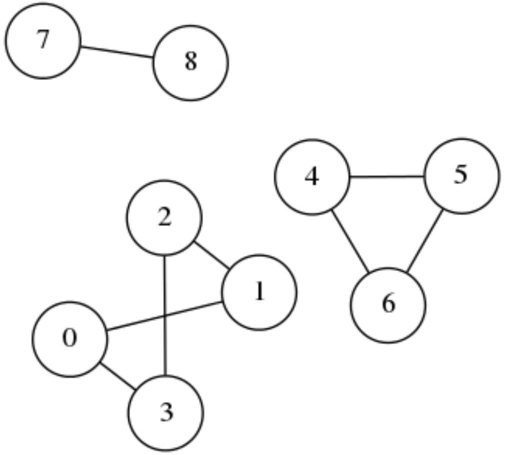
PROBLEM SOLVING 7

Disjoint Sets, Minimum Spanning Trees, Kruskal’s Algorithm

CS340

Determining Connected Components

- How could we do this?




```
CONNECTED-COMPONENTS(G)
for each vertex v ∈ G.V
    MAKE-SET(v)
for each edge (u, v) ∈ G.E
    if FIND-SET(u) ≠ FIND-SET(v)
        UNION(u, v)
```

```
SAME-COMPONENT(u, v)
if FIND-SET(u) == FIND-SET(v)
    return TRUE
else return FALSE
```

Looks like a tree, but can be done in an array:

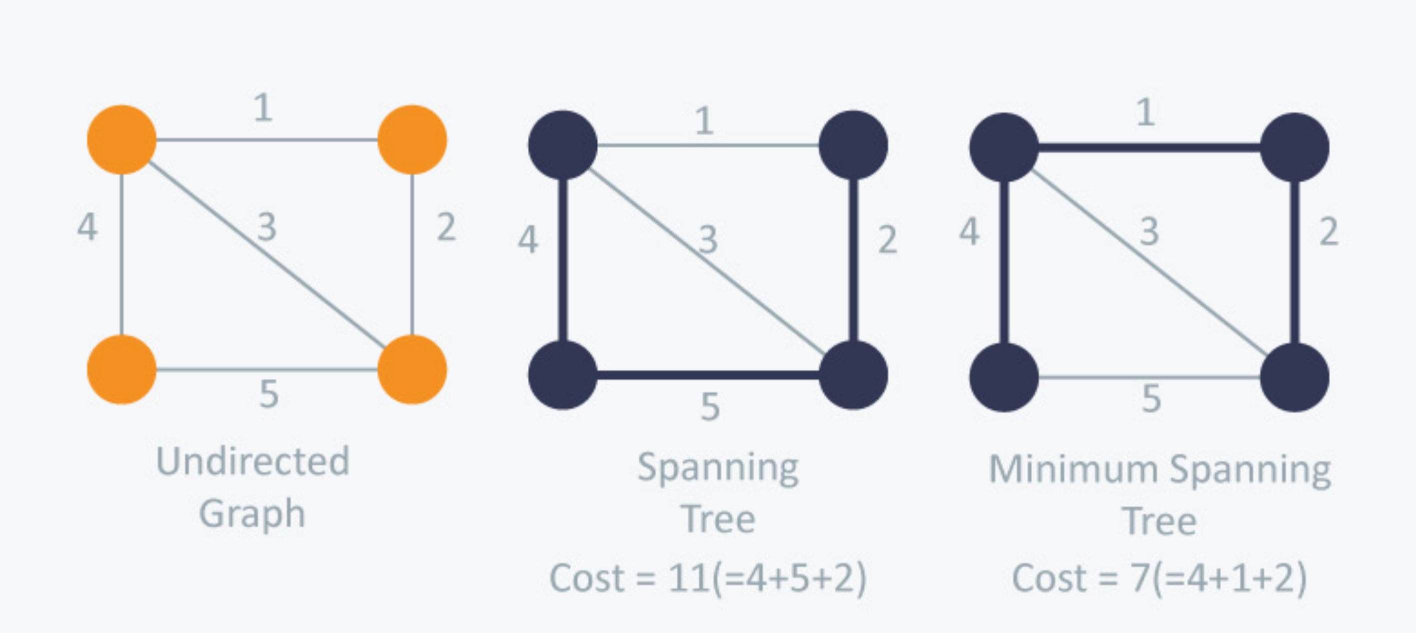
Connected Components Algorithm

- During the execution of CONNECTED-COMPONENTS on an undirected graph with k connected components,
  - How many times is MAKE-SET called?
  - How many times is FIND-SET called?
  - How many times is UNION called?
  - Express your answers in terms of |V|, |E|, and k.

```
CONNECTED-COMPONENTS(G)
for each vertex v ∈ G.V
    MAKE-SET(v)
for each edge (u, v) ∈ G.E
    if FIND-SET(u) ≠ FIND-SET(v)
        UNION(u, v)
```

```
SAME-COMPONENT(u, v)
if FIND-SET(u) == FIND-SET(v)
    return TRUE
else return FALSE
```

MST Example



1
4
3
2
5
Undirected Graph
1
4
3
2
5
Spanning Tree
Cost = 11(=4+5+2)
1
4
3
2
5
Minimum Spanning Tree
Cost = 7(=4+1+2)

Kruskal’s Algorithm

To make a minimum spanning tree:

Sort the edges, smallest to largest

A = empty set

For each edge **e** in sorted order

    If **e** doesn’t cause a cycle, add the edge to set A

Return A

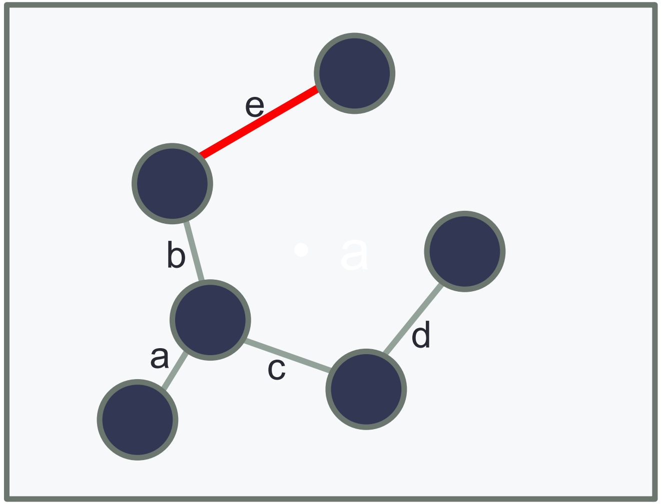
It’s just a loop!

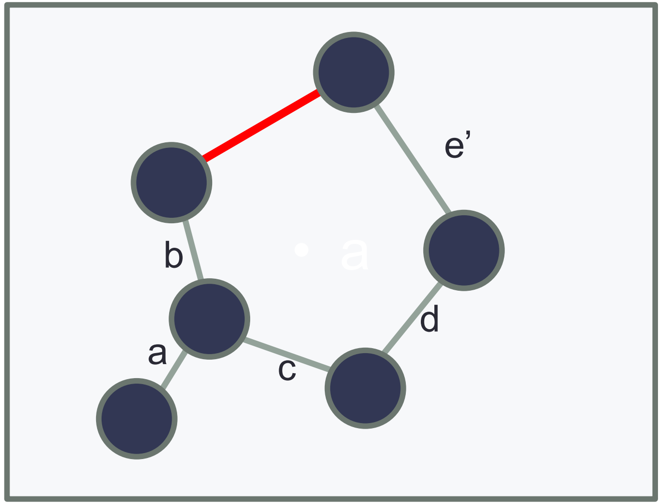
T:
MST Returned by Kruskal's Algorithm

S:
An MST with lower weight than T
W(T) > W(S)

S':
W(e) ≤ W(e')

W(S`) ≤ W(S)





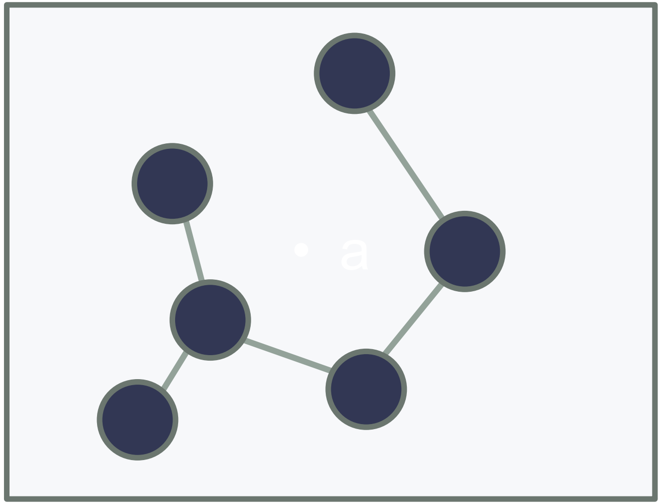
When S' = T:
W(T) ≤ W(S)A contradiction

MST Questions

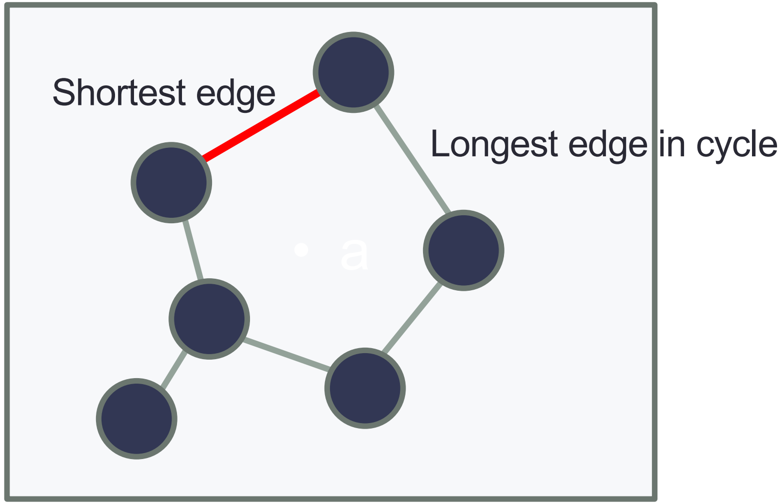
Prove or find a counter example:

1. Must the minimum weight edge be part of a minimum spanning tree?
    1. Must the 2 minimum weight edges?
    2. Must the 3 minimum weight edges?
2. 2 ways of proving?

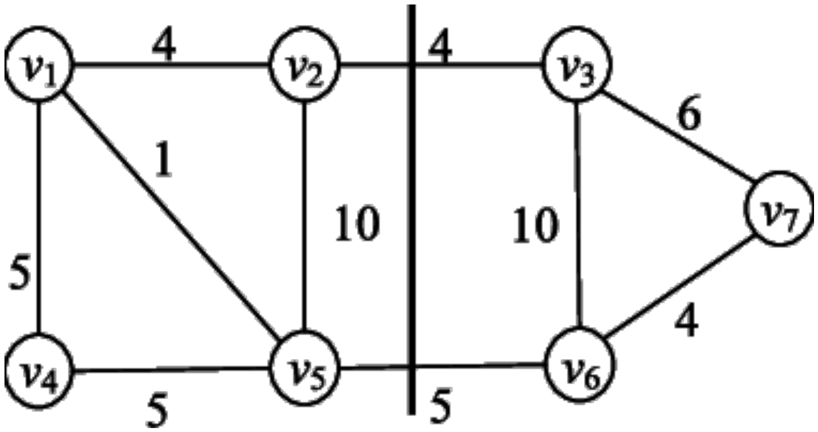
If MST does not include the shortest edge:



1. We can add shortest edge, creating a cycle.
2. Then remove the longest edge in that cycle.
3. We still have a minimum spanning tree, but it is more minimum than the tree without the shortest edge.



Crossing a cut proof



The light edge crossing any cut is part of the MST.

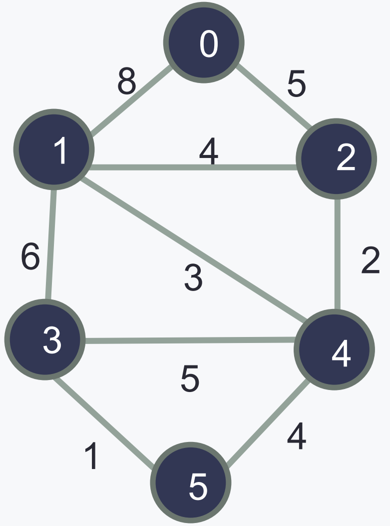
MST Questions

Prove or find a counter example:

1. Can the maximum weight edge be part of a minimum spanning tree?
2. For a given graph, is the minimum spanning tree unique?
    1. If edges are distinct?
    2. If edges are not distinct?

Interview Questions

Run Kruskal's algorithm on this graph.



Edge	Weight
(3,5)	1
(2,4)	2
(1,4)	3
(1,2)	4
(4,5)	4
(0,2)	5
(3,4)	5
(1,3)	6
(0,1)	8

Interview Questions

- Let G be a weighted graph with edge weights greater than one and G' be the graph constructed by squaring the weights of edges in G. How is MST of G different from MST of G’?
- If 5 is added to each edge weight, how does the MST change?

Interview Questions

- Let G be a connected undirected graph of 100 vertices and 300 edges. The weight of a minimum spanning tree of G is 500. When the weight of each edge of G is increased by five, what is the weight of a minimum spanning tree?
- Suppose we have an undirected graph with weights that can be either positive or negative. Does Kruskal’s algorithm produce a MST for such a graph?
- What is time complexity of Kruskal’s algorithm?
  - $\lg x^y = y \lg x \rightarrow |E| = |V|^2 \rightarrow E \lg E \rightarrow E \lg V^2 \rightarrow 2 E \lg V \rightarrow E \lg V$

Interview questions

- Suppose that all edge weights in a graph are integers in the range from 1 to |V|.
- How fast can you make Kruskal's algorithm run?
- How about if we just redo the weights to be from 1 to |V|?
- Can we "throw a hashmap at it"?

```
MST-KRUSKAL(G, w)
A = ∅
for each vertex v ∈ G.V
    MAKE-SET(v)
sort the edges of G.E into nondecreasing order by weight w
for each edge (u, v) ∈ G.E, taken in nondecreasing order by weight
    if FIND-SET(u) ≠ FIND-SET(v)
        A = A ∪ {(u, v)}
        UNION(u, v)
return A
```

The Opposite of Kruskal?

a. MAYBE-MST-A(G, w)

sort the edges into nonincreasing order of edge weights w
$T = E$
for each edge e, taken in nonincreasing order by weight
    if $T - \{e\}$ is a connected graph
        $T = T - \{e\}$
return T

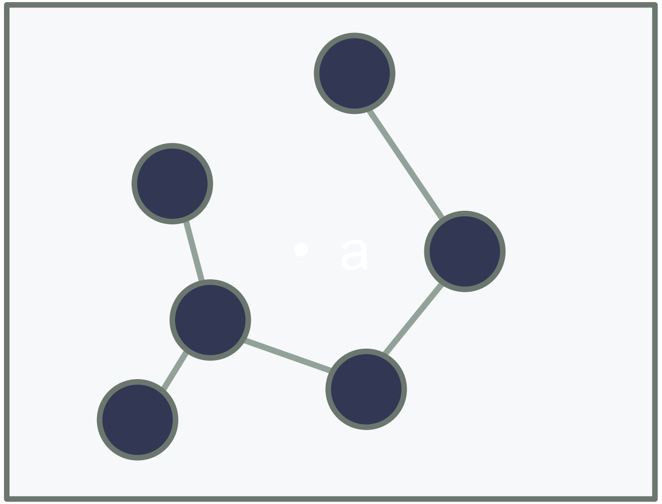
Is it a spanning tree?
Is it minimal?

More random than Kruskal?

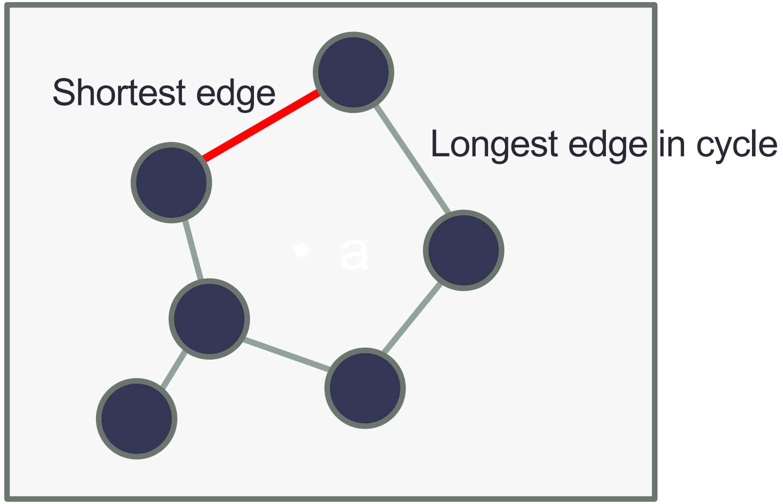
b. MAYBE-MST-B(G, w)

```
T = ∅
for each edge e, taken in arbitrary order
    if T ∪ {e} has no cycles
        T = T ∪ {e}
return T
```

If MST does not include the shortest edge:



1. We can add shortest edge, creating a cycle.
2. Then remove the longest edge in that cycle.
3. We still have a minimum spanning tree, but it is more minimum than the tree without the shortest edge.



Another idea?

c. MAYBE-MST-C(G, w)

$T = \emptyset$
for each edge e, taken in arbitrary order
    $T = T \cup \{e\}$
    if T has a cycle c
        let e' be a maximum-weight edge on c
        $T = T - \{e'\}$
return T

How to program this??

Interview Question

- If we have an MST and the weight of an edge e is reduced, what is an efficient way to calculate the new MST?

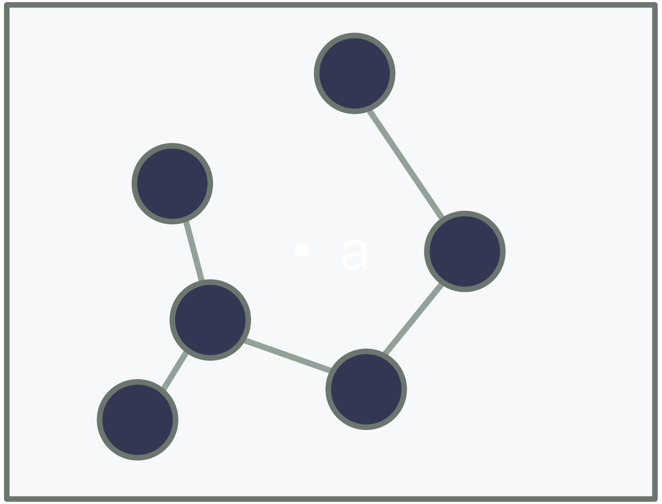
Interview Question

- What if we have a graph G and an MST T. The weight of an edge that is not part of the current MST is increased. How can we compute the new MST?

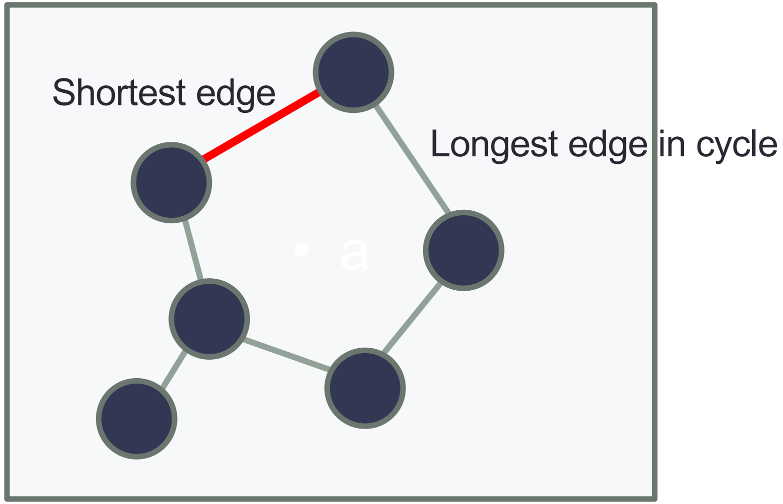
Interview Question

- Describe an efficient algorithm to update the minimum spanning tree when the weight of one edge e not in T is decreased.
  - You may want to add this edge to the MST.
  - How can you decide?

If MST does not include the shortest edge:



1. We can add shortest edge, creating a cycle.
2. Then remove the longest edge in that cycle.
3. We still have a minimum spanning tree, but it is more minimum than the tree without the shortest edge.



Run Kruskal's Algorithm