COMPSCI330 Design and Analysis of Algorithms
Midterm Exam

Guidelines

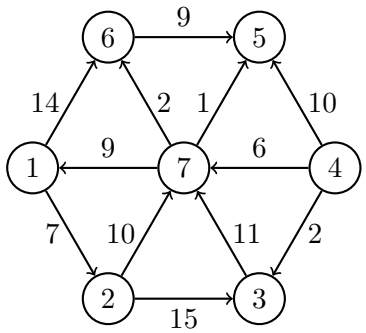
- **Describing Algorithms** If you are asked to provide an algorithm, you should clearly define each step of the procedure, and then analyze its overall running time. There is no need to write pseudo-code; an unambiguous description of your algorithm in plain text will suffice. If the running time of your algorithm is worse than the suggested running time, you might receive partial credits.
- **Timing** Exam starts at 3:05 pm and ends at 4:20 pm.

Name: _____

Duke ID: _____

Problem 1 (Graph Algorithms)**.** (25 points)

(a) Run Dijkstra's algorithm on the following directed graph, starting from vertex 1. The numbers in the circles are the indices for nodes, and the numbers near the arrows are the weights of the edges.



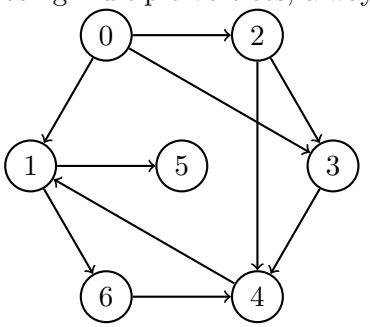
(a.1) (5 points) Give the ordering in which the vertices are visited in Dijkstras algorithm. (When there are multiple options, the algorithm chooses the vertex with smaller index.) Also list the final shortest path distance for all vertices.

Ordering	1	2	3	4	5	6	7
Vertex	1						
Distance	0						

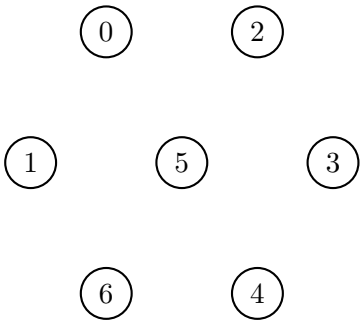
(a.2) (5 points) After vertex 6 is visited by the algorithm, what are the distance values for all vertices (if a vertex cannot yet be reached, the distance is $+\infty$?)

Vertex	1	2	3	4	5	6	7
Distance	0						

(b) Run DFS on the following graph, starting from vertex 0. When you have the option of choosing multiple vertices, always choose the vertex with smaller index first.



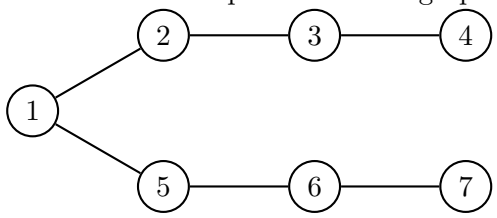
(b.1)(5 points) Draw the DFS tree. (Feel free to use the vertices we provided below, if you need to change your solution, please point clearly where your solution is.)



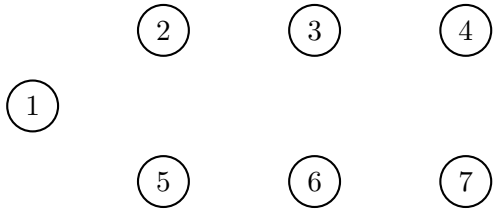
(b.2)(5 points) List the pre-order and post-order.

(b.3) (5 points) For edges not on the DFS tree, classify them as forward edge, backward edge or crossing edge.

Problem 2 (Graph Examples)**.** (25 points)
(a) (10 points) Construct an undirected graph with 7 vertices and a starting vertex 1, such that **one of** its BFS tree looks like the figure below, and the graph has as many edges as possible. You should draw the graph with all the edges, label the vertices as in the given figure, and list the BFS order of the graph (you need to list the specific BFS order that would lead to the given BFS tree). (You do not need to prove that the graph you draw has the maximum number of edges possible.)

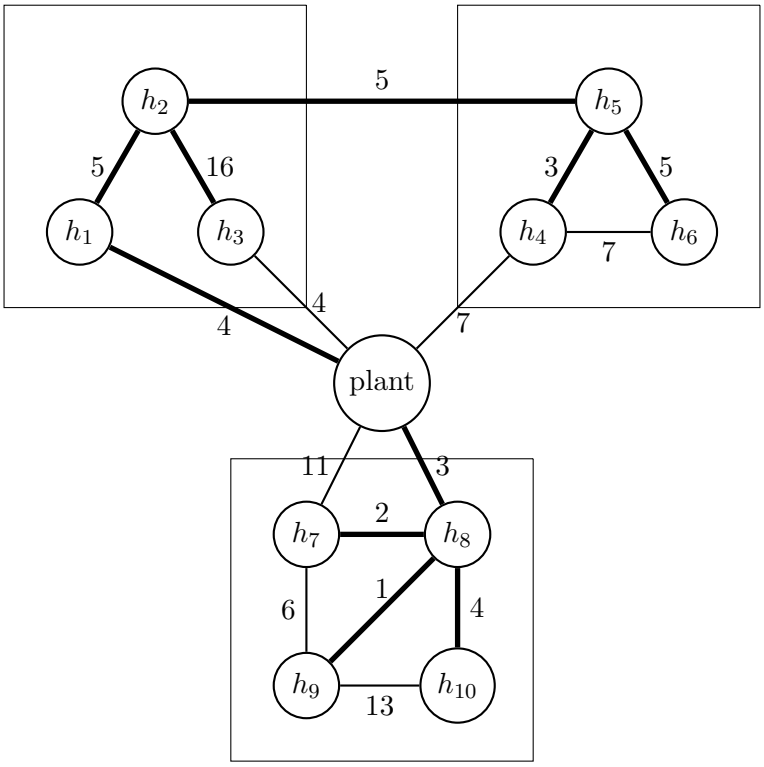


Feel free to use these vertices. If you prefer to draw the entire graph, please clearly indicate where it is.



(b) (15 points) Construct a weighted directed graph with 5 vertices (with no negative cycles), such that when running Bellman-Ford algorithm starting from vertex 1, the distance to vertex 5 changes 4 times. Draw the graph, and list the values of $d[5, i]$ for $i = 0, 1, 2, 3, 4, 5$.

Problem 3 (Electricity Supply)**.** (25 points) There are k communities that have n houses in total. In this problem, our goal is to connect all of these houses to an power plant. We are given a graph (see figure below) with $n+1$ vertices representing the power plant (vertex 0) and the houses (remaining n vertices). There is an undirected edge (u, v) with weight $w(u, v) > 0$ if we can build a connection between vertex i and vertex j. A house has electricity as long as it is connected with the power plant (by a path). We also know which houses belong to the same community.



To prevent loss of power, each community also has its own generator (located in one of the houses, which house it is does not matter for this problem). Therefore, when we are building the connections, we want to make sure that for every community, even if all the connections with outside (either with the power plant or with houses in another community) fail, all the houses within the community are still connected. The thicker lines in the figure above illustrate the optimal solution satisfying this requirement.

Design an algorithm that finds the minimum cost way of connecting all the houses to the power plant, while satisfying the constraint in the above paragraph. You can assume there is always a way to connect the houses while satisfying the constraint.

(a) (10 points) Describe your algorithm (it should not be slower than $O(m \log n)$, where m is the number of edges in the graph).

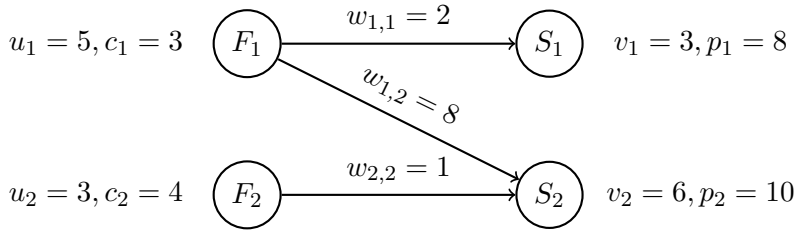
(b) (5 points) Analyze the running time of your algorithm.

(c) (10 points) Prove the correctness of your algorithm.

(This page is left blank intentionally.)

Problem 4 (Shipping Products)**.** (25 points) Company X is trying to figure out how to ship their product from factories to shops. There are n_1 factories and n_2 shops. Factory F_i $(i = 1, 2, ..., n_1)$ can produce at most u_i products at the cost of c_i per unit. Shop S_j $(j = 1, 2, ..., n_2)$ can sell at most v_j products at the price of p_j per unit. However, transportation is only available for a limited pairs of factories and shops - you are given a list of possible transportation routes $E = \{(i_1, j_1), (i_2, j_2), ..., (i_m, j_m)\}$. Each route $(i, j) \in E$ has a cost of $w_{i,j}$ per unit, where transporting each product from factory F_i to shop S_j costs $w_{i,j}$. See Figure below for an example. In this example, the optimal solution would produce 3 products at each of F_1 and F_2, and transport them to S_1 and S_2 correspondingly. The total profit is 24.

Note: All of $u_i, c_i, v_j, p_j, w_{i,j}$ are constants given to you as input, these are not the variables. You need to define your own variables. Your LP needs to work for all instances of the problem, not just the example given.



(a) (10 points) Write a linear program to compute the maximum profit that the company can make. Profit is equal to the total selling price, minus the total production cost and total transportation cost.

(b) (10 points) Suppose you can overload the factories so that factory F_i can produce at most u_i' extra products, at the additional cost of $c_i' > c_i$ (that is, the first u_i products still cost c_i each, but the next u_i' products will cost c_i' each). Modify your LP so that it computes the maximum profit in this setting. (For the example before, you can think about $u_1' = 1, c_1' = 4$, and $u_2' = 4, c_2' = 7$. In this case the optimal solution would produce 3 products at F_1 and 6 products at F_2. The total profit for this solution is 30.)

(c) (5 points) If in the setting of part (b), we did not have the assumption that $c_i' > c_i$, does your LP in part (b) still work? If not, give a feasible solution to your LP that does not correspond to an actual feasible scenario in the problem. (You will only get points for this part if your solution to (b) is at least close to correct.)