Name: Chirag Telang

ID: 109410601

CSCI 3104, Algorithms **Due March 12, 2021**

Problem Set 7 (50 points) **Spring 2021, CU-Boulder**

Advice 1: For every problem in this class, you must justify your answer: show how you arrived at it and why it is correct. If there are assumptions you need to make along the way, state those clearly.
Advice 2: Verbal reasoning is typically insufficient for full credit. Instead, write a logical argument, in the style of a mathematical proof.

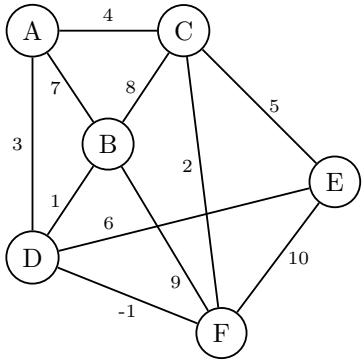
Instructions for submitting your solution:

- The solutions **should be typed** and we cannot accept hand-written solutions. Here's a short intro to Latex.
- You should submit your work through **Gradescope** only.
- The easiest way to access Gradescope is through our Canvas page. There is a Gradescope button in the left menu.
- Gradescope will only accept **.pdf** files.
- It is vital that you match each problem part with your work. Skip to 1:40 to just see the matching info.

Name: Chirag Telang
ID: 109410601

CSCI 3104, Algorithms **Due March 12, 2021**
Problem Set 7 (50 points) **Spring 2021, CU-Boulder**

1. Consider the following graph:



(a) Use Kruskal's algorithm to compute the MST. It will suffice to indicate the order in which edges are added to the MST.

(b) Now use Prim's algorithm starting at node A to compute the MST, again by indicating the order in which edges are added to the MST.

(c) Is it possible to change the starting node for Prim's algorithm such that it adds edges to the MST in the same order as Kruskal's algorithm? If so, which starting node(s) would work? Justify your answer.

Note: For parts (a) and (b), let (Node1, Node2) represent the edge between two nodes Node1 and Node2. Therefore, your answer should have the form: (Node1, Node2), (Node4, Node5), etc.

Solution:

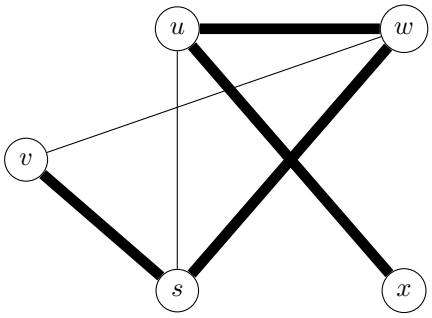
(a) (D, F), (D, B), (C, F), (A, D), (C, E)

(b) (A, D), (D, F), (D, B), (C, F), (C, E)

(c) Yes, it possible to change the starting node for Prim's algorithm such that it adds edges to the MST in the same order as Kruskal's algorithm. For this to work, one would have to choose either D or F as the starting node. Starting with D, your distance vector would be [A=3, B=1, $C = \infty$, E=6, F=-1]. Since F has the lowest cost, you would go to F (edge D, F) and then your distance vector becomes [A=3, B=1, C=2, E=6]. B has the lowest cost, so you would go to B (edge D, B) and then your distance vector becomes [A=3, C=2, E=6]. Now, C has the lowest cost, so you would go to C (edge F, C) and then your distance vector becomes [A=3, E=5]. Next, you would go to A (edge D, A) and your new distances vector is just [E=5]. Finally, you would choose E (edge C, E) and the path concludes. The final order is (D, F), (D, B), (C, F), (A, D), (C, E) – which is identical to Kruskal's algorithm. This is the same case if you were to start with node F.

Name: Chirag Telang
ID: 109410601

2. Consider the undirected, unweighted graph $G = (V, E)$ with $V = \{s, u, v, w, x\}$ and $E = \{(s, u), (s, v), (s, w), (u, w), (u, x), (v, w)\}$, and let $T \subset E$ be $T = \{(s, v), (s, w), (u, w), (u, x)\}$. This is pictured below with T represented by wide edges.



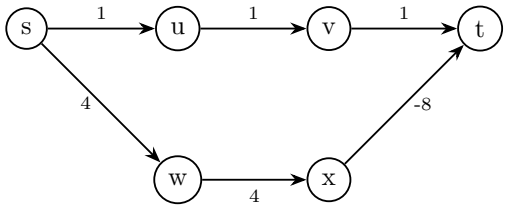
Demonstrate that T cannot be output by BFS with start vertex s.

Solution:

In Breadth First Search, we explore all the adjacent edges. Since there isn't an edge between s and u, T cannot be the output by BFS with start vertex s. Using BFS in the given tree would go from s to v, s to w, s to u, and then finally u to x. Therefore, in order for T to be a valid output by BFS, T would have to be $T = \{(s, v), (s, w), (s, u), (u, x)\}$.

Name: Chirag Telang
ID: 109410601

3. Given the following directed graph $G = (V, E)$ with starting and ending vertices $s, t \in V$, show that Dijkstra's algorithm does *not* find the shortest path between s and t.



Solution:

If you were to use Dijkstra's algorithm, the path you would take is s, u, v, t – with cost 3. However, since there is a negative cost from x to t, if you were to use the path s, w, x, t, you would get a cost of 0. Therefore, since the second path has a lower cost, Dijkstra's algorithm does not find the shortest path between s and t.

Name: Chirag Telang
ID: 109410601

4. Given a graph, implement Prim's algorithm via Python 3. The input of the Graph class is an Adjacency Matrix. Complete the Prim function. The Prim fucntion should return the weight sum of the minimum spanning tree starting from node 0.

   The file `graph.py` is provided; use this file to construct your solution and upload with the same name. You may add class variables and methods but **DO NOT MODIFY THE PROVIDED FUNCTION OR CLASS PROTOTYPES.**.

   Here is an example of how your code will be called:

   Sample input:

```
g = Graph([ [0, 10, 11, 33, 60],
            [10, 0, 22, 14, 57],
            [11, 22, 0, 11, 17],
            [33, 14, 11, 0, 9],
            [60, 57, 17, 9, 0]])

assert g.Prim() == 41
```