L25: Minimum Spanning Trees (MST) and Disjoint Sets

Alex Steiger
CompSci 201: Spring 2024
4/15/2024

1

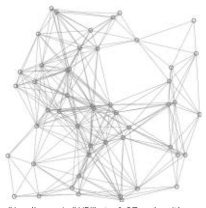
Logistics, coming up

- This Wednesday, 4/17
  - Midterm exam 3
  - APT 9 (last APTs) - extended to Thursday 4/18
- This Friday, 4/19
  - Semester / Final review in discussion
- Next Monday, 4/22
  - Project P6: Route (last project) due
- Tuesday after next, 4/30
  - Final exam, 9 am

2

Exploring a node with Dijkstra's Algorithm, Pseudocode

While unexplored nodes remain
- Explore current = the closest unexplored node
- For each neighbor:
  - Update shortest path to neighbor if shorter to go through current

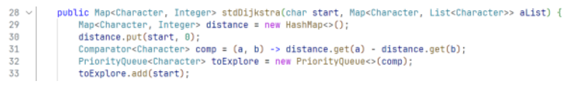

wikipedia.org/wiki/Dijkstra%27s_algorithm

Just like BFS (explore closer nodes first) except… now we need to account for weights.

3

Practical Dijkstra Initialization

Add vertices to the queue once they are actually reached/visited.



```java
public Map<Character, Integer> stdDijkstra(char start, Map<Character, List<Character>> aList) {
    Map<Character, Integer> distance = new HashMap<>();
    distance.put(start, 0);
    Comparator<Character> comp = (a, b) -> distance.get(a) - distance.get(b);
    PriorityQueue<Character> toExplore = new PriorityQueue<>(comp);
    toExplore.add(start);
```

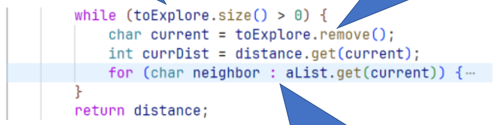
Don’t need to add anything for all nodes yet.

4

Practical Dijkstra search loop

Keep searching while there are unexplored nodes.

Choose to explore from the *next closest (to* start*) unexplored node to start* at each iteration.



```java
while (toExplore.size() > 0) {
    char current = toExplore.remove();
    int currDist = distance.get(current);
    for (char neighbor : aList.get(current)) {…
}
return distance;
```

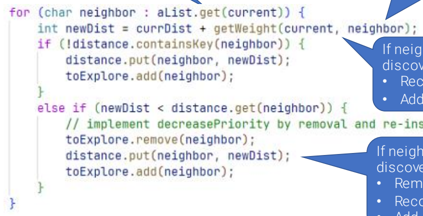
Search all neighbors of current. If you find a ***shorter path*** to neighbor through current, update to reflect that.

5

Details: Checking each neighbor

All neighbors of current node

Distance to neighbor through current = distance to current + weight on edge from current to neighbor



```java
for (char neighbor : aList.get(current)) {
    int newDist = currDist + getWeight(current, neighbor);
    if (!distance.containsKey(neighbor)) {
        distance.put(neighbor, newDist);
        toExplore.add(neighbor);
    }
    else if (newDist < distance.get(neighbor)) {
        // implement decreasePriority by removal and re-insertion
        toExplore.remove(neighbor);
        distance.put(neighbor, newDist);
        toExplore.add(neighbor);
    }
}
```

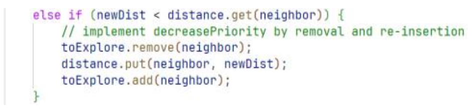
If neighbor newly discovered:
- Record new distance
- Add to priority queue

If neighbor already discovered, update:
- Remove from PQ
- Record new distance
- Add back to PQ

6

Implementing decreasePriority

- Most standard library binary heaps (including java.util) don't support an efficient update/decrease priority operation.



```
else if (newDist < distance.get(neighbor)) {
    // implement decreasePriority by removal and re-insertion
    toExplore.remove(neighbor);
    distance.put(neighbor, newDist);
    toExplore.add(neighbor);
}
```

- Our code works, but is O(N) time
  - Java's PQ takes O(N) to remove ***given*** node (O(log N) for smallest)
  - Other PQ implementations support O(log N)-time decreasePriority, but they are not in Java library

7

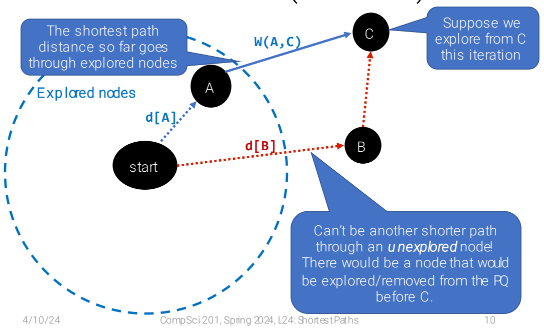
Is Dijkstra's algorithm guaranteed to be correct? (Informal)

- **Claim.** Distance is correct shortest path distance for all nodes ***explored*** so far, and shortest path distance ***through explored nodes*** for all others.

- Formal proof is ***by induction***, see CompSci 230.
  - Assume the property is true up to some point in the algorithm, then…
  - Consider the next node we explore:

9

Is Dijkstra's algorithm guaranteed to be correct? (Informal)


The shortest path distance so far goes through explored nodes
Explored nodes
W(A,C)
C
Suppose we explore from C this iteration
A
d[A]
start
d[B]
B
Can't be another shorter path through an *unexplored* node! There would be a node that would be explored/removed from the PQ before C.

10

L25-WOTO1-Dijkstra-Sp24

Hi, Alexander. When you submit this form, the owner will see your name and email address.

* Required

1

NetID *

solutions

2

At each iteration, Dijkstra's algorithm will explore the next unexplored node that... *

○ was discovered most recently

○ was discovered earliest

[✓ is closest to the starting node]

○ is closest to the ending node

3

Suppose every edge in the graph has equal weight. Then Dijkstra's algorithm would... *

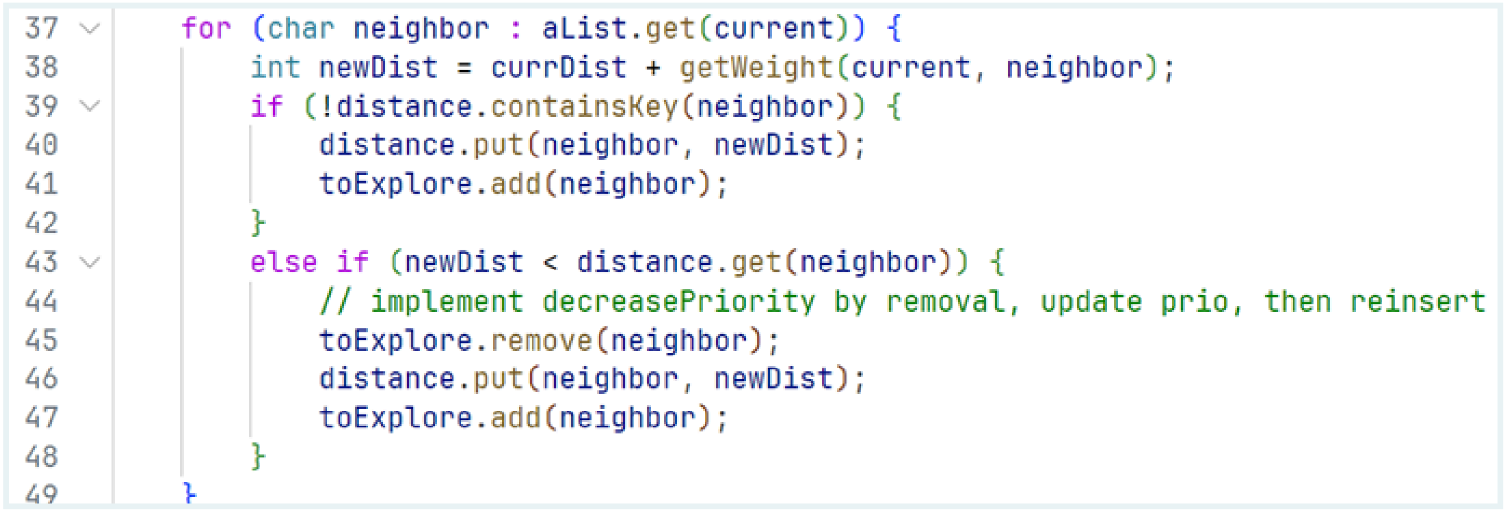
○ Explore nodes in the order of a depth-first search (DFS)

[✓ Explore nodes in the order of a breadth-first search (BFS)]

○ Explore nodes in an order different than DFS or BFS

4

The best explanation of the if statement on line 39 is... *

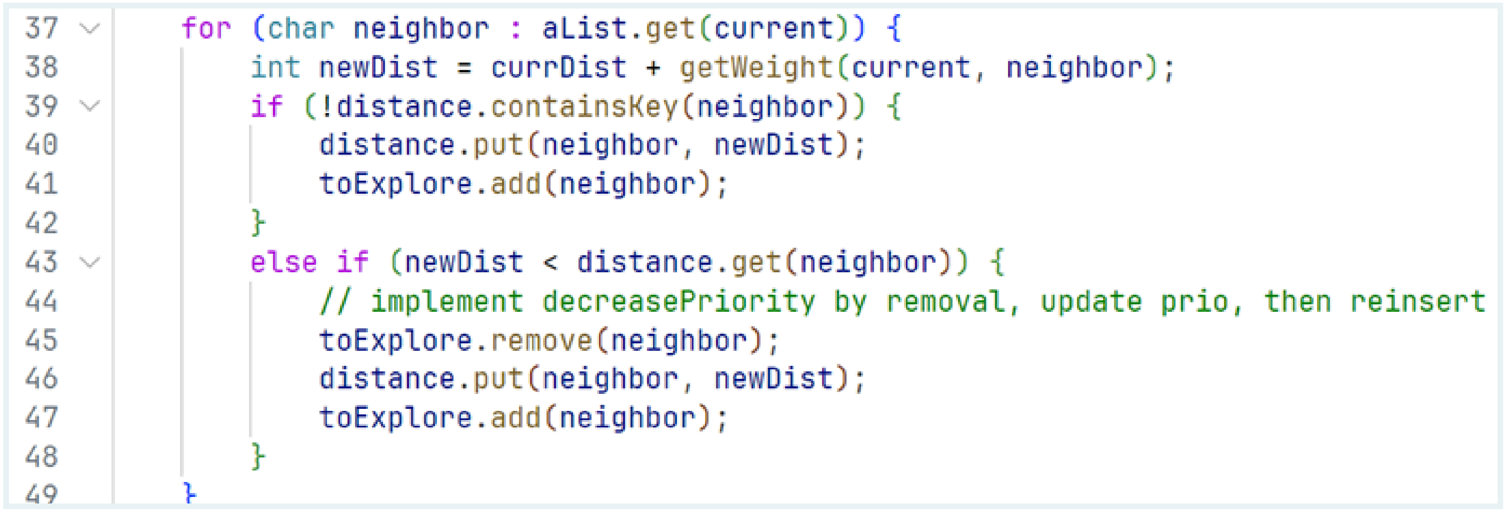


```
37      for (char neighbor : aList.get(current)) {
38          int newDist = currDist + getWeight(current, neighbor);
39          if (!distance.containsKey(neighbor)) {
40              distance.put(neighbor, newDist);
41              toExplore.add(neighbor);
42          }
43          else if (newDist < distance.get(neighbor)) {
44              // implement decreasePriority by removal, update prio, then reinsert
45              toExplore.remove(neighbor);
46              distance.put(neighbor, newDist);
47              toExplore.add(neighbor);
48          }
49      }
```

- If neighbor is newly discovered or is closer to start than current
- If neighbor is newly discovered or the path through current to neighbor is shorter
- [✓ If neighbor is newly discovered]
- If neighbor is not in the graph or is closer to start than current
- If neighbor is not in the graph or the path through current to neighbor is shorter
- If the path through current to neighbor is shorter

5

The best explanation of the if statement on line 43 is... *



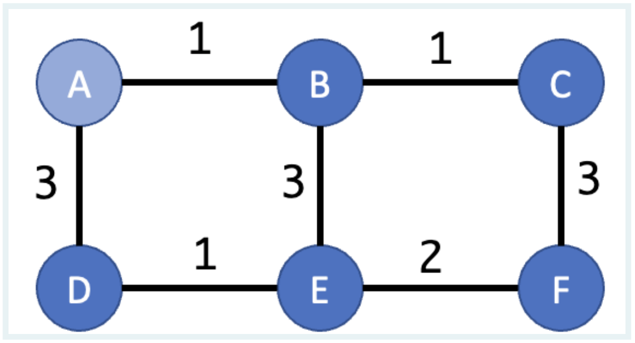
```java
    for (char neighbor : aList.get(current)) {
        int newDist = currDist + getWeight(current, neighbor);
        if (!distance.containsKey(neighbor)) {
            distance.put(neighbor, newDist);
            toExplore.add(neighbor);
        }
        else if (newDist < distance.get(neighbor)) {
            // implement decreasePriority by removal, update prio, then reinsert
            toExplore.remove(neighbor);
            distance.put(neighbor, newDist);
            toExplore.add(neighbor);
        }
    }
```

- If neighbor is newly discovered or is closer to start than current
- If neighbor is newly discovered or the path through current to neighbor is shorter
- If neighbor is newly discovered
- If neighbor is not in the graph or is closer to start than current
- If neighbor is not in the graph or the path through current to neighbor is shorter
- [✓ If the path through current to neighbor is shorter]

6

Consider the weighted undirected graph diagrammed with edges labeled by their weight.

Starting from A, in what order will Dijkstra's algorithm explore (remove from the priority queue) the nodes in the graph? *



[✓ A, B, C, D, E, F]

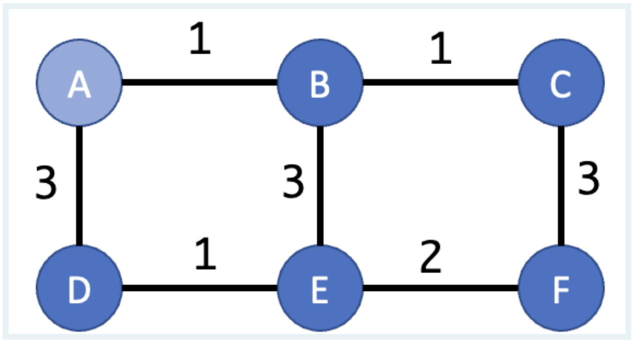
A, B, C, F, E, D

A, B, D, C, E, F

A, B, D, E, C, F

7

Again starting from A in the same graph, which path from A to E will Dijkstra's algorithm record as the shortest path? Hint: See the if statements above in problems 4-5. *

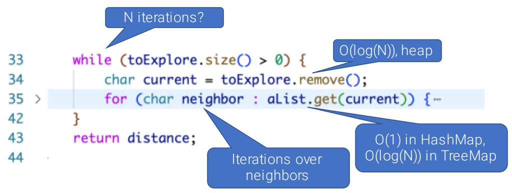


[✓ A, B, E]

A, D, E

○ Might be either, depends on tie breaking in the priority queue

Runtime Complexity of Dijkstra's Algorithm (with N nodes, M edges) *assuming O(log N) decreasePriority*



```
while (toExplore.size() > 0) {
    char current = toExplore.remove();
    for (char neighbor : aList.get(current)) {⋯
}
return distance;
```

N iterations?
O(log(N)), heap
O(1) in HashMap, O(log(N)) in TreeMap
Iterations over neighbors

Like BFS, consider each node once and each edge twice, takes O(log N) time for each: O((N+M)log(N))

11

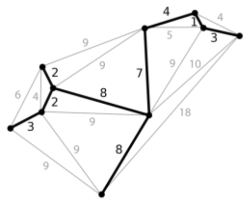
Minimum Spanning Tree (MST) and Greedy Graph Algorithms

12

Minimum Spanning Tree (MST) Problem

- Given N nodes and M edges, each with a weight/cost…
- Find a set of edges that connect *all* the nodes with minimum total cost (will be a tree)

Weighted undirected graph with:
- Edges labeled with weights/costs
- Minimum spanning tree highlighted



13

Motivating/Applying Minimum Spanning Tree

- Create a connected cable/data network with the least cable/cost/energy possible.
- City planning: Connect several metro stops with least tunneling
- Image Segmentation
- Clustering

https://slideplayer.com/slide/11413693/

14

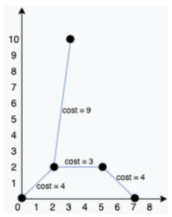
Example MST Problem

leetcode.com/problems/min-cost-to-connect-all-points

You are given an array points representing integer coordinates of some points on a 2D-plane, where points[i] = [x_i, y_i].

The cost of connecting two points [x_i, y_i] and [x_j, y_j] is the **manhattan distance** between them: $|x_i - x_j| + |y_i - y_j|$, where |val| denotes the absolute value of val.

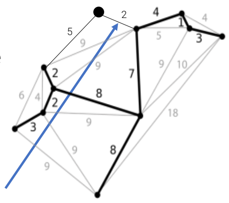
Return *the minimum cost to make all points connected.* All points are connected if there is **exactly one** simple path between any two points.



15

Intuitive Inductive Reasoning

- Suppose we have the MST on N-1 vertices.
- We consider the next vertex to get the MST on N vertices.
  - Must use the cost 2 or the cost 5 edge *regardless* of the rest of the MST
  - Might as well use the cheaper cost 2 edge



16

Greedy Optimization: Prim's Algorithm

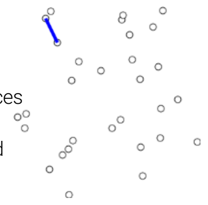
- Initialize?
  - Choose an arbitrary vertex
- Partial solution?
  - MST connecting *subset* of the vertices.
- Greedy step?
  - Choose the cheapest / least weight edge that connects a new vertex to the partial solution.

17

Visualizing Prim's Algorithm

In the visualization:

- Edges between all pairs of vertices
- Weights are implicit by distances
- Algorithm greedily grows by choosing closest unconnected vertex

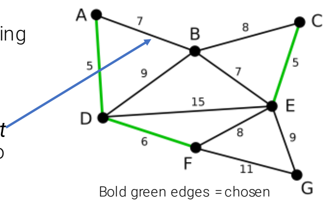


By Shiyu Ji - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=54420894

18

More Intuitive Inductive Reasoning

- Suppose we have chosen some spanning trees so far.
- Must connect all of them, might as well choose the *cheapest* edge connecting two trees.



Bold green edges = chosen

Public Domain, https://commons.wikimedia.org/w/index.php?curid=644030

19

Greedy Optimization Again: Kruskal's Algorithm

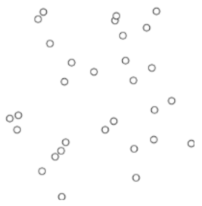
- Initialize?
  - All nodes in ***disjoint sets***
- Partial solution?
  - Forest of spanning trees in disjoint sets
- Greedy step?
  - Choose the cheapest / least weight edge that connects two disjoint sets / trees, connect them.

20

Visualizing Kruskal's Algorithm

In the visualization:

- Edges between all pairs of vertices
- Weights are implicit by distances
- Algorithm greedily grows by cheapest edge that connects disjoint sets/trees.



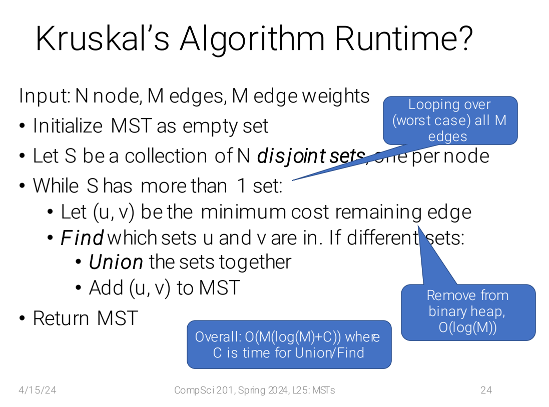
By Shiyu Ji - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=54420894

21

Kruskal's Algorithm in *Pseudocode*

```
Input: N node, M edges, M edge weights
• Initialize MST as empty set
• Let S be a collection of N disjoint sets, one per node
• While S has more than 1 set:
    • Let (u, v) be the minimum cost remaining edge
    • Find which sets u and v are in. If different sets:
        • Union the sets together
        • Add (u, v) to MST
• Return MST
```

23



Kruskal's Algorithm Runtime?

Input: N node, M edges, M edge weights
- Initialize MST as empty set
- Let S be a collection of N ***disjoint sets***, one per node
- While S has more than 1 set:
  - Let (u, v) be the minimum cost remaining edge
  - ***Find*** which sets u and v are in. If different sets:
    - ***Union*** the sets together
    - Add (u, v) to MST
- Return MST

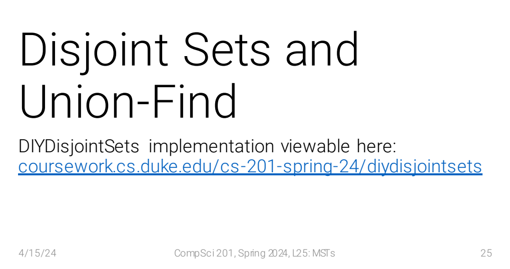
Looping over (worst case) all M edges

Remove from binary heap, O(log(M))

Overall: O(M(log(M)+C)) where C is time for Union/Find

4/15/24 CompSci 201, Spring 2024, L25: MSTs 24

24

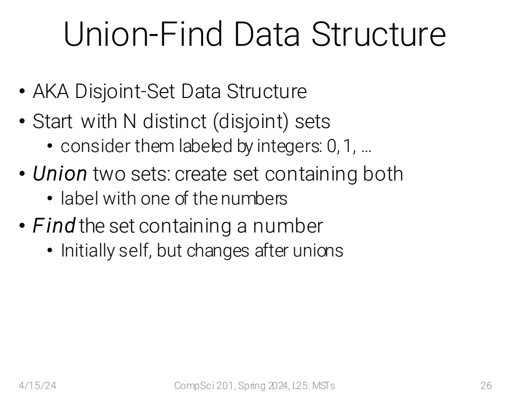


Disjoint Sets and Union-Find

DIYDisjointSets implementation viewable here:
coursework.cs.duke.edu/cs-201-spring-24/diydisjointsets

4/15/24 CompSci 201, Spring 2024, L25: MSTs 25

25



Union-Find Data Structure

- AKA Disjoint-Set Data Structure
- Start with N distinct (disjoint) sets
  - consider them labeled by integers: 0, 1, …
- *Union* two sets: create set containing both
  - label with one of the numbers
- *Find* the set containing a number
  - Initially self, but changes after unions

4/15/24 CompSci 201, Spring 2024, L25: MSTs 26

26

L25-WOTO2-MST-Sp24

Hi, Alexander. When you submit this form, the owner will see your name and email address.

* Required

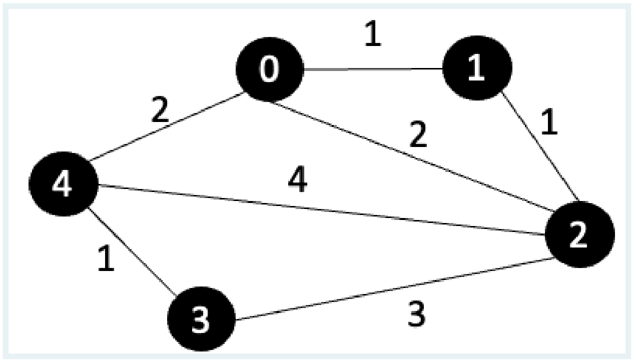
1

NetID *

solutions

2

For the weighted undirected graph pictured below, what is the total cost of the minimum spanning tree? *



○ 3

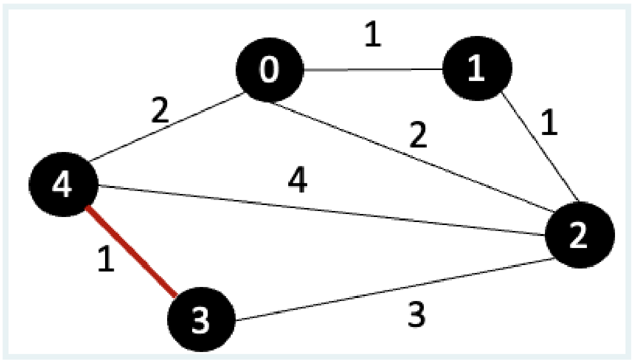
[✓ 5]

○ 8

○ 10

3

Suppose we are running Prim's algorithm, and we have so far selected the edge between nodes 3 and 4 shown in bolded red. What edge will we greedily select next? *



○ Edge between nodes 0 and 1

○ Edge between nodes 0 and 2

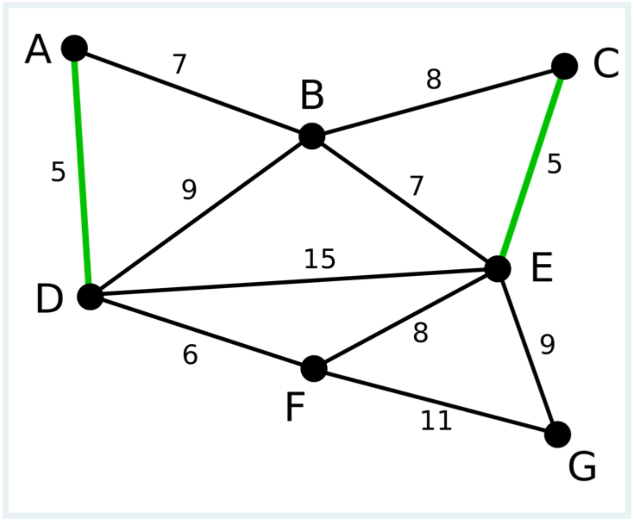
[✓ Edge between nodes 0 and 4]

○ Edge between nodes 2 and 3

4

Suppose we are running Kruskal's algorithm. So far we have selected the edges shown in bolded green: (A, D) and (C, E).

How many disjoint sets/trees are there remaining at this point? *



○ 0

○ 2

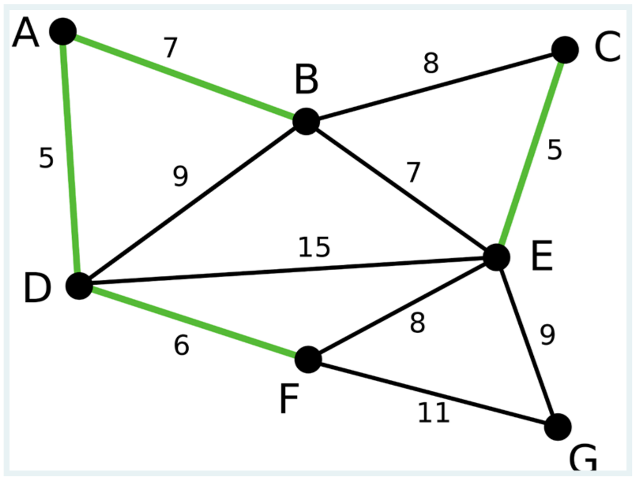
○ 3

[✓ 5]

5

Suppose we are running Kruskal's algorithm. So far we have selected the edges shown in bolded green: (A, D), (A, B), (D, F), and (C, E).

Which edge will the algorithm select next? *



◯ (B, C)

[✓ (B, E)]

◯ (D, E)

◯ (E, G)

◯ (F, E)

6

True or false: In a given iteration, Kruskal's algorithm will always select/add to the MST the next lowest weight edge. *

◯ True

[✓ False]

Microsoft 365