CSCI 570 - HW 1

Due Sunday Feb. 02 (by 23:30)

1. Arrange the following functions in increasing order of growth rate with $g(n)$ following $f(n)$ in your list if and only if $f(n) = O(g(n))$

$$2^{\log n}, \quad (\sqrt{2})^{\log n}, \quad n\,(\log n)^3, \quad 2^{\sqrt{2\log n}}, \quad 2^{2^n}, \quad n\log n, \quad 2^{n^2}$$

2. Give a linear time algorithm based on BFS to detect whether a given undirected graph contains a cycle. If the graph contains a cycle, then your algorithm should output one. It should not output all cycles in the graph, just one of them. You are NOT allowed to modify BFS, but rather use it as a black box. Explain why your algorithm has a linear time runtime complexity.

3. Let $G = (V, E)$ be a connected undirected graph and let v be a vertex in G. Let T be the depth-first search tree of G starting from v, and let U be the breadth-first search tree of G starting from v. Prove that the height of T is at least as great as the height of U.

4. A binary tree is a rooted tree in which each node has at most two children. Show by *induction* that in any nonempty binary tree the number of nodes with two children is exactly one less than the number of leaves.

5. Prove that a complete graph K_5 is not a planar graph. You can use facts regarding planar graphs proven in the textbook.

6. Suppose we perform a sequence of n operations on a data structure in which the i^{th} operation costs i if i is an exact power of 2, and 1 otherwise. Use aggregate analysis to determine the amortized cost per operation.

7. We have argued in the lecture that if the table size is doubled when it's full, then the amortized cost per insert is acceptable. Fred Hacker claims that this consumes too much space. He wants to try to increase the size with every insert by just two over the previous size. What is the amortized cost per insertion in Fred's table?

8. You are given a weighted graph G, two designated vertices *s* and *t*. Your goal is to find a path from *s* to *t* in which the minimum edge weight is maximized i.e. if there are two paths with weights 10→1→5 and 2→7→3 then the second path is considered better since the minimum weight (2) is greater than the minimum weight of the first (1). Describe an efficient algorithm to solve this problem and show its complexity.

9. When we have two sorted lists of numbers in non-descending order, and we need to merge them into one sorted list, we can simply compare the first two elements of the lists, extract the smaller one and attach it to the end of the new list, and repeat until one of the two original lists become empty, then we attach the remaining numbers to the end of the new list and it's done. This takes linear time. Now, try to give an algorithm using O(*n* log *k*) time to merge *k* sorted lists (you can also assume that they contain numbers in non-descending order) into one sorted list, where *n* is the total number of elements in **all** the input lists. Use a binary heap for *k*-way merging.