COMPSCI330 Design and Analysis of Algorithms
Final Exam 2019

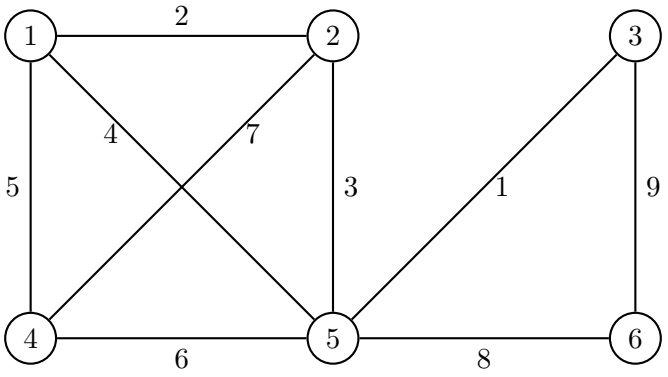
Guidelines

- **Describing Algorithms** If you are asked to provide an algorithm, you should clearly define each step of the procedure and 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.
- **Proof of Correctness** You are only required to prove the correctness of your algorithm if there is a subproblem that asks you to do that.
- **Timing** Exam starts at 2:00 pm and ends at 5:00 pm.
- **Score** The exam has 5 problems and each problem has 20 points. The total number of points is 100.

Name: _____ Duke ID: _____

Problem 1 (Warm-up)**.** (20 points)
(a) (10 points) Run Prim and Kruskal on the following graph. For Prim's algorithm, start at vertex 1. List the edges in the order that they are added into the minimum spanning tree.

Please write an edge as a pair of vertices, like (1,2). Also, please always write the vertex with smaller index in front (that is, write (1,2) instead of (2,1), even though both represent the same edge).



Ordering	1	2	3	4	5
Prim's					
Kruskal					

(b) (10 points) Solve the following recurrence relation:

$$T(n) = \sum_{i=1}^{\log_3 n} T(n/3^i) + n.$$

In this problem you can assume n is a power of 3 and $T(1) = 0$. The sum expands to

$$T(n) = T(n/3) + T(n/9) + \cdots + T(1) + n.$$

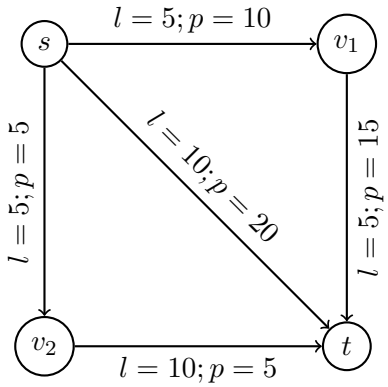
Problem 2 (Meeting Scheduling)**.** (20 points) There are n groups of people trying to schedule meetings at the same time, and there are m meeting rooms available. Each group i ($i = 1, 2, ..., n$) has a_i people, and each room j ($j = 1, 2, ..., m$) has a capacity of b_j. Group i can be scheduled to use room j if and only if $a_i \leq b_j$ (i.e., the number of people is no larger than the capacity of the room).

Of these n groups, the first k ($1 \leq k \leq n$) groups have high priority and they need to be scheduled (there will always be at least one way to schedule them). Design an algorithm that schedules all the high priority group, and schedules as many other groups as possible.
(a) (8 points) Describe your algorithm and analyze its running time. Your algorithm should run in $O(n \log n + m \log m)$ time.

(b) (12 points) Prove the correctness of the algorithm you designed in part (a).

Problem 3 (Flat Path)**.** (20 points) You are driving in a hilly landscape. The map of this area can be described as a directed graph G with n vertices and m edges. Your goal is to go from vertex s to t.

Each edge (u, v) in the graph has a length $l(u, v) > 0$, and a slope $p(u, v) \geq 0$. Your goal is to find a path that (1) has the minimum length; and (2) among all the paths that have minimum length, minimizes the maximum slope along the path.



Example: For the graph on the left, there are three paths from s to t. The path we are looking for is (s, v_1, t), with length 10 and maximum slope 15. The reason why (s, v_2, t) is bad is that it has total length 15, so it is not a shortest path. For the path (s, t), even though it has the same length 10, its maximum slope is 20, which is worse than that of (s, v_1, t).

(a) (12 points) Design an algorithm for this problem and analyze its running time. Your algorithm should run in $O(m + n \log n)$ time.

(b) (8 points) Prove the correctness of the algorithm you designed in part (a). Note that if you modified a classical algorithm you need to show what changes in the proof.

Problem 4 (Shared-Bus Take II)**.** (20 points) Remember how you helped the famous ride-sharing company Ubyft to created a new "shared-bus" service during the first midterm? Well that didn't work very well. Turns out that the customers don't like to specify two points, they just want to give a single destination. Ubyft is redesigning their algorithm and they need your help.

The bus runs on a path, each point on the path is denoted by its distance from the starting point. In this new problem, there are n passengers. They all board the bus at the starting point 0. Passenger i specifies a destination x_i where the passenger wants to go. You can assume that the x_i's are already sorted in increasing order.

If passenger i leaves the bus at a position y, then the "unhappiness" of the passenger is $(x_i - y)^2$.

Similar to the previous problem, there are no fixed bus stops. Instead, the stops will be decided after gathering the user information. The bus can make at most k stops, and your goal is to design an algorithm that minimizes the sum of unhappiness of all the passengers.

As an example, if there are 4 passengers with with $x_1 = 1, x_2 = 18, x_3 = 19, x_4 = 23$, and $k = 2$, then the optimal solution will make two stops at points 1 and 20. The first stop serves the first passenger; the second stop serves the remaining three passengers. The unhappiness for passengers 1, 2, 3, 4 are 0, 4, 1, 9 respectively and the total unhappiness is 14.

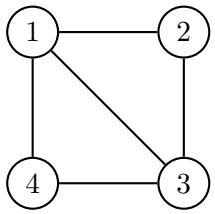
(a) (12 points) Define the state (sub-problems), write the transition function and specify the base cases.

(b) (8 points) Design an algorithm for finding the locations where the bus needs to stop. (Your algorithm only needs to output the optimal total unhappiness of the passengers). Analyze the running time of your algorithm. (Your algorithm should run in time $O(kn^2)$.)

Problem 5 (DOUBLE DOMINATING SET)**.** (20 points)

In the DOMINATING SET problem, we are given an undirected graph $G = (V, E)$ with n vertices and a number k ($1 \leq k \leq n$). A vertex u dominates itself and all of its neighbors. That is, vertex u dominates vertex v if $v = u$ or v is adjacent to u. A set S of the vertices is called a *dominating set* if every vertex $v \in V$ is dominated by at least one vertex $u \in S$. DOMINATING SET problem asks you to check whether there is a dominating set of size k in graph G. It is well-known that DOMINATING SET is an NP-complete problem.

In this problem, we consider a variant called DOUBLE DOMINATING SET. In this problem, the input is an undirected graph $G' = (V', E')$ with n' vertices, and a number k' ($1 \leq k' \leq n'$). A set $S' \subset V'$ is called a *double dominating set*, if every vertex $v \in V'$ is dominated by at least two vertices in S'.



Example: For the graph on the left, vertex $\{1\}$ is a dominating set of size 1; vertices $\{2, 4\}$ form a dominating set of size 2. However, neither $\{1\}$ nor $\{2, 4\}$ is a double dominating set. The set $\{1, 3\}$ is a double dominating set of size 2.

(a) (3 points) To show DOUBLE DOMINATING SET is NP-hard based on the fact that DOMINATING SET is NP-complete, what is the correct direction of reduction? (Please answer in the form A to B)

---------- TO ----------

(b) (5 points) Prove that DOUBLE DOMINATING SET is in NP.

(c) (12 points) Do a reduction (related to the DOMINATING SET problem) to show DOUBLE DOMINATING SET is NP-hard. (Hint: The intended solution only creates 2 extra vertices in the new instance. Duplicating the graph seems like a natural idea but I couldn't make it work.)