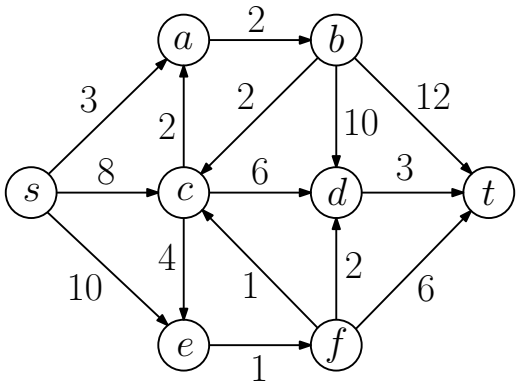
Problem Set 2

CMSC 27230: Honors Theory of Algorithms

Assigned January 18, Due January 23

Problem 1: Dijkstra's Algorithm, Kruskal's Algorithm, and Prim's Algorithm (30 points)

(a) 15 points: Use Dijkstra's algorithm to find the shortest path from s to t in the following graph G. For your answer, first give the state of the priority queue after we add in the edges



going out from s. Then give the following data for each new vertex that is reached.

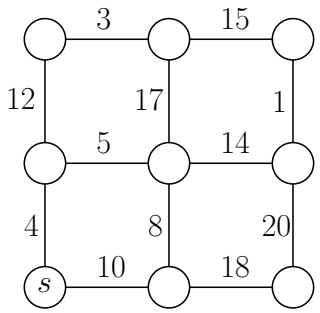
(a) The name of the vertex v which is reached and the length of the shortest path from s to v.

(b) The state of the priority queue after we add in the edges which go out from v.

After giving this data, give all of the shortest paths from s to t.

Note: For this problem, you do not need to show the heap structure of the priority queue. Instead, you can represent the priority queue as a sorted list. Also, you may either use a priority queue with edges or a priority queue with vertices.

(b) 15 points: Use Kruskal's algorithm and Prim's algorithm (starting from the vertex s) to find the minimum spanning tree in the following graph. For your answer, give the order in which the edges of G are considered for Kruskal's algorithm, the resulting minimum spanning tree T of G, and the order in which Prim's algorithm finds the edges of T if we start from the vertex s. Note that for this last part you only need to order the edges in T.



Problem 2: Restaurant Rush (Rephrasing of Problem 13 in Chapter 4 of Kleinberg-Tardos, 20 points)

You are the owner of a restaurant and there is suddenly a rush of hungry people. Unfortunately, there is currently only one chef in the kitchen so the people are going to have to wait for a while, but you'd like to minimize the total unhappiness.

To model this problem assume that there are n tables. The orders for table i will take time $l_i > 0$ to cook. If their orders come at a time t_i, the total unhappiness of table i will be $u_i t_i$ where $u_i > 0$. Your goal is to minimize the total unhappiness $\sum_{i=1}^{n} u_i t_i$ where $t_i = l_i + \sum_{j:\text{Table } j \text{ is served before table } i} l_j$.

Give an algorithm for this problem, prove that your algorithm is correct, and analyze its runtime.

Problem 3: Supervisory Committee (Rephrasing of Problem 15 in Chapter 4 of Kleinberg-Tardos, 25 points)

You have n part-time employees, each of whom work during an interval $[a_i, b_i]$. You don't have time to supervise all of your employees, so you would like to promote a few of your employees to managers to help supervise the remaining employees. Your employees don't need to be supervised all of the time, but you would like every employeee to overlap with at least one of your managers. What is the minimum number of managers you need to accomplish this and how can you choose these managers?

Give a polynomial time algorithm for this problem, prove that it is correct, and analyze its running time.

Note: We can express this problem mathematically as follows: What is the minimum size of a subset $I \subseteq [n]$ such that for all $j \in [n]$ there exists an $i \in I$ such that $[a_i, b_i] \cap [a_j, b_j] \neq \emptyset$?

Problem 4: Treasure Scramble (25 points)

You are an adventurer trying to grab treasure from an ancient ruin before it collapses. There are n treasures, each with a value v_i and a time t_i after which it can no longer be taken. At each time t from 1 to T, you can take one treasure i which is still available, i.e, you have not yet taken this treasure and $t \leq t_i$. What is the maximum total value of treasure you can take and which treasure should you take at each time to achieve this?

For example, if there are five treasures with the following values and times

1. $v_1 = 1$ and $t_1 = 4$
2. $v_2 = 6$ and $t_2 = 3$
3. $v_3 = 3$ and $t_3 = 1$
4. $v_4 = 2$ and $t_4 = 2$
5. $v_5 = 5$ and $t_5 = 3$

then it is optimal to take treasure 3 at time 1, treasures 2 and 5 at times 2 and 3, and treasure 1 at time 4 for a total value of $3 + 6 + 5 + 1 = 15$.

Give an algorithm for solving this problem, prove that your algorithm is correct, and analyze its runtime. Hint is available.

Hint

4. It may help to work backwards from the end.