Computing Science 204 (A1, A2)
Final Examination: Version A
Wednesday December 11, 2002

Department of Computing Science
University of Alberta

INSTRUCTORS: G. LIN and I. E. LEONARD **TIME: 180 MINUTES**

LAST NAME: ______________________________

FIRST NAME: ______________________________

INSTRUCTIONS:

- **Put your name above. Print clearly.**
- **Make sure there are 7 problems, on 11 pages (78 marks in total).**
- **Closed book: Only Pen, Pencil, & Eraser allowed.**
- **Use space below the questions to write your answers.**

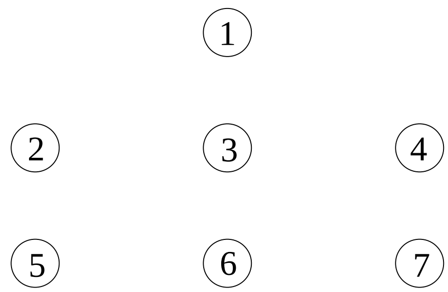
Name:____________________________________ **I.D. #:**____________________

Question One. [15 = 1 + 4 + 1 + 4 + 4 + 1 marks]

The following arcs (an *arc* is a directed edge) specify a weighted directed 7-vertex graph D, where $(i, j; w)$ means there is an arc from vertex i to vertex j of weight w :

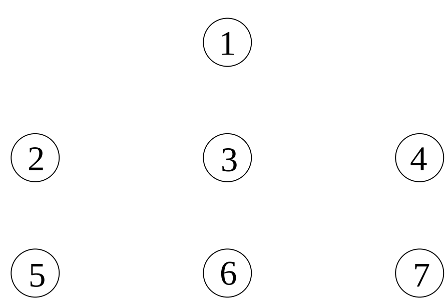
$$(1,2;3),\ (1,4;2),\ (2,3;3),\ (2,5;2),\ (3,1;8),\ (3,4;1),\ (4,6;1),\ (4,7;9),\ (5,6;4),\ (6,2;1),\ (6,3;7),\ (6,7;8).$$

(a) Draw the graph D using the following arrangement of vertices. Draw each arc as an arrow and label it with its weight.



(b) With vertex 1 as the starting vertex, use Dijkstra's algorithm to find all shortest paths in the graph from vertex 1, showing at each iteration (each while-loop body execution) the values of $d[i]$ and $p[i]$ for every vertex i.

(c) Draw the shortest paths computed in (b) in the following:



Name:_______________________________________ **I.D. #:**______________________

Question One. [15 = 1 + 4 + 1 + 4 + 4 + 1 marks] ...

Change the weight of arc $(3, 4)$ from 1 to -5. From now on we consider the graph D with this new arc $(3, 4; -5)$.

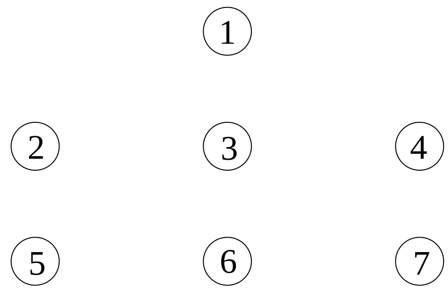
(d) Can we still apply Dijkstra's algorithm to compute all shortest paths from vertex 1? If your answer is yes, show at each iteration the values $d[i]$ and $p[i]$ for every vertex i and draw the shortest paths similar to part (b). If your answer is no, state the reason why the algorithm fails.

(e) Use the Bellman-Ford algorithm and the ordering of the arcs specified above in the statement of the problem to find all shortest paths in the graph D starting from vertex 1, showing at each iteration (each for-loop body execution) the values $d[i]$ and $p[i]$ for every vertex i.

Name:______________________________ **I.D. #:**____________________

Question One. [15 = 1 + 4 + 1 + 4 + 4 + 1 marks] ...

(f) If the Bellman-Ford algorithm returns the shortest paths, then draw them in the following. Otherwise draw a cycle of negative weight that is detected by the algorithm.



Name:____ **I.D. #:**______________________

Question Two. [10 = 5 + 5 marks]

Let $G = (V, E)$ be a simple, undirected and connected graph containing at least 3 vertices.

(a) Prove or disprove that a bridge is incident with at least one cut vertex (articulation point).

(b) Prove or disprove that a cut vertex is incident with at least one bridge.

Name:____ **I.D. #:**______________________

Question Three. [10 = 3 + 3 + 4 marks]

The graph $G = (V, E, w)$ is a simple, undirected, connected, weighted graph, where the weight function $w : E \longrightarrow \mathbb{N}$ assigns a positive weight to each of the edges.

It is known that three edges e_1, e_2 and e_3 all have weight 1, and all of the other edges have weight greater than 1.

It is also known that these three edges cannot coexist in any MST (minimum spanning tree) for G.

(a) Is there some MST for G that contains the edge e_1? If your answer is yes, show how to produce such an MST. If your answer is no, prove that no MST for G can contain e_1.

(b) Is there some MST for G that contains both edge e_1 and edge e_2? If your answer is yes, show how to produce such an MST. If your answer is no, prove that no MST for G can contain both e_1 and e_2.

(c) Prove or disprove that the edges e_1, e_2 and e_3 lie on a simple cycle and the cycle contains exactly these three edges.

Name:______________________________________ **I.D. #:**______________________

Question Four. [**8 = 4 + 4 marks**]

Let $G = (V, E)$ be a simple, undirected and connected graph represented by a given adjacency list structure.

(a) The BFS and DFS algorithms are both executed using this adjacency list structure and both start with the *same* vertex v_0. Is it true that the height of the resulting BFS spanning tree is no greater than the height of the resulting DFS spanning tree? If your answer is yes, state the reason. If your answer is no, give an example.

(b) The BFS and DFS algorithms are both executed again starting with *different* vertices but still using the same adjacency list structure. Is it true that the height of the resulting BFS spanning tree is no greater than the height of the resulting DFS spanning tree? If your answer is yes, state the reason. If your answer is no, give an example.

Name:______________________________________ **I.D. #:**______________________

Question Five. [11 marks: O 4 marks, Ω 7 marks]

Assuming that all keys are distinct, show that the best-case running time of `Heapsort` applied to arrays of length n is in $\Theta(n \log n)$.

Name:____ **I.D. #:**______________________

Question Six. [11 = 5 + 3 + 3 marks]

Given a set S of n positive integers $x_1, x_2, x_3, \ldots, x_n$, and another positive integer B, the *Subset Sum* problem asks if there is a subset of integers from S which sum to exactly B.

For example, if $S = \{1, 2, 4, 6\}$ and $B = 7$, then the answer is **yes** since $1 + 2 + 4 = 7$.

(a) Design a dynamic programming algorithm to solve the Subset Sum problem. The running time of your algorithm should be in $O(n \times B)$.

Note: If you are not able to design the algorithm as required in part (a), you can still answer the 2 questions below, assuming that you have an algorithm with running time $O(n \times B)$.

(b) What is the size of the above instance of the Subset Sum algorithm?

(c) Is the dynamic programming algorithm you designed a polynomial time algorithm?

Name:______________________________________ **I.D. #:**______________________

Question Seven. [13 = 3 + 3 + 7 marks]

In a simple, undirected and connected graph $G = (V, E)$, the *Independent Set* problem asks for a maximum subset of pairwise non-adjacent vertices. The *Vertex Cover* problem asks for a minimum subset of vertices such that for every edge in E, at least one of the end-vertices of the edge is in the subset.

(a) State the decision problem for the Independent Set problem.

(b) State the decision problem for the Vertex Cover problem.

(c) Assuming that the decision problem for the Independent Set problem is NP-complete, prove that the decision problem for the Vertex Cover problem is NP-complete.

Do not write on this page, for instructor's use only.

15 pts	1	
10 pts	2	
10 pts	3	
8 pts	4	
11 pts	5	
11 pts	6	
13 pts	7	
78 pts	Total	