CS 61B Spring 2022 — Disjoint Sets, Graph Algorithms

Exam Prep Discussion 14: April 11, 2022

1 Disjoint Sets

Here is a video walkthrough of the solutions.

For each of the arrays below, write whether this could be the array representation of a weighted quick union with path compression and explain your reasoning.

```
      i:   0   1   2   3   4   5   6   7   8   9
      ---------------------------------------
A. a[i]:   1   2   3   0   1   1   1   4   4   5
B. a[i]:   9   0   0   0   0   0   9   9   9 -10
C. a[i]:   1   2   3   4   5   6   7   8   9 -10
D. a[i]: -10   0   0   0   0   1   1   1   6   2
E. a[i]: -10   0   0   0   0   1   1   1   6   8
F. a[i]:  -7   0   0   1   1   3   3  -3   7   7
```

Solution:

A. Impossible: has a cycle 0-1, 1-2, 2-3, and 3-0 in the parent-link representation.

B. Impossible: the nodes 1, 2, 3, 4, and 5 must link to 0 when 0 is a root; hence, 0 would not link to 9 because 0 is the root of the larger tree.

C. Impossible: tree rooted at 9 has height 9 >lg 10.

D. Possible: 8-6, 7-1, 6-1, 5-1, 9-2, 3-0, 4-0, 2-0, 1-0.

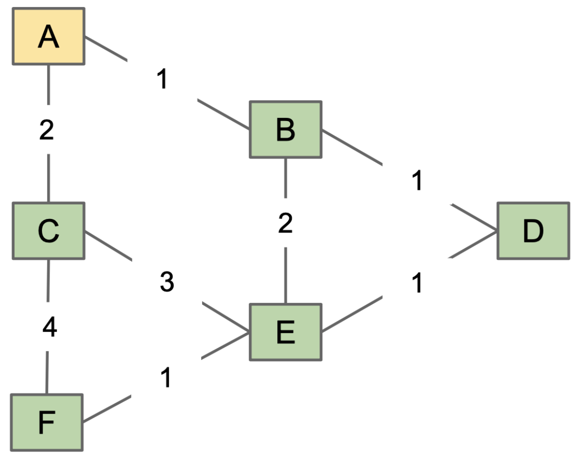
E. Impossible: tree rooted at 0 has height 4 >lg 10.

F. Impossible: tree rooted at 0 has height 3 >lg 7.

2 Graph Algorithms

Here is a video walkthrough for part a, and **here** is a video walkthrough for part b.

a) (4 Points). Suppose we have the graph below.



For all parts, break ties alphabetically. We recommend drawing this graph on paper! Recall that SPT stands for shortest paths tree.

i) (1 Points). Run Dijkstra's from vertex `A` on the graph above.

1. **(0.5 Points).** What is the ordering that vertices are visited? Format your answer as a space separated list, e.g. `A B C D E F`.

   Order:

   Solution:
   Order: A B C D E F

2. **(0.5 Points).** What edges are included in the SPT rooted at A?

   ☐ AB ☐ AC ☐ BD ☐ BE ☐ CE ☐ CF ☐ DE ☐ EF

   Solution:
   ■ AB ■ AC ■ BD ■ BE ☐ CE ☐ CF ☐ DE ■ EF

ii) (1 Points). Run A* from vertex `A` to `F` on the graph above. Let the heuristic of a vertex `v` be the number of edges between `v` and `F` in the BFS tree rooted at `F` (you may need to run BFS to find this). For instance, the heuristic of D is 2.

1. **(0.5 Points).** What is the ordering that vertices are visited? Format your answer as a space separated list, e.g. `A B C D E F`.

   Order:

   Solution:
   Order: A B C D E F

2. **(0.5 Points).** What edges comprise the shortest path?

☐ AB ☐ AC ☐ BD ☐ BE ☐ CE ☐ CF ☐ DE ☐ EF

Solution:
■ AB ☐ AC ☐ BD ■ BE ☐ CE ☐ CF ☐ DE ■ EF

iii) (1 Points). Run Prim's from vertex `A` on the graph above.

1. **(0.5 Points).** What is the ordering that vertices are visited? Format your answer as a space separated list, e.g. `A B C D E F`.

   Order:

   Solution:
   Order: A B D E F C

2. **(0.5 Points).** What edges are included in the MST rooted at A?

   ☐ AB ☐ AC ☐ BD ☐ BE ☐ CE ☐ CF ☐ DE ☐ EF

   Solution:
   ■ AB ■ AC ■ BD ☐ BE ☐ CE ☐ CF ■ DE ☐ EF

iv) (0.5 Points). Increase the weight of one edge by 1 so that the SPT found by Dijkstra's is the same as the MST found by Prim's. Assume we've run Prim's and Dijkstra's from vertex `A` on the graph above.

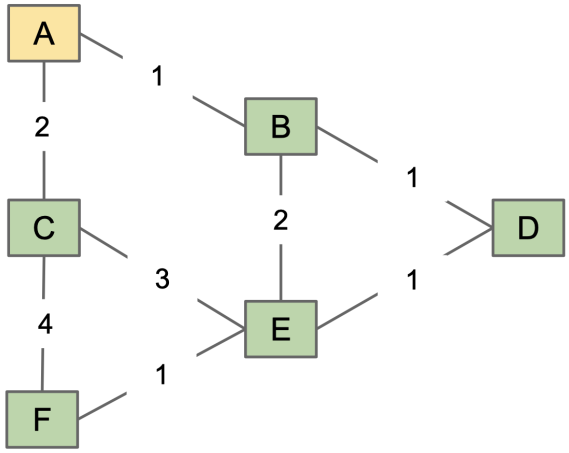
If the SPT found by Dijkstra's is already the same as the MST found by Prim's, select the option "no change needed". If it's impossible for them to be equal after increasing the weight of one edge by 1, select the option "impossible".

◯ AB ◯ AC ◯ BD ◯ BE ◯ CE ◯ CF ◯ DE ◯ EF
◯ no change needed ◯ impossible

Solution:
◯ AB ◯ AC ◯ BD √ **BE** ◯ CE ◯ CF ◯ DE ◯ EF
◯ no change needed ◯ impossible

b) (2 Points). Multiple SPTs

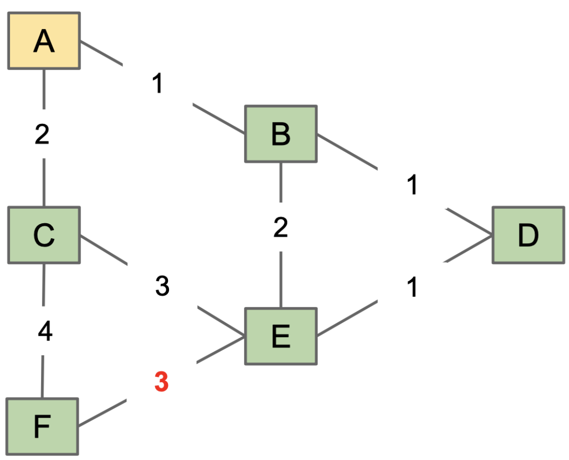
i) (0.5 Points). Using the graph from the previous part, which has been recopied for your convenience below, how many different **SPTs** rooted at A exist? Recall there can be multiple SPTs for a given start vertex if there are multiple paths of equal, minimum cost to at least one end vertex.



Number of SPTs: ◯ 0 ◯ 1 ◯ 2 ◯ 3 ◯ 4 ◯ 5 ◯ 6 ◯ 7 ◯ 8

Solution:
Number of SPTs: ◯ 0 ◯ 1 √ **2** ◯ 3 ◯ 4 ◯ 5 ◯ 6 ◯ 7 ◯ 8

ii) (0.5 Points). Now, suppose we change the edge weight of EF to 3. How many SPTs are there now? The altered graph has been drawn for your convenience below. Hint: The number of SPTs should *increase* from part i to part ii.



Number of SPTs: ◯ 0 ◯ 1 ◯ 2 ◯ 3 ◯ 4 ◯ 5 ◯ 6 ◯ 7 ◯ 8

Solution:
Number of SPTs: ◯ 0 ◯ 1 ◯ 2 ◯ 3 √ **4** ◯ 5 ◯ 6 ◯ 7 ◯ 8

iii) (1 Point). Finally, using the same graph from part b.ii, change *one* edge weight to further **increase** the total number of SPTs rooted at A.

Edge: ◯ AB ◯ AC ◯ BD ◯ BE ◯ CE ◯ CF ◯ DE ◯ EF

Solution:
Edge: ◯ AB ◯ AC ◯ BD ◯ BE √ **CE** ◯ CF ◯ DE ◯ EF

New weight: ◯ 0 ◯ 1 ◯ 2 ◯ 3 ◯ 4 ◯ 5 ◯ 6 ◯ 7 ◯ 8

Solution:
New weight: ◯ 0 √ **1** ◯ 2 ◯ 3 ◯ 4 ◯ 5 ◯ 6 ◯ 7 ◯ 8

3 Multiple MSTs

Here is a video walkthrough of all parts of this problem.

Recall a graph can have multiple MSTs if there are multiple spanning trees of minimum weight.

(a) For each subpart below, select the correct option and justify your answer. If you select "never" or "always," provide a short explanation. If you select "sometimes", provide two graphs that fulfill the given properties — one with multiple MSTs and one without. Assume G is an undirected, connected graph.

1. If **none** the edge weights are **identical**, there will

   ◯ never be multiple MSTs in G.

   ◯ sometimes be multiple MSTs in G.

   ◯ always be multiple MSTs in G.

   Justification:

2. If **some** of the edge weights are **identical**, there will

   ◯ never be multiple MSTs in G.

   ◯ sometimes be multiple MSTs in G.

   ◯ always be multiple MSTs in G.

   Justification:

3. If **all** of the edge weights are **identical**, there will

   ◯ never be multiple MSTs in G.

   ◯ sometimes be multiple MSTs in G.

   ◯ always be multiple MSTs in G.

   Justification:

Solution:

1. If **none** the edge weights are **identical**, there will

    ■ never be multiple MSTs in G.

    ○ sometimes be multiple MSTs in G.

    ○ always be multiple MSTs in G.

Justification:
To prove this, we can leverage the cut property. Recall the cut property states that the cheapest edge in any cut is in *some* MST. However, if the cheapest edge in any cut is unique, then we get a stronger claim — the cheapest edge must be in *the* MST. As such, if none of the edge weights are identical, i.e. they are all unique, then the cheapest edge in any cut will always be unique, and we will only have one MST.

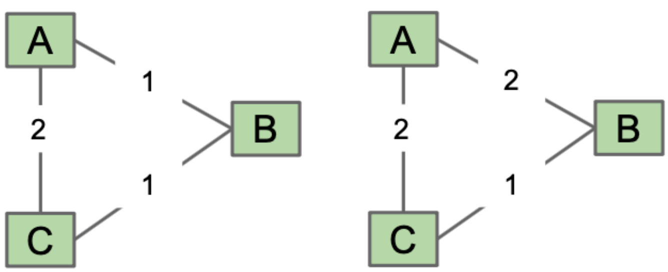
2. If **some** of the edge weights are **identical**, there will

    ○ never be multiple MSTs in G.

    ■ sometimes be multiple MSTs in G.

    ○ always be multiple MSTs in G.

Justification:



In the graph on the left, the only MST is [AB, BC]. In the graph on the right, two MSTs exist — [AB, BC] and [AC, BC].

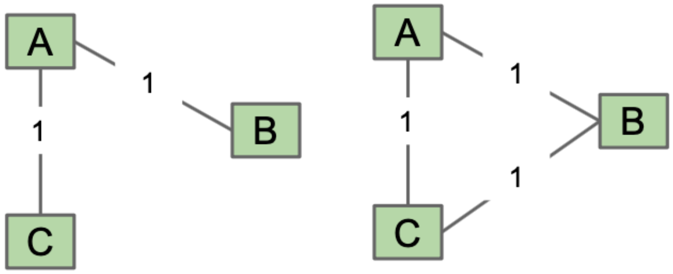
3. If **all** of the edge weights are **identical**, there will

    ○ never be multiple MSTs in G.

    ■ sometimes be multiple MSTs in G.

    ○ always be multiple MSTs in G.

Justification:



In the graph on the left, the only MST is [AB, AC]. Note that for any tree, we only have one MST, since the tree itself is the MST! In the graph on the right, three MSTs exist — [AB, BC], [AC, BC], and [AB, AC].

(b) Suppose we have a connected, undirected graph G with N vertices and N edges, where all the **edge weights are identical**. Find the maximum and minimum number of MSTs in G and explain your reasoning.

Minimum: _________
Maximum: _________

Justification:

Solution: Minimum: 3, Maximum: N

Justification: Notice that if all the edge weights are the same, an MST is just a spanning tree. Let's begin by creating a tree, i.e. a connected graph with $N-1$ edges. Now, notice that there is only one spanning tree, since the graph is itself a tree.

As such, the problem reduces to: how many spanning trees can the insertion of one edge create? If we add an edge to a tree, it will create a cycle that can be of length at minimum 3 and at maximum N. Then, notice that we can only remove **any** edge from a cycle to create a spanning tree, so we have at minimum 3 and at maximum N possible MSTs in G.

(c) It is possible that Prim's and Kruskal's find **different** MSTs on the same graph `G` (as an added exercise, construct a graph where this is the case!). Given any graph `G` with integer edge weights, modify `G` to **ensure** that Prim's and Kruskal's will always find the same MST. You may not modify Prim's or

Kruskal's.
Hint: Look at subpart 1 of part a.

Solution: To ensure that Prim's and Kruskal's will always produce the same MST, notice that if `G` has unique edges, only one MST can exist, and Prim's and Kruskal's will always find that MST! So, what if we modify `G` to ensure that all the edge weights are unique?

To achieve this, let's strategically add a small, unique `offset` between 0 and 1, exclusive, to each edge. It is important that we choose an `offset` between 0 and 1 so that this added value doesn't change the MST, since all the edge weights are integers. It is also important that the offset is unique for each edge, because then we ensure each weight is distinct. Pseudocode for such a change is shown below:

```
E = number of edges in the graph
offset = 0
for edge in graph:
    edge.weight += offset
    offset += 1 / E
```

In regard to the added exercise, here is a simple graph G where Prim's and Kruskal's produce different MSTs. Prim's starting from A will select AD, BD, and CD, whereas Kruskals will select AD, BC, and BD.