Name: Luke Joyce

ID: 107355873

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**

Problem Set 5a (11 points) **Fall 2019, CU-Boulder**

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.
- If you don't have an account on it, sign up for one using your CU email. You should have gotten an email to sign up. If your name based CU email doesn't work, try the identikey@colorado.edu version.
- Gradescope will only accept **.pdf** files (except for code files that should be submitted separately on Gradescope if a problem set has them) and **try to fit your work in the box provided**.
- You cannot submit a pdf which has less pages than what we provided you as Gradescope won't allow it.
- Verbal reasoning is typically insufficient for full credit. Instead, write a logical argument, in the style of a mathematical proof.
- 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.
- You may work with other students. However, **all solutions must be written independently and in your own words.** Referencing solutions of any sort is strictly prohibited. You must explicitly cite any sources, as well as any collaborators.

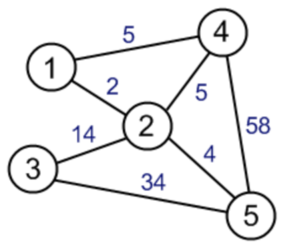
Name: Luke Joyce

ID: 107355873

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**

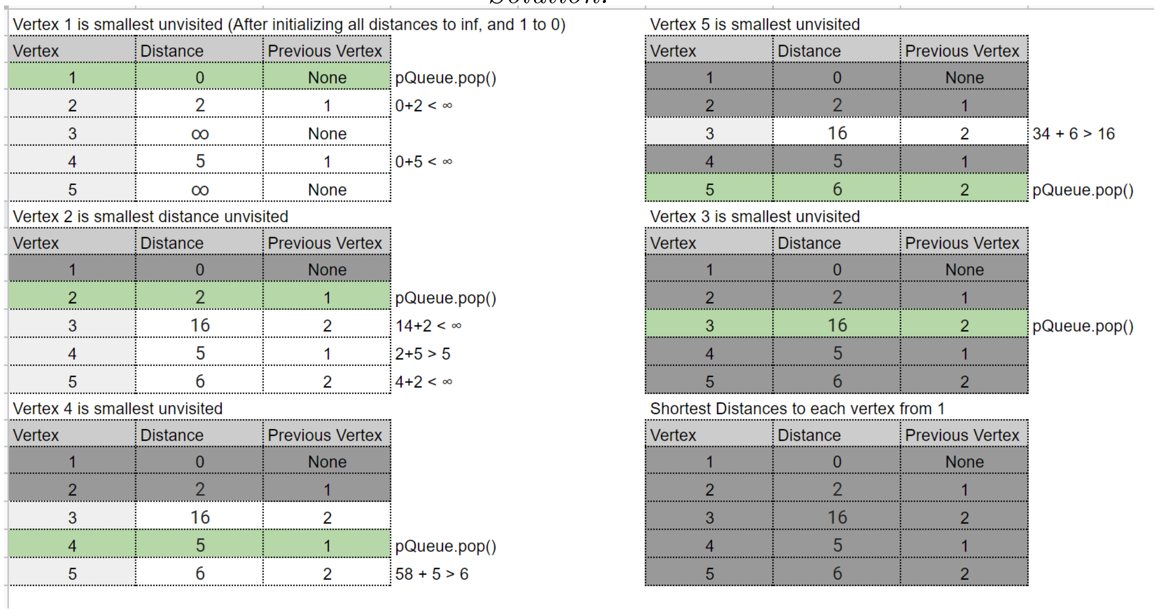
Problem Set 5a (11 points) **Fall 2019, CU-Boulder**

1. (7 pts) Consider the following weighted graph.



(a) (2 pts) Run Dijkstra's algorithm on this graph to obtain a tree of shortest paths. Use vertex 1 as the source vertex.

Solution.



Vertex 1 is smallest unvisited (After initializing all distances to inf, and 1 to 0)

Vertex	Distance	Previous Vertex	
1	0	None	pQueue.pop()
2	2	1	0+2 < ∞
3	∞	None	
4	5	1	0+5 < ∞
5	∞	None	

Vertex 2 is smallest distance unvisited

Vertex	Distance	Previous Vertex	
1	0	None	
2	2	1	pQueue.pop()
3	16	2	14+2 < ∞
4	5	1	2+5 > 5
5	6	2	4+2 < ∞

Vertex 4 is smallest unvisited

Vertex	Distance	Previous Vertex	
1	0	None	
2	2	1	
3	16	2	
4	5	1	pQueue.pop()
5	6	2	58 + 5 > 6

Vertex 5 is smallest unvisited

Vertex	Distance	Previous Vertex	
1	0	None	
2	2	1	
3	16	2	34 + 6 > 16
4	5	1	
5	6	2	pQueue.pop()

Vertex 3 is smallest unvisited

Vertex	Distance	Previous Vertex	
1	0	None	
2	2	1	
3	16	2	pQueue.pop()
4	5	1	
5	6	2	

Shortest Distances to each vertex from 1

Vertex	Distance	Previous Vertex
1	0	None
2	2	1
3	16	2
4	5	1
5	6	2

Name: Luke Joyce
ID: 107355873

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 5a (11 points) **Fall 2019, CU-Boulder**

(b) (2 pts) Run Kruskal's algorithm on this graph to obtain a minimum spanning tree.

Solution.

```
Priority queue of edges:
Edge:   Cost:   |   Start by checking first in queue:
1 <-> 2 [2]     |   1 <-> 2 [2]  (Spanning tree empty) Add to tree
2 <-> 5 [4]     |   2 <-> 5 [4]  (2 spans from 1) Add to tree
1 <-> 4 [5]     |   1 <-> 4 [5]  (1 spans from 1) Add to tree
2 <-> 4 [5]     |   2 <-> 4 [5]  (both vertices span from 1) Skip
2 <-> 3 [14]    |   2 <-> 3 [14] (2 spans from 1) Add to tree
3 <-> 5 [34]    |   3 <-> 5 [34] (both vertices span from 1) Skip
4 <-> 5 [58]    |   4 <-> 5 [58] (both vertices span from 1) Skip

Spanning tree consists of {1 <-> 2, 2 <-> 5, 1 <-> 4, 2 <-> 3}
```

(c) (1 pts) Is the tree of shortest paths produced by Dijkstra's algorithm a minimum spanning tree? Justify your answer.

Solution.

The tree of shortest paths produced by Dijkstra's Algorithm is indeed a minimum spanning tree. This is true because the total weight of the shortest paths tree is 25, and the total weight of the minimum spanning tree is also 25. The minimum spanning tree and the shortest paths tree are also the same tree.

(d) (2 pts) Find two vertices u and v, where the $u-v$ path in the Kruskal tree is not a shortest $u-v$ path.

Solution.

Although the MST and SPT that I found were the same, there is a second MST that consists of edge $2 <-> 4$ instead of edge $1 <-> 4$, since both those edges are the same weight and allow the tree to consist of all vertices without any loops. Let us call this second MST $MST2$. In $MST2$, the path from 1 to 4 goes through 2, and the weight of that path is 7, while the SPT I found shows that the shortest path from 1 to 4 is 5.

$\therefore$ there exists two vertices u and v where the $u-v$ path in the Kruskal tree is not the shortest $u-v$ path.

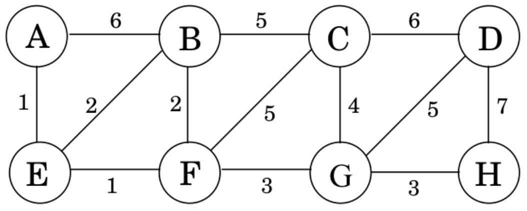
Name: Luke Joyce
ID: 107355873

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 5a (11 points) **Fall 2019, CU-Boulder**

2. (1 pt) Provide a brief description of what the *find(v)* and *union(A,B)* features of the union-find algorithm produce.

   Solution.
   find(v) returns the domain that v is a part of. In Kruskal's algorithm for example, find(v) takes in vertex v as its input, and returns the tree that v is a part of. Union(A,B) takes in two trees as its input and joins them together to make one big tree. In Kruskal's algorithm, when visiting a new edge, $u-v$, find(u) and find(v) are performed, and if they are part of the same tree, Union(A,B) is skipped. If they are part of different trees, Union(A,B) is performed, taking the tree that u is a part of as A, and the tree that v is a part of as B. Those trees are then conjoined to make a larger spanning tree. The point of Union(A,B) in this is to check if adding $u-v$ will form a loop or not.

3. (3 pts) Identify three edges in the following graph G that won't be included in any MST of G. Provide a 3-4 sentence explanation of your answer.



   Solution.
   Three edges that will never be in any MST of G are $D-H$, $D-C$, and $B-C$. $D-H$ will never make the cut because H in the priority queue, $G-D$ and $G-H$ will be considered first. There is no possibility of them forming a loop because $C-D$ will not be considered before them. That being said, since $G-D$ will be added to the list along with $G-C$ before $C-D$, $C-D$ will not be added or else it will form a loop. $B-C$ will not be added because or else it would form a loop with $F-B$, $F-G$, and $G-C$.