Final

Note

This is a closed-book exam. Each problem accounts for 10 points, unless otherwise marked.

Problems

1. For each of the following pairs of functions, determine whether $f(n) = O(g(n))$ and/or $f(n) = \Omega(g(n))$. Justify your answers.

	$f(n)$	$g(n)$
(a)	$(\log n)^{\log n}$	$\frac{n}{\log n}$
(b)	$n^3 2^n$	3^n

   Note: the suggested solution to this midterm exam problem is not quite complete. Please try to do better.

2. Consider rearranging the following array into a max heap using the *bottom-up* approach.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	8	3	5	1	14	6	7	11	4	10	12	13	15	9

   Please show the result (i.e., the contents of the array) after a new element is added to the current collection of heaps (at the bottom) until the entire array has become a heap.

3. Prove that the sum of the heights of all nodes in a complete binary tree with n nodes is at most $n - 1$. You may assume it is known that the sum of the heights of all nodes in a *full* binary tree of height h is $2^{h+1} - h - 2$. (Note: a single-node tree has height 0.)

4. Compute the *next* table as in the KMP algorithm for string $B[1..11] = abaababababaa$. Please show how $next[7]$ and $next[11]$ are computed from using preceding entries in the table.

5. Given as input a connected undirected graph G, a spanning tree T of G, and a vertex v, design an algorithm to determine whether T is a valid DFS tree of G rooted at v. In other words, determine whether T can be the output of DFS under some order of the edges starting with v. Please present your algorithm in an adequate pseudo code and make assumptions wherever necessary. Explain why the algorithm is correct and give an analysis of its time complexity. The more efficient your algorithm is, the more points you get for this problem.

6. What is wrong with the following algorithm for computing the minimum-cost spanning tree of a given weighted undirected graph (assumed to be connected)?

   > If the input is just a single-node graph, return the single node. Otherwise, divide the graph into two subgraphs, recursively compute their minimum-cost spanning trees, and then connect the two spanning trees with an edge between the two subgraphs that has the minimum weight.

7. Let $G = (V, E)$ be a connected weighted undirected graph and T be a minimum-cost spanning tree (MCST) of G. Suppose that the cost of one edge $\{u, v\}$ in G is *increased*; $\{u, v\}$ may or may not belong to T. Design an algorithm either to find a new MCST or to determine that T is still an MCST. The more efficient your algorithm is, the more points you will be credited for this problem. Explain why your algorithm is correct and analyze its time complexity.

8. Let $G = (V, E)$ be a directed graph, and let T be a DFS tree of G. Prove that the intersection of the edges of T with the edges of any strongly connected component of G form a subtree (rather than two or more separate subtrees) of T.

9. Below is a solution to the single-source shortest path problem using the dynamic programming approach, which we have discussed in class:

   Denote by $D^l(u)$ the length of a shortest path from v (the source) to u containing *at most* l edges; particularly, $D^{n-1}(u)$ is the length of a shortest path from v to u (with no restrictions).

$$D^1(u) = \begin{cases} \mathit{length}(v, u) & \text{if } (v, u) \in E \\ 0 & \text{if } u = v \\ \infty & \text{otherwise} \end{cases}$$

$$D^l(u) = \min\{D^{l-1}(u), \min_{(u',u)\in E}\{D^{l-1}(u') + \mathit{length}(u', u)\}\},$$
$$2 \le l \le n - 1$$

Please explain why the solution allows edges with a negative weight (as long as there is no cycle with a negative weight). How is this different from Dijkstra's algorithm? Please explain.

10. The subgraph isomorphism problem is as follows.

    > Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, does G_1 have a subgraph that is isomorphic to G_2? (Two graphs are isomorphic if there exists a one-one correspondence between the two sets of vertices of the two graphs that preserves adjacency, i.e., if there is an edge between two vertices of the first graph, then there is also an edge between the two corresponding vertices in the second graph, and vice versa.)

    Prove that the subgraph isomorphism problem is NP-complete. (Hint: the proof of NP-hardness is by reduction from the Hamiltonian cycle problem to this problem. Given the input graph $G = (V, E)$ of the Hamiltonian cycle problem, we set $G_1 = (V_1, E_1)$ to be precisely the same as G and $G_2 = (V_2, E_2)$ to be a ring of $|V_1|$ nodes for the input to the subgraph isomorphism problem.)

Appendix

- The Hamiltonian cycle problem: given an undirected graph G, does G have a Hamiltonian cycle? (A Hamiltonian cycle in a graph is a cycle that contains each vertex, except the starting vertex of the cycle, exactly once.)

  The Hamiltonian cycle problem is NP-complete.