COMPSCI330 Design and Analysis of Algorithms
Midterm Exam 2

Guidelines

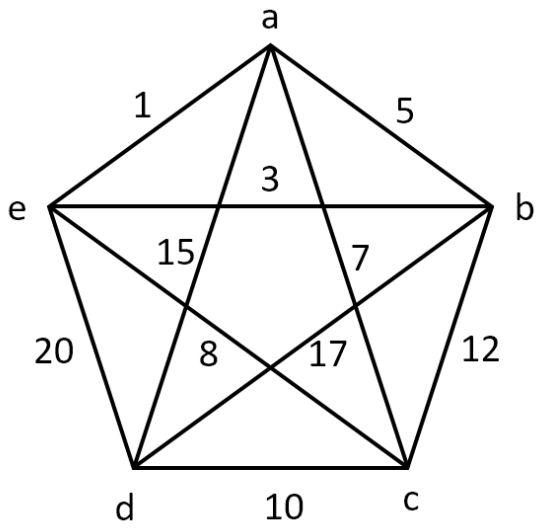
- **Describing Algorithms** If you are asked to provide an algorithm, you should clearly define each step of the procedure, and then analyze its overall running time. There is no need to write pseudo-code; an unambiguous description of your algorithm in plain text will suffice. If the running time of your algorithm is worse than the suggested running time, you might receive partial credits.
- **Using Existing Algorithms** If you use any of the graph algorithms we covered in class, you just need to state the name of the algorithm, which graph you run it on (if the problem involves multiple graphs), and its running time. You do not need to write pseudo-code or explain how the algorithm works.
- **Timing** Exam starts at 3:05 pm and ends at 4:20 pm.

Name: _____ Duke ID: _____

For grading use only:

Problem	1a	1b	2a	2b	3	4	Total
Score							

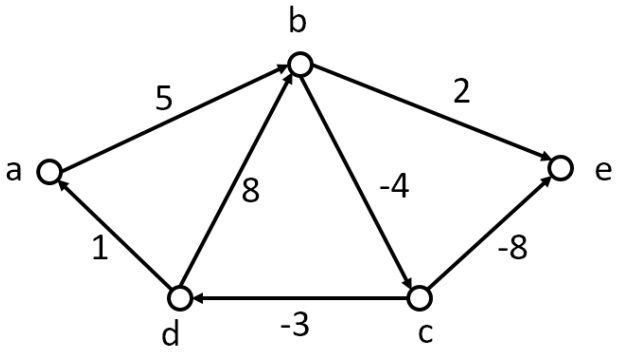
Problem 1 (Graph Algorithms)**.** (25 points)
(a) Minimum Spanning Trees: compute the minimum spanning tree of the following graph:



(a.1) (5 points) List the edges added by the Prim's algorithm (in the order that they are added), when the starting vertex is chosen to be a.

(a.2) (5 points) List the edges added by the Kruskal's algorithm (in the order that they are added).

(b) Shortest Path with negative edge weights. In this problem we will run Bellman-Ford algorithm on the following directed graph:



(b.1) (10 points) Starting from vertex a, please fill in the following table for the Bellman-Ford algorithm. (Recall that $d[u, i]$ is the shortest path distance from u to i that uses at most i edges.)

u	a	b	c	d	e
d[u,0]	0	∞	∞	∞	∞
d[u,1]					
d[u,2]					
d[u,3]					
d[u,4]					
d[u,5]					

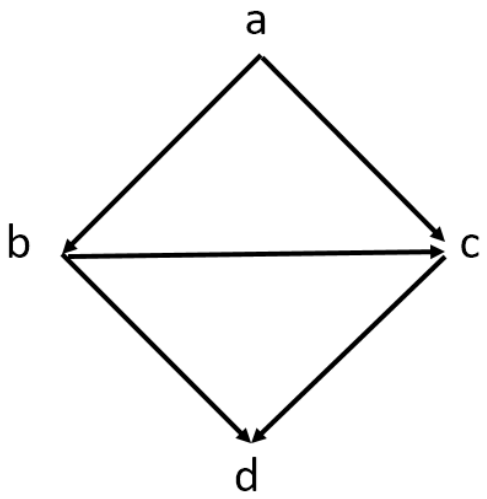
(b.2) (5 points) Does the graph have a negative cycle (Yes/No)? If your answer is Yes, explain how Bellman-Ford algorithm detected there is a negative cycle; if your answer is No, explain how Bellman-Ford algorithm certified that there is no negative cycle.

Problem 2 (Graph Examples)**.** (25 points)
(a) (10 points) Give an undirected graph G that meets the following criteria:

1. All edges have different weights except exactly two edges e_1 and e_2 that have the same weight.
2. Edges e_1 and e_2 do not share a common vertex.
3. There are two different minimum spanning trees in G.
4. G has at most 5 vertices.

Draw the graph, label the vertices, write the weights next to the edges. Point out edges e_1, e_2, and list the edges for two different minimum spanning trees of the graph.

(b) (15 points) Give an example of a **directed** graph with a starting vertex a, such that no DFS tree is a BFS tree. More precisely, the graph may have multiple DFS trees and multiple BFS trees due to the ordering of edges in running the algorithms. What we are asking is a graph where none of the multiple DFS trees is equal to any of the multiple BFS trees. The following graph is an example that does not work:



In this graph there are 3 different DFS trees: 1. (a, b), (b,d),(b,c); 2. (a,b), (b,c), (c,d); 3. (a,c), (c, d), (a,b). There are also 2 different BFS trees: 1. (a,b), (a,c), (b,d); 2. (a,c), (a,b), (c,d). This does not work because the third DFS tree is equal to the second BFS tree (two graphs are equal if they have the same set of vertices and edges).

Draw the **directed** graph, and explain why no DFS trees can be a BFS tree (you can do this by listing all the DFS/BFS trees, or you can do this by write a short proof, such as "any DFS tree must contain one of these edges but no BFS tree contain any of these edges"). Your graph should contain no more than 5 vertices.

(This page is left blank, use it to answer 2b or as scratch paper.)

Problem 3 (Toll roads)**.** (25 points) Rong is trying to drive to a nearby city. The map is given as a directed graph $G = (V, E)$ with n vertices and m edges. The starting point is vertex s and the end point is vertex t. Each edge (u, v) represents a road, and its weight $w[u, v]$ is the amount of time it takes to go from u to v.

Some roads (a subset of edges $E' \subset E$) have tolls - you need to pay 1 dollar to use one of these roads; the other roads do not have tolls. Whether a road has toll or not is given as input, and can be checked in constant time. Rong forgot to bring his wallet and he only has 1 dollar with him, so he can only use at most 1 road with toll. You need to design an algorithm that helps Rong find the shortest path (in terms of travel time) from s to t that uses at most 1 road with toll.

Design the algorithm and analyze its running time. Your algorithm should run in time $O(m + n \log n)$. If you designed an algorithm with slower (but still polynomial) running time, you will get 10 points.

(This page is left blank.)

Problem 4 (Weighted Matching)**.** (a) (10 points) Consider the classroom assignment problem again where we try to assign courses to classrooms. We form a bipartite graph between courses and classrooms, such that if a course can be assigned to a classroom, there is an edge between them. In addition, the edges now have weights $0 \leq w_{i,j} \leq C$: a higher weight means the classroom is more suitable for the course. Write a linear program that finds a matching between courses and classrooms that maximizes the sum of weights of matching edges. (In this problem you do not need to prove the linear programming will return an integer solution.)

(b) (15 points) Suppose now we want to first maximize the number of courses that have classrooms, and after that maximize the sum of weights of matching edges. Describe how to modify the linear program above to solve this new problem. (As an example, suppose there are two courses x_1, x_2 and two classrooms y_1, y_2, the weight for $(x_1, y_1), (x_2, y_2)$ are both equal to 1, the weight for (x_1, y_2) is equal to 100 and (x_2, y_1) is not an edge. Then the solution to the previous problem will just take the edge (x_1, y_2), while the solution to this problem should take edges (x_1, y_1) and (x_2, y_2).) To get all 15 points you need to use a *single* linear program to solve this problem.