Final

Note

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

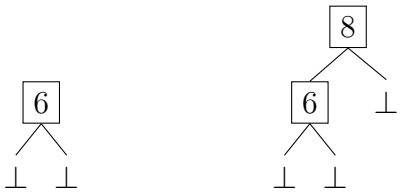
Problems

1. The set of all (non-empty) full binary trees that store non-negative integer key values may be defined inductively as follows.

   (a) $node(k, \perp, \perp, 0)$, for any non-negative integer k, is a full binary tree of height 0.

   (b) If t_l and t_r are full binary trees of height h, then $node(k, t_l, t_r, h+1)$, for any non-negative integer k, is a full binary tree of height $h+1$.

   Please give a similar inductive definition for the set of all (non-empty) complete binary trees that store non-negative integer key values. For instance, $node(6, \perp, \perp, 0)$ is a single-node complete binary tree storing key value 6 and $node(8, node(6, \perp, \perp, 0), \perp, 1)$ is a complete binary tree with two nodes — the root and its left child, storing key values 8 and 6 repsectively. Pictorially, they may be depicted as below.



2. Please give a binary de Bruijn sequence of 2^4 bits, which is a *cyclic* sequence of 16 bits $a_1 a_2 \cdots a_{16}$ such that each binary sequence of size 4 appears somewhere in the sequence. Explain how you can systematically produce the de Bruijn sequence.

3. Design an algorithm that, given a weighted directed graph, detects the existence of a negative-weight cycle (the sum of the weights of its edges is negative). Please present your algorithm in adequate pseudocode and make assumptions wherever necessary. Explain why your algorithm is correct and give an analysis of its time complexity. The more efficient your algorithm is, the more points you will be credited for this problem. (Hint: adapt Floyd's algorithm for all-pair shortest paths.)

4. Prove that, if the costs of all edges in a connected weighted undirected graph are distinct, then the graph has a unique minimum-cost spanning tree.

5. 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 *updated*; $\{u, v\}$ may or may not belong to T. Prove that T is still an MCST of G under any of the following two conditions:

   (a) $\{u, v\}$ belongs to T and its cost decreases or

   (b) $\{u, v\}$ does not belong to T and its cost increases.

   You may assume that the costs of all edges are distinct before and after the cost update to $\{u, v\}$.

6. Below is the algorithm discussed in class for determining the strongly connected components (SCCs) of a directed graph. The algorithm is based on depth-first search (DFS). To explore the neighbors of a particular node v on which the SCC procedure is invoked, the algorithm visits each neighbor w of v via some edge (v, w). In terms of the DFS tree (which is implicit in the algorithm), the edge (v, w) may be classified as a tree edge, forward edge, back edge, or cross edge. How does the algorithm handle these different cases? Please explain by referring to the code and pointing out the actions taken for each case.

```
Algorithm Strongly_Connected_Components(G, n);
begin
   for every vertex v of G do
       v.DFS_Number := 0;
       v.component := 0;
   Current_Component := 0;  DFS_N := n;
   while v.DFS_Number = 0 for some v do
       SCC(v)
end

procedure SCC(v);
begin
   v.DFS_Number := DFS_N;
   DFS_N := DFS_N - 1;
   insert v into Stack;
   v.high := v.DFS_Number;
   for all edges (v, w) do
      if w.DFS_Number = 0 then
         SCC(w);
         v.high := max(v.high, w.high)
      else if w.DFS_Number > v.DFS_Number
                and w.component = 0 then
           v.high := max(v.high, w.DFS_Number)
   if v.high = v.DFS_Number then
      Current_Component := Current_Component + 1;
      repeat
```

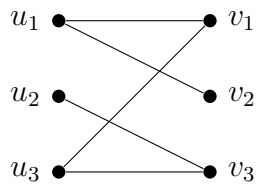
remove x from the top of *Stack*;
$x.component := Current\_Component$
until $x = v$
end

7. Consider designing by dynamic programming an algorithm that, given as input a sequence of distinct numbers, determines the length of a longest increasing subsequence in the input sequence. For instance, if the input sequence is $1, 3, 11, 5, 12, 14, 7, 9, 15$, then a longest subsequence is $1, 3, 5, 7, 9, 15$ whose length is 6 (another longest subsequence is $1, 3, 11, 12, 14, 15$).

   (a) Formulate the solution using recurrence relations.

   (b) Present the algorithm in suitable pseudocode, based on the previous recursive formulation. What is the time complexity of your algorithm?

8. The bipartite matching problem (the maximum-cardinality matching problem for bipartite graphs) can be reduced to the network flow problem, which in turn can be reduced to linear programming. Please illustrate the reductions, using the graph below as input to the bipartite matching problem.



   (a) Draw the network resulted from the conversion of the bipartite graph and show a maximum flow of the resulting network.

   (b) Give the linear-programming objective function and constraints for the network.

9. In the proof (discussed in class) of the NP-hardness of the clique problem by reduction from the SAT problem, we convert an arbitrary boolean expression in CNF (input of the SAT problem) to an input graph to the clique problem.

   (a) Please illustrate the conversion by drawing the graph that will be obtained from the following boolean expression:

$$(\overline{x} + \overline{z}) \cdot (x + y + \overline{z}) \cdot (\overline{x} + \overline{y} + z).$$

   (b) The original boolean expression is satisfiable. As a demonstration of how the reduction works, please use the resulting graph to argue that the original boolean expression is indeed satisfiable.

10. Solve one of the following two problems. (Note: if you try to solve both problems, I will randomly pick one of them to grade.)

(a) 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.

(b) The traveling salesman problem is as follows.

> Given a weighted complete graph $G = (V, E)$ (representing a set of cities and the distances between all pairs of cities) and a number D, does there exist a circuit (traveling-salesman tour) that includes all the vertices (cities) and has a total length $\leq D$?

Prove that the traveling salesman problem is NP-complete.

Appendix

- Below is a theorem useful for discovering an MCST of a connected weighted undirected graph $G = (V, E)$:

  Let V_1 and V_2 be a partition of V and $E(V_1, V_2)$ be the set of edges connecting nodes in V_1 to nodes in V_2. An edge with the minimum weight in $E(V_1, V_2)$ must be in an MCST of the given G.

- 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.