Name: Alex Book

ID: 108073300

Collaboration: Varun Narayanswamy, Michael Hasenkamp, Jamie Foster, Irvin Carbajal

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: Alex Book
ID: 108073300
Collaboration: Varun Narayanswamy, Michael Hasenkamp, Jamie Foster, Irvin Carbajal

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.

```
DijkstrasModified(G,S) //G is a graph with edges and vertices G=(V,E), S is the source vertex
  for each V in G: //set all vertices to have maximum possible distance/cost, no parent node, and minimum value for maximum edge length in path
    dist(V)=$\infty$
    prev(V)=null
    maxEdge(V)=0
  dist(s)=0
  for V in G:
    Q.add(V) //add all vertices to Q, a minimum priority queue of vertices
  while Q ! empty:
    u=Q.pop()
    for each v in u.adjacent:
      d=dist(u)+$E_{u,v}$
      if d <dist(v):
        maxEdge(v)=max(maxEdge(u),$E_{u,v}$)
        dist(v)=d
        prev(v)=u
  return G //return the graph with all vertices having minimum distance/cost, appropriate parent node, and correct value for maximum edge length in path
```

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

Solution.
Proof by induction:
For set S with starting value S_0, for each vertex u $\in$ S, dist(u) is the shortest path from S_0 to u and maxEdge(u) is the maximum edge weight along that path.

Base Case: |S|=1
Since we have S=S_0, P_{S_0}=dist(S_0)=0, and M_{S_0}=maxEdge(S_0)=0, we have the shortest possible path to S_0 and the maximum edge weight along that path. Base case holds.

Inductive Hypothesis:
Assume the claim holds for |S|=k, so there are k solved vertices in S for k$\geq$1.

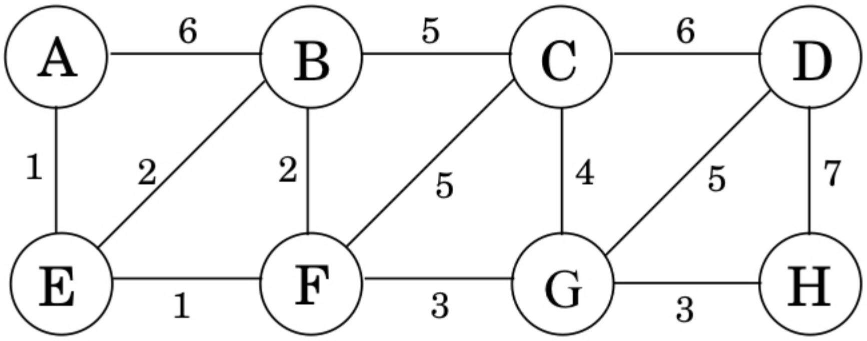
Inductive Step:
Grow |S| to k+1 by adding vertex v. Let u represent the parent vertex of v (so u $\in$ S, as only solved vertices have had their adjacent vertices' set to non-∞). Let dist$'$(v)=dist(u)+$E_{u,v}$ (because this is not necessarily the minimum possible path to v, but can be seen as the prospective minimum weight path to v). If the weight of $E_{u,v}$ is greater than the heaviest weighted edge in the path to u (maxEdge(u)), then maxEdge(v) is to be set equal to the weight of $E_{u,v}$. Otherwise, maxEdge(v) is to be set to equal to maxEdge(u), as v then shares the same heaviest weighted edge as its parent. To finish this proof by induction (and show that maxEdge(v) was set correctly), we must prove that dist$'$(v) is the minimum possible path to v.

Proof by contradiction:
Assume that dist$'$(v) is not the minimum path from S_0 to v. Consider another path P from S_0 to v. This path P must have a weight less than dist$'$(v). Let $E_{a,b}$ be another edge leaving S (where a$\in$S and b$\notin$S), the first edge beyond vertex a that makes up P.
weight(P) $\geq$ dist(a)+$E_{a,b}$ (because all edge weights are positive)
dist(a)+$E_{a,b}$ $\geq$ dist$'$(b) (because this is not necessarily the minimum possible path to b, but can be seen as the prospective minimum weight path to b)
dist$'$(b) $\geq$ dist$'$(v) (as the algorithm chose to add v to S, not b, so v must have been at the front of the minimum priority queue).
Piecing together these three inequalities, we get that weight(P) $\geq$ dist$'$(v). Therefore, dist$'$(v) must be the minimum possible path from S_0 to v and maxEdge(v) must store the maximum edge weight along that path, so there are k+l solved vertices for |S|=k+1. $\square$

Name: Alex Book
ID: 108073300
Collaboration: Varun Narayanswamy, Michael Hasenkamp, Jamie Foster, Irvin Carbajal

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

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.
   Note: listed 'adjacent' vertices are also still in the minimum priority queue.

   root=A
   A: cost=0
   adjacent:
   B: cost=6 ($6<\infty$, so update cost and parent=A)
   E: cost=1 ($1<\infty$, so update cost and parent=A)

   u=E
   adjacent:
   B: cost=2 ($2<6$, so update cost and parent=E)
   F: cost=1 ($1<\infty$, so update cost and parent=E)

   u=F
   adjacent:
   B: cost=2 ($2=2$, so don't update cost and parent still=E)
   C: cost=5 ($5<\infty$, so update cost and parent=F)
   G: cost=3 ($3<\infty$, so update cost and parent=F)

u=B
adjacent:
C: cost=5 (5=5, so don't update cost and parent still=F)

u=G
adjacent:
C: cost=4 ($4<5$, so update cost and parent=G)
D: cost=5 ($5<\infty$, so update cost and parent=G)
H: cost=3 ($3<\infty$, so update cost and parent=G)

u=H
adjacent:
D: cost=7 ($7>5$, so don't update cost and parent still=G)

u=C
adjacent:
D: cost=6 ($6>5$, so don't update cost and parent still=G)

u=D
There are no adjacent vertices that are in the minimum priority queue

The queue of vertices is now empty, so the algorithm returns the MST. The edges were added in the following order:
A-E
E-B
E-F
F-G
G-C
G-D
G-H

(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.

Solution.
Note: The sets I use in my cut justifications are divided by whether or not the vertices (and the edges connecting them) are in the MST already, as confirmed by Rhonda in 9/23 office hours.

Since the lowest weight edges are A-E and E-F, so the algorithm should add A-E, breaking the tie by choosing the left-most option. I decide arbitrarily to use the sets (A) and (E,B,C,D,F,G,H) to perform a cut. The edge options are A-B with a weight of 6, or A-E with a weight of 1. Therefore I add edge A-E to the MST and remove it from the queue.

The next lowest weight edge is E-F, which the algorithm should add. I perform a cut using the sets (A,E) and (B,C,D,F,G,H). The edge options are A-B (weight of 6), E-B (weight of 2), and E-F (weight of 1). Therefore I add edge E-F to the MST and remove it from the queue.

The next lowest weight edges are E-B and F-B, so the algorithm should add E-B, breaking the tie by choosing the left-most option. I perform a cut using the sets (A,E,F) and (B,C,D,G,H). The edge options are A-B (weight of 6), E-B (weight of 2), F-B (weight of 2), F-C (weight of 5), and F-G (weight of 3). Therefore I add edge E-B to the MST (breaking the tie by choosing the left-most option) and remove it from the queue.

Edge F-B would form a cycle, so it is not considered for the MST and removed from the queue.

The next lowest weight edges are F-G and G-H, so the algorithm should add F-G, breaking the tie by choosing the left-most option. I perform a cut using the sets (A,E,F,B) and (C,D,G,H). The edge options are B-C (weight of 5), F-C (weight of 5), and F-G (weight of 3). Therefore I add edge F-G to the MST and remove it from the queue.

The next lowest weight edge is G-H. I perform a cut using the sets (A,E,F,B,G) and (C,D,H). The edge options are B-C (weight of 5), F-C (weight of 5), G-C (weight of 4), G-D (weight of 5), and G-H (weight of 3). Therefore I add edge G-H to the MST and remove it from the queue.

The next lowest weight edge is G-C. I perform a cut using the sets (A,E,F,B,G,H) and (C,D). The edge options are B-C (weight of 5), F-C (weight of 5), G-C (weight of 4), G-D (weight of 5), and H-D (weight of 7). Therefore I add edge G-C to the MST and remove it from the queue.

Edge B-C would form a cycle, so it is not considered for the MST and removed from the queue.

Edge F-C would form a cycle, so it is not considered for the MST and removed from the queue.

The next lowest weight edge is G-D. I perform a cut using the sets (A,E,F,B,G,H,C) and (D). The edge options are C-D (weight of 6), G-D (weight of 5), and H-D (weight of 7). Therefore I add edge G-D to the MST and remove it from the queue.

Edge A-B would form a cycle, so it is not considered for the MST and removed from the queue.

Edge C-D would form a cycle, so it is not considered for the MST and removed from the queue.

Edge H-D would form a cycle, so it is not considered for the MST and removed from the queue.

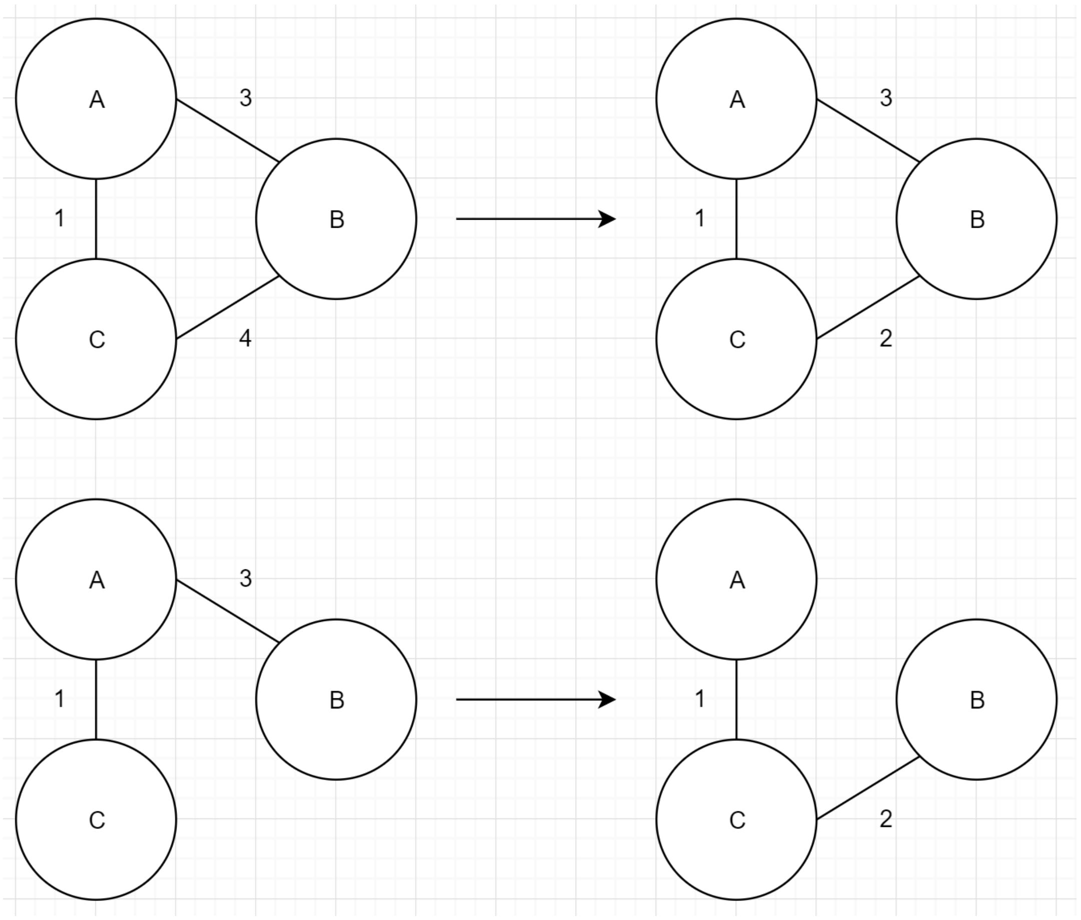
The queue of edges is now empty, so the algorithm returns the MST. The edges were added in the following order:
A-E
E-F
E-B
F-G
G-H
G-C
G-D

Name: Alex Book
ID: 108073300
Collaboration: Varun Narayanswamy, Michael Hasenkamp, Jamie Foster, Irvin Carbajal

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

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.
Counterexample:
In the following example, the MST for the given graph includes edges A-C and A-B. After decreasing edge C-B by some constant c (in this case, c=2), the MST changes, now including edges A-C and C-B. Therefore, reducing one of the edges in a graph by a constant c will not necessarily keep the same MST (dependent upon the value of c in relation to the edge weights in the graph).



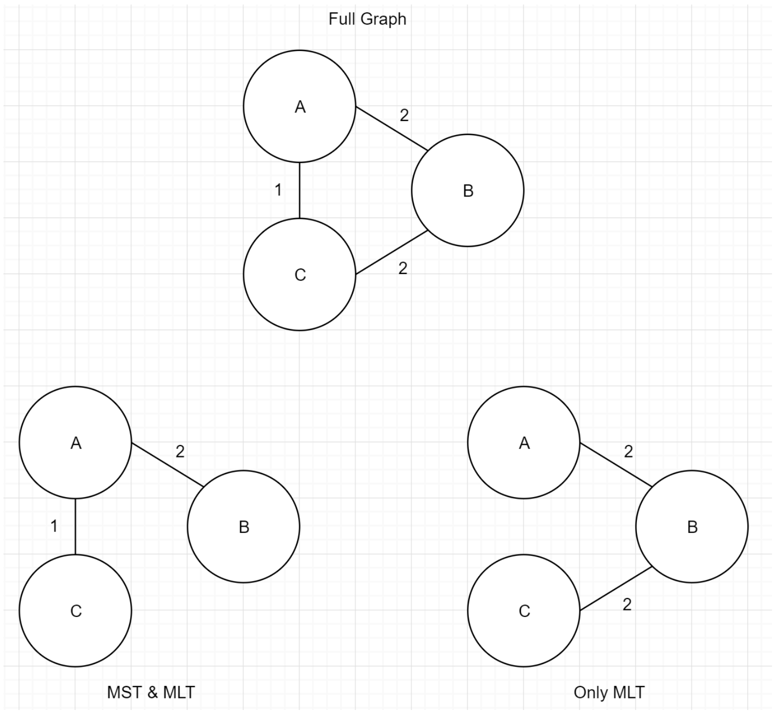
Name: Alex Book
ID: 108073300
Collaboration: Varun Narayanswamy, Michael Hasenkamp, Jamie Foster, Irvin Carbajal

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.
   Counterexample:
   In the following example, the top graphic is the original tree, the bottom left graphic is both an MST and MLT of the graph, and the bottom right graphic is an MLT but not an MST of the graph. Therefore, not every MLT is an MST.



Name: Alex Book
ID: 108073300
Collaboration: Varun Narayanswamy, Michael Hasenkamp, Jamie Foster, Irvin Carbajal

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

(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 by contradiction:
Let spanning tree T be an MST, spanning tree T' be an MLT, edge e be the heaviest edge in T, and edge e' be the heaviest edge in T'. Assume T is not an MLT, which means that $e>e'$. So, you should be able to replace e with e' (the minimum limiting edge), lowering the total cost for T. However, since T is an MST, it already has the minimum total cost of any spanning tree, and its cost cannot be lowered. Therefore, by contradiction, T must be an MLT. Every MST of a given graph is also an MLT of that graph.

Name: Alex Book
ID: 108073300
Collaboration: Varun Narayanswamy, Michael Hasenkamp, Jamie Foster, Irvin Carbajal

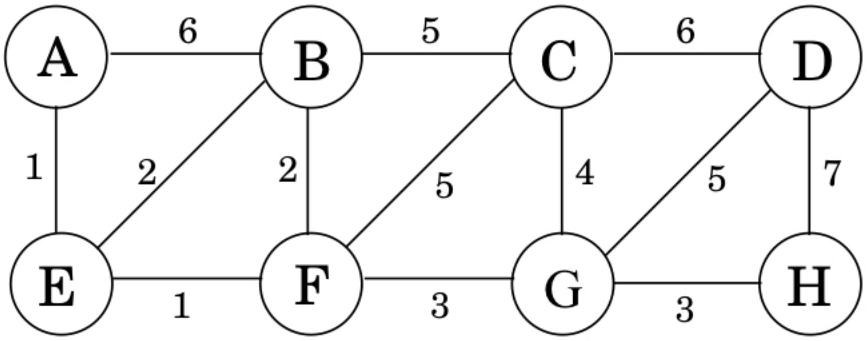
CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 4b (45 points) **Fall 2019, CU-Boulder**

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.
   text

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.
   text

Name: Alex Book
ID: 108073300
Collaboration: Varun Narayanswamy, Michael Hasenkamp, Jamie Foster, Irvin Carbajal

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

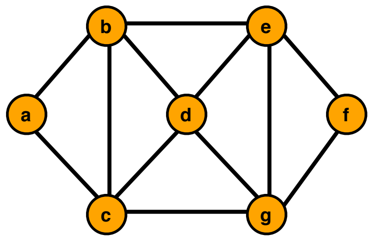
(b) Run Prim's starting from vertex A and find the MST. You can break the ties however you want. Draw the MST that you found and also find it's total weight. Is the total weight same as what you get from the above?

Solution.
text

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.
text