Name: Kevina Wong

ID: 10917904

**CSCI 3104, Algorithms** **Profs. Chen & Grochow**

**Problem Set 7 – Due Thur Mar 12 11:55pm** **Spring 2020, CU-Boulder**

*Advice 1*: 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.

*Advice 2*: Informal reasoning is typically insufficient for full credit. Instead, write a logical argument, in the style of a mathematical proof.

**Instructions for submitting your solutions**:

- All submissions must be typed.
- You should submit your work through the **class Canvas page** only.
- 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 allot at least as many pages per problem (or subproblem) as are allotted in this template.

Quicklinks: 1 2 3 3a 3b 3c 3d

Name: Kevina Wong
ID: 10917904

**CSCI 3104, Algorithms** **Profs. Chen & Grochow**
**Problem Set 7 – Due Thur Mar 12 11:55pm** **Spring 2020, CU-Boulder**

1. *Let $G(V, E, w)$ be a weighted graph. The* **Cycle Property** *provides conditions for when an edge is* ***not*** *included in any minimum spanning tree (MST) of $G$. The* ***Cycle Property*** *is stated as follows. Let $C$ be a cycle in $G$, and let edge $e = (u, v)$ be a maximum-cost edge on $C$. Then the edge $e$ does not belong to any MST of $G$.*

   *Use an exchange argument to show why this property holds. You may assume that all edge costs are distinct. [****Note:*** *You may freely use any of the tree properties covered in Michael's Tree Notes on Canvas.]*

   **Answer:**

   Suppose the following is true:

   - G is a weighted graph with cycle C.
   - $e_m = (u, v)$ is a maximum weighted edge of G.
   - S is a spanning tree of the nodes in G that contains edge $e_m$.

   **Goal:** We want to show that S not a MST (ie. it does not have the minimum cost possible).

   **Exchange Property:** If we wanted to prove this using the exchange property, we would need to show that if we swap the maximum edge, $e_m$, for another less weighted edge, $e_o$, we would still have a spanning tree.

   **Proof:** Suppose that our claim is that our spanning tree S is an MST, and suppose that we remove $e_m$ that connects $u$ to $v$ from S. We would then have two disconnected graphs. Recall that $e_m$ was an edge within the cycle, C. In order to recombine the two disconnected graphs, we would have to reconnect nodes $u$ to $v$. Since $e_m$ was the maximum weighted edge and was in a cycle, we can thus confirm that there exists another edge $e_o$ that is able to connect two nodes within the cycle, C, such that it will connect the two disconnected graphs, $u'$ and $v'$, but with a lower weight. This creates another possible spanning tree, S'. S' is connected and has a cheaper weight. By proof by contradiction, we can see that it is not true that S is an MST. This proves that the cycle property holds.

2. *Let $G(V, E, w)$ be a connected, weighted graph. A cut of $G$ is a set $C \subset E$ of edges such that removing the edges of $C$ from $G$ disconnects the graph. The* **Cut Property** *provides conditions for when an edge* ***must*** *be included in a minimum spanning tree (MST) of $G$. The* ***Cut Property*** *is stated as follows. Let $C \subset E$ be a cut. Suppose $e = (u, v)$ is the minimum-weight edge in $C$. Then every MST contains the edge $e$.*

   *Use an exchange argument to show why this property holds. You may assume that all edge costs are distinct. [**Note:** You may freely use any of the tree properties covered in Michael's Tree Notes on Canvas.]*

   **Answer:**

   Suppose the following is true:

   - G(V,E,w) is a connected, weighted graph.
   - All edge costs are distinct.
   - Let C be a subset of edges such that when one of these edges are removed, it disconnects the graph.
   - Let $e_m = (u, v)$ be the minimum cost edge in C.
   - S is a spanning tree that does not contain $e_m$.
   - Let N be a subset of nodes.

   **Goal:** We want to show that S is not a MST (ie. does not have the minimum cost possible)

   **Exchange Property:** If we wanted to use exchange argument to prove the cut property, we need to show that an edge, $e_b$, in S can be exchanged with the minimum cost edge, $e_m$, to produce another tree S' such that it is still a spanning tree and has a lower cost than the original S.

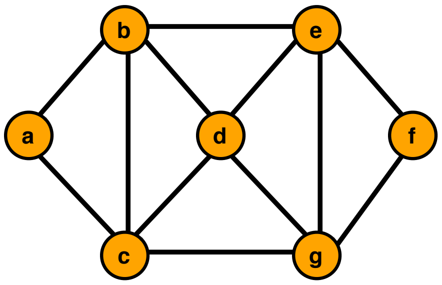
   **Proof:** Suppose we claim that S is an MST. Recall that edge $e_m$ connects nodes $u$ and $v$. Also recall one of the properties of a spanning tree: a spanning tree connects every vertex. Since $e_m$ isn't in the graph, we are guaranteed other edges, $e_b$, in C such that they connect the graph by connecting nodes, $u'$ to $v'$, both of which are edges in the original spanning tree. If we exchange $e_m$ for $e_b$, it would yield another spanning tree, S', such that it has a lower weight (since $e_b$ has been exchanged for a lower weight $e_m$), is connected, and consisted of the original nodes in G. Because S' exists and has a lower weight, we have contradicted our original statement the S would be an MST. Thus, we can conclude that every MST would contain the edge $e_m$.

Name: Kevina Wong
ID: 10917904

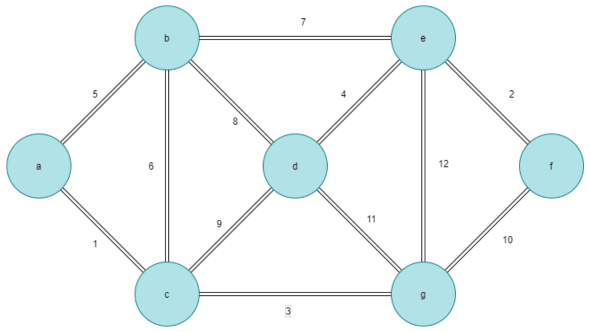
**CSCI 3104, Algorithms** **Profs. Chen & Grochow**
**Problem Set 7 – Due Thur Mar 12 11:55pm** **Spring 2020, CU-Boulder**

3. *Given the following unweighted graph, you're asked to assign a positive integer weight to each edge, such that the following properties are true regarding minimum spanning trees (MST) and single-source shortest path (SSSP) trees:*

   - *All edge weights are distinct*
   - *The MST is distinct from any of the seven SSSP trees. Note that there is a SSSP tree for each node as the source.*
   - *The order in which Prim's algorithm adds the safe edges is different from the order in which Kruskal's algorithm adds them.*



   (a) *Put your picture of the graph (the same one as above)* ***with your edge weights*** *here. Remember that all the edge weights should be distinct:*



Name: Kevina Wong

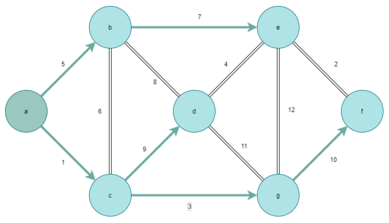
ID: 10917904

**CSCI 3104, Algorithms** **Profs. Chen & Grochow**

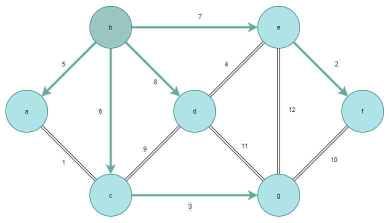
**Problem Set 7 – Due Thur Mar 12 11:55pm** **Spring 2020, CU-Boulder**

(b) *Draw the seven SSSP trees (one for each starting vertex). In each tree, indicate which vertex you used as the starting vertex. For one such tree, clearly articulate the steps taken to select the first three edges using **Dijkstra's algorithm**.*

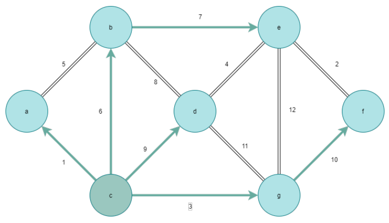
a



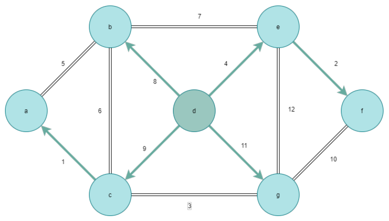
b



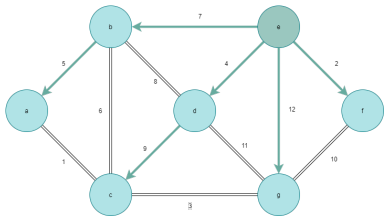
c



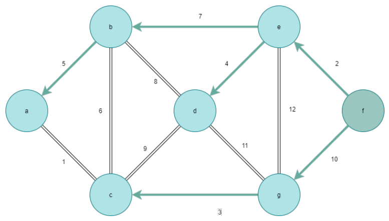
d



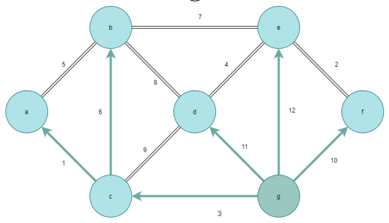
e



f



g



**Steps taken to select the first three edges - starting node g:**

**First Edge:** First, starting at G, we would have an options G $\rightarrow$ C with a cost of 3, G $\rightarrow$ D with a cost of 10, G $\rightarrow$ E with a cost of 12, and G $\rightarrow$ F with a cost of 10. Dijkstra's would pick the path G $\rightarrow$ C because it's the lowest cost path.

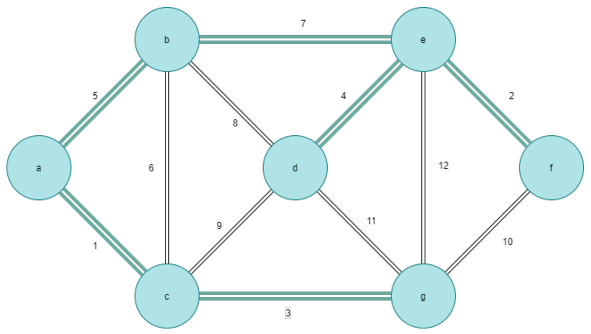
**Second Edge:** Then, it would hae the options C $\rightarrow$ A with a cost of 4, C $\rightarrow$ B with a cost of 9, C $\rightarrow$ D with a cost of 12, G $\rightarrow$ D with a cost of 10, G $\rightarrow$ E with a cost of 12, and G $\rightarrow$ F with a cost of 10. Dijkstra's next selection would be to take the path from C $\rightarrow$ A because it is the next shortest path.

**Third Edge:** Now, we have the options A $\rightarrow$ B with a cost of 9, C $\rightarrow$ B with a cost of 9, C $\rightarrow$ D with a cost of 12, G $\rightarrow$ D with a cost of 10, G $\rightarrow$ E with a cost of 12, and G $\rightarrow$ F with a cost of 10. Dijkstra's would pick either the path from A $\rightarrow$ B or c $\rightarrow$ B. It doesn't matter which one is chosen since they both have the same weight.

| | Vertex | Cost |
|---|---|---|
| 1 | g | 0 (Start) |
| 2 | c | 3 |
| 3 | a | 4 |
| 4 | b | 9 |
| 5 | f | 10 |
| 6 | e | 12 |

(c) *Use Prim's algorithm to compute the MST. List the order in which Prim's algorithm adds the edges, and draw the MST. Furthermore, clearly articulate the steps the algorithm takes as it selects the first three edges.*

**Answer:**
Prim's algorithm works in a way that it always selects the lowest cost immediate path. This mean that of it's options, it will always pick the path that is the lowest cost. To contrast this with Dijkstra's, Dijkstra's always picks the path with the shortest total path; whereas Prim's will pick the shortest immediate path.



*The MST is indicated by a thicker blue edge.*

**Order:**
*The first three edges that the algorithm selects is* ***bolded*** *below. Assume that we are starting at node a.*

| | Vertex | Cost | Visited? | Safe Edge? |
|---|---|---|---|---|
| 1 | a | 0 (Start) | No | Yes |
| **2** | **a → c** | **1** | **No** | **Yes** |
| **3** | **c → g** | **3** | **No** | **Yes** |
| **4** | **a → b** | **5** | **No** | **Yes** |
| 5 | b → c | 6 | Yes | No |
| 6 | b → e | 7 | No | Yes |
| 7 | e → f | 2 | No | Yes |
| 8 | e → d | 4 | No | Yes |
| 9 | b → d | 8 | Yes | No |
| 10 | c → d | 9 | Yes | No |
| 11 | g → f | 10 | Yes | No |
| 12 | d → g | 11 | Yes | No |
| 13 | e → g | 12 | No | No |

Name: Kevina Wong

ID: 10917904

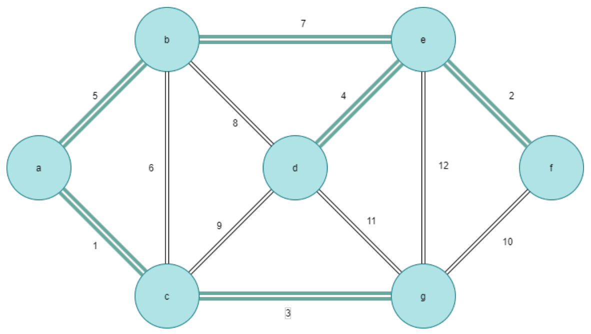
**CSCI 3104, Algorithms** **Profs. Chen & Grochow**

**Problem Set 7 – Due Thur Mar 12 11:55pm** **Spring 2020, CU-Boulder**

(d) *List the order in which Kruskal's algorithm adds the edges. Furthermore, clearly articulate the steps the algorithm takes as it selects the first three edges.*

**Answer:**

Kruskal's algorithm works in a way that it always selects the lowest cost edge. Id doesn't add nodes to a queue and select the shortest path that way, but rather, it will just pick the shortest edge out of all of the edges. Thus, there is no "starting" node.



*The MST is indicated by a thicker blue edge.*

**Order:**

*The first three edges that the algorithm selects is* ***bolded*** *below.*

| | Vertex | Cost | Safe Edge? |
|---|---|---|---|
| **1** | **a → c** | **1** | **Yes** |
| **2** | **e → f** | **2** | **Yes** |
| **3** | **c → g** | **3** | **Yes** |
| 4 | d → e | 4 | Yes |
| 5 | a → c | 6 | Yes |
| 6 | b → e | 7 | Yes |
| 7 | b → d | 8 | No |
| 8 | c → d | 9 | No |
| 9 | g → f | 10 | No |
| 10 | d → g | 11 | No |
| 11 | g → e | 12 | No |