

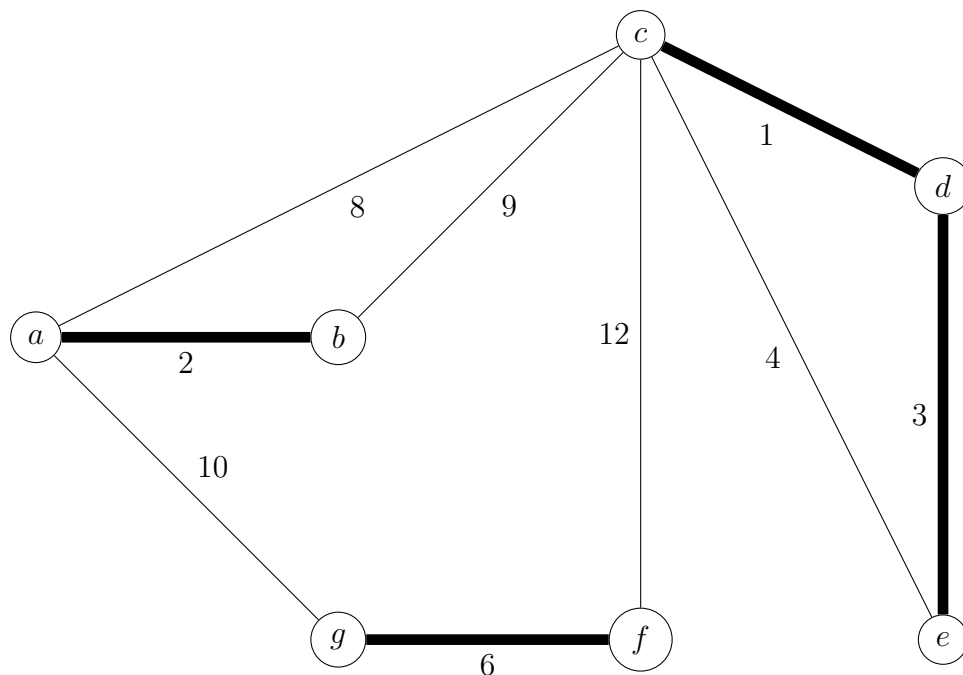
Instructions

- 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 pdf_latex 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.



Edge ce is useless because it connects two vertices in component $\{c, d, e\}$, creating a cycle.

Edge ac is safe because it is the minimum weight edge that connects components $\{a, b\}$ and $\{c, d, e\}$ without creating a cycle.

Edge ag is safe because it is the minimum weight edge that connects components $\{a, b\}$ and $\{g, f\}$ without creating a cycle.

Edges bc and cf are undecided. They aren't useless because they don't connect two

vertices in the same component (i.e. they don't create a cycle), but they also aren't safe because they aren't the minimum weight edges that connect the components.

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.

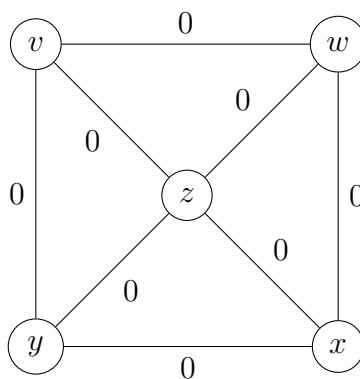
To complete the first part of this problem, we must identify how each algorithm adds edges to the intermediate spanning forest during execution. That is, we must identify how the edges of the intermediate spanning forests are stored.

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 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.

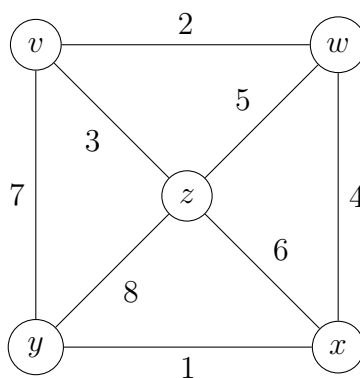
In Kruskal's algorithm it is pretty clear when an edge is added to the intermediate spanning forests. Observe that we actually add edges to a variable F which stores the edges of the intermediate spanning forest as it grows.

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 component to begin and so we don't need an edge to add it at any step. However, one can observe that $P[s] = \text{NULL}$ 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.



- (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.**



Let Q be a minimum priority queue, P be the set of edges in the MST found by Prim's algorithm, and the starting vertex be v . Then:

Start:

$$P = \{\}$$

$$Q = \{(v, 0), (z, \infty), (y, \infty), (w, \infty), (x, \infty)\}$$

Step 1:

v is the starting vertex and first in the queue with weight zero, so it is added to set P . We then update our queue with the weights with vertices connecting to component $\{(NULL, v)\}$

$$P = \{(NULL, v)\}$$

$$Q = \{(vw, 2), (vz, 3), (vy, 7), (x, \infty)\}$$

Step 2:

vw is the next edge to be added to P , as it is the the first element in our priority queue and is safe. x is updated as weight 4 from w ,

$$P = \{(NULL, v), (vw)\}$$

$$Q = \{(vz, 3), (wx, 4), (vy, 7)\}$$

Step 3:

vz is the next edge to be added to P , as it is the first element in our priority queue and is safe.

$$P = \{(NULL, v), (vw), (vz)\}$$

$$Q = \{(wx, 4), (vy, 7)\}$$

Step 4:

wx is the next edge to be added to P , as it is the first element in our priority queue and is safe. y is updated as weight 1 from x .

$$P = \{(NULL, v), (vw), (vz), (wx)\}$$

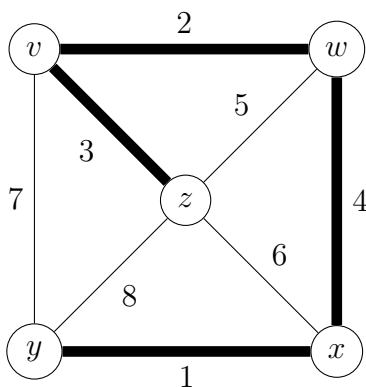
$$Q = \{(xy, 1)\}$$

Step 5:

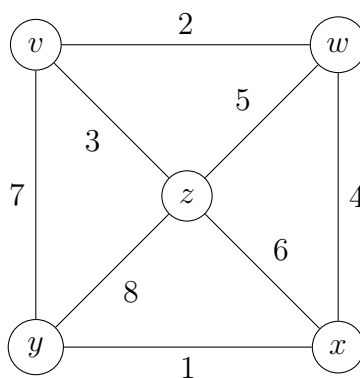
xy is the last edge to be added and our final MST is

$$P = \{(NULL, v), (vw), (vz), (wx), (xy)\}$$

$$Q = \{\}$$



- (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).



Let E be the set of edges in G , sorted in ascending order of weight, and let F be the set of edges in the MST found by Kruskal's algorithm.

Then we have:

$$F = \{\}$$

$$E = \{(xy, 1), (vw, 2), (vz, 3), (wx, 4), (wz, 5), (xz, 6), (vy, 7), (yz, 8)\}$$

Step 1:

xy is safe and the lowest weight edge, so we add component $\{x, y\}$ to F . So we have:

$$F = \{\{x, y\}\}$$

$$E = \{(vw, 2), (vz, 3), (wx, 4), (wz, 5), (xz, 6), (vy, 7), (yz, 8)\}$$

Step 2:

vw is safe and the lowest weight edge, so we add component $\{v, w\}$ to F . So we have:

$$F = \{\{x, y\}, \{v, w\}\}$$
$$E = \{(vz, 3), (wx, 4), (wz, 5), (xz, 6), (vy, 7), (yz, 8)\}$$

Step 3:

vz is safe and the lowest weight edge. vz connects to component $\{v, w\}$, so $\{v, w\} = \{v, w\} \cup \{z\}$

$$F = \{\{x, y\}, \{v, w, z\}\}$$
$$E = \{(wx, 4), (wz, 5), (xz, 6), (vy, 7), (yz, 8)\}$$

Step 4:

wx is safe and the lowest weight edge. wx connects the components $\{x, y\}$ and $\{v, w, z\}$, so this becomes $\{x, y\} \cup \{v, w, z\}$

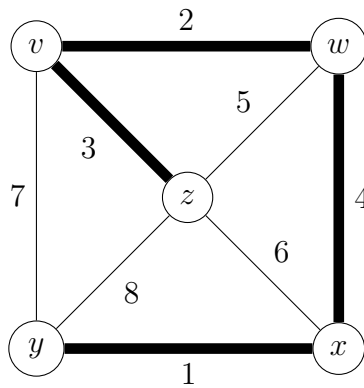
$$F = \{\{x, y, v, w, z\}\}$$
$$E = \{(wz, 5), (xz, 6), (vy, 7), (yz, 8)\}$$

Step 5:

The rest of the edges in E are useless because they connect two vertices in component $\{x, y, v, w, z\}$, creating a cycle. This must be the case as all vertices in graph G are included as a single component in set F , so we are done.

Note the algorithm as defined in Week7.pdf would check the rest of the edges one-by-one and find edge wz is useless because it connects two vertices in the same component, edge xz is useless because it connects two vertices in the same component, and so on.

The final MST is:



This is the same MST found by Prim's algorithm in part 2(a), however we note the order in which the edges are added differ between the algorithms, as desired.