Name: Willem Scott
ID: 108759616

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.

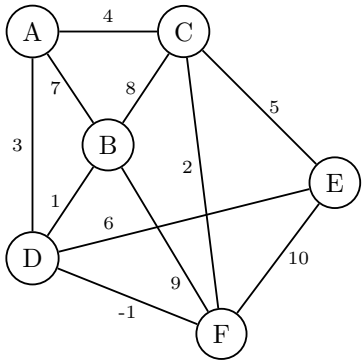
Instructions for submitting your solution:

- 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: Willem Scott
ID: 108759616

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.

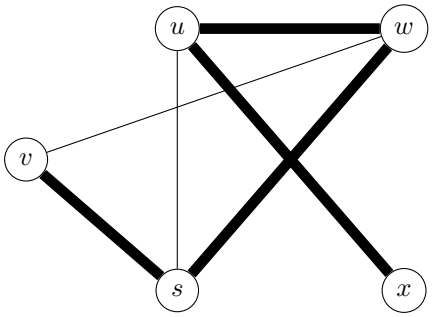
Solution:

(a) The edges included in the spanning tree are $[(D, F), (D, B), (C, F), (A, D), (C, E)]$ in that order.

(b) Prim's algorithm starting at node A yields $[(A, D), (D, F), (D, B), (F, C), (C, E)]$

(c) Using Prim's algorithm starting at D will yield the same order for the edges in the MST.

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.

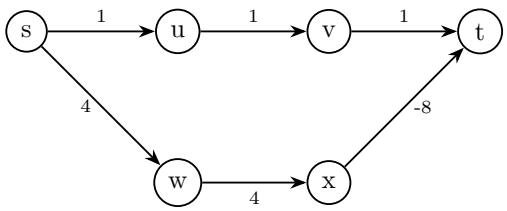


Demonstrate that T cannot be output by BFS with start vertex s.

Solution:

By the definition of a BFS, all children will be searched before grandchildren, and so forth. The children of node s are v, w, u, and will therefore be searched before the grandchildren of s which are the children of w. Because of this, u will be found to be a child of s before it will be found as a grandchild of w, therefore any BFS will yield the edge (s, u) and will never yield (u, w)

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:
Dijkstra's algorithm will first follow the path s, u, v, t before attempting to traverse s, w, since the total weight of the first path it finds is less than the weight of the first edge of its first alternative path. The algorithm is not designed to be used with negative numbers, and as such will not find this path, given that it expects all alternative paths to have a cumulative cost of at least 4. The algorithm terminates before it considers the alternative path starting with 4, and will not compute all paths if it is sure it has found the smallest.

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
```