Graphs & WQU

Discussion 14

Announcements

- **Project 3 Checkpoint** due Monday 04/25
- **Lab 14** due Monday 04/25
- **Last Weekly Survey** due Tuesday 04/26
- **Project 3** due Friday 04/29

Review

Disjoint Sets

```
public interface DisjointSet {
     void connect (x, y); // Connects nodes x and y (you may also see union)
     boolean isConnected(x, y); // Returns true if x and y are connected
}
```

QuickFind uses an array of integers to track which set each element belongs to.

QuickUnion stores the parent of each node rather than the set to which it belongs and merges sets by setting the parent of one root to the other.

WeightedQuickUnion does the same as QuickUnion except it decides which set is merged into which by size, reducing stringiness.

WeightedQuickUnion with Path Compression sets the parent of each node to the set's root whenever find() is called on it.

Disjoint Sets

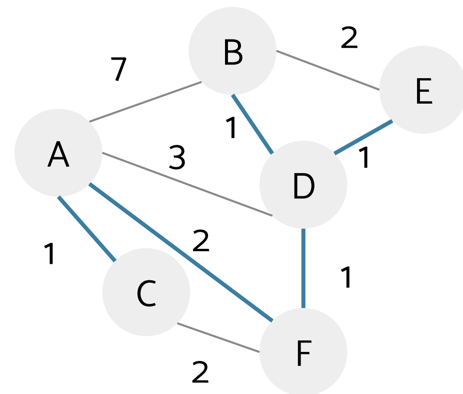
```
public interface DisjointSet {
     void connect (x, y); // Connects nodes x and y (you may also see union)
     boolean isConnected(x, y); // Returns true if x and y are connected
}
```

Implementation	Constructor	connect()	isConnected()
QuickUnion	Θ(N)	O(N)	O(N)
QuickFind	Θ(N)	O(N)	O(1)
Weighted Quick Union	Θ(N)	O(log N)	O(log N)
WQU with Path Compression	Θ(N)	O(log N) Θ(1)*	O(log N) Θ(1)*

Minimum Spanning Trees

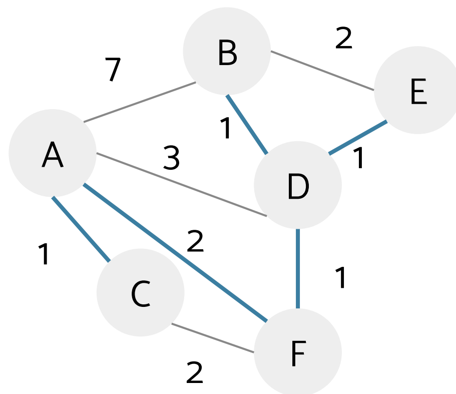
Minimum Spanning Trees are set of edges that connect all the nodes in a graph while being of the smallest possible weight.

MSTs may not be unique if there are multiple edges of the same weight.



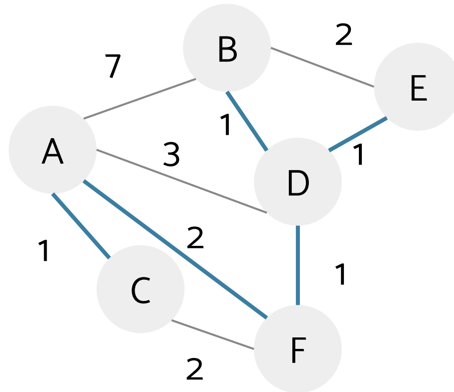
Prim's Algorithm

```
Start with any node.
Add that node to the set of nodes in the MST.
While there are still nodes not in the MST:
     Add the lightest edge that leads to a node that is unvisited.
     Add the new node to the set of nodes in the MST.
```



Kruskal's Algorithm

```
While there are still nodes not in the MST:
    Add the lightest edge that doesn't create a loop.
    Add the new node to the set of nodes in the MST.
```

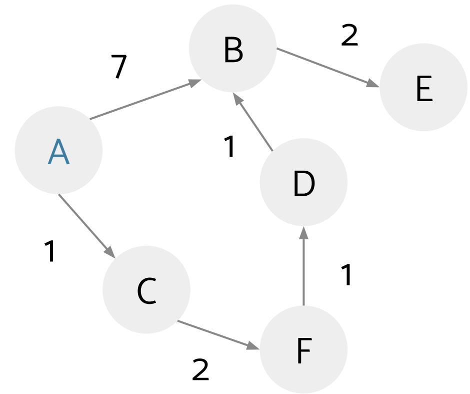


Dijkstra's Algorithm

Dijkstra's algorithm is a method of finding the shortest path from one node to every other node in the graph. You use a priority queue that sorts points based off of their distance to the root node.

Steps:

1. Pop node from the top of the queue - this is the current node.
2. Add/update distances of all of the children of the current node.
3. Re-sort the priority queue.
4. Finalize the distance to the current node from the root.
5. Repeat.

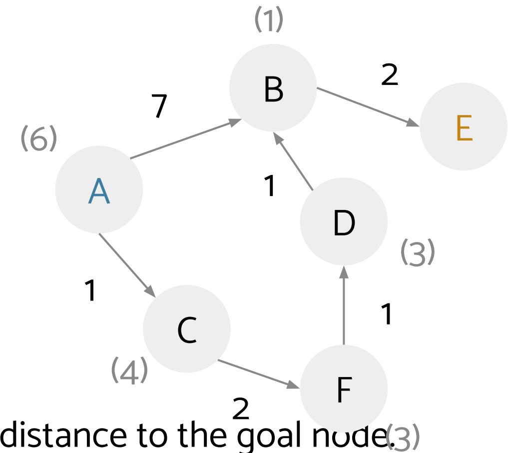


A*

A* is a method of finding the shortest path from one node to a specific other node in the graph. It operates very similarly to Dijkstra's except for the fact that we use a (given) heuristic to which path is the best to our goal point.

Steps:

1. Pop node from the top of the queue - this is the current node.
2. Add/update distances of all of the children of the current node. This distance will be the sum of the distance up to that child node and our guess of how far away the goal node is (our heuristic).
3. Re-sort the priority queue.
4. Check if we've hit the goal node (if so we stop).
5. Repeat.



To be admissible, our heuristic is that it must be less than or equal to the true distance to the goal node.

Worksheet

1A Weight Times

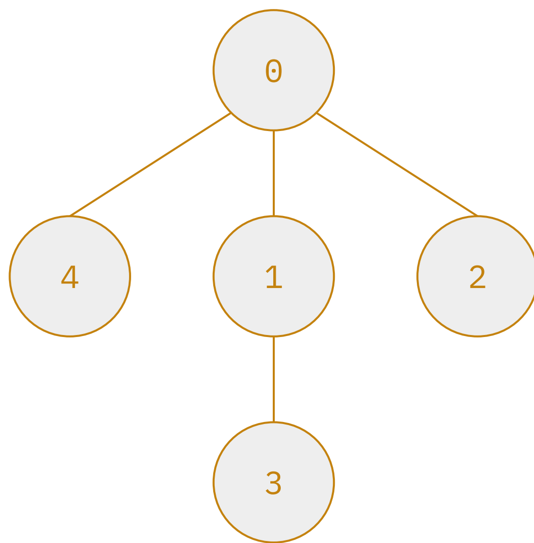
Draw the Weighted Quick Union object that results from the following method calls

```
connect(1, 3);
connect(0, 4);
connect(0, 1);
connect(0, 2);
```

1A Weight Times

Draw the Weighted Quick Union object that results from the following method calls

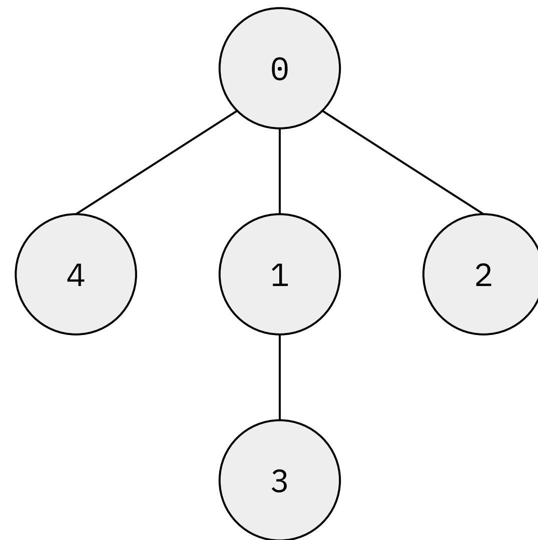
```
connect(1, 3);
connect(0, 4);
connect(0, 1);
connect(0, 2);
```



1B Weight Times

What is the resulting array of Weighted Quick Union after the calls in part A are executed?

```
connect(1, 3);
connect(0, 4);
connect(0, 1);
connect(0, 2);
```

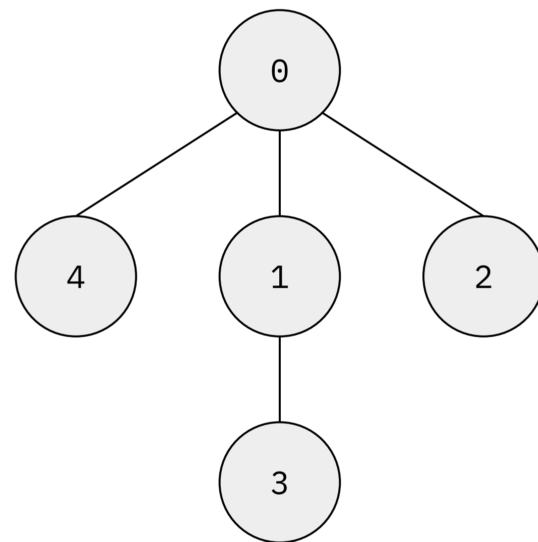


1B Weight Times

What is the resulting array of Weighted Quick Union after the calls in part A are executed?

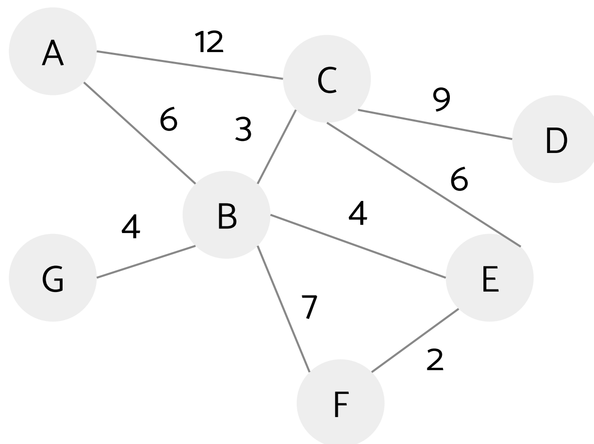
```
connect(1, 3);
connect(0, 4);
connect(0, 1);
connect(0, 2);

arr = [-5, 0, 0, 1, 0]
```



2A Minimum Spanning Trees

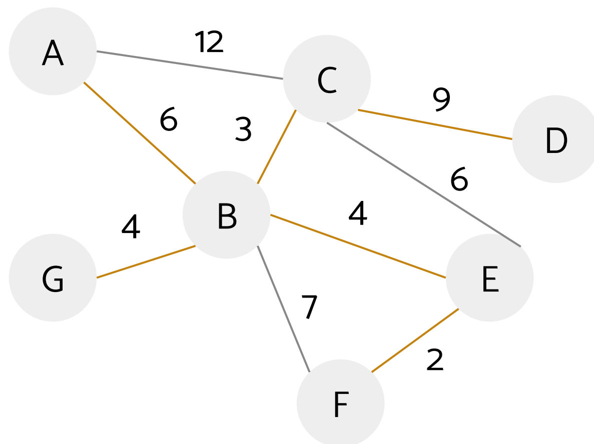
Perform Prim's Algorithm.



2A Minimum Spanning Trees

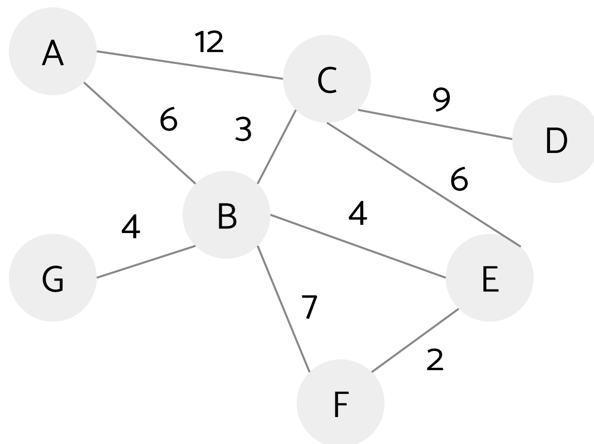
Perform Prim's Algorithm.

AB, BC, BE, EF, BG, CD



2A Minimum Spanning Trees

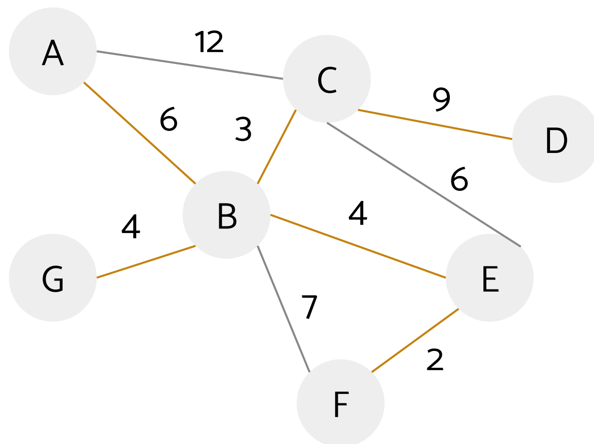
Perform Kruskal's Algorithm.



2A Minimum Spanning Trees

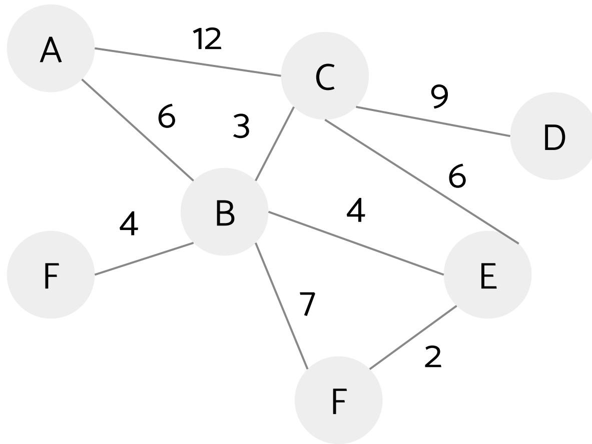
Perform Kruskal's Algorithm.

EF, BC, BE, BG, AB, CD



2B Minimum Spanning Trees

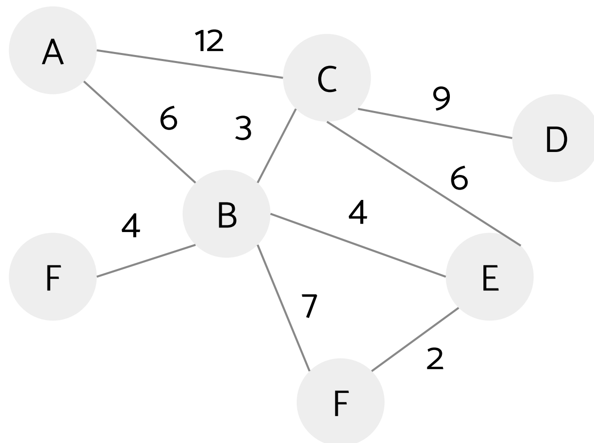
Is there any vertex for which the shortest paths three from that vertex is the same as the Prim MST?



2B Minimum Spanning Trees

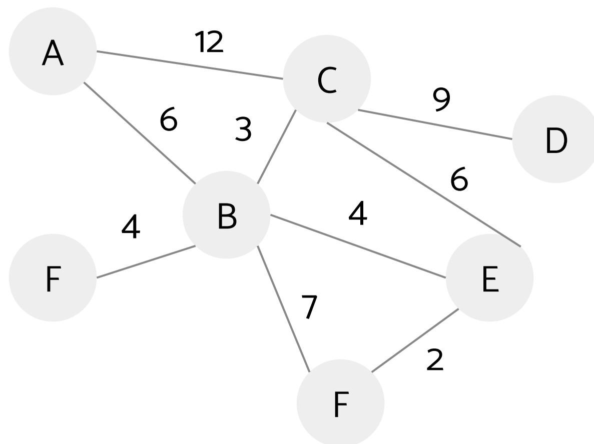
Is there any vertex for which the shortest paths three from that vertex is the same as the Prim MST?

B, A, G



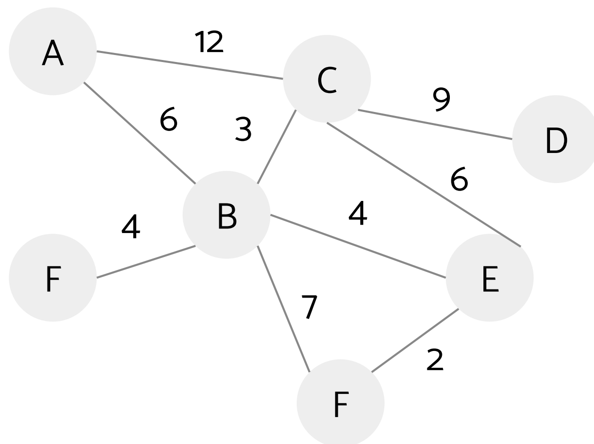
2C Minimum Spanning Trees

True or False: Adding 1 to the smallest edge of a graph G with unique edge weights must change the total weight of the MST.



2C Minimum Spanning Trees

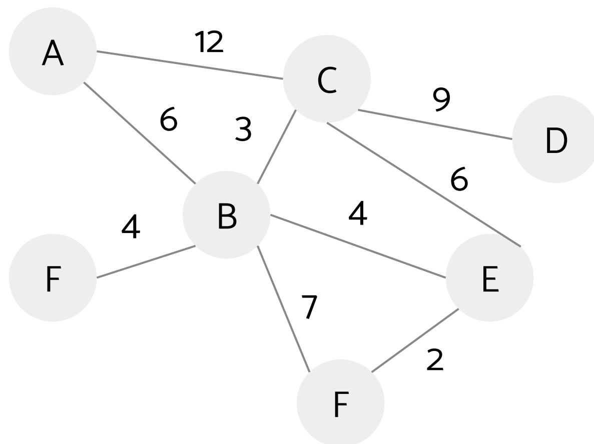
True or False: Adding 1 to the smallest edge of a graph G with unique edge weights must change the total weight of the MST.



True - either this smallest edge is still included (increasing the weight by one) or some larger edge takes its place (increasing the weight by less than one) since there is no other edge of equal weight

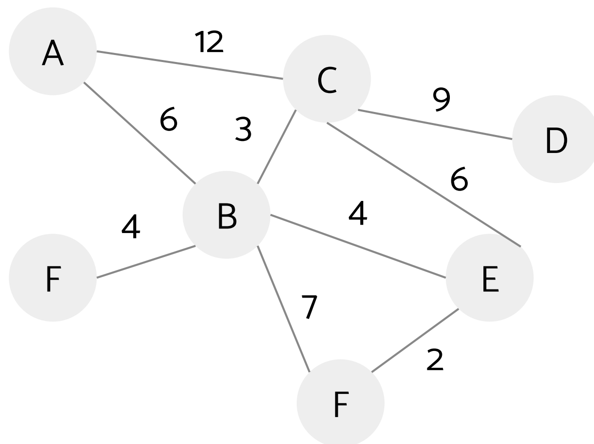
2D Minimum Spanning Trees

True or False: The shortest path from vertex A to vertex B in graph G is the same as the shortest path from A to B using only edges on T, where T is the MST of G.



2D Minimum Spanning Trees

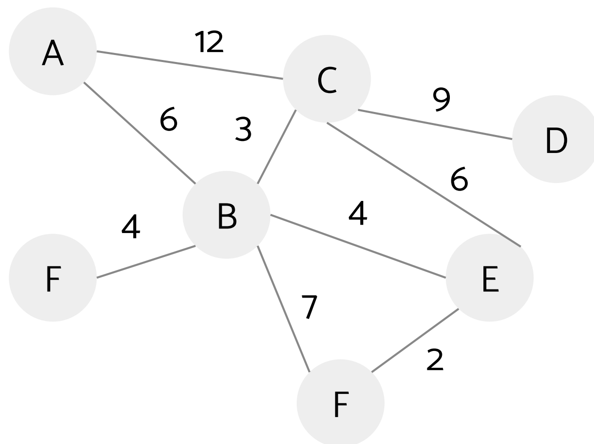
True or False: The shortest path from vertex A to vertex B in graph G is the same as the shortest path from A to B using only edges on T, where T is the MST of G.



False - Consider vertices C and E in the given graph

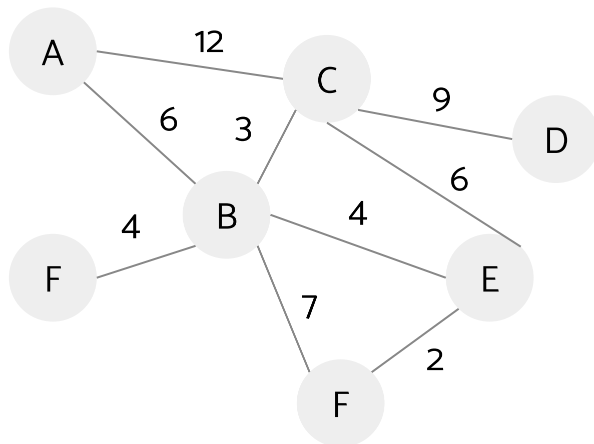
2E Minimum Spanning Trees

True or False: Given any cut, the maximum-weight crossing edge is in the maximum spanning tree



2E Minimum Spanning Trees

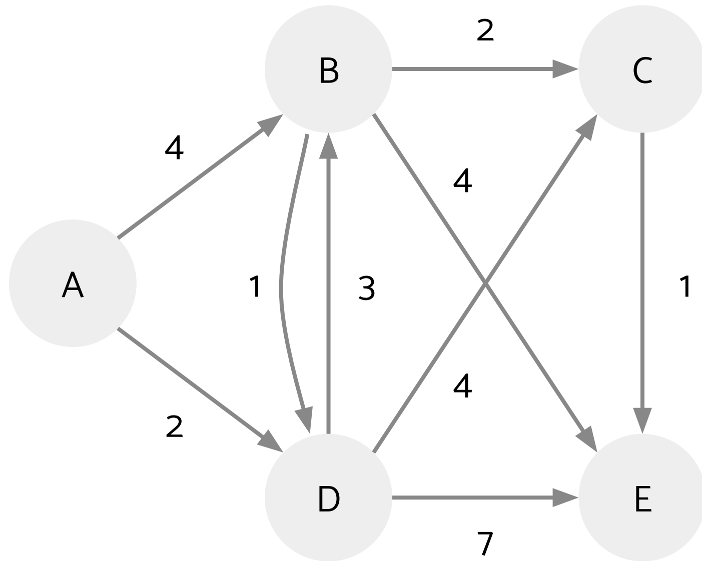
True or False: Given any cut, the maximum-weight crossing edge is in the maximum spanning tree



True - we can use the cut property proof as seen in the class, but replace smallest with largest

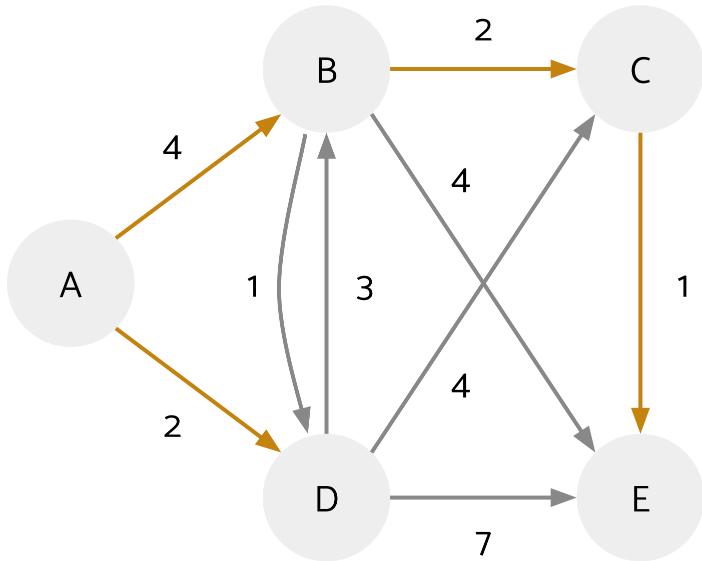
3A Dijkstra's Algorithm

Run Dijkstra's Algorithm starting from node A.



3A Dijkstra's Algorithm

Run Dijkstra's Algorithm starting from node A.



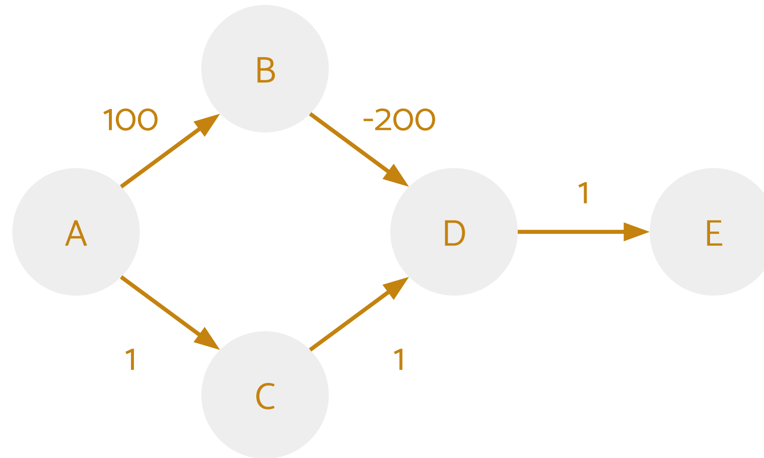
v	Init	Pop A	Pop D	Pop B	Pop C	Pop E
A	**0**	0	0	0	0	0
B	∞	4	4	**4**	4	4
C	∞	∞	6	6	**6**	6
D	∞	2	**2**	2	2	2
E	∞	∞	9	8	7	**7**

3B Dijkstra's Algorithm

What must be true about Dijkstra's to guarantee we return the shortest path? Draw an example where this fails.

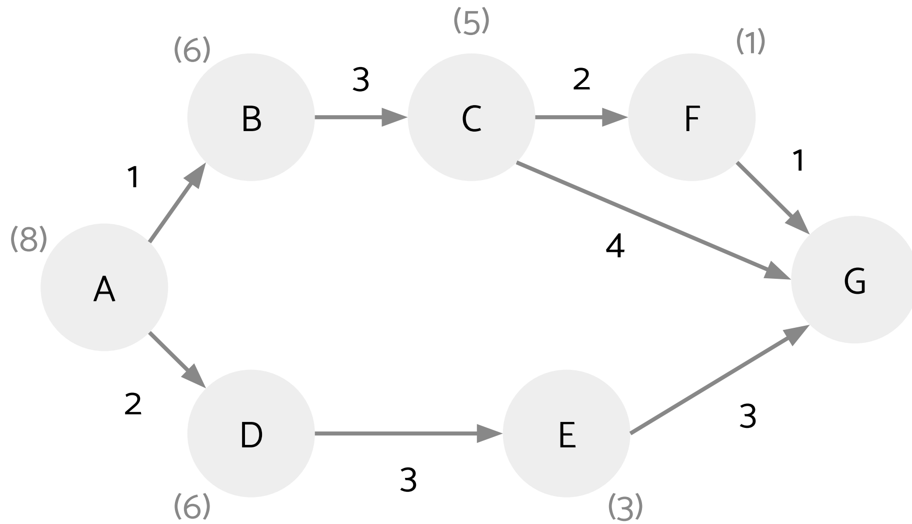
3B Dijkstra's Algorithm

What must be true about Dijkstra's to guarantee we return the shortest path? Draw an example where this fails.



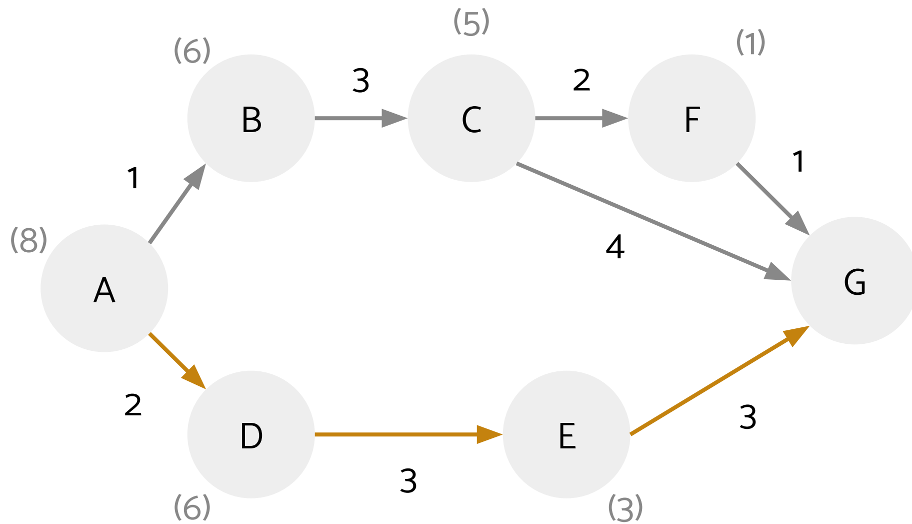
4A A* Search

What would A* return as the shortest path from A to G?



4A A* Search

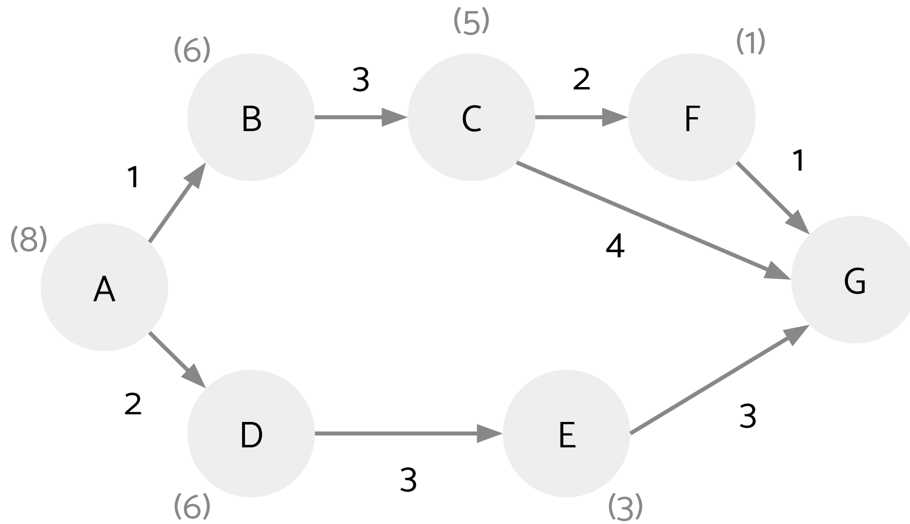
What would A* return as the shortest path from A to G?



v	Init	Pop A	Pop B	Pop D	Pop E	Pop G
A	**0**	0	0	0	0	
B	∞	**1**	1	1	1	
C	∞	∞	4	4	4	
D	∞	2	**2**	2	2	
E	∞	∞	∞	**5**	5	
F	∞	∞	∞	∞	∞	
G	∞	∞	∞	∞	**8**	

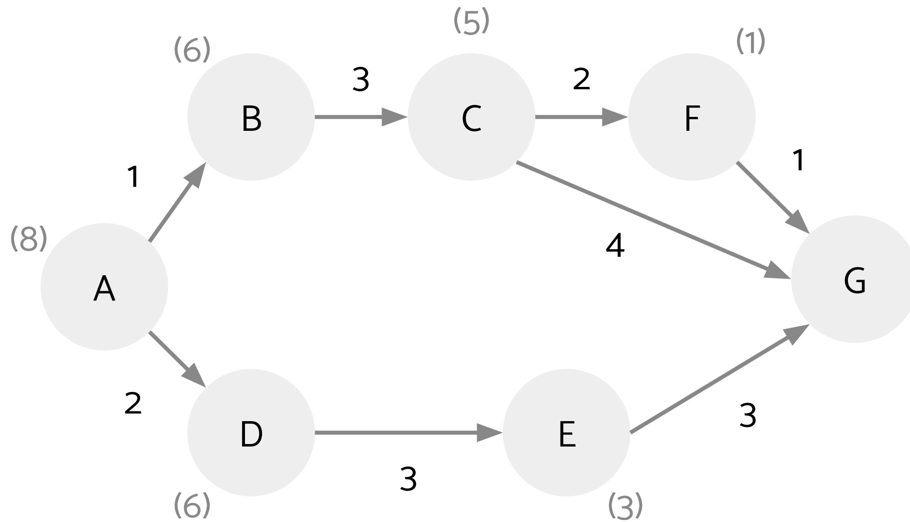
4B A* Search

Is this heuristic admissible?



4B A* Search

Is this heuristic admissible?



No, h(C, G) = 5, but the shortest path from C to G has length 3.