CSCI 3104 Algorithms Name: Connor O'Reilly
Problem Set 7
Fall 2020, CU Boulder ID: 107054811

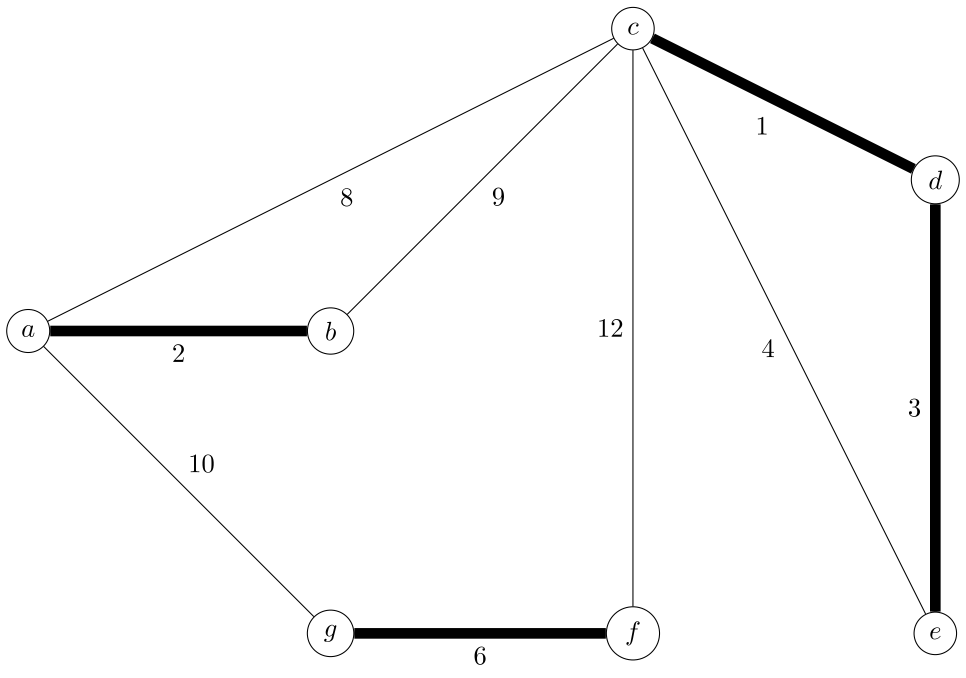
Instructions

- This problem set is **open book**: you may refer to the lectured material found on Canvas and the recommended books to help you answer the questions.
- This problem set is an **individual effort**. You must arrive at your answers independently and write them up in your own words. Your solutions should reflect your understanding of the content.
- **Posting questions to message boards or tutoring services including, but not limited to, Chegg, StackExchange, etc., is STRICTLY PROHIBITED. Doing so is a violation of the Honor Code.**
- Your solutions must be submitted typed in LaTeX, **handwritten work is not accepted**. If you want to include a diagram then we do accept photos or scans of hand-drawn diagrams included with an appropriate `\includegraphics` command. It is your responsibility to ensure that the photos you obtain are in a format that pdflatex understands, such as JPEG.
- The template tex file has carefully placed comments (% symbols) to help you find where to insert your answers. There is also a **STUDENT DATA** section in which you should input your name and ID, this will remove the warnings in the footer about commands which have not been edited. You may have to add additional packages to the preamble if you use advanced LaTeX constructs.
- You must CITE any outside sources you use, including websites and other people with whom you have collaborated. You do not need to cite a CA, TA, or course instructor.
- Take care with time, we do not usually accept problem sets submitted late.
- Take care to upload the correct pdf with the correct images inserted in the correct places (if applicable).
- Check your pdf before upload.
- **Check your pdf before upload.**
- **CHECK YOUR PDF BEFORE UPLOAD.**

Quicklinks: 1 2 (2a) (2b)

Problem 1

To solve this problem, consider the undirected, weighted graph G shown below. The wide edges represent (the edges of) an intermediate spanning forest F, obtained by running the generic minimum spanning tree algorithm in *Week7.pdf Algorithm 1.* The intermediate spanning forest F has three components: $\{a, b\}$, $\{g, f\}$ and $\{c, d, e\}$. For each non-wide edge in the graph, determine whether the edge is safe, useless or undecided.



Considering the forest with components $\{a, b\}$, $\{g, f\}$ and $\{c, d, e\}$ we can initially determine that edge (c, e) with $w(c, e) = 4$. Both endpoints c and e are both part of the same component $\{c, d, e\}$ and would create a cycle. Now consider component $\{g, f\}$, there are only two edges which connect this component with other components of the ISF these are edges (a, g) and (c, f). Endpoints of both edges do not share the same components and therefore cannot be considered useless. Edge (a, g) has the smallest weight of all edges from components $\{a, b\}$ and $\{c, d, e\}$ so we can declare this edge as a safe edge. Following there are three edges that connect component $\{a, b\}$ to other components. These edges are $(a, c), (b, c)$ and (a, g). All edges contain endpoints that do not share components therefore they cannot be useless, edge (a, c) has the smallest weight with $w(a, c) = 8$ and therefore can be considered safe. Finally we are left with edges (b, c) and (f, c). As stated earlier both edges do not have endpoints which share the same component, and they are not the minimum weight edge so they cannot be the safe so we determine that edges (b, c) and (f, c) are undecided.

Problem 2

Your work for Problem 2 will be divided up into parts (a) and (b) on the following pages. On this page we offer some useful insight and context. The goal of this problem is to construct an example of a weighted graph $G = (V, E, w)$ where there exists a source vertex s such that Prim's algorithm (starting at s) adds the edges of the MST in a different order than Kruskal's algorithm.

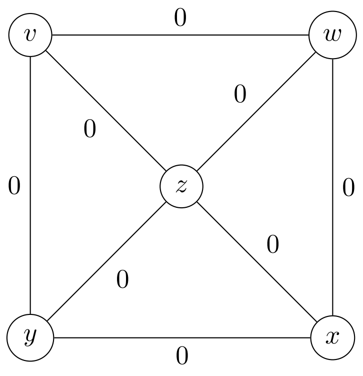
To complete the first part of this problem, we must identify how each algorithm adds edges to the intermediate spanning forest during execution. That is, we must identify how the edges of the intermediate spanning forests are stored.

Recall that both Prim's and Kruskal's algorithms build an intermediate spanning forest edge-by-edge until it is connected. The initial intermediate spanning forest in both algorithms is a collection of isolated vertices. The the two algorithms differ from each other in deciding which edge to add at each step. Prim's algorithm starts with a source vertex s and always adds a safe edge leaving the component of the current intermediate spanning forest that contains s. Kruskal's algorithm always adds the minimum weight safe edge in the current intermediate spanning forest. We often say that an edge is "added to the MST" when it is added to the intermediate spanning forest because the intermediate spanning forest is a subgraph of the MST.

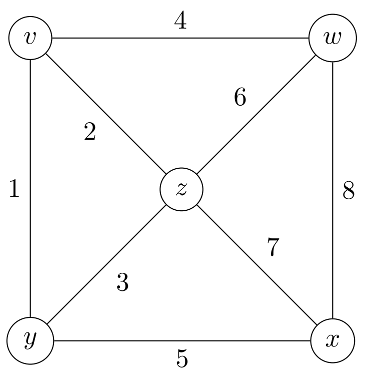
In Kruskal's algorithm it is pretty clear when an edge is added to the intermediate spanning forests. Observe that we actually add edges to a variable F which stores the edges of the intermediate spanning forest as it grows.

In Prim's algorithm, the story isn't as simple. Observe that a vertex $x \in V$ is added to the connected component which contains the source vertex s when it is removed from the queue Q. The only exception is the source vertex itself which is already in the appropriate component to begin with. Recall that in class and the notes, we argued that each vertex is removed from the queue at most once. To add a vertex x to the component in the current intermediate spanning forest, we must add an appropriate safe edge to the intermediate spanning forest. The way we keep track of the edge added is by storing it in $P[x]$. More specifically, we assume that once a vertex is added to the component containing s (i.e. removed from the queue) then $(P[x], x)$ is added to our intermediate spanning forest. Observe that we will never update $P[x]$ after vertex x is removed from the queue. At any step of the algorithm, we can retrieve the edges in our current intermediate spanning forest by looking at all entries of P that correspond to vertices already removed from the queue. Note, the source vertex started in the correct connected to begin and so we don't need an edge to add it at any step. However, one can observe that $P[s] = NULL$ throughout the algorithm. This makes sense considering an MST has exactly $|V| - 1$ edges.

In parts (2a) and (2b), using the graph pictured below, assign the weights 1, 2, 3, 4, 5, 6, 7, 8 to the edges of the graph such that every edge gets a different weight. You will need to copy this diagram for both parts (2a) and (2b). **Note:** Since the edge weights are all distinct, there is only one minimum spanning tree for the graph. So you should get the same minimum spanning tree in both (2a) and (2b). The only difference is that the edges should be added in different orders.

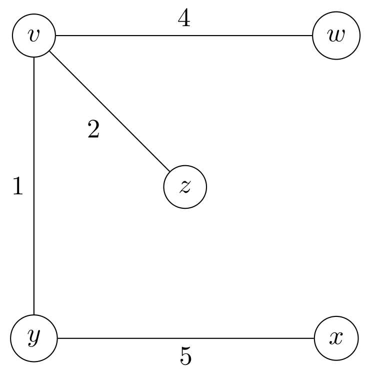

v
w
z
y
x
0
0
0
0
0
0
0
0

(2a) Adjust the figure below to display your edge weights, replacing 0 with digits of your choice. Then use Prim's algorithm *as defined in Week7.pdf* to compute the MST. For your answer you must give a starting vertex and the order in which Prim's algorithm adds the edges to the MST (that is, adds an edge to the current intermediate spanning forest that will eventually be contained in the MST). **For each edge added to the MST, clearly indicate both the edge and its weight.**

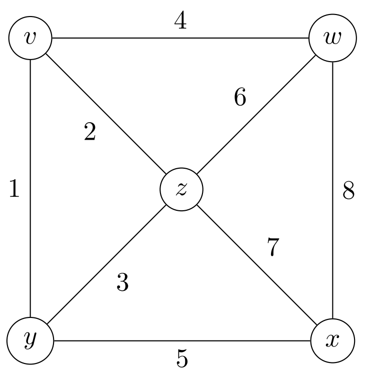


```
Starting vertex: v
    First:  edge (v,y)  w(v,y) = 1
    Second: edge (v,z)  w(v,z) = 2
    Third:  edge (v,w)  w(v,w) = 4
    Fourth: edge (y,x)  w(y,x) = 5

            Image of MST located below
```



(2b) Adjust the figure below to display your edge weights *which must be the same as for (2a)*, replacing 0 with digits of your choice. Then use Kruskal's algorithm *as defined in Week7.pdf* to compute the MST. For your answer you must give the order in which Kruskal's algorithm adds the edges to the MST. **For each edge added to the MST, clearly indicate both the edge and its weight.** You do not need to specify the state of the algorithm during its execution (i.e. we do not need to see the state of the disjoint sets data structure in your answer).



```
Turned out to be the same order as Prim's
        First:  edge (v,y)  w(v,y) = 1
        Second: edge (v,z)  w(v,z) = 2
        Third:  edge (v,w)  w(v,w) = 4
        Fourth: edge (y,x)  w(y,x) = 5

                Image of MST located below
```