Name: Felipe Lima

ID: 109290055

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.

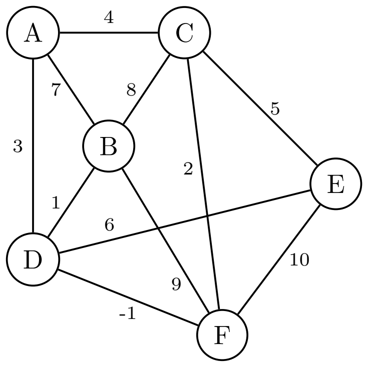
Name: Felipe Lima

ID: 109290055

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) $n = 6$, run until T has $n - 1$ edges
Kruskal's order will be:

$$T = (D, F)(D, B)(C, F)(A, D)(C, E)$$

(b) Starting at node A we will have:

$$T = (A, D)(D, F)(D, B)(F, C)(C, E)$$

(c) Yes, if we pick D as starting node, the Prim's algorithm will generate the following form:

$$T = (D, F)(D, B)(C, F)(A, D)(C, E)$$

Which is the same as the Kruskal's algorithm form found in question a

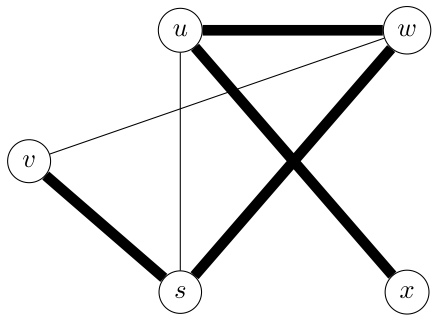
Name: Felipe Lima

ID: 109290055

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

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

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:
Since in a BFS we "explore" all the adjacent edges, T could not be outputted by BFS if we were to start on vertex S, because there should be an edge between nodes S and U in the BFS.

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.

$$\begin{tikzcd} s \arrow[r, "1"] \arrow[dr, "4"'] & u \arrow[r, "1"] & v \arrow[r, "1"] & t \\ & w \arrow[r, "4"'] & x \arrow[ur, "-8"'] & \end{tikzcd}$$

[= *

Solution:
In the given graph, Dijkstra's algorithm will not find the shortest path between s and t because it will mistakenly pick the path via nodes u and v. This path will generate a distance of 3, however, the path via nodes w and x will generate a distance of 0. Therefore, since $0 < 3$, Dijkstra's algorithm does not find the shortest path in the above situation.

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
```