Name: Nikolaas Bender
ID: 106977096

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: Nikolaas Bender
ID: 106977096

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.)

Solution.

```
max dist = 0;
$dist(s) = 0$;
$dist(v) = \infty$;
$Q$ = min priority queue;
Q.add(v);
while Q !empty do
    u = Q.pop();
    for each V in u.adj do
        d = dist(u) + $e_{u,v}$;
        if $d <$ dist(v) then
            dist(v) = d;
            if $e_{current,parent} >$ max dist then
                max dist = d;
            end
            prev(v) = u;
        end
    end
end
return max dist;
```

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

Solution. Assume Dijkstra's algorithm is correct.

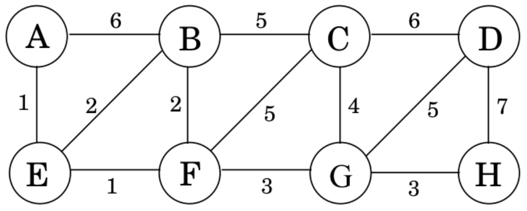
Loop invariant:
max dist stores the largest edge along the path found through dijkstra's algorithm

init:
max dist = 0

maintenance:
compare parent edge to max dist because parent edge isn't changing

termination:
return max dist

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?

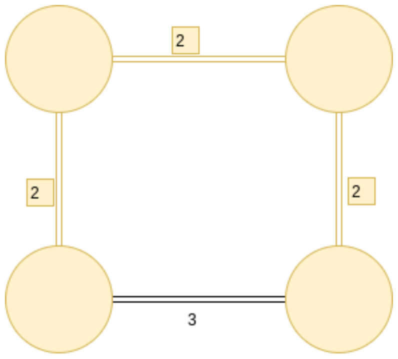
Solution. In this order doing a breadth first search with Prim's
$A \to E$
$E \to F$
$F \to B$
$F \to G$
$G \to H$
$G \to C$
$G \to D$

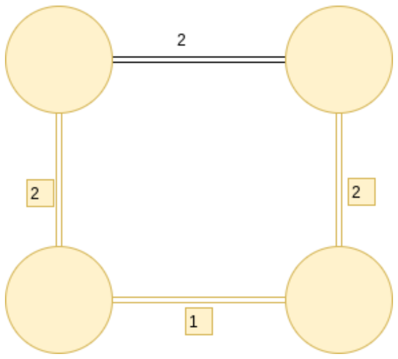
(b) (7 pts) In what order would Kruskal's add the edges to the MST? For each edge added by Kruskal's sequentially, give a cut that justifies it's addition.

Solution. $A \to E$ one of the least weighty edges
$E \to F$ the other least weighty edge connects F to graph
$F \to B$ connects B to the rest of the graph
$F \to G$ connects G to the rest of the graph
$G \to H$ connects H to the rest of the graph
$G \to C$ connects C to the rest of the graph
$G \to D$ connects D to the rest of the graph

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 G by a constant $\mathbf{c}$? Prove your answer.

Solution. No, because G can change the mst might change
see this example





Name: $\boxed{\text{Nikolaas Bender}}$
ID: $\boxed{106977096}$

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

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. They are the same because both are found through the same algorithm. They both use a minimum priority queue and stop when a tree is formed that spans to all vertices of the graph.
   Because the algorithm goes through the edges in ascending order and stops on the heaviest edge that connects the graph.
   As long as the metric for choosing edges is the same for when finding the minimum limiting tree and the minimum spanning tree then the two will be the same. The only caveat is that there might be more than MLTs and MLTs for a graph G however the set of MLTs and MSTs will be the same.

(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. If we are able to replace the heaviest edge of T with and edge from T' then T was not an MST of G and now $T' = T$.

Name: Nikolaas Bender
ID: 106977096

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 :

   PS4/mst_graph_q2.jpg

   (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.

   `graph_mst.pdf`

   Solution.