Name: Michael Rogers

ID: 105667404

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

Problem Set 4b (45 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: Michael Rogers
ID: 105667404

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 4b (45 points) **Fall 2019, CU-Boulder**

1. (10 pts) For a directed graph with positive weights, we define the max-weight of a path from s to d as the maximum of the edge weights along the path. For example, if the path from A to D has edges and weights of $e_{AB} = 5$, $e_{BC} = 4$, and $e_{CD} = 1$, the length of the path is defined as $e_{AB} + e_{BC} + e_{CD}$, and the max-weight is 5.

   (a) (5 pts) Give an algorithm to compute the smallest max-weight paths from a source vertex s to all other vertices. In this problem, you are changing the definition of length of the path from A to D to $max(e_{AB}, e_{BC}, e_{CD})$ (Hint: Your algorithm should be a modification of Dijkstra's algorithm presented in Lecture.)

```
def shortestPath(G,s):
   dist(s) = 0
   dist(v) = Infinity for all v-s
   max(v) = 0 // Set max in a path from s to v to 0
   Q = Min priority queue
   Q.add(All v in G)
   while Q ! empty:
      u = Q.pop()
      for each v in u.adj:
         d = dist(u) + e(u,v)
         if d > dist(u):
            dist(u) = d
            prev(v) = u
            if e(u,v) > max(d):
               max(d) = e(u,v)
   return G
```

(b) (5 pts) Prove the correctness of your algorithm.

Solution. Proof:

Base Case: Before the algorithm executes, the max edge in a given path is zero given by the line of code max(v) = 0. This means that for and path from s to some v = 0 which makes sense because no paths have been explored. Thus our base case holds.

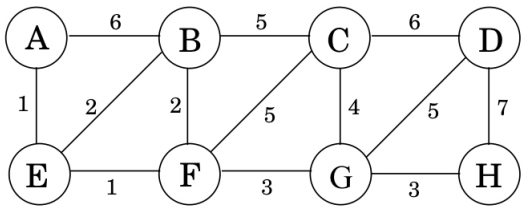
Inductive Hypothesis: Assume that for some path P, at the $1, 2, 3, ..., k^{th}$ iteration of the algorithm, the max(P) = max(d)

Inductive Step: Then we can say at the $k + 1^{th}$ iteration if $\exists\ e(u, v)$ S.T. $e(u, v) > max(d)$ then the max(d) = e(u,v). The algorithm will increase the max value associated with the path at the $k + 1^{th}$ iteration and this will continue until the end of the path is reached. This in turn will result in max(d) holding the max value in some P at the end of the $k + 1^{th}$ iteration.

$\therefore$ shortestPath(G,s) stores the smallest max weight of a given path P

Name: Michael Rogers
ID: 105667404

2. (11 pts) Based on the following graph :



(a) (4 pts) In what order would Prim's algorithm add edges to the MST if we start at vertex A?

Solution. (A,E),(E,F),(E,B),(F,G),(G,H),(G,C),(G,D)

(b) (7 pts) In what order Kruskal's would add the edges to the MST? For each edge added by Kruskal's sequentially, give a cut that justifies it's addition.

Edge Added	Cut Partitions
(A,E)	1= {A,B,C,D}, 2 = {E,F,G,H}
(E,F)	1= {A,E}, 2 = {B,F,C,G,D,H}
(E,B)	1= {A,E,F}, 2 = {B,C,G,D,H}
(F,G)	1= {A,E,B,F}, 2 = {C,G,D,H}
(G,H)	1= {A,E,F,B,C,G}, 2 = {D,H}
(G,C)	1= {A,B,E,F,G}, 2 = {C,D,H}
(G,D)	1= {A,B,C,E,F,G,H}, 2 = {D}

3. (10 pts) Let T be a MST of a given graph G. Will T still be the MST if we reduce the weight of exactly one of the edges in T by a constant **c**? Prove your answer.

   Solution.

   $$T \subset G$$

   By definition, a MST is a spanning tree with minimum weight that contains all verticies in G, where G=(E,V). Since T = min(G), that implies that

   $$T \leq T' \Rightarrow T = T - e + e' \leq T'$$

   where T' is any other spanning tree $\subset$ G, e' is some edge ce

   Since $e' = ce$ and c reduces e by a factor of c then it must be the case that $c < 1$ for all $e \geq 1 \Rightarrow e' = \frac{e}{c}$

   This would cause T to reduce in overall weight. Since $T \leq T'$, and $e' \leq e$ then it must be the case that $T - e + e \leq T - e + e'$

   $$\therefore T \leq T'$$

   And our hypothesis holds that T must still be the MST of G.

4. (14 pts) One of the uses of MSTs is finding a set of edges that span a network for minimum cost. Network problems could also have another type of objective: designing a spanning tree for which the most expensive edge is minimized. Specifically, let $G = (V, E)$ be a connected graph with n vertices, m edges, and positive edge costs that you may assume are all distinct. Let $T = (V, E)$ be a spanning tree of G; we define the **limiting edge** of T to be the edge of T with the greatest cost. A spanning tree T of G is a minimum-limiting spanning tree if there is no spanning tree T of G with a cheaper limiting edge.

   (a) (7 pts) Is every minimum-limiting tree of G an MST of G? Prove or give a counterexample.

   Solution. Every minimum limiting spanning tree is not necissarily a MST.

   Spose' we have 2 spanning trees of G with edges of weights $S_1 = \{10, 8, 9, 7\}$ and $S_2 = \{11, 3, 2, 1\}$

   The overall weight of S_1 is 34 while the overall weight of S_2 is 17. The limiting edge of S_1 is 10 which is cheaper than S_2's 11, but S_2's overall weight is less, so it would not be possible that the minimum limiting spanning tree is the MST of the graph.

(b) (7 pts) Prove that every MST of G is a minimum-limiting tree of G. [**Hint:** Let T be an MST of G, and let T' be a minimum-limiting tree of G. If T is not a minimum-limiting tree, can we replace the heaviest edge of T? Think about how to use T' here.]

Solution. Proof:

Exchange Argument:

Spose' T is an MST of G and T' is a minimum limiting tree of G. Say there is some $e \in T$ and $e' \in T'$ If T is not a minimum limiting tree, then the largest values in T and T' respectivley must be $e > e'$

$T = \{e_1, e_2, ..., e\}, T' = \{e_1, e_2, ..., e'\}$, Where e and e' are the limiting values of T and T' respectivley. If $T = \{e_1, e_2, ..., e'\}, T' = \{e_1, e_2, ..., e\}$ then e will either create a cycle in the MST or contradict a cut made by kruskals in finding the MST.

In order for our MST to be a MLT, $\exists e \in T$ S.T. $e \leq e'$ If we exchange e for e', then we see that T is still a MST, and also an MLT. But in order for T to be an MST, the overall weight of the tree $T \leq T'$. Unless $e' \in T$ this inequality is not possible. Therefore $e = e'$ and T must be a MST as well as a MLT of G.

Name: Michael Rogers

ID: 105667404

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

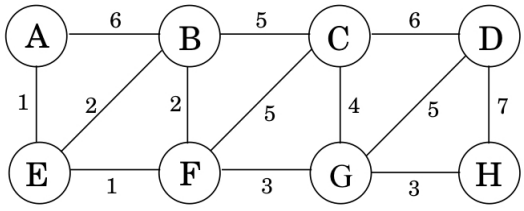
Problem Set 4b (45 points) **Fall 2019, CU-Boulder**

Ungraded questions - These questions are for your practice. We won't grade them or provide a typed solution but we are open to discuss these in our OHs and you should take feed backs on your approach during the OHs. These questions are part of the syllabus.

1. Suppose you are given the minimum spanning tree T of a given graph G (with n vertices and m edges) and a new edge $e = (u, v)$ of weight w that will be added to G. Give an efficient algorithm to find the MST of the graph $G \cup e$, and prove its correctness. Your algorithm should run in $O(n)$ time.

   Solution.

2. Based on the following graph :



   (a) Run Kruskal's and find the MST. You can break the ties however you want. Draw the MST that you found and also find it's total weight.

   Solution.

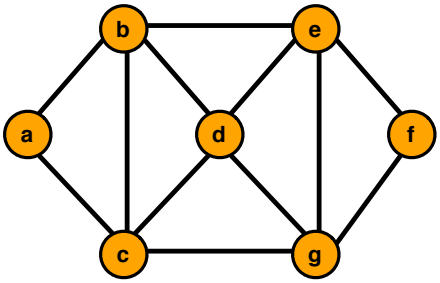
(b) Run Prim's starting from vertex A and find the MST. You can break the ties however you want. Draw the MST that you found and also find it's total weight. Is the total weight same as what you get from the above?

Solution.

3. Consider the following unweighted graph, and assign the edge weights (using positive integer weights only), such that the following conditions are true regarding minimum spanning trees (MST) and single-source shortest path (SSSP) trees:

   - The MST is distinct from any of the seven SSSP trees.
   - The order in which Prim's algorithm adds the safe edges is different from the order in which Kruskal's algorithm adds them.

   Justify your solution by (i) giving the edges weights, (ii) showing the corresponding MST and all the SSSP trees, and (iii) giving the order in which edges are added by each of the three algorithms.



Solution.