CSCI 3104 Spring 2022
Instructor: Profs. Chen and Layer

Problem Set 3

Due Date ..... **February 8, 2022**
Name ..... **Chengming Li**
Student ID **109251991**
Collaborators ..... **N/A**

Contents

1 Instructions

- The solutions **must be typed**, using proper mathematical notation. We cannot accept hand-written solutions. Here's a short intro to LaTeX.
- You should submit your work through the **class Canvas page** only. Please submit one PDF file, compiled using this LaTeX template.
- You may not need a full page for your solutions; pagebreaks are there to help Gradescope automatically find where each problem is. Even if you do not attempt every problem, please submit this document with no fewer pages than the blank template (or Gradescope has issues with it).
- You are welcome and encouraged to collaborate with your classmates, as well as consult outside resources. You must **cite your sources in this document. Copying from any source is an Honor Code violation. Furthermore, all submissions must be in your own words and reflect your understanding of the material.** If there is any confusion about this policy, it is your responsibility to clarify before the due date.
- Posting to **any** service including, but not limited to Chegg, Reddit, StackExchange, etc., for help on an assignment is a violation of the Honor Code.
- You **must** virtually sign the Honor Code (see Section 2). Failure to do so will result in your assignment not being graded.

2 Honor Code (Make Sure to Virtually Sign)

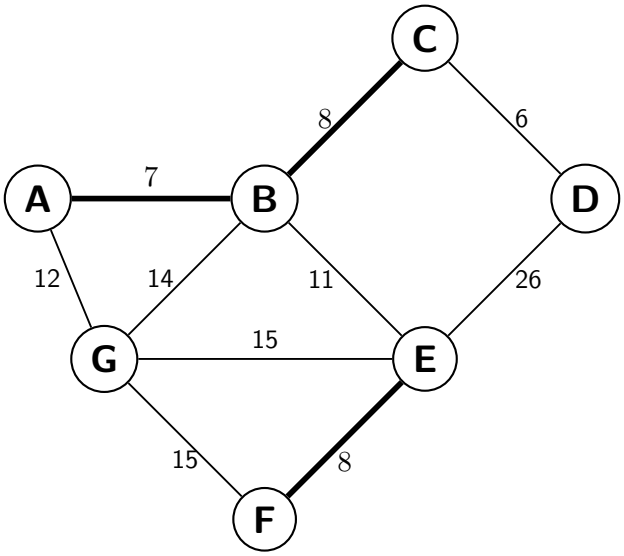
Problem 1.

- My submission is in my own words and reflects my understanding of the material.
- Any collaborations and external sources have been clearly cited in this document.
- I have not posted to external services including, but not limited to Chegg, Reddit, StackExchange, etc.
- I have neither copied nor provided others solutions they can copy.

Agreed (signature here). I agree to the above, Chengming Li □

3 Standard 7 - MST: safe and useless edges

Problem 2. Consider the weighted graph $G(V, E, w)$ below. Let $\mathcal{F} = \{\{A, B\}, \{B, C\}, \{E, F\}\}$ be an intermediate spanning forest (indicated by the thick edges below). Label each edge that is **not** in $\mathcal{F}$ as safe, useless, or undecided. Provide a 1-2 sentence explanation for each such edge.



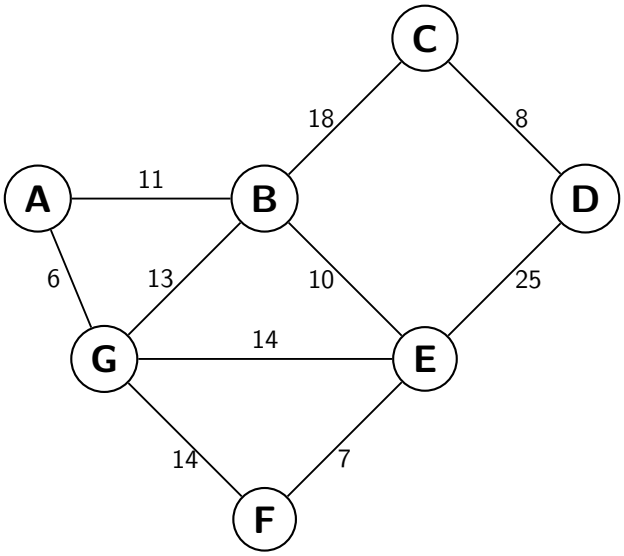
Answer.

- we have two component in an intermediate spanning forest. {A,B,C} ,{E,F}
- {A,G} is **safe** edge. Beacause for all the edge leaving component {G}, edge{A,G}, edge{B,G}, edge{E,G} and edge{F,G}, {A,G} has minimum weight among these edges. As {A,G} is a light edge with exactly one end point belonging to {G}, we have by Corollary 61 that {A,G} is a **safe** edge with respect to $\mathcal{F}$
- {B,E} is **safe** edge. Beacause for all the edge leaving component {E,F}, edge{F,G}, edge{B,E}, edge{E,G} and edge{D,E}, {B,E} has minimum weight among these edges. As {B,E} is a light edge with exactly one end point belonging to {E,F}, we have by Corollary 61 that {B,E} is a **safe** edge with respect to $\mathcal{F}$
- {C,D} is **safe** edge. Beacause for all the edge leaving component {A,B,C}, edge {A,G},edge{B,G}, edge{B,E} and edge{C,D}, {C,D} has minimum weight among these edges. As {C,D} is a light edge with exactly one end point belonging to {A,B,C}, we have by Corollary 61 that {C,D} is a **safe** edge with respect to $\mathcal{F}$
- {B,G} is **useless** edge. While the edge {B,G} connects the component {G} and {A,B,C}, {B,G} is not a minimum-weight edge leaving component {A,B,C}.And this edge will not result in a subgraph of some minimum-weight spanning tree of G, or it will result in a cycle in the graph.
- {F,G} is **useless** edge. While the edge {F,G} connects the component {G} and {E,F}, {F,G} is not a minimum-weight edge leaving component {E,F}. And this edge will not result in a subgraph of some minimum-weight spanning tree of G, or it will result in a cycle in the graph.
- {E,G} is **useless** edge. While the edge {E,G} connects the component {G} and {E,F}, {E,G} is not a minimum-weight edge leaving component {E,F}. And this edge will not result in a subgraph of some minimum-weight spanning tree of G, or it will result in a cycle in the graph.
- {D,E} is **useless** edge. While the edge {D,E} connects the component {D} and {E,F}, {D,E} is not a minimum-weight edge leaving component {E,F}. And this edge will not result in a subgraph of some minimum-weight spanning tree of G, or it will result in a cycle in the graph.
- There is **no undecided** edge since no edge we add to the immediate spanning forest will result in a cycle.

□

4 Standard 8- Kruskal's Algorithm

Problem 3. Consider the weighted graph $G(V, E, w)$ below. Clearly list the order in which Kruskal's algorithm adds edges to a minimum-weight spanning tree for G. Additionally, clearly articulate the steps that Kruskal's algorithm takes as it selects the first **three** edges.



Answer. • **Cite: adapted from Michaellevet's note.**

- We initialize the intermediate spanning forest $\mathcal{F}$ to be the empty graph (the graph on no edges). We also place the edges of G into a priority queue, which we call Q.
  So: Q = [({A,G },6), ({E,F },7), ({C,D },8), ({B,E },10), ({A,B },11), ({B,G },13), ({E,G },14), ({F,G},14), ({B,C },18), ({D,E },25)]

- We poll from Q, which returns the edge {A, G}. Note that w({A, G}) = 6. As A and G are on different components of $\mathcal{F}$ (didn't result in a cycle), we **add** the edge {B, C} to $\mathcal{F}$.
  So: Q = [({E,F },7), ({C,D },8), ({B,E },10), ({A,B },11), ({B,G },13), ({E,G },14), ({F,G},14), ({B,C },18), ({D,E },25)]

- We poll from Q, which returns the edge {E, F}. Note that w({E, F}) = 7. As E and F are on different components of $\mathcal{F}$ (didn't result in a cycle), we **add** the edge {E, F} to $\mathcal{F}$.
  So: Q = [({C,D },8), ({B,E },10), ({A,B },11), ({B,G },13), ({E,G },14), ({F,G},14), ({B,C },18), ({D,E },25)]

- We poll from Q, which returns the edge {C, D}. Note that w({C, D}) = 8. As C and D are on different components of $\mathcal{F}$ (didn't result in a cycle), we **add** the edge {C, D} to $\mathcal{F}$.
  So: Q = [({B,E },10), ({A,B },11), ({B,G },13), ({E,G },14), ({F,G},14), ({B,C },18), ({D,E },25)]

- We poll from Q, which returns the edge {B,E}. Note that w({B, E}) = 10. As B and E are on different components of $\mathcal{F}$ (didn't result in a cycle), we **add** the edge {B, E} to $\mathcal{F}$.
  So: Q = [({A,B },11), ({B,G },13), ({E,G },14), ({F,G},14), ({B,C },18), ({D,E },25)]

- We poll from Q, which returns the edge {A,B}. Note that w({A, B}) = 11. As A and B are on different components of $\mathcal{F}$ (didn't result in a cycle), we **add** the edge {A, B} to $\mathcal{F}$.
  So: Q = [({B,G },13), ({E,G },14), ({F,G},14), ({B,C },18), ({D,E },25)]

- We poll from Q, which returns the edge {B,G}. Note that w({B, G}) = 13. As B and G are on the same components of $\mathcal{F}$ (result in a cycle), we **do not add** the edge {B, G} to $\mathcal{F}$.
  So: Q = [({E,G },14), ({F,G},14), ({B,C },18), ({D,E },25)]

- We poll from Q, which returns the edge {E,G}. Note that w({E, G}) = 14. As E and G are on the same components of $\mathcal{F}$ (result in a cycle), we **do not add** the edge {E, G} to $\mathcal{F}$.
  So: Q = [({F,G},14), ({B,C },18), ({D,E },25)]

- We poll from Q, which returns the edge {F,G}. Note that w({F, G}) = 14. As F and G are on the same components of $\mathcal{F}$ (result in a cycle), we **do not add** the edge {F, G} to $\mathcal{F}$.
  So: Q = [({B,C },18), ({D,E },25)]

- We poll from Q, which returns the edge {B,C}. Note that w({B, C}) = 18. As B and C are on different components of $\mathcal{F}$ (didn't result in a cycle), we **add** the edge {B, C} to $\mathcal{F}$.
  So: Q = [({D,E },25)]

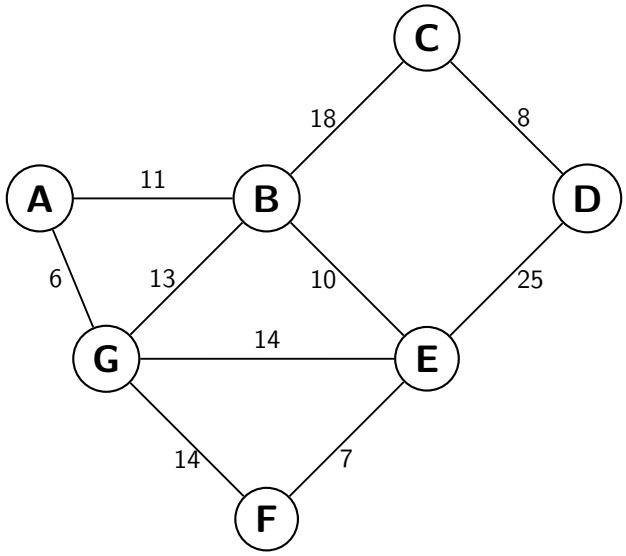
- We poll from Q, which returns the edge {D,E}. Note that w({D, E}) = 25. As D and E are on the same components of $\mathcal{F}$ (result in a cycle), we **do not add** the edge {D, E} to $\mathcal{F}$.
  So: Q = [ ]

- As we have 7 vertex and 6 edges on $\mathcal{F}$, the algorithm terminates and return $\mathcal{F}$ as minimum-weight spanning tree.

□

5 Standard 9- Prim's Algorithm

Problem 4. Consider the weighted graph $G(V, E, w)$ below. Clearly list the order in which Prim's algorithm, **using the source vertex** A, adds edges to a minimum-weight spanning tree for G. Additionally, clearly articulate the steps that Prim's algorithm takes as it selects the first **three** edges.



Answer. • **Cite: adapted from Michaellevet's note.**

- The algorithm first initialize an intermediate spanning tree $\mathcal{F}$ with all the vertex in weighted graph G, but no edges in it.
  Then, algorithm initialize an priority queue Q (ascending order) to contains the edge incident to the source vertex A.
  So: Q:[({A,G },6), ({A,B },11)]

- Algorithm poll from Q and mark it as processed, which returns the edge {A,G}. Note that w({A, G}) = 6. As A and G has exactly one end point on the component containing A (didn't result in a cycle), algorithm **add** the edge {A,G} to $\mathcal{F}$. We then push into the priority queue the unprocessed edges incident to G (edges that not already in Q, and unprocessed)
  So: Q:[({A,B },11), ({B,G },13), ({E,G },14), ({F,G },14)]

- Algorithm poll from Q and mark it as processed, which returns the edge {A,B}. Note that w({A, B}) = 11. As A and B has exactly one end point on the component containing A (didn't result in a cycle), algorithm **add** the edge {A,B} to $\mathcal{F}$. We then push into the priority queue the unprocessed edges incident to B (edges that not already in Q, and unprocessed)
  So: Q:[({B,E },10), ({B,G },13), ({E,G },14), ({F,G },14), ({B,C },18),]

- Algorithm poll from Q and mark it as processed, which returns the edge {B,E}. Note that w({B, E}) = 11. As B and E has exactly one end point on the component containing A (didn't result in a cycle), algorithm **add** the edge {B,E} to $\mathcal{F}$. We then push into the priority queue the unprocessed edges incident to E (edges that not already in Q, and unprocessed)
  So: Q:[({E,F }, 7) ({B,G },13), ({E,G },14), ({F,G },14), ({B,C },18), ({D,E },25)]

- Algorithm poll from Q and mark it as processed, which returns the edge {E,F}. Note that w({E, F}) = 11. As E and F has exactly one end point on the component containing A (didn't result in a cycle), algorithm **add** the edge {E,F} to $\mathcal{F}$. We then push into the priority queue the unprocessed edges incident to F (edges that not already in Q, and unprocessed)
  So: Q:[({B,G },13), ({E,G },14), ({F,G },14), ({B,C },18), ({D,E },25)]

- Algorithm poll from Q and mark it as processed, which returns the edge {B,G}. Note that w({B, G}) = 13. As both B and G belong to the component containing A (result in a cycle), algorithm **do not add** the

edge {B,G} to $\mathcal{F}$. Then, nothing will be pushed into the Q. So: Q:[({E,G },14), ({F,G },14), ({B,C },18), ({D,E },25)]

- The algorithm **do not add** ({E,G },14) to $\mathcal{F}$, which result in a cycle if it is included. And both E and G belong to the component containing A
  So: Q:[({F,G },14), ({B,C },18), ({D,E },25)]
- The algorithm **do not add** ({F,G },14) to $\mathcal{F}$, which result in a cycle if it is included. And both F and G belong to the component containing A
  So: Q:[({B,C },18), ({D,E },25)]
- The algorithm **add** ({B,C},18) to $\mathcal{F}$ because B and C has exactly one end point on the component containing A (didn't result in a cycle). Then, We then push into the priority queue the unprocessed edges incident to C (edges that not already in Q, and unprocessed)

  So: Q:[({C,D },8), ({D,E },25)]
- The algorithm **add** ({C,D},8) to $\mathcal{F}$ because B and C has exactly one end point on the component containing A (didn't result in a cycle). Then, We then push into the priority queue the unprocessed edges incident to D (edges that not already in Q, and unprocessed)

  So: Q:[({D,E },25)]
- The algorithm **do not add** ({D,E },25) to $\mathcal{F}$, which result in a cycle if it is included. And both D and E belong to the component containing A
  So: Q:[ ]
- Algorithm terminates, because we have 7 vertex and $7 - 1 = 6$ edges in $\mathcal{F}$. Then algorithm will return $\mathcal{F}$ as result.

□