CS2040S: Data Structures and Algorithms

Discussion Group Problems for Week 13

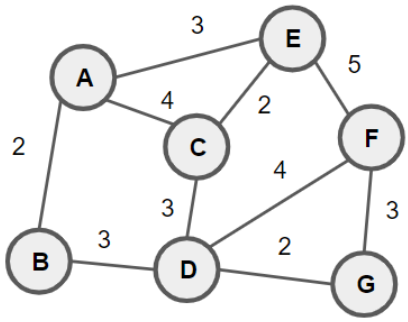
For: April 10–April 14

Goals:

- MST
- Dynamic Programming

Problem 1. MST Review

Problem 1.a.



Can you use the cycle and cut property of MST that we learnt in class to determine which edges must be in the MST?

Perform Prim's, Kruskal's, and Boruvka's (optional) MST algorithm on the graph above.

Problem 1.b. Henry The Hacker has some ideas for a faster MST algorithm! Recently, he read about *Fibonacci Heaps*. A Fibonacci heap is a priority queue that implements insert, delete, and decreaseKey in $O(1)$ amortized time, and implements extractMin in $O(\log n)$ amortized time, assuming n elements in the heap. If you run Prim's Algorithm on a graph of V nodes and E edges using a Fibonacci Heap, what is the running time?

Problem 2. Divide-and-Conquer MST

Henry The Hacker has invented a new divide-and-conquer algorithm for finding a minimum spanning tree! The algorithm is super simple! It only involves 4 steps:

1. Find the median edge weight w_e.
2. Partition the edges using the edges into those $\leq w_e$ and those $> w_e$ into two graphs G_a and G_b.
3. Recursively find the MST for G_a and G_b, yielding trees T_a and T_b.
4. Combine the edges from T_a and T_b, and recursively find the MST for the resulting graph.

How well do you think this algorithm works? (What would happen if it continued to a base case of 1 edge?)

After a little bit of thought, Henry realizes that he can stop the recursion early (i.e., not going down to a base case of 1). Instead, if the number of edges in the graph $E < 4V$, then the recursion terminates and he uses Prim's algorithm to find the MST.

Why do you think this algorithm will return a valid MST (unlike the one we saw in lecture)?

Problem 3. Road Trip

Relevant Kattis Problems:

- https://open.kattis.com/problems/adventuremoving4
- https://open.kattis.com/problems/highwayhassle
- https://open.kattis.com/problems/roadtrip

You are going on a road trip. You get in your trusty car and drive to Panglossia, where you will spend a nice vacation by the beach.

You have already found the best route from your home to Panglossia (using Dijkstra's Algorithm). Next, you need to determine where you can buy petrol along the way. On the road between your home and Panglossia, there are a set of n petrol stations, the last of which is in Panglossia itself. Assume that your trip is complete when you reach the last station.

By searching on the internet, you find for each station, its location on the road and the cost of petrol. That is, the input to the problem is n stations, $s_0, s_1, \ldots, s_{n-1}$, along with $c(s_i)$, which specifies the cost of petrol at station s_i, and $d(s_i)$, which specifies the distance from your home to station s_i.

The tank of your car has a capacity of L liters. You begin your trip at home with a full tank of petrol. Your car uses exactly 1 liter per kilometer. Along the way, you must ensure that your car always has petrol (though you may arrive at a station just as you run out of petrol). Note that you do not need to fill your tank at a station. You can buy any amount of petrol, as long as it's within the capacity of your tank.

Your job is to determine **how much petrol to buy at each station** along the way so as to minimize the cost of your trip.

You may assume, for simplicity, that L and all the given distances are integers, i.e., all the quantities are integers.

Here are two examples, a simple one and a more complicated one.

Example 1: Your tank holds 6 liters, and there are 3 stations:

```
5km: $1
6km: $2
7km: $4
```

In this case, the best solution costs one dollar, where you purchase one liter of petrol at the first station at a cost of 1 per liter.

Example 2: Your tank holds 20 liters, and there are 10 stations:

```
7km:  $3
17km: $5
20km: $2
23km: $1
39km: $5
48km: $4
66km: $9
83km: $9
88km: $4
92km: $1
```

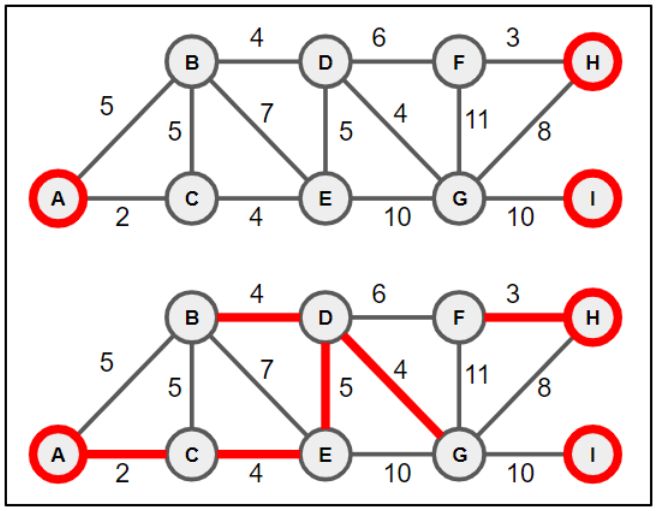
In this case, the best solution is to purchase petrol at each station as follows, for a total cost of 327 dollars:

```
0
0
3
20
5
20
15
5
4
0
```

Hint: To solve this problem, think about how to calculate $DP(s_j, k)$, the minimum cost to get from station s_j to the destination, assuming you have k liters of petrol left.

Problem 4. Power Plant

Given a network of power plants, houses and potential cables to be laid, along with its associated cost, find the cheapest way to connect every house to at least one power plant. The connections can be routed through other houses if required.



In the diagram above, the top graph shows the initial layout of the power plants (modelled as red nodes), houses (modelled as gray nodes) and cables (modelled as bidirectional edges with its associated cost). The highlighted edges shown in the bottom graph shows an optimal way to connect every house to at least one power plant. Come up with an algorithm to find the minimum required cost. You may assume that there exists at least one way to do so.

Problem 5. Traffic Reduction

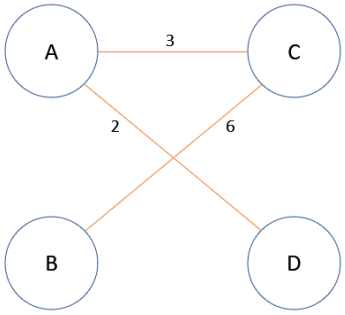
In order to reduce traffics, the Singapore government has passed a new law - cars are not allowed to go in circles. If you drive around in a circle, you will be charged a fine. Your job is to deploy a set of cameras to monitor the roads and catch drivers that are violating the new law.

For this problem, you are given a road map of Singapore as a connected, undirected graph $G = (V, E)$. For each road (i.e., edge) e, you are given the associated cost (i.e., weight) $w(e)$ of building a camera station that can detect when the same car passes twice. Your job is to find the cheapest set of roads (i.e., edges) to deploy a camera on such that you can detect any car that makes a circle.

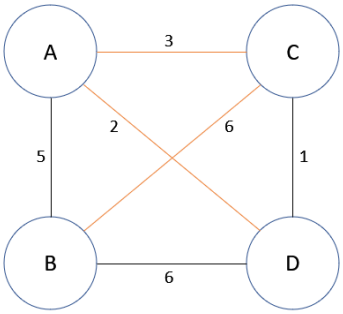
Problem 6. Let's play a game *(Just for fun)*

Now that we've reached the end of the semester, let's have some fun with an adaptation of the Stackelberg MST Game. You are now playing this game with the prof, and he is here to assess whether you have mastered CS2040S, here's how the game works:

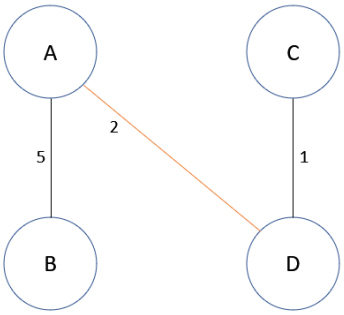
1. Prof draws an undirected weighted connected graph G_1 with $E = \theta(V)$. All edges that Player 1 draws will be **orange** edges.
2. You add any number of weighted edges to G_1 to create G_2. Adding duplicate edges is not allowed. All edges that you draw will be **black** edges.
3. Prof then selects an MST of graph G_2. This MST is M_1.
4. Your score is the sum of weights of the **black** edges in M_1.
5. Swap roles and repeat Step 2, 3, and 4 using the same graph G_1.
6. Whoever gets the higher score wins! (If it's a tie, you win!)



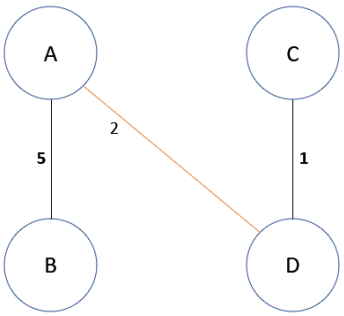
Player 1 constructs graph G_1



Player 2 draws additional edges to construct G_2



Player 1 selects an MST of G_2,which is M_1



Player 2 scores $5 + 1 = 6$

How can you win prof and show him that you have mastered CS2040S?

Problem 7. Espionage (optional)

You are a spy, currently on a mission to obtain intel on the final examination that will be unleashed upon this world by the dreaded *Htes Treblig* within the next 25 days. At the moment, you have a message to send to HQ in the form of a **weighted tree** of N vertices, where all of the edge weights are positive integers.

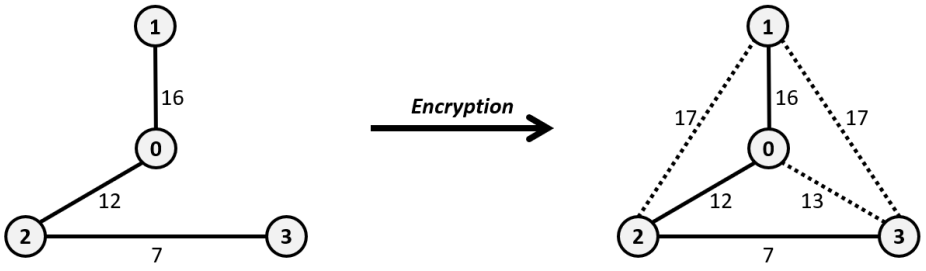
To ensure your message is not at risk of being intercepted, you need to encrypt your message. The way you decide to do this is by inserting additional edges (with integer weights) to your tree. The resulting graph, which represents the encrypted message, must satisfy the following properties:

- The resulting graph is a **complete graph** of N vertices (i.e. every pair of vertices in the graph must have exactly 1 edge between them).
- The initial tree is **the unique Minimum Spanning Tree** of the resulting graph.

Inserting edges with large weights will make the encryption process more complicated, so you would like to avoid them wherever possible. The cost of the encryption process is defined as the sum of the weights of the additional edges that are inserted into the graph during the encryption process.

Given the initial message, which consists of the number of vertices in the tree, N, and a list of $N-1$ edges of the tree, **output the minimum possible cost** of encrypting this message.

For example, let the initial message be the tree on the left, consisting of $N = 4$ vertices. The graph on the right represents the optimal way to encrypt the message with a total cost of $17+17+13 = 47$. Therefore, the expected output is 47.



Initial Graph *Encrypted Graph*

- This is it for CS2040S this semester. All the best! -