a graph that is obtained by ignoring the direction of the edges or replacing each directed edge by an undirected edge. In class we proved the following theorem: In any directed graph $G = (V, E)$, there is an independent set S of the underlying undirected graph of G such that in the original directed graph G any node of V can be reached by a node from S via a path of length ≤ 2 (Note that nodes of S can be reached by a path of length zero). We defined a *tournament graph* to be a directed graph that its underlying undirected graph is a complete graph (i.e. ignoring directions there is an edge between every pair of nodes).

   (a) First use the above theorem to prove that there is a node in the tournament graph that can reach any other node by a path of length ≤ 2. We call this node the *king*.

   (b) Describe an algorithm that given any tournament graph as input finds the king of the the graph.

   (c) Analyze the running time of your algorithm.

2. Let $K_n = (V, E)$ denote the complete undirected graph with n vertices (namely, every two vertices are connected), and let n be an even number.

   (a) Prove that the set of edges E of K_n can be partitioned into sets $E_1, \ldots, E_{n/2}$ such that each graph $G_i = (V, E_i)$ is a spanning tree of K_n. (A spanning tree is a connected subgraph that contains all vertices and no cycles).

   (b) Give a recursive algorithm that given n as input outputs all $n/2$ spanning trees. (Each spanning tree can be represented by a set of $n - 1$ edges, so your algorithm should output the parts E_i.)

3. Consider the problem of finding a *local minimum* in an array. The value $A[i]$ is a local minimum if and only if $A[i] \leq A[i-1]$ and $A[i] \leq A[i+1]$. Give an efficient $O(\log n)$-time algorithm to find a local minimum in an array A in which $A[1] = A[n] = +\infty$. State the runtime and explain its correctness.

   For example, in the following array every highlighted element is a local minimum.

$+\infty$	10	23	17	20	1	2	3	4	-3	5	$+\infty$

   Hint: You may first try to solve the following problem. You are given an array that only consists of 0's and 1's. We know that $A[1] = 0 \leq A[2] \leq A[3] \leq \ldots \leq A[n-1] \leq A[n] = 1$. Find an efficient algorithm to find i such that $A[i] = 0$ and $A[i+1] = 1$. Use the idea of this algorithm to solve the main problem. You don't need to include the solution to hint problem in your final submission.

4. Design a $O(n \log n)$ algorithm for sorting using the idea of heap data structure that we have seen in class. The input to your algorithm is an array of numbers and the output should be the sorted array. If your algorithm needs to use any heap functions, you also need to provide the algorithm for those functions.

   We will soon study *greedy algorithms* in class. These are algorithms that build up a solution in small steps, choosing a decision at each step to optimize some underlying criterion. In other words, a greedy algorithm generates the partial solution incrementally and commits to the partial solution. We will see that *merge sort* is not a greedy algorithm because in merge sort until the last phase, elements might not be in their final place. However, *bubble sort* is a greedy algorithm, it builds the final sorted array element by element and it is $O(n^2)$ algorithm. Given this introduction to greedy algorithms, do you think your sorting algorithm that uses heap data structure is a greedy algorithm?