CS2040S
Data Structures and Algorithms
(e-learning edition)

All about minimum spanning trees…

Roadmap

Today: Minimum Spanning Trees

- Prim’s Algorithm
- Kruskal’s Algorithm
- Boruvka’s Algorithm

Variations:

- Constant weight edges
- Bounded integer edge weights
- Directed graphs
- Maximum Spanning Tree
- Steiner Tree

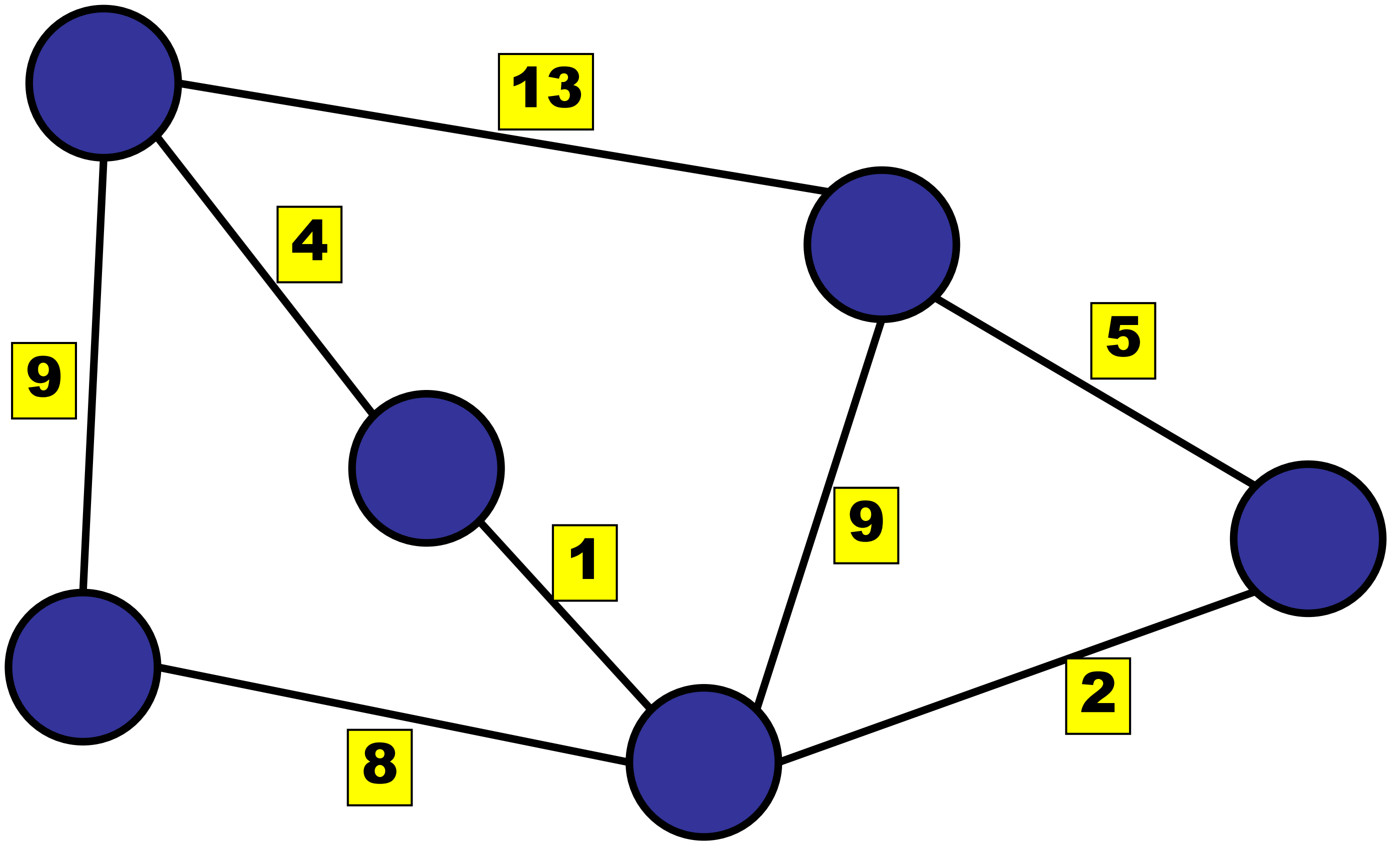
Roadmap

Minimum Spanning Trees

- **The MST Problem**
- Basic Properties of an MST
- Generic MST Algorithm
- Prim's Algorithm
- Kruskal's Algorithm
- Boruvka's Algorithm
- Variations

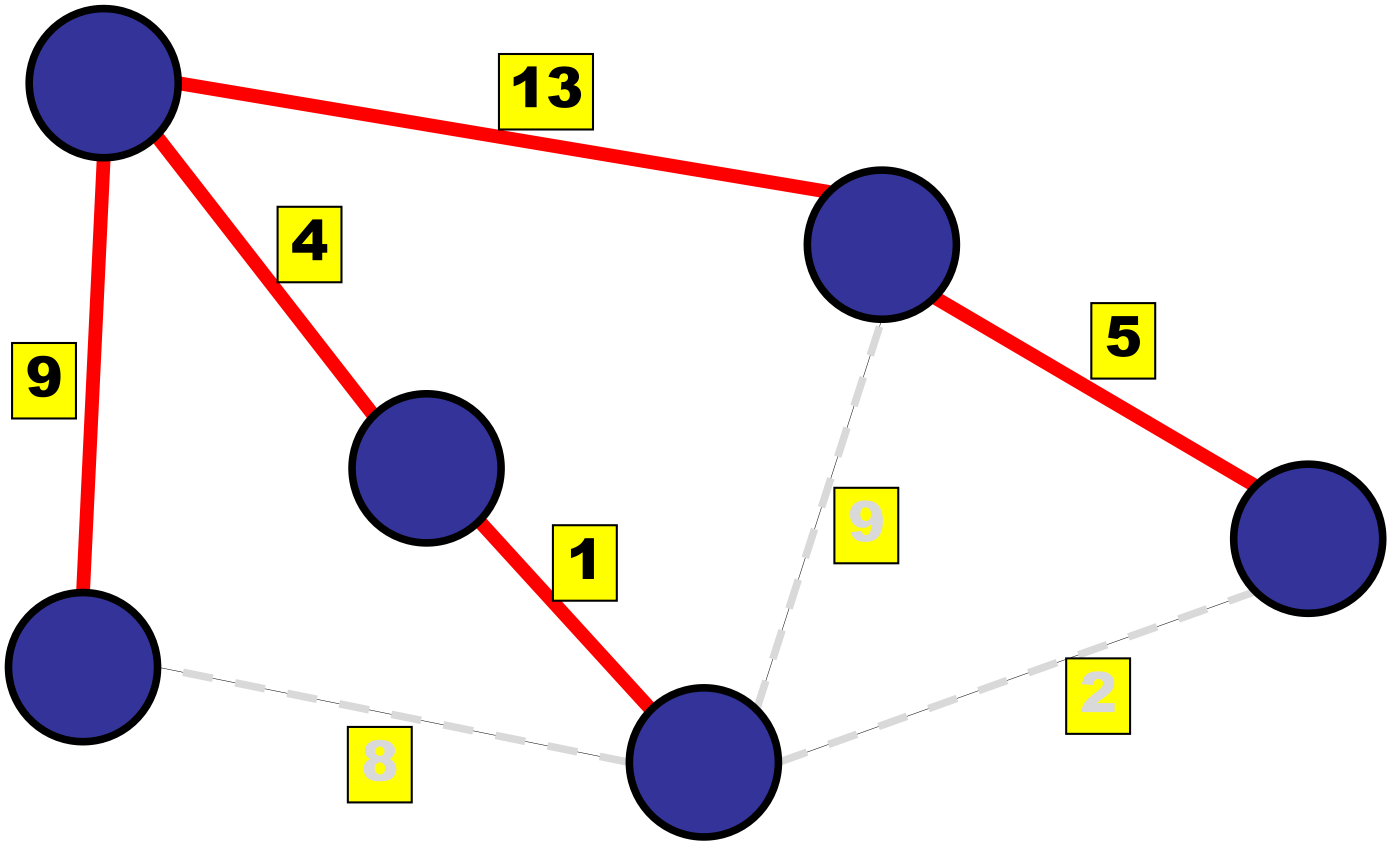
Spanning Tree

Weighted, undirected graph:



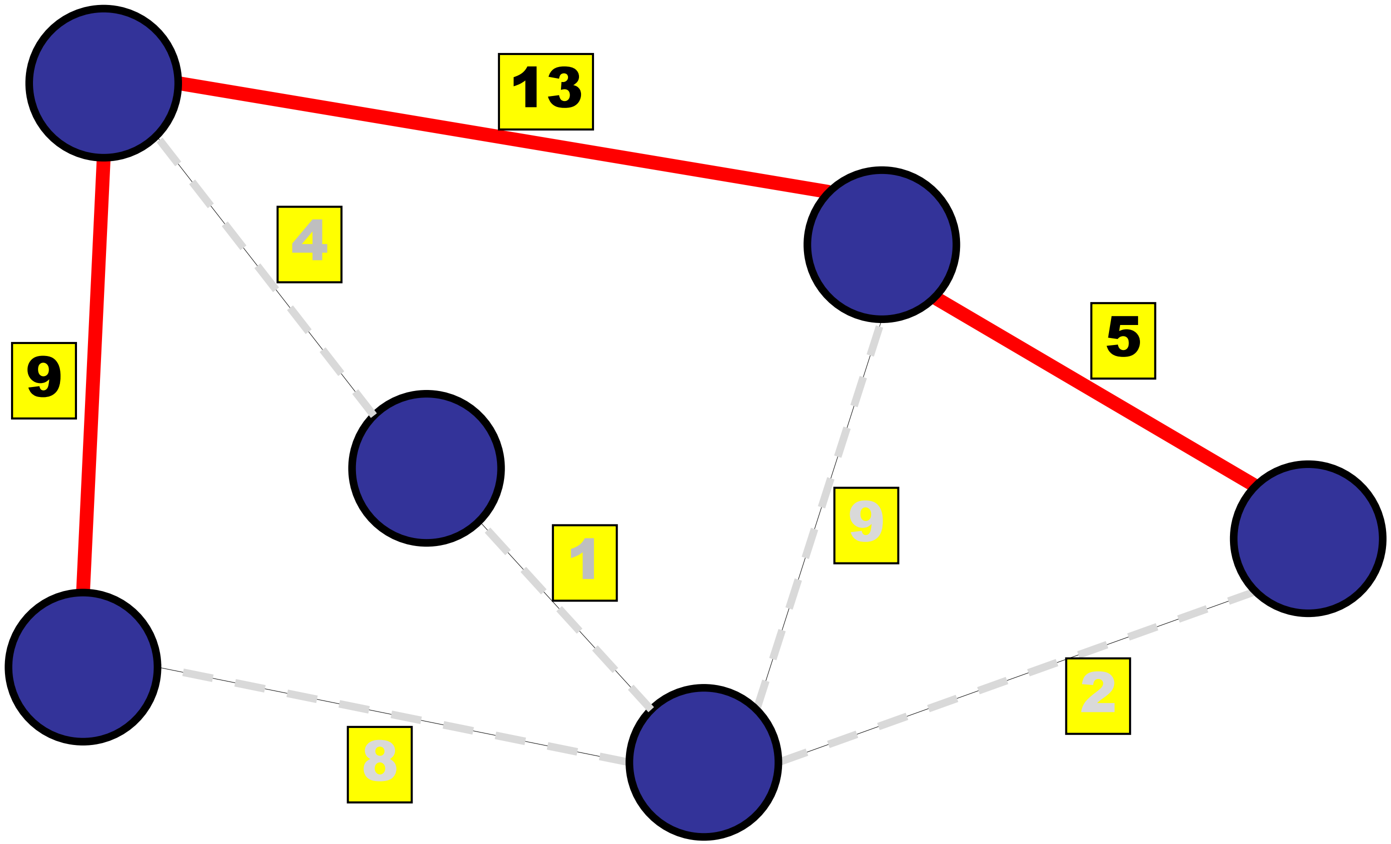
Spanning Tree

Definition: a spanning tree is an acyclic subset of the edges that connects all nodes



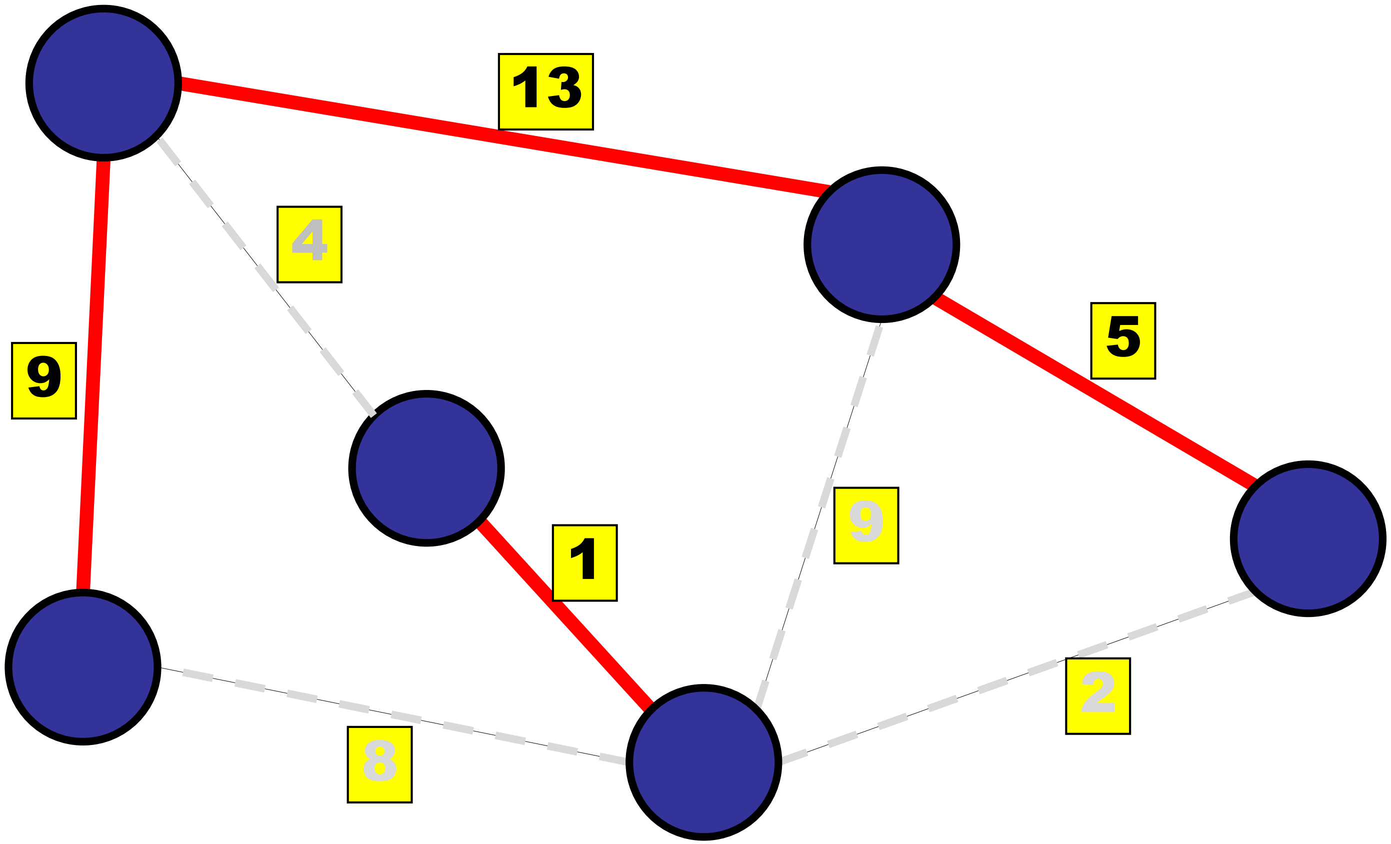
Spanning Tree

NOT a spanning tree...



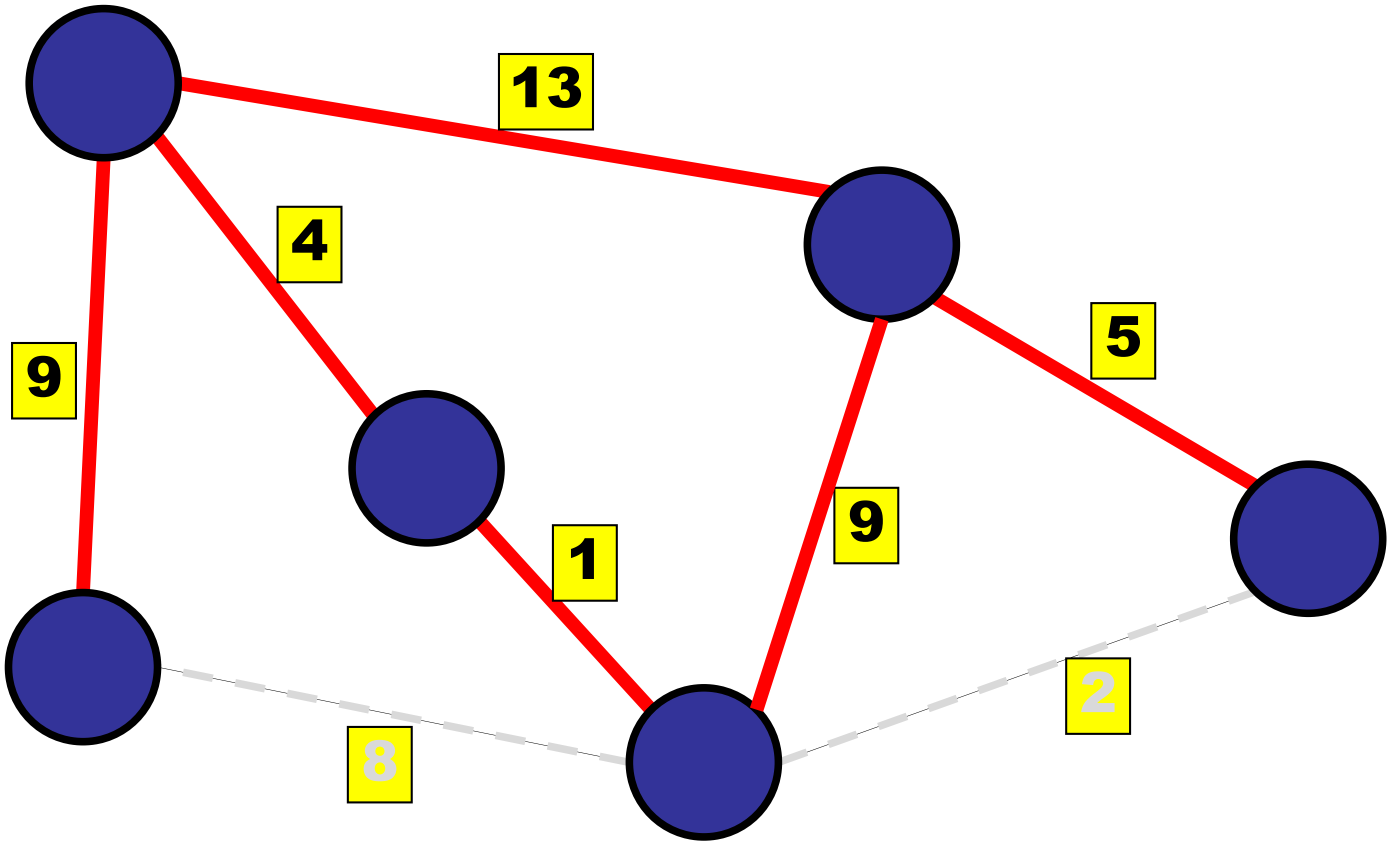
Spanning Tree

NOT a spanning tree…



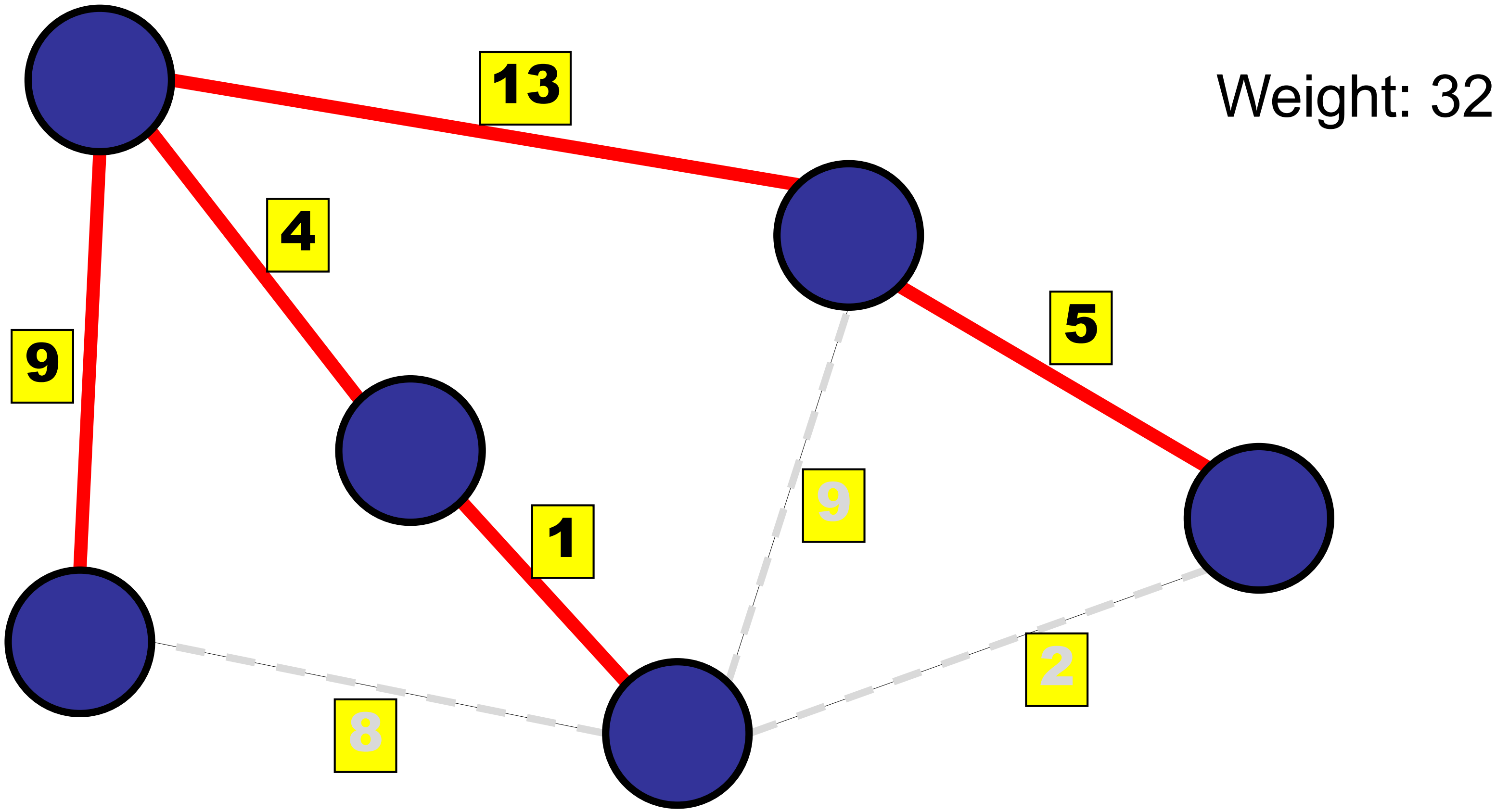
Spanning Tree

NOT a spanning tree…



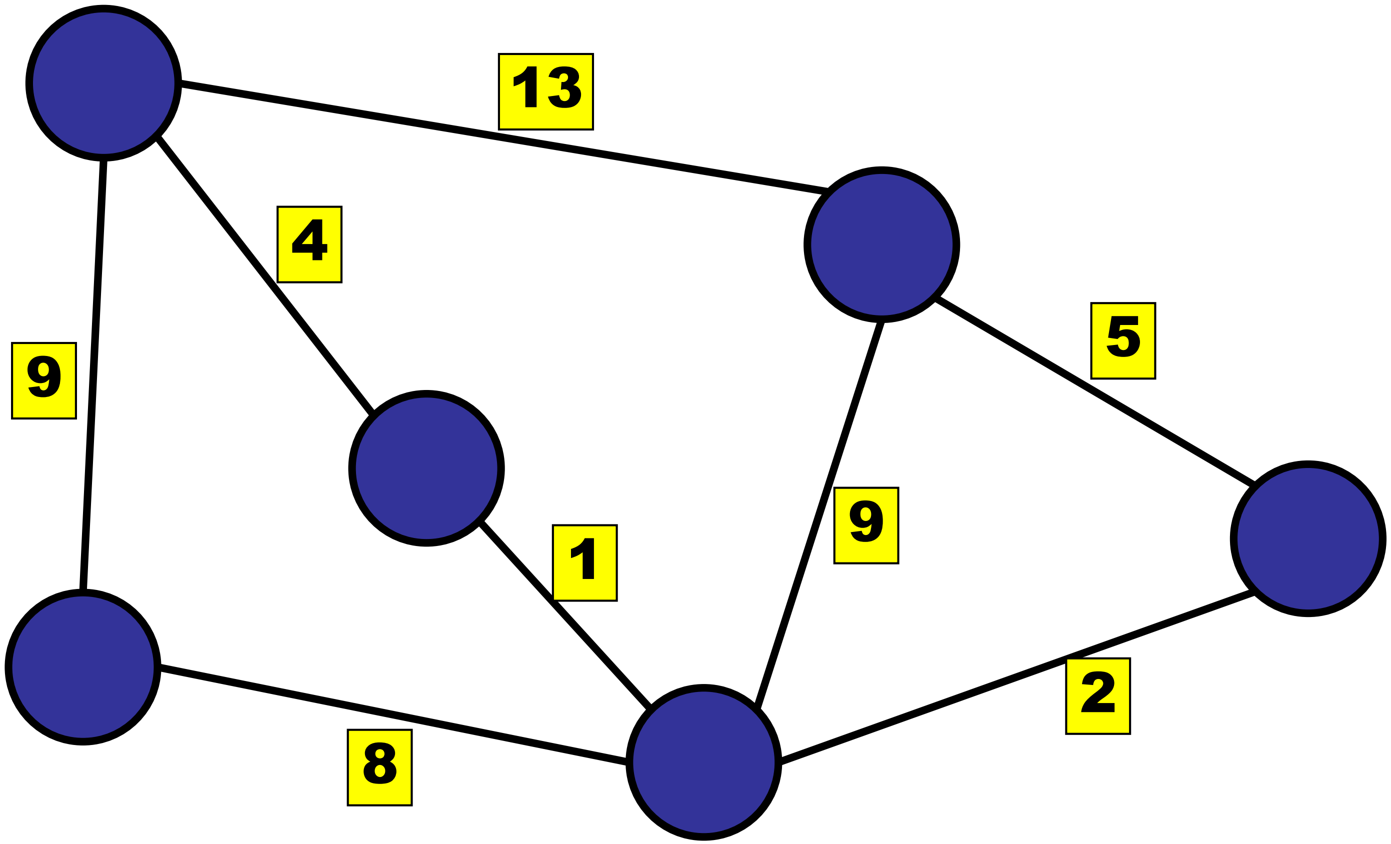
Spanning Tree

Definition: a spanning tree is an acyclic subset of the edges that connects all nodes



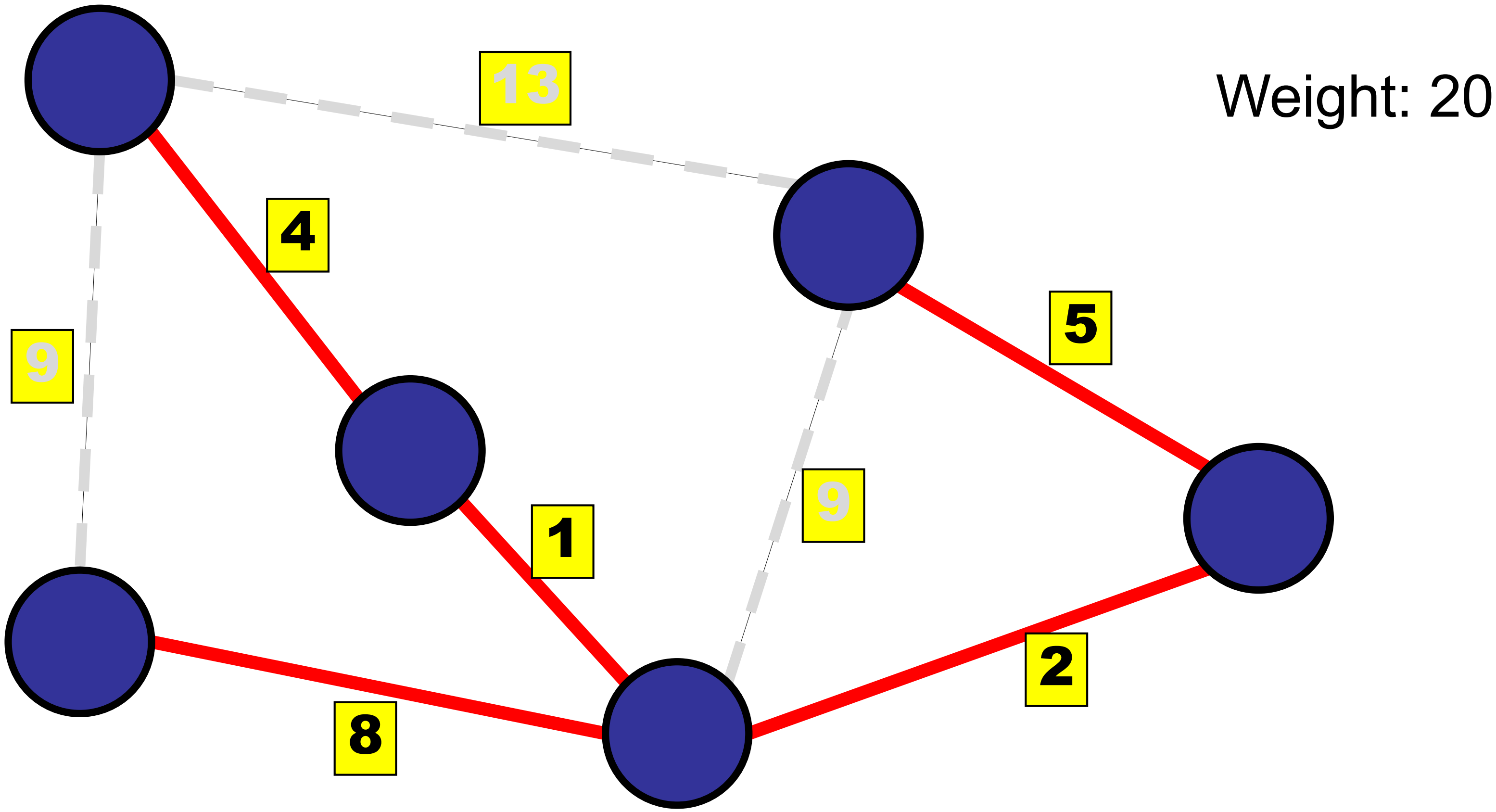
Minimum Spanning Tree

Definition: a spanning tree with minimum weight



Minimum Spanning Tree

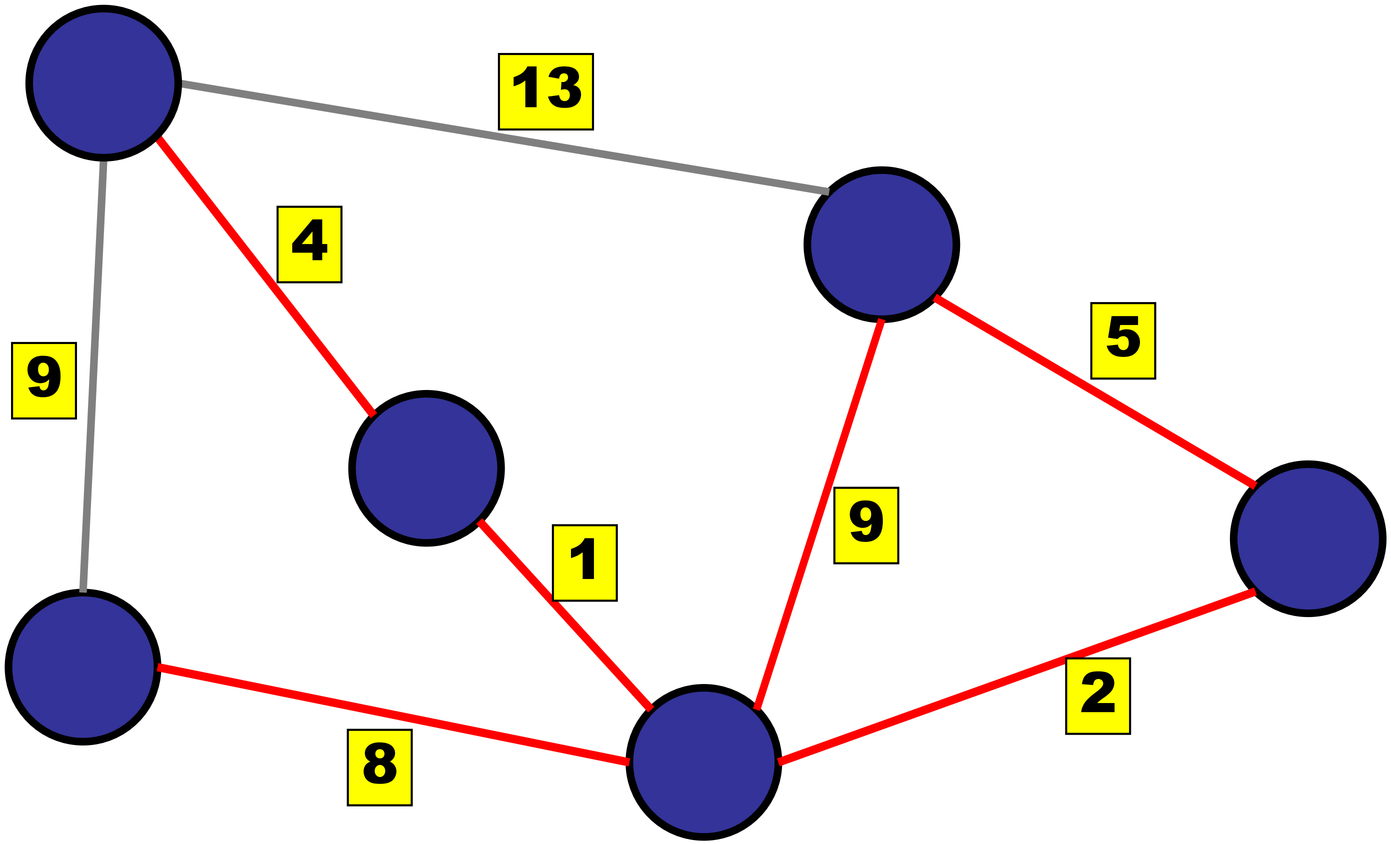
Definition: a spanning tree with minimum weight



Minimum Spanning Tree

Note: no cycles

Why? If there were cycles, we could remove one edge and reduce the weight!

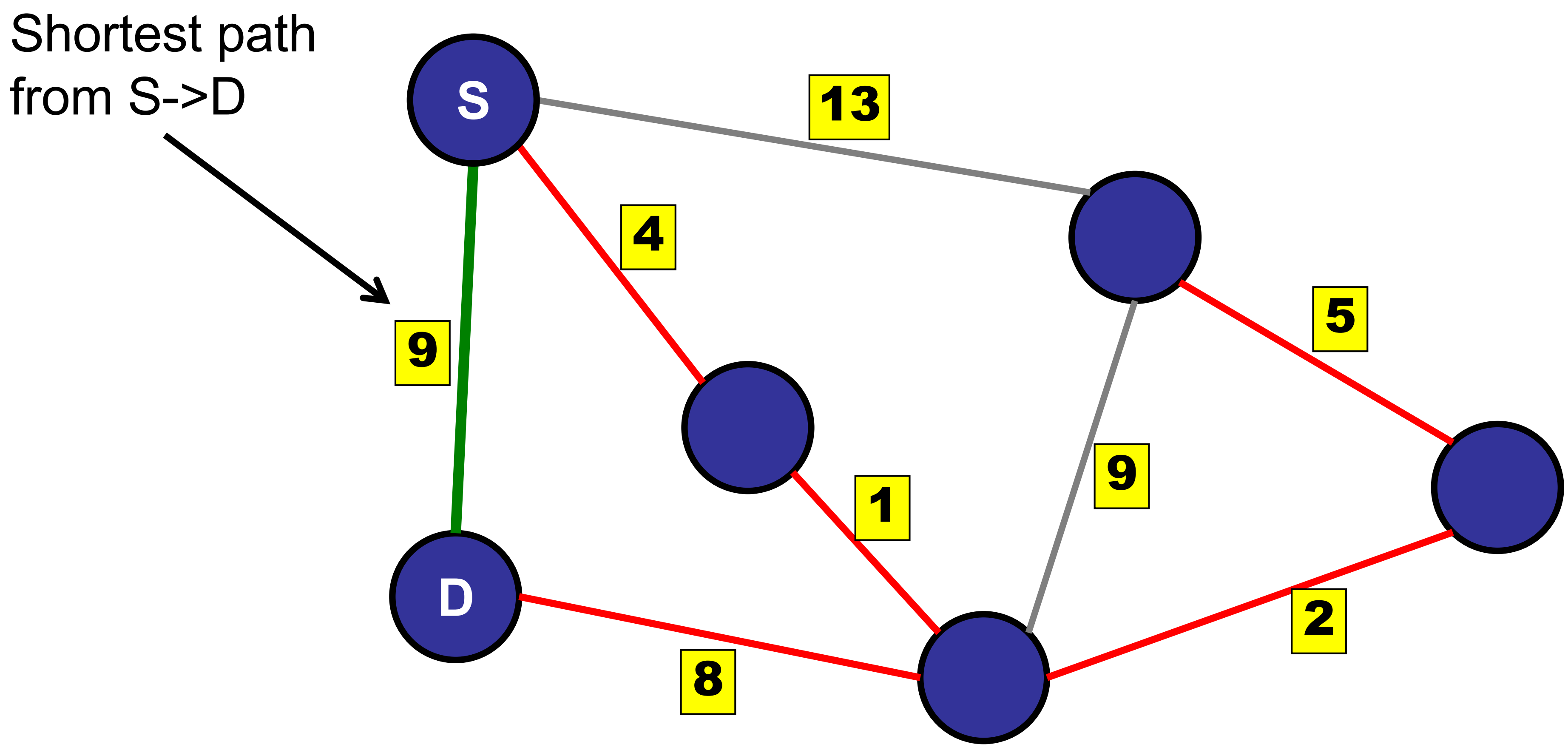


Can we use MST to find shortest paths?

1. Yes
2. Only on connected graphs.
3. Only on dense graphs.
4. No.
5. I need to see a picture.

Minimum Spanning Tree

Not the same a shortest paths:



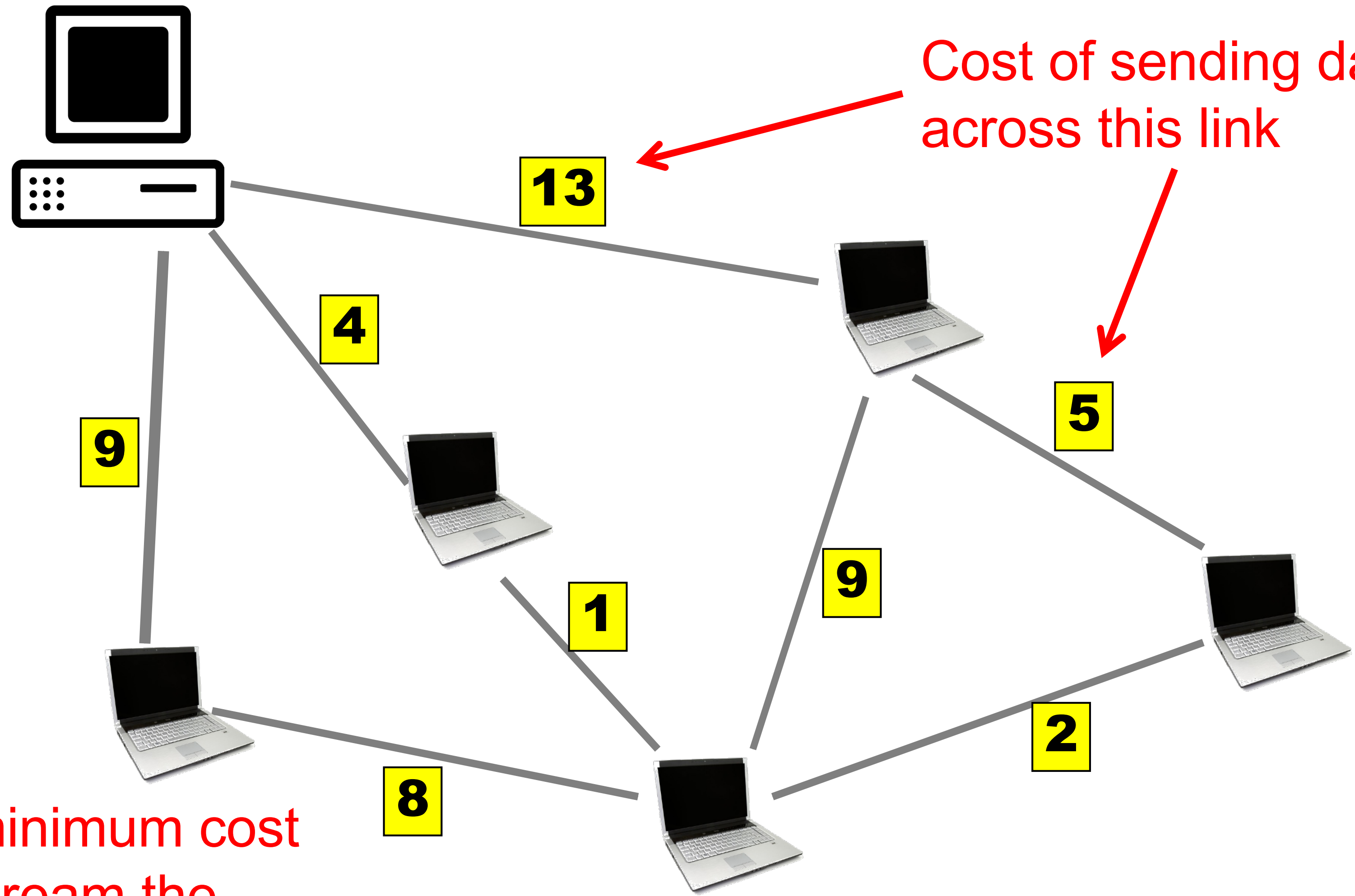
Applications of MST

Many applications:

- Network design
  - Telephone networks
  - Electrical networks
  - Computer networks
  - Ethernet autoconfig
  - Road networks
  - Bottleneck paths

Data distribution

Stream a movie over the internet:



Cost of sending data across this link

What is the minimum cost **network** to stream the movie to all the users?

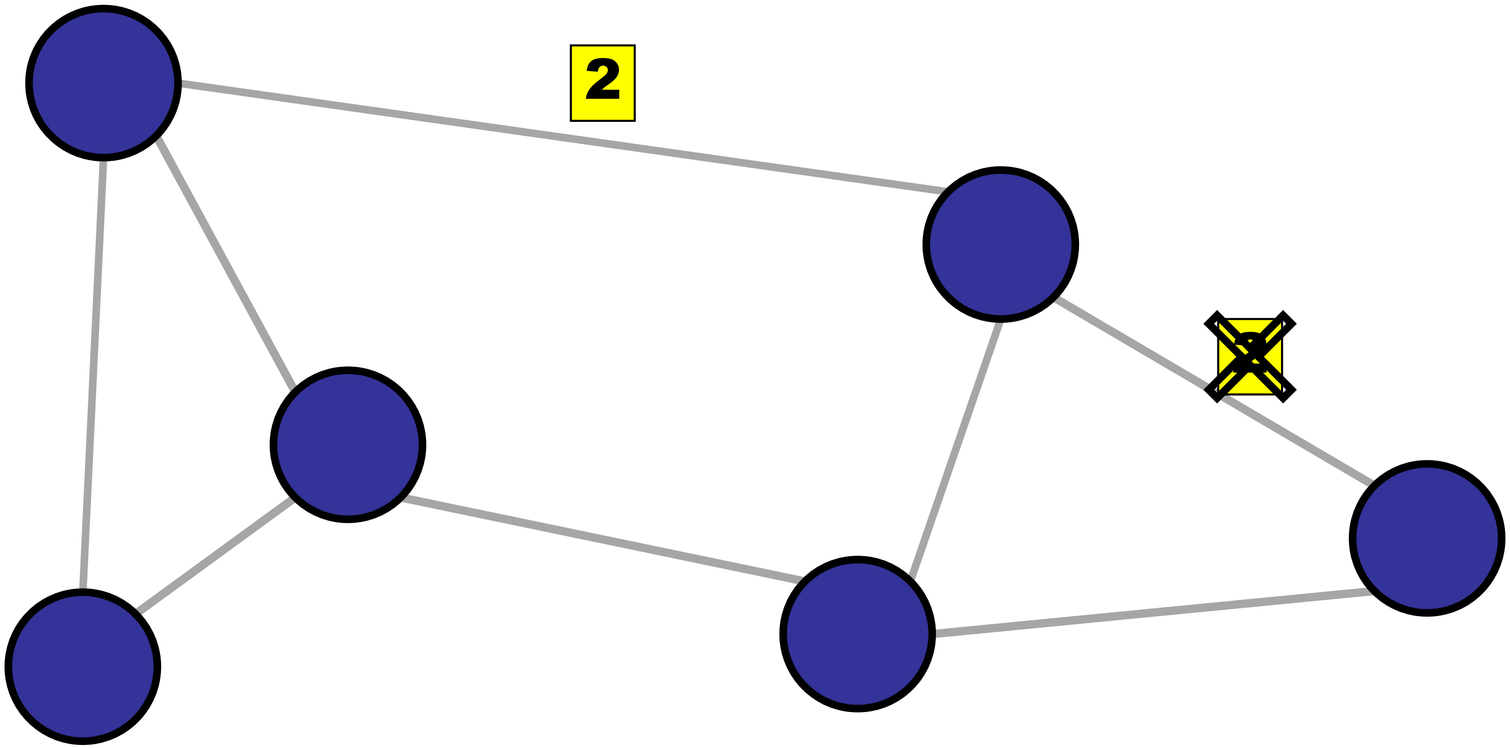
Applications of MST

Many applications:

- Many other
  - Error correcting codes
  - Face verification
  - Cluster analysis
  - Image registration

Assumption

All edge weights are distinct. (Simplification…)



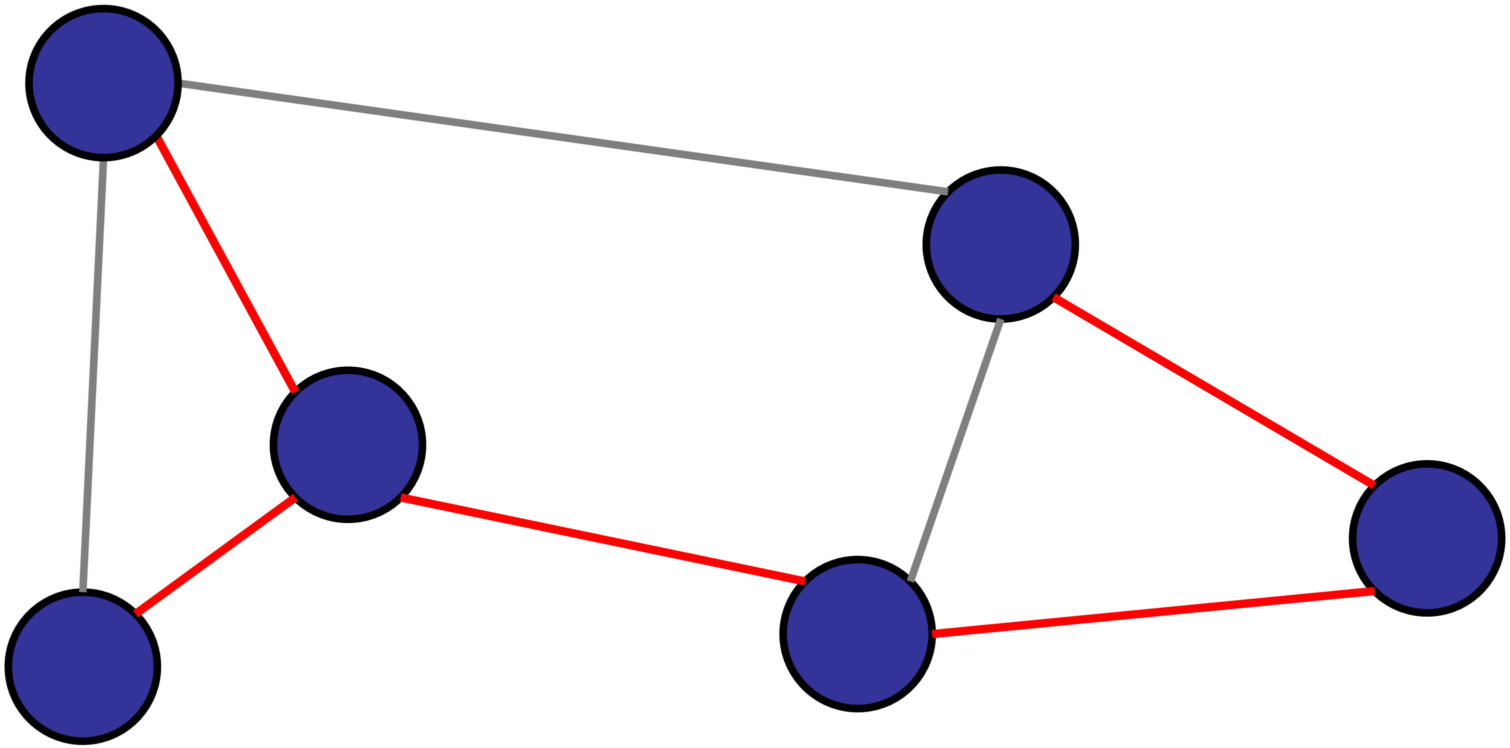
Roadmap

Minimum Spanning Trees

- The MST Problem
- **Basic Properties of an MST**
- Generic MST Algorithm
- Prim’s Algorithm
- Kruskal’s Algorithm
- Boruvka’s Algorithm
- Variations

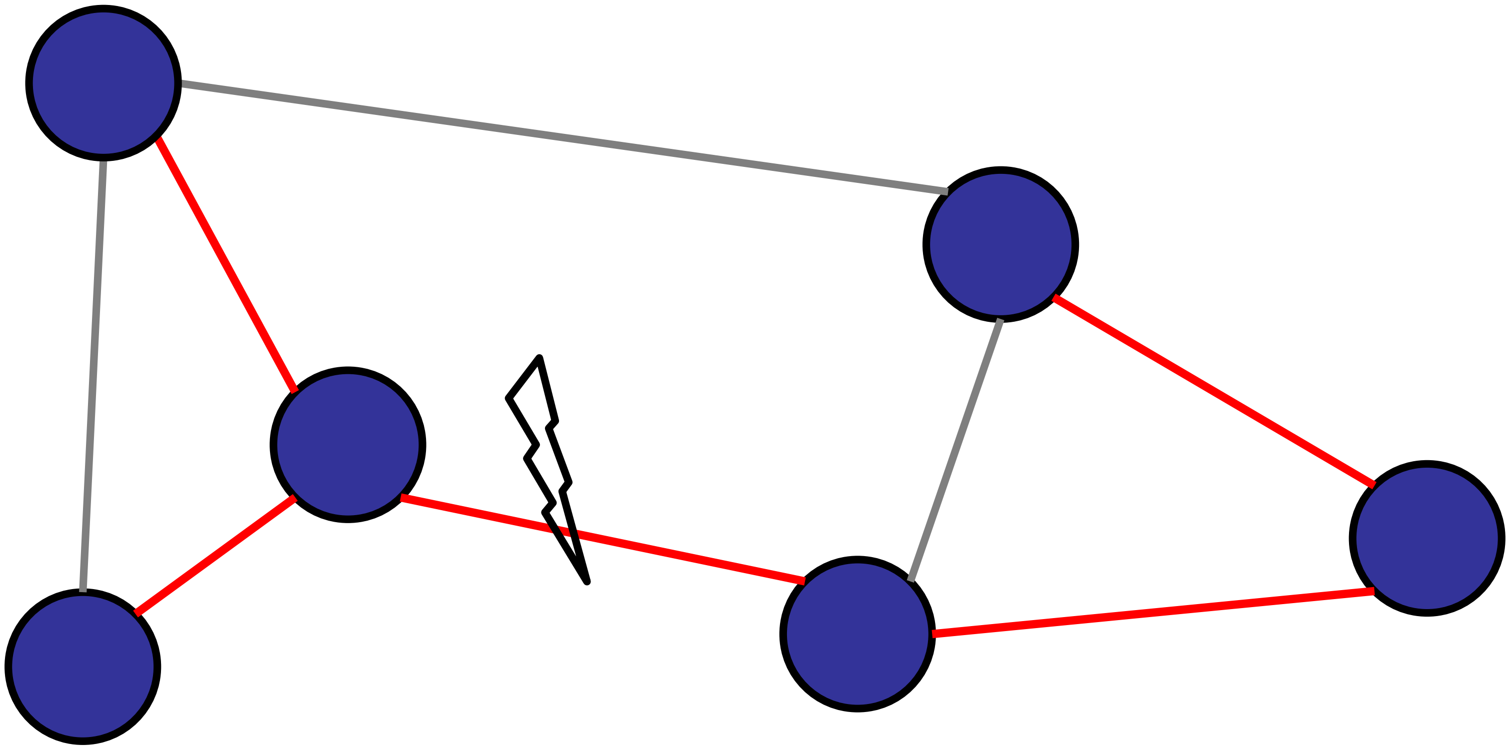
Properties of MST

Property 1: No cycles



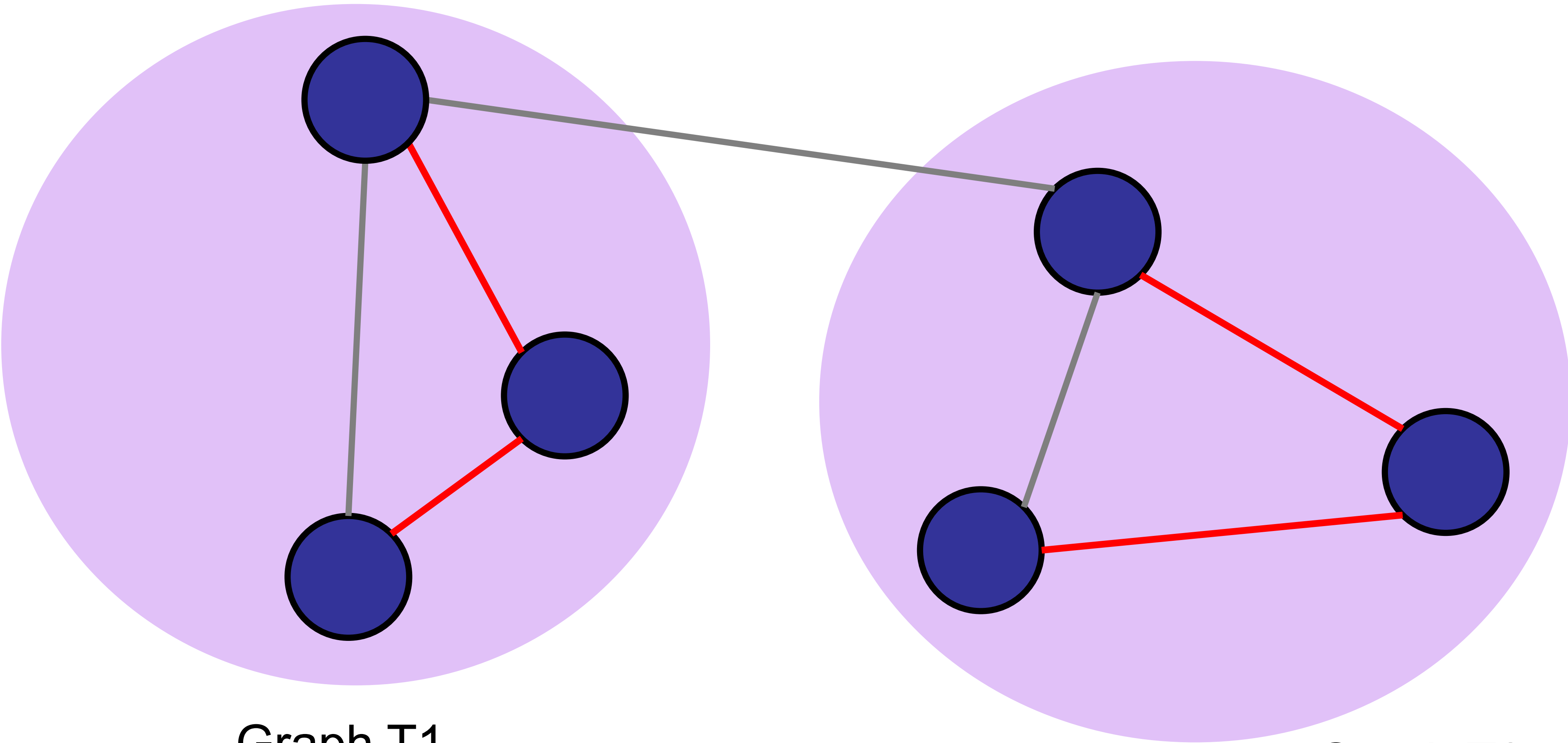
Properties of MST

What happens if you cut an MST?



Properties of MST

What happens if you cut an MST?

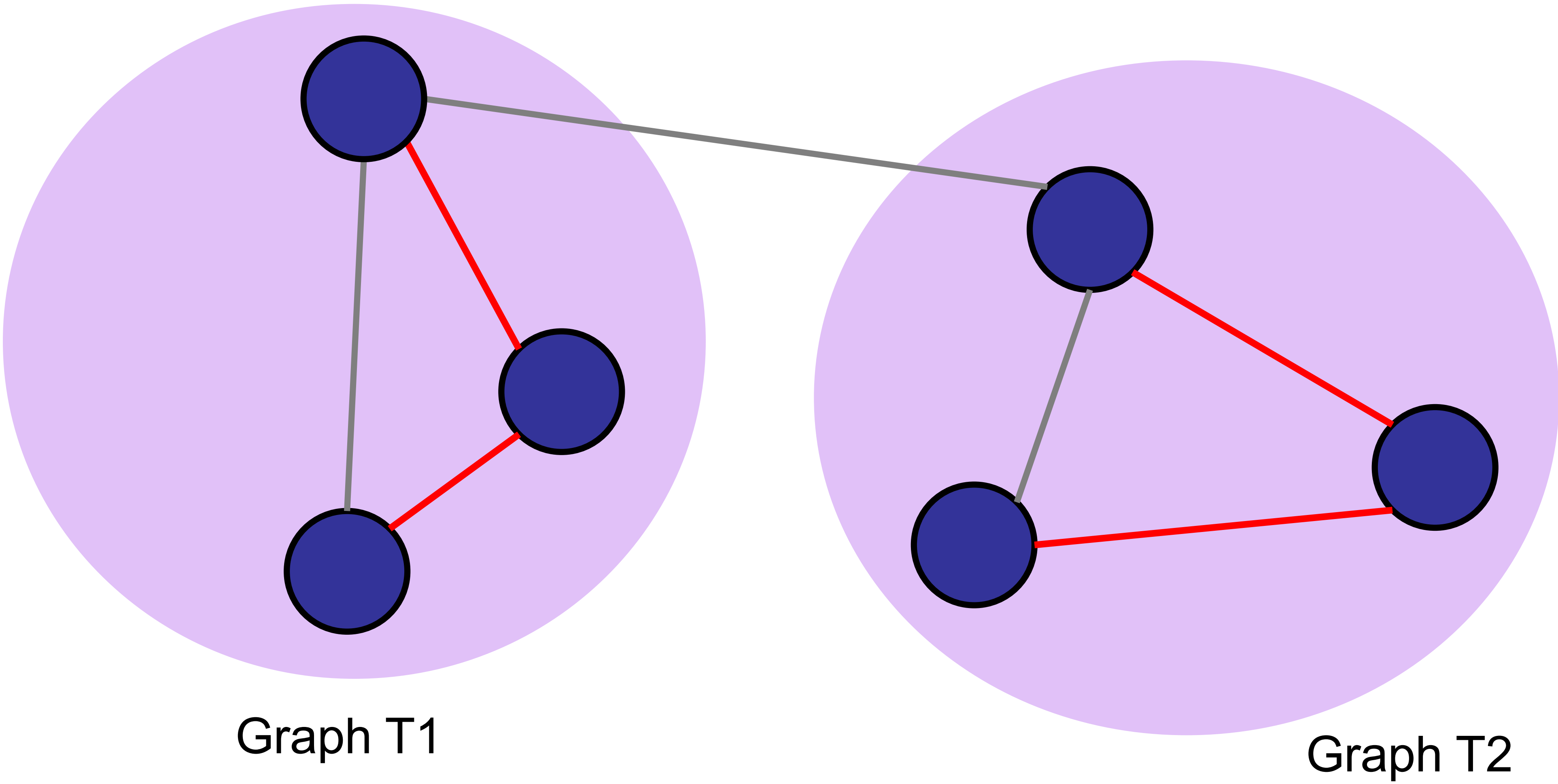


Graph T1

Graph T2

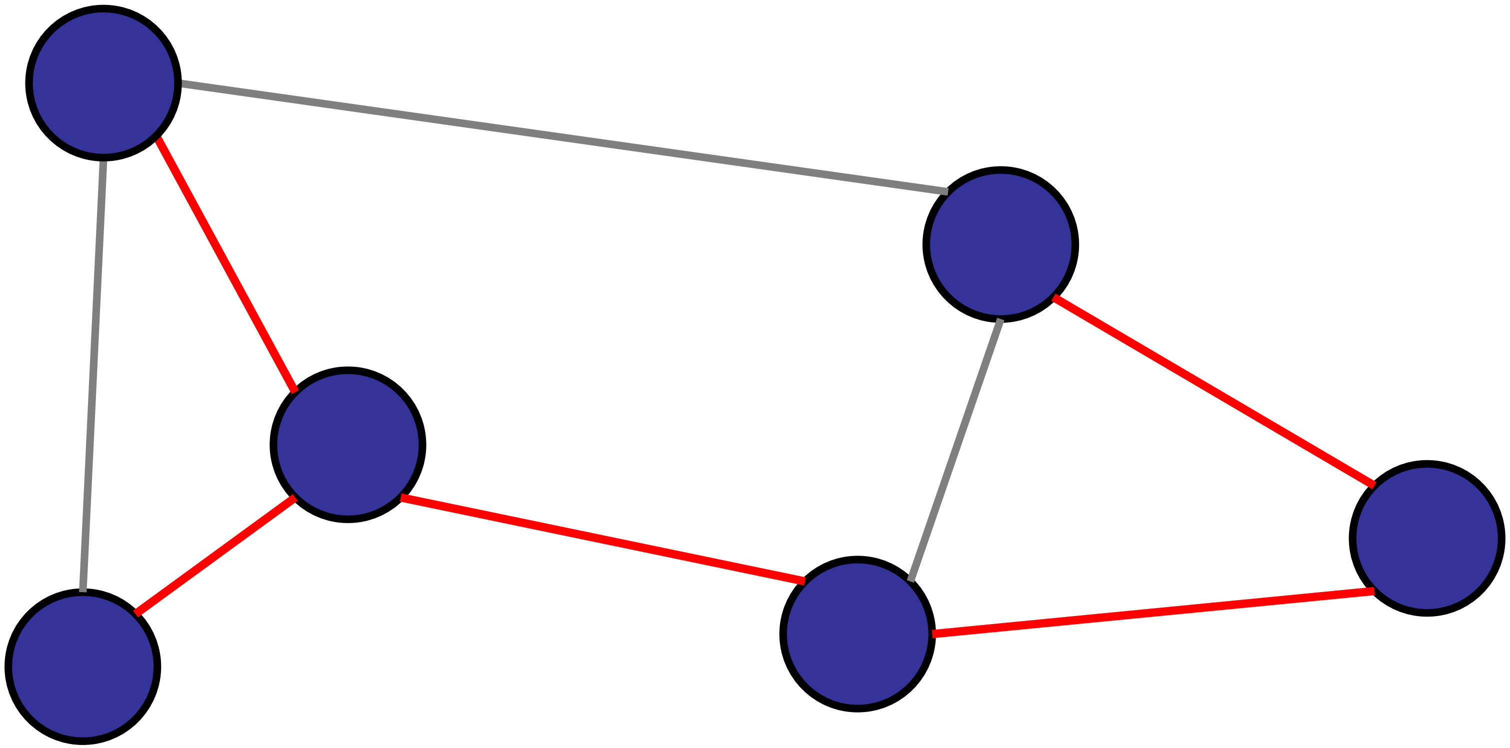
Properties of MST

Theorem: T1 is an MST and T2 is an MST.



Properties of MST

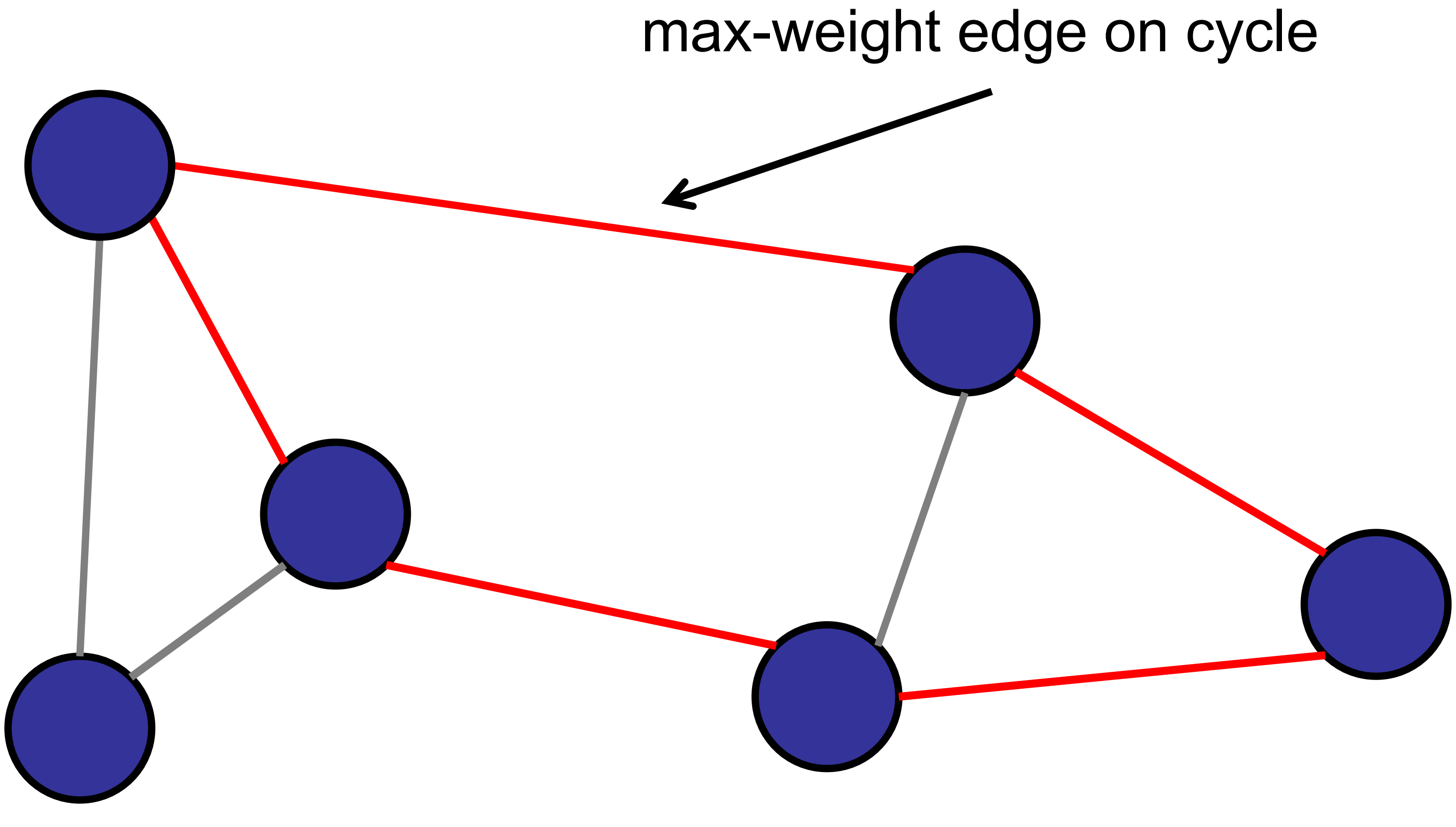
Property 2: If you cut an MST, the two pieces are both MSTs.



Overlapping sub-problems! Dynamic programming? Yes, but better…

Properties of MST

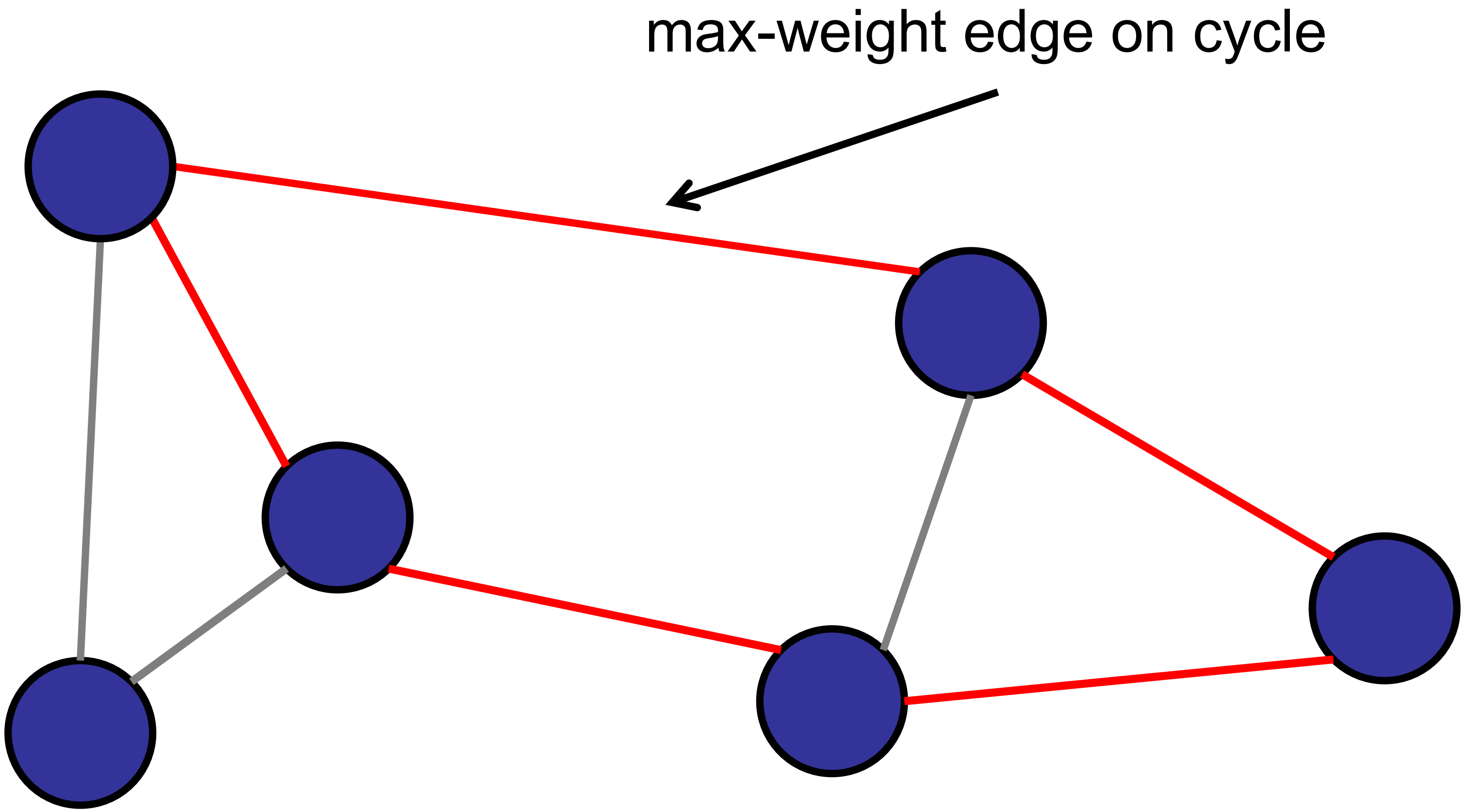
Property 3: Cycle property



Properties of MST

Property 3: Cycle property

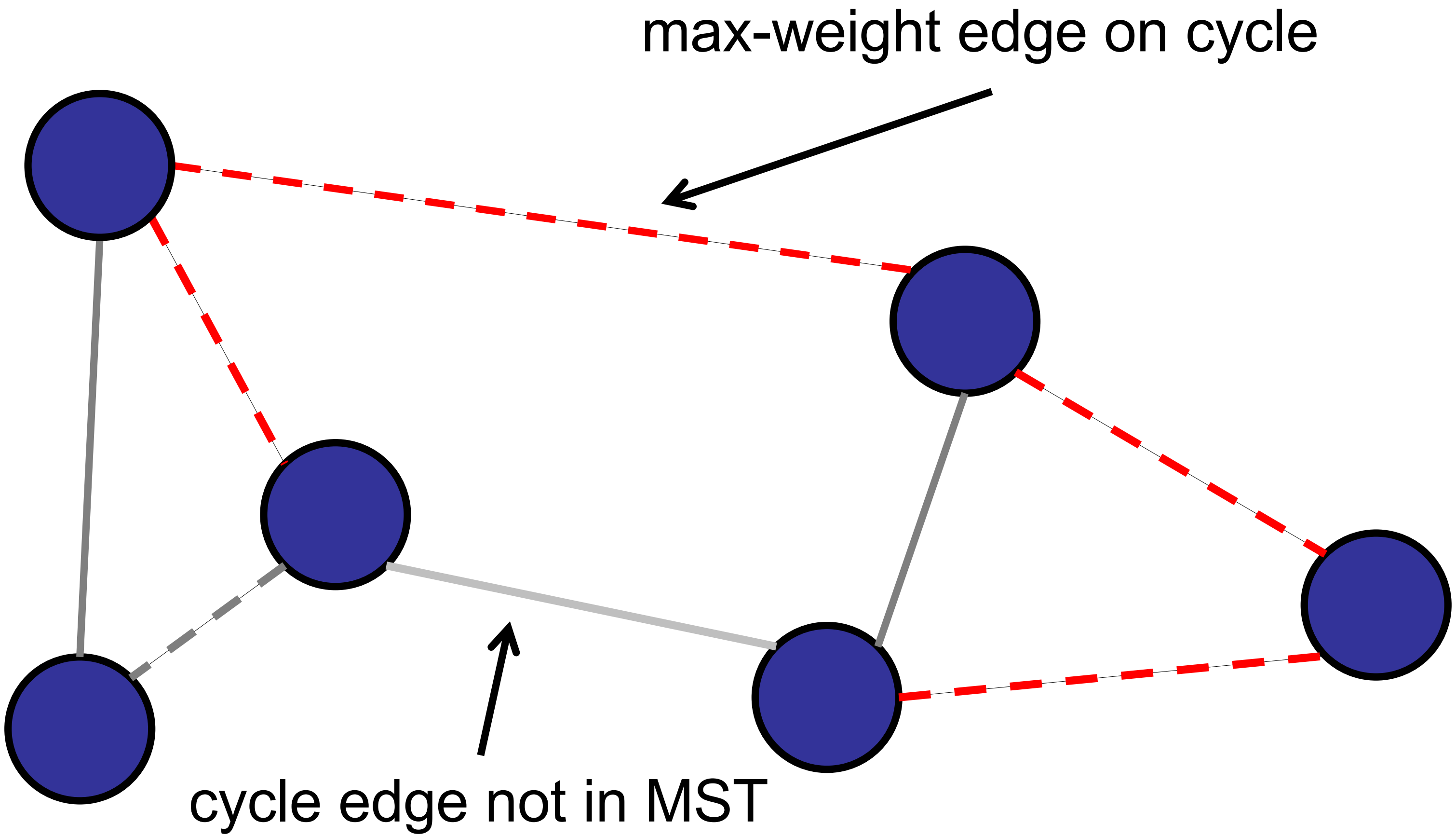
For every cycle, the maximum weight edge is ***not*** in the MST.



Properties of MST

Proof: Cut-and-paste

Assume heavy edge is in the MST.

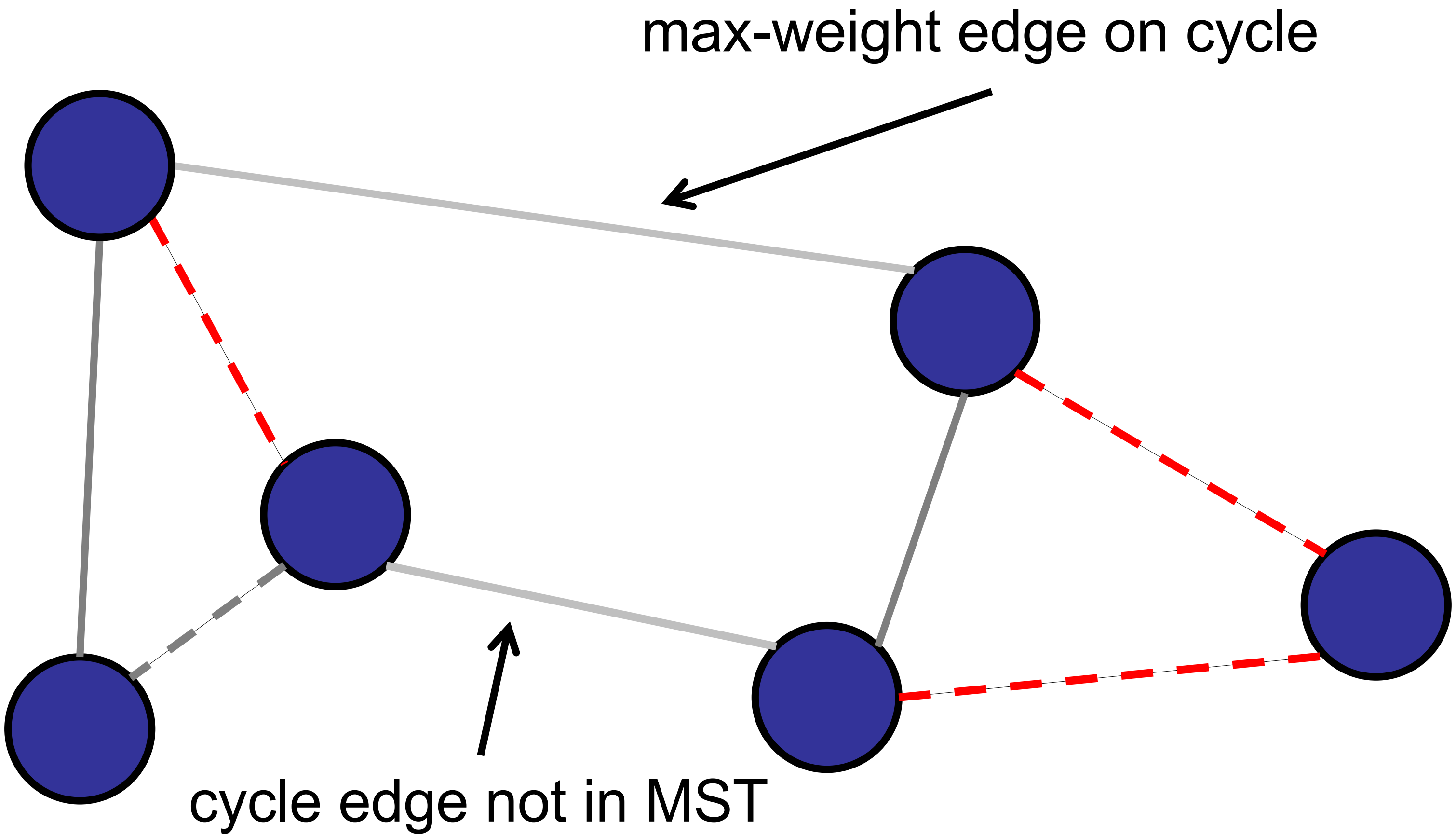


Properties of MST

Proof: Cut-and-paste

Assume heavy edge is in the MST.

Remove max-weight edge; cuts graph.

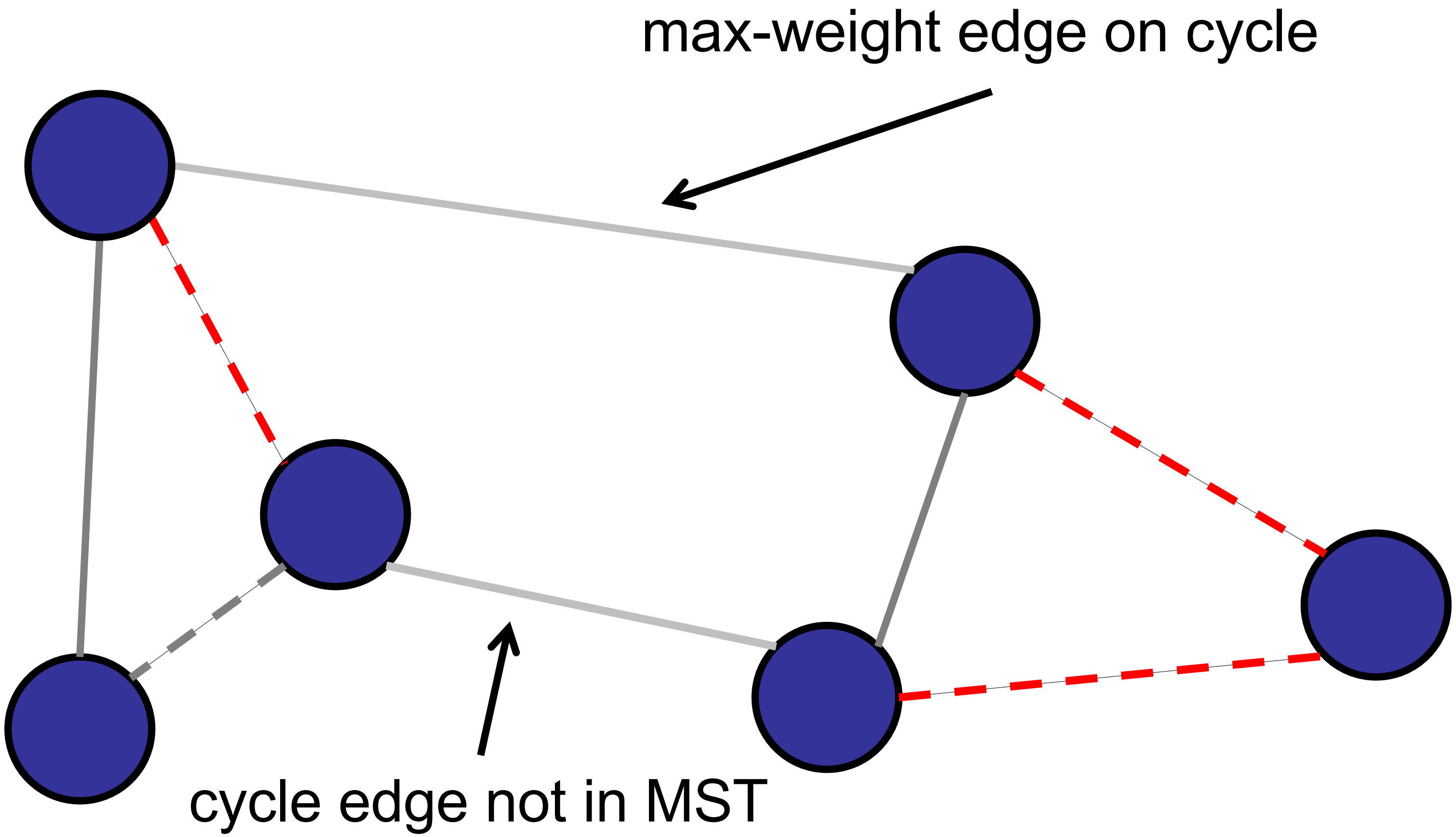


Properties of MST

Proof: Cut-and-paste

Some other cycle edge crosses the cut.

(Even # of cycle edges across cut.)

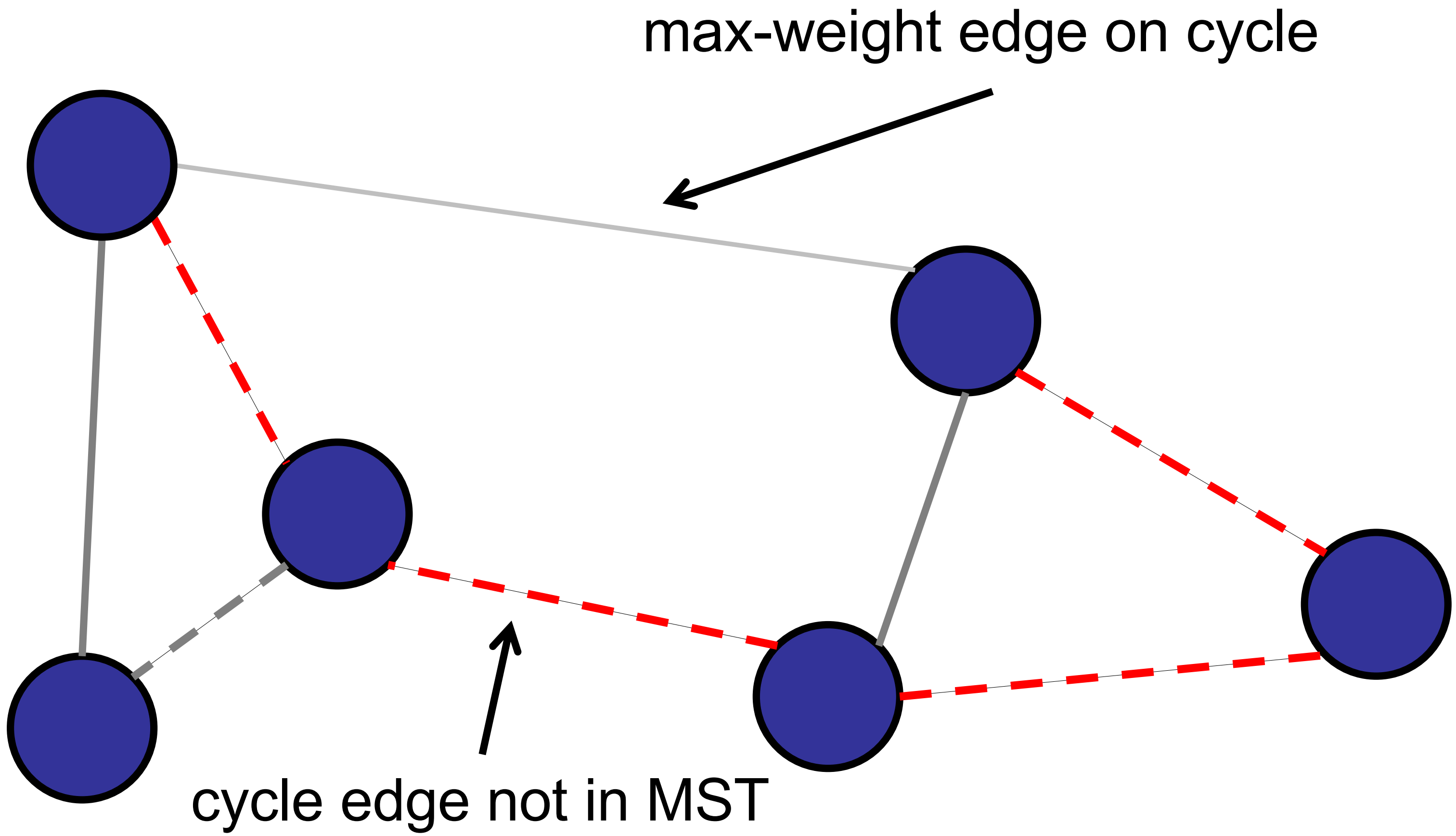


Properties of MST

Proof: Cut-and-paste

Replace heavy edge with lighter edge.

Re-creates a spanning tree.

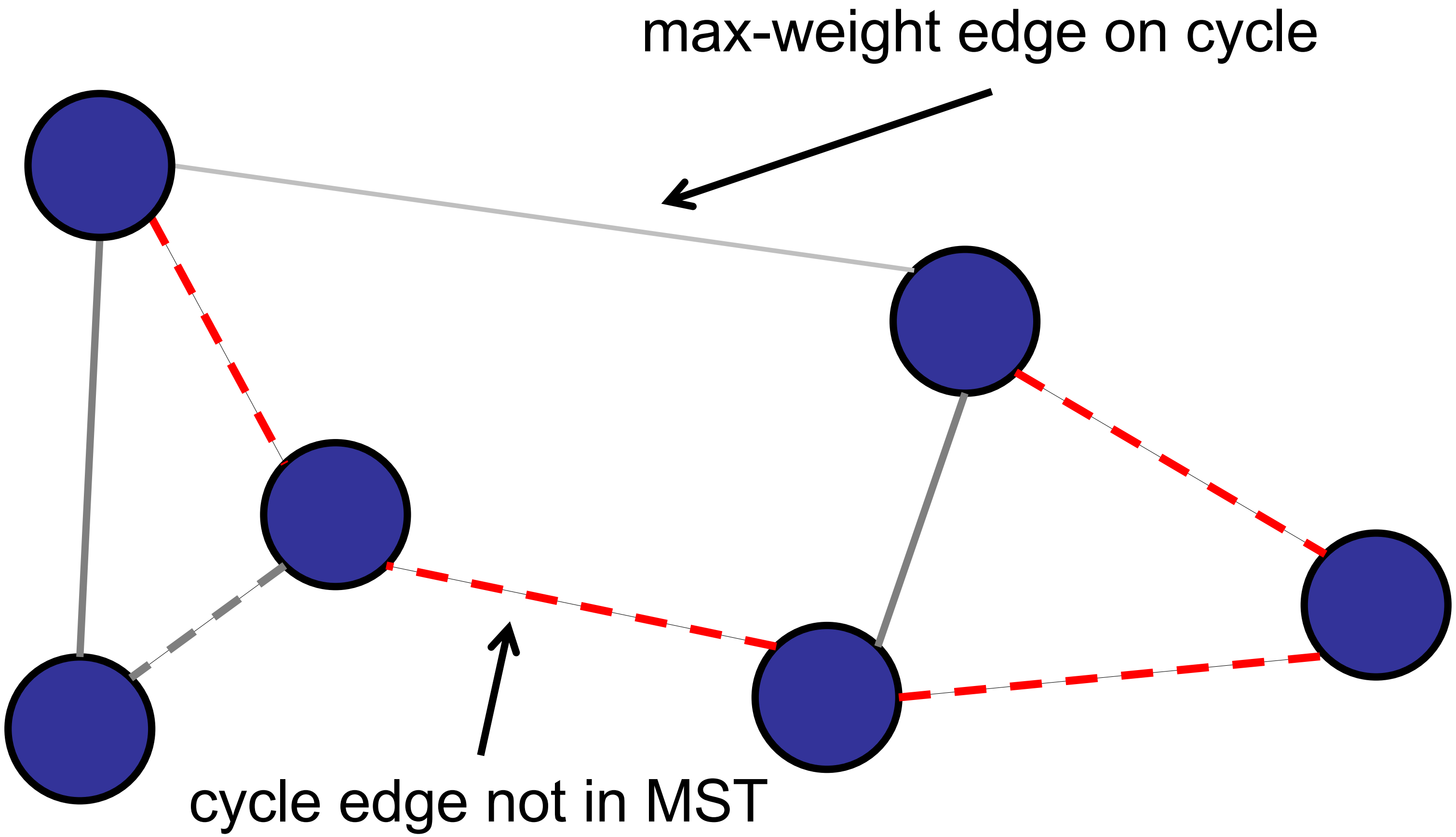


Properties of MST

Proof: Cut-and-paste

Replace heavy edge with lighter edge.

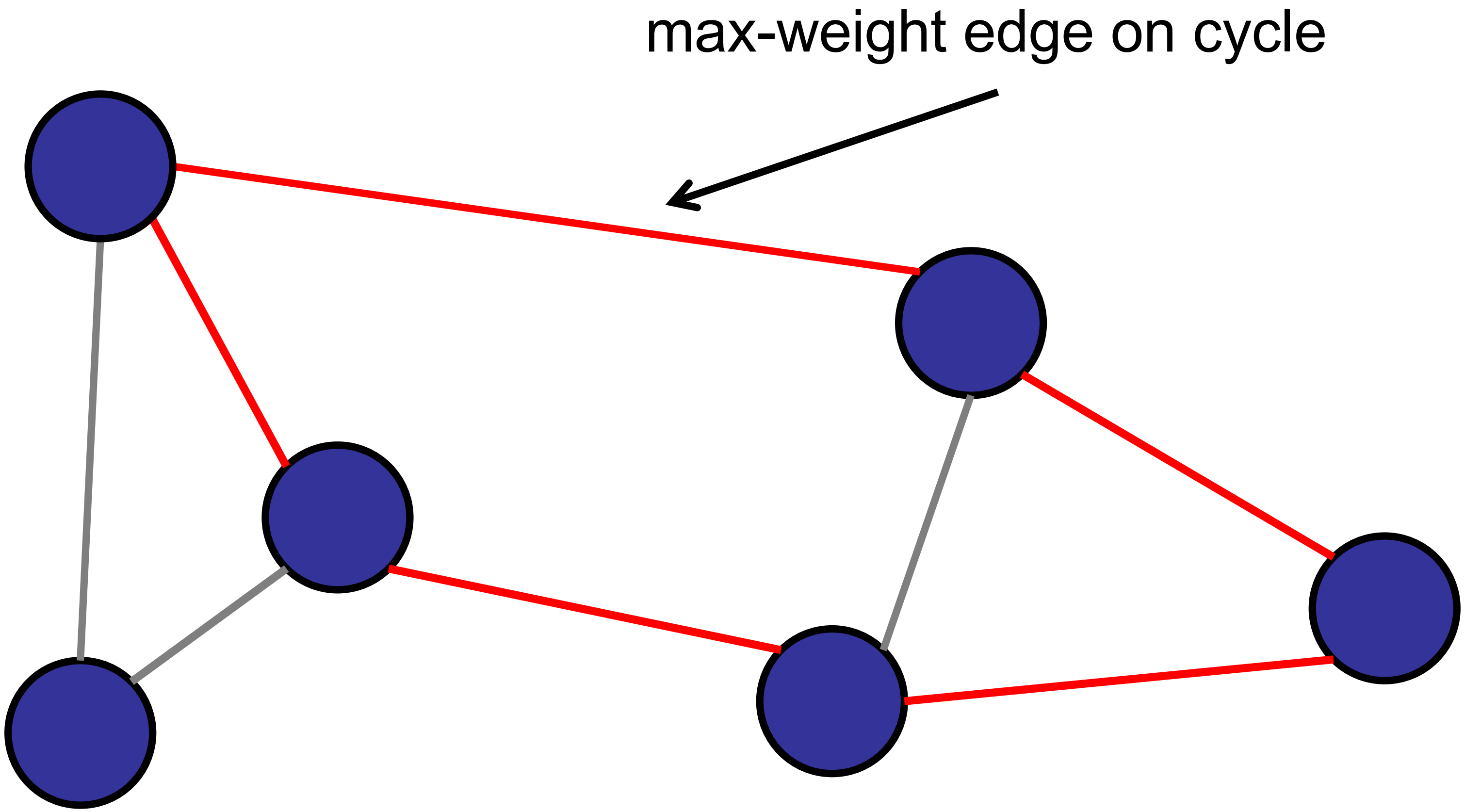
Less weight! Contradiction…



Properties of MST

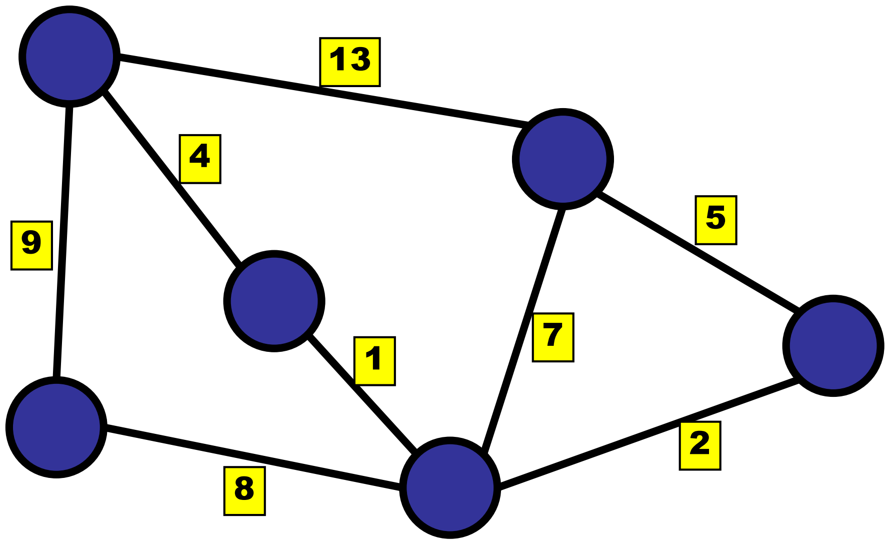
Property 3: Cycle property

For every cycle, the maximum weight edge is ***<u>not</u>*** in the MST.



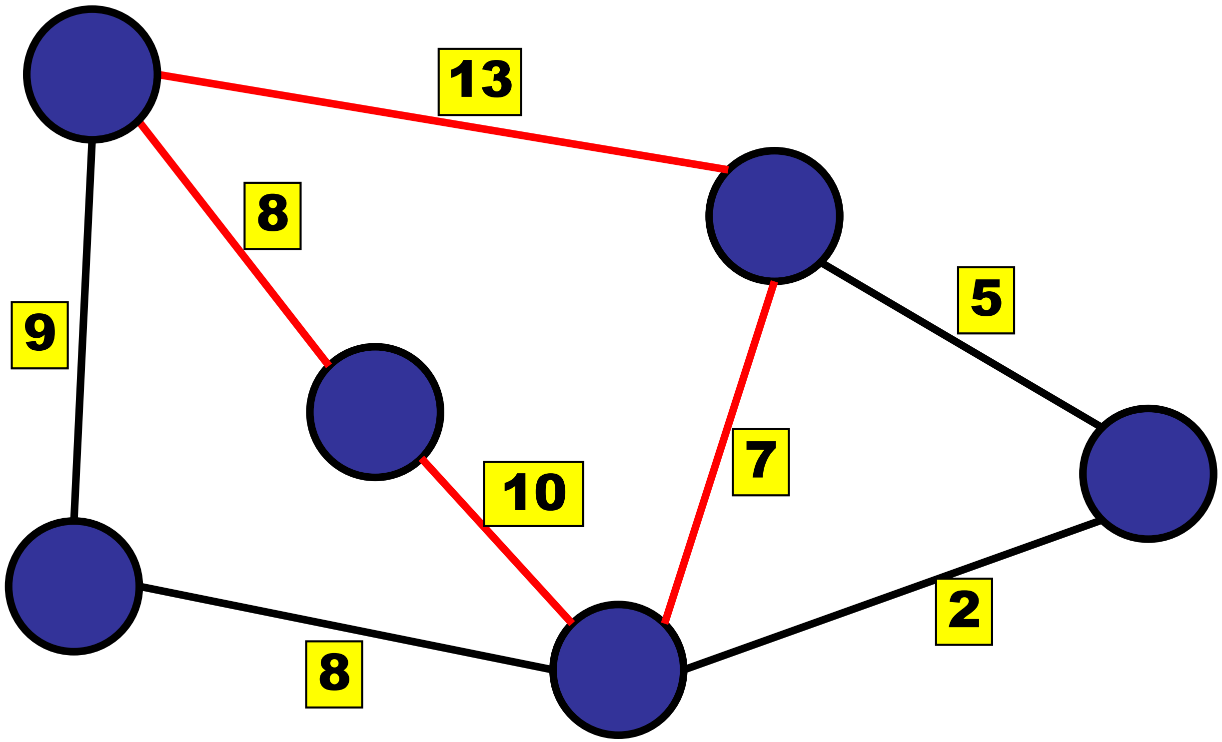
True or False:
For every cycle, the minimum weight edge is always in the MST.

1. True
2. False
3. I don't know.



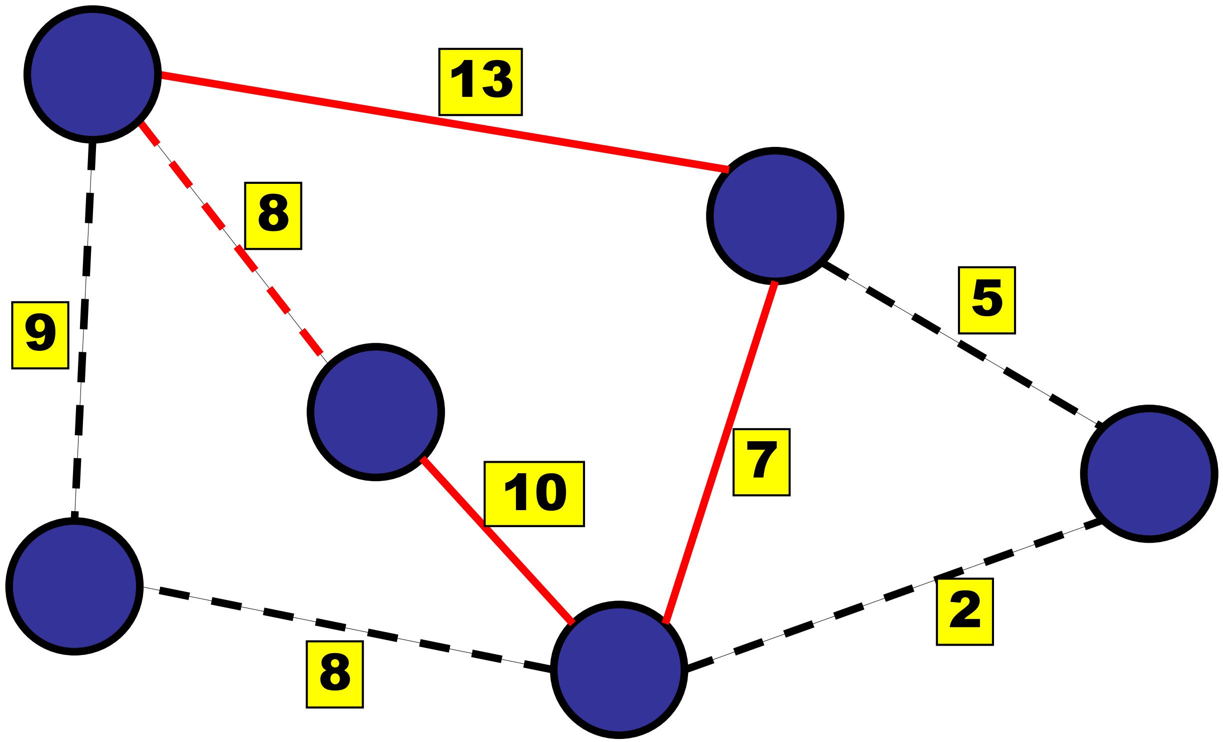
Properties of MST

For every cycle, the minimum weight edge may or may *not* be in the MST.



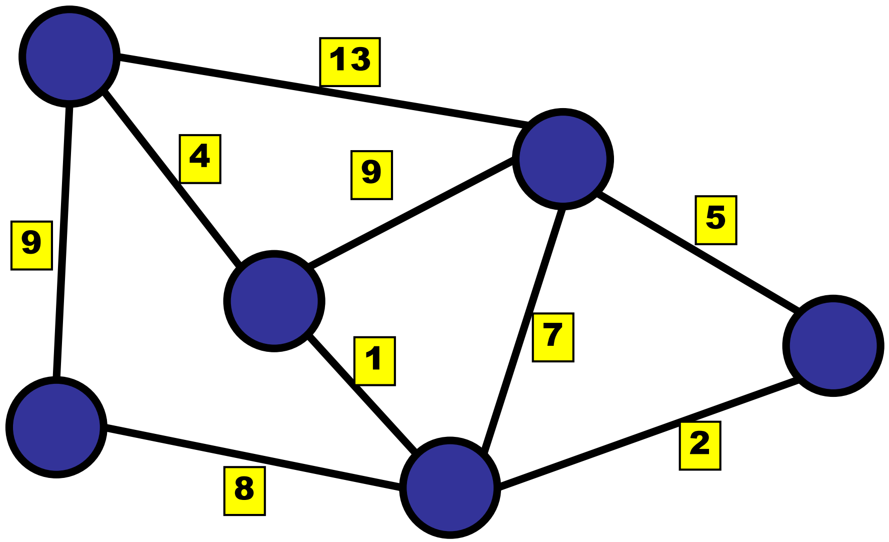
Properties of MST

For every cycle, the minimum weight edge may or may *not* be in the MST.



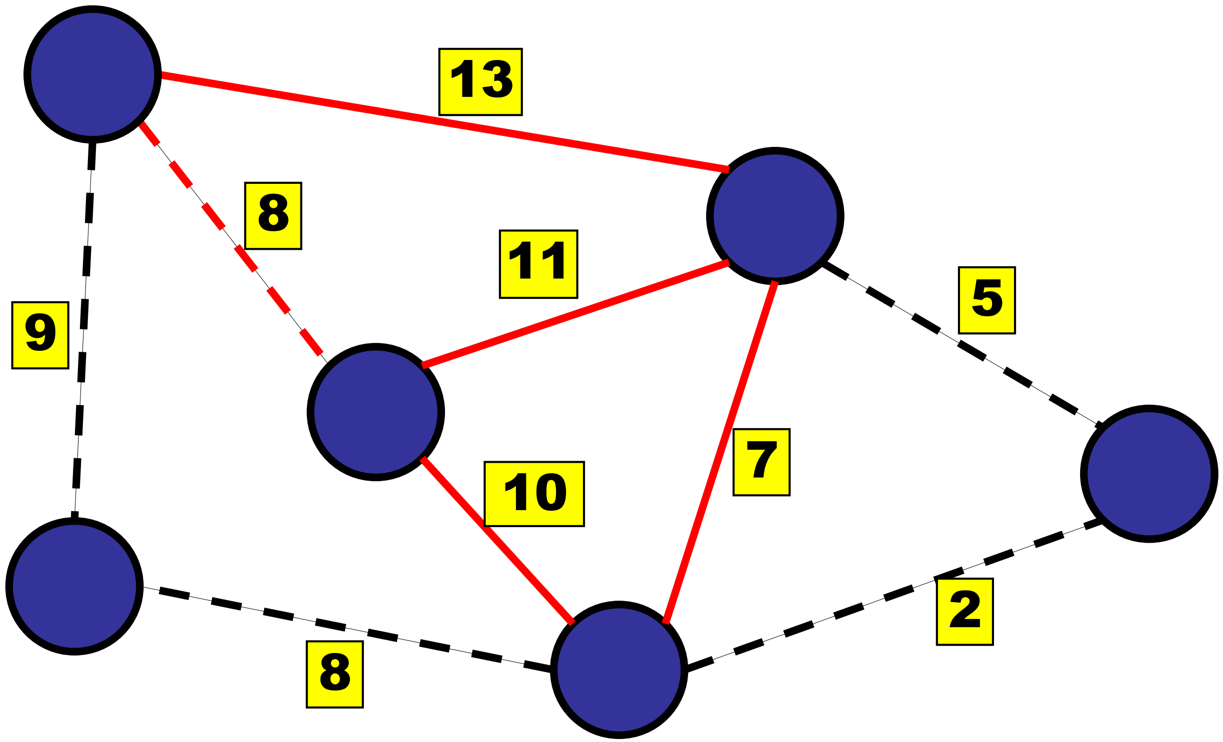
True or False:
For every cycle, at least one edge is always in the MST.

1. True
2. False
3. I don't know.



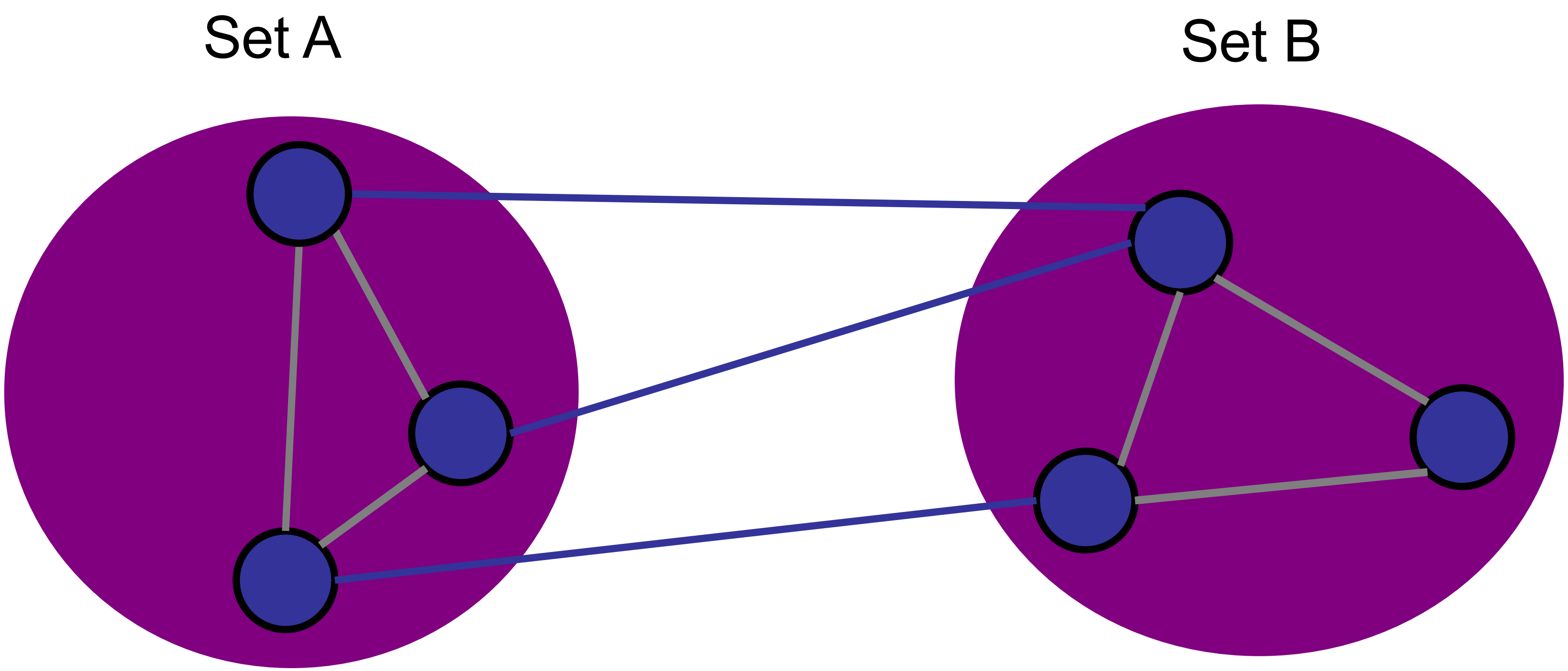
Properties of MST

For every cycle, it is possible that NO edges are in the MST.



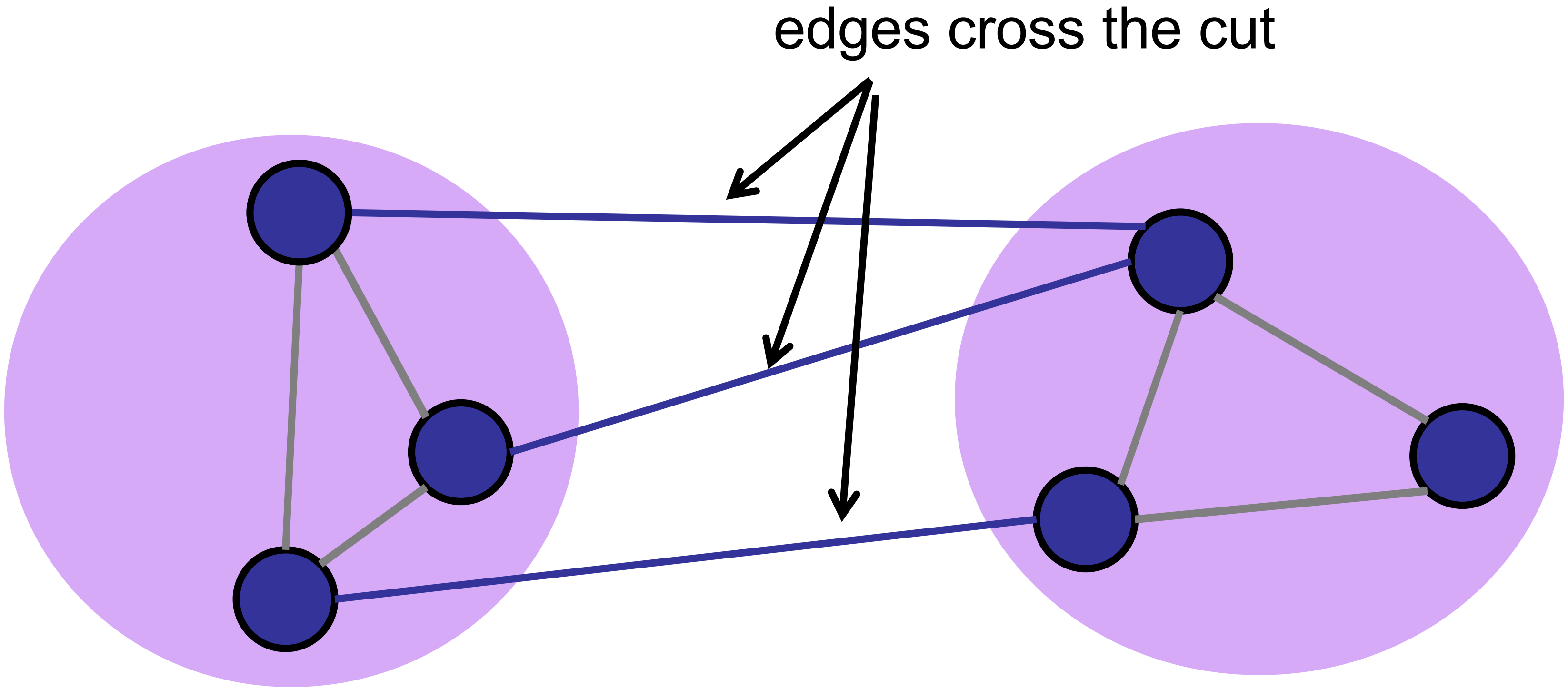
Properties of MST

Definition: A *cut* of a graph $G=(V,E)$ is a partition of the vertices V into two disjoint subsets.



Properties of MST

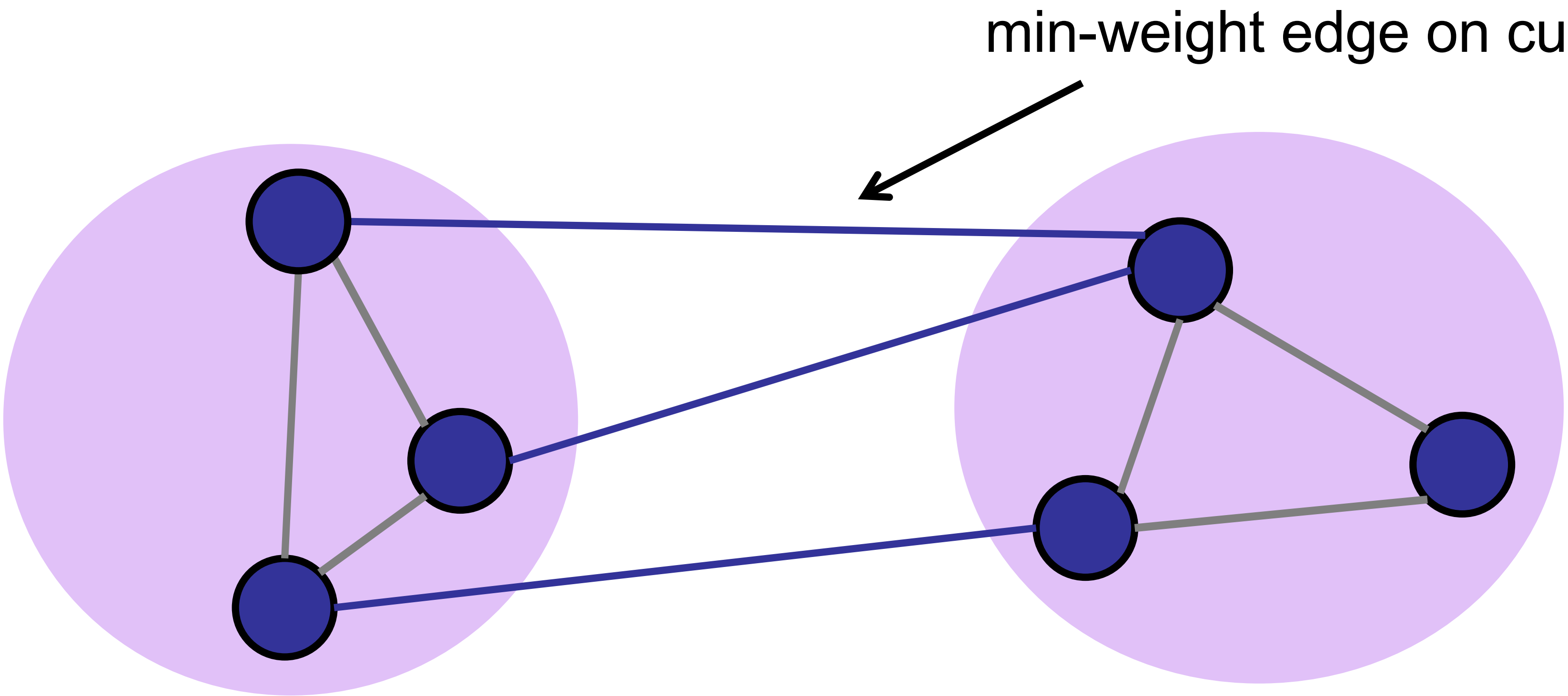
Definition: An edge *crosses a cut* if it has one vertex in each of the two sets.



Properties of MST

Property 4: Cut property

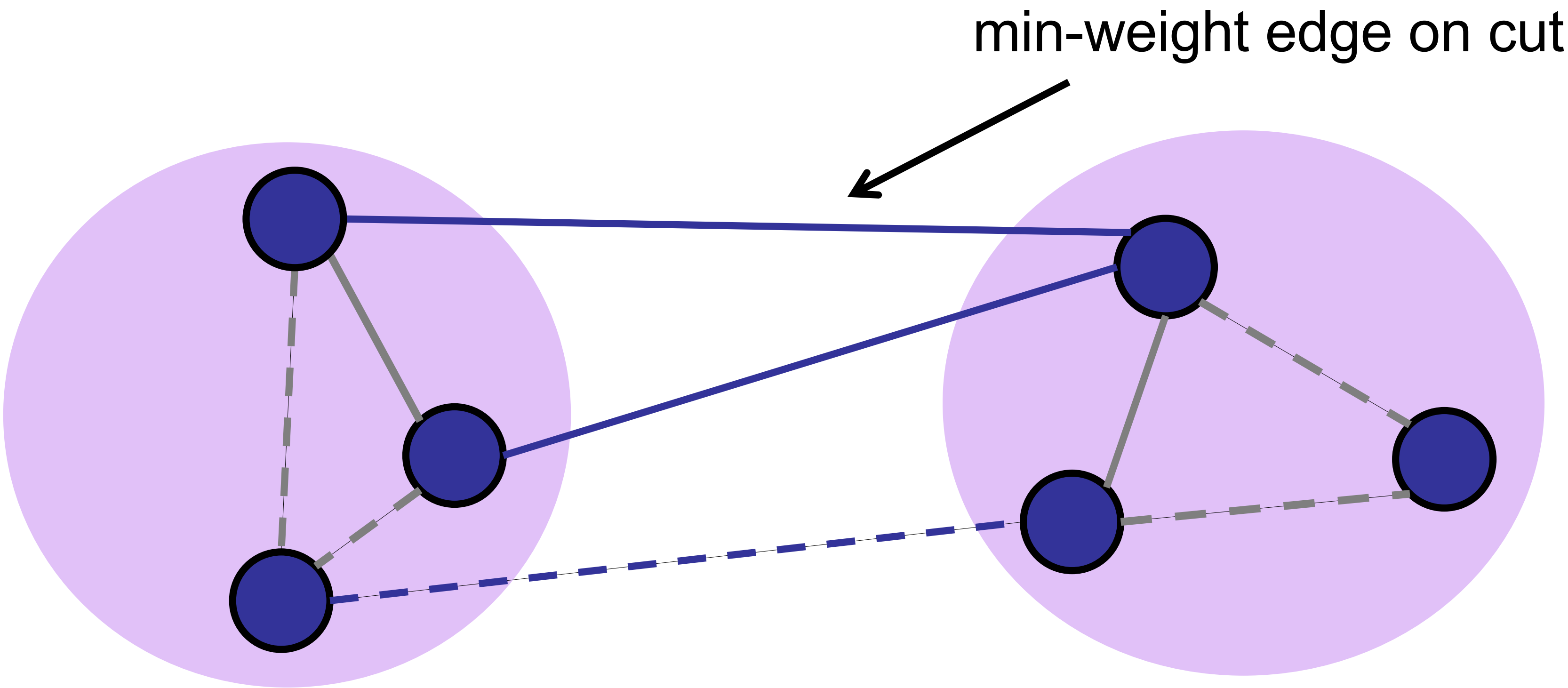
For every partition of the nodes, the minimum weight edge across the cut *is* in the MST.



Properties of MST

Proof: Cut-and-paste

Assume not.

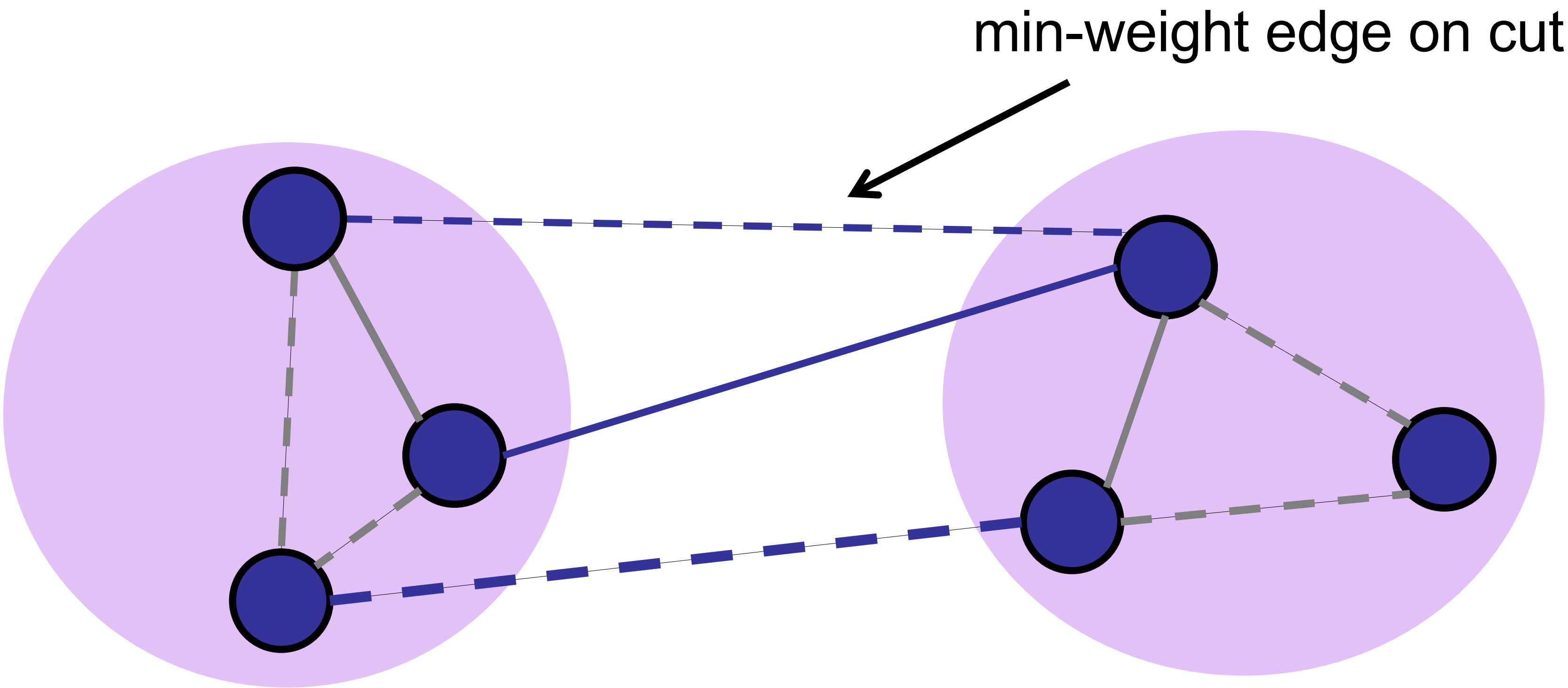


Properties of MST

Proof: Cut-and-paste

Assume not.

Add min-weight edge on cut.



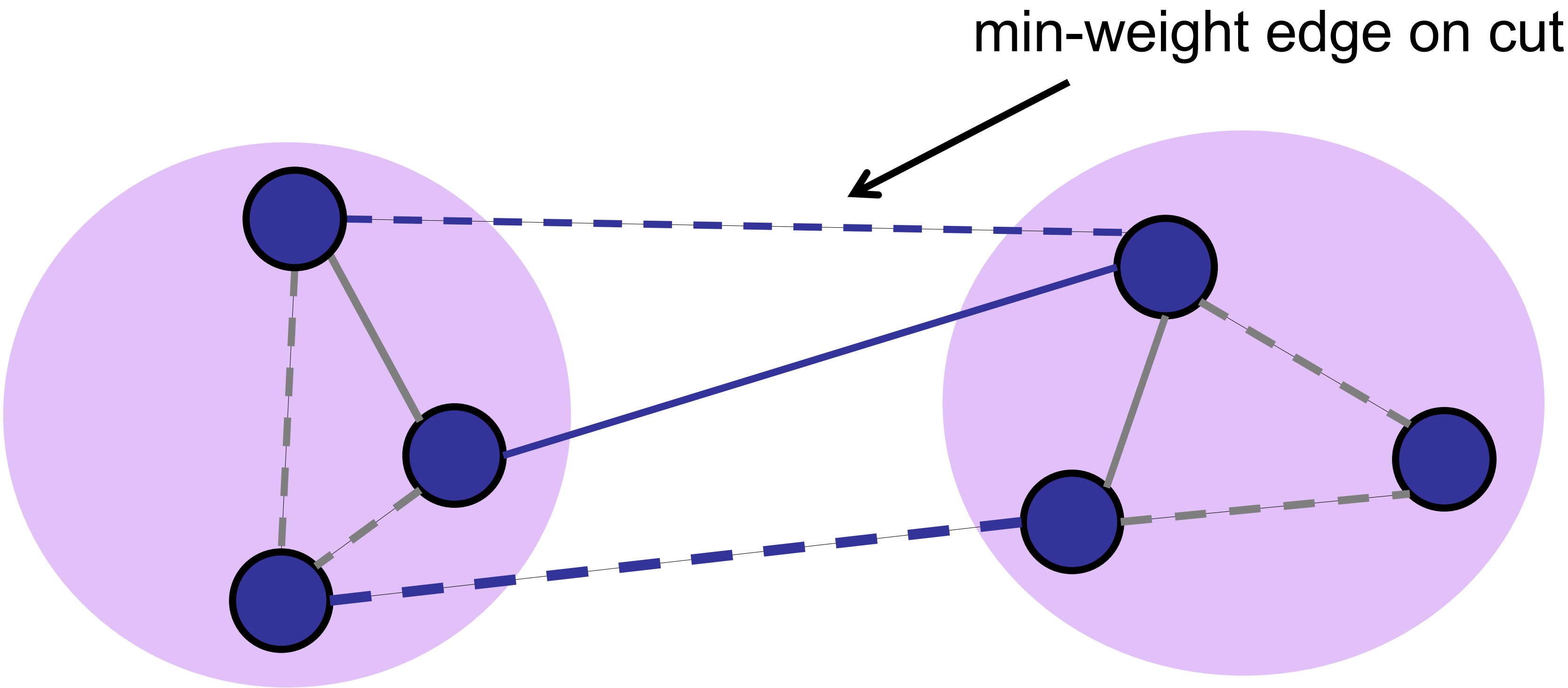
Properties of MST

Proof: Cut-and-paste

Assume not.

Add min-weight edge on cut.

Oops, creates a cycle!



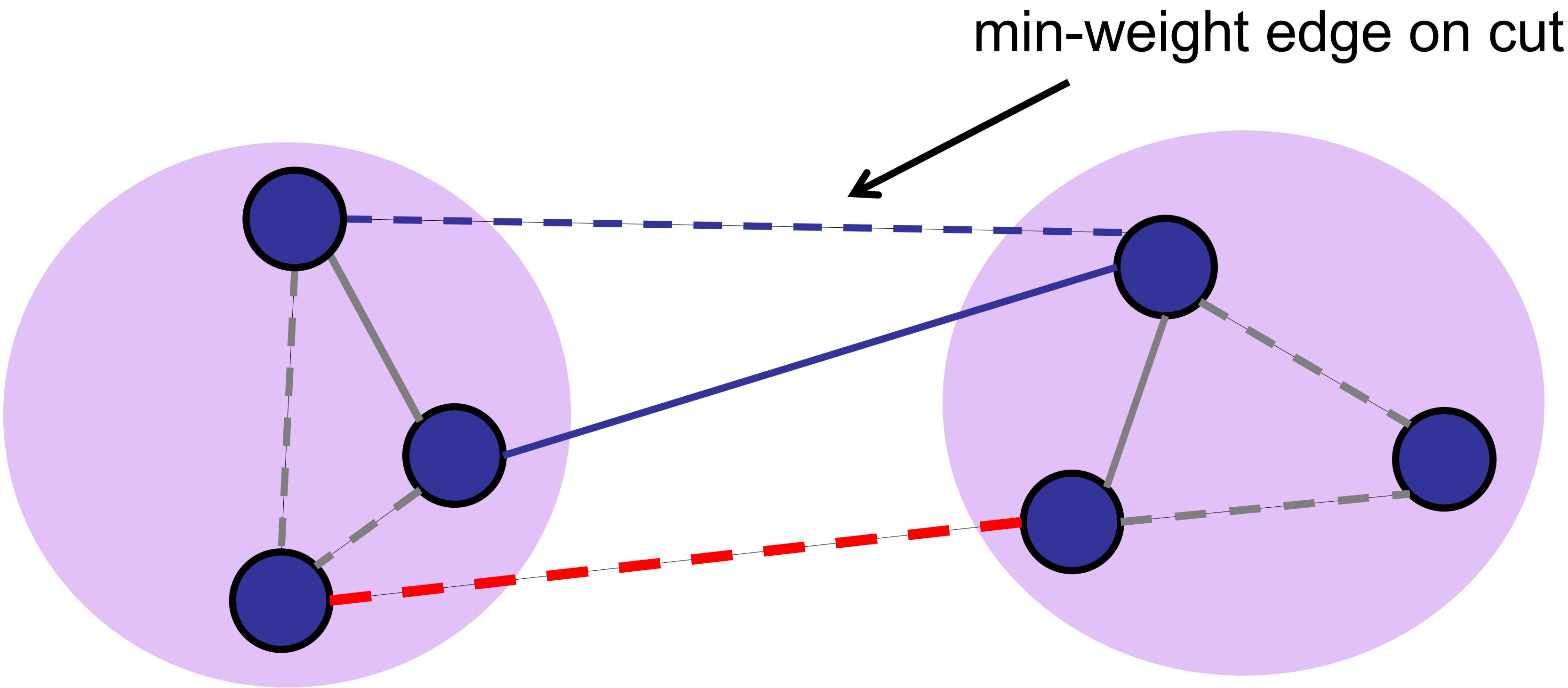
Properties of MST

Proof: Cut-and-paste

Assume not.

Add min-weight edge on cut.

Remove heaviest edge on cycle.



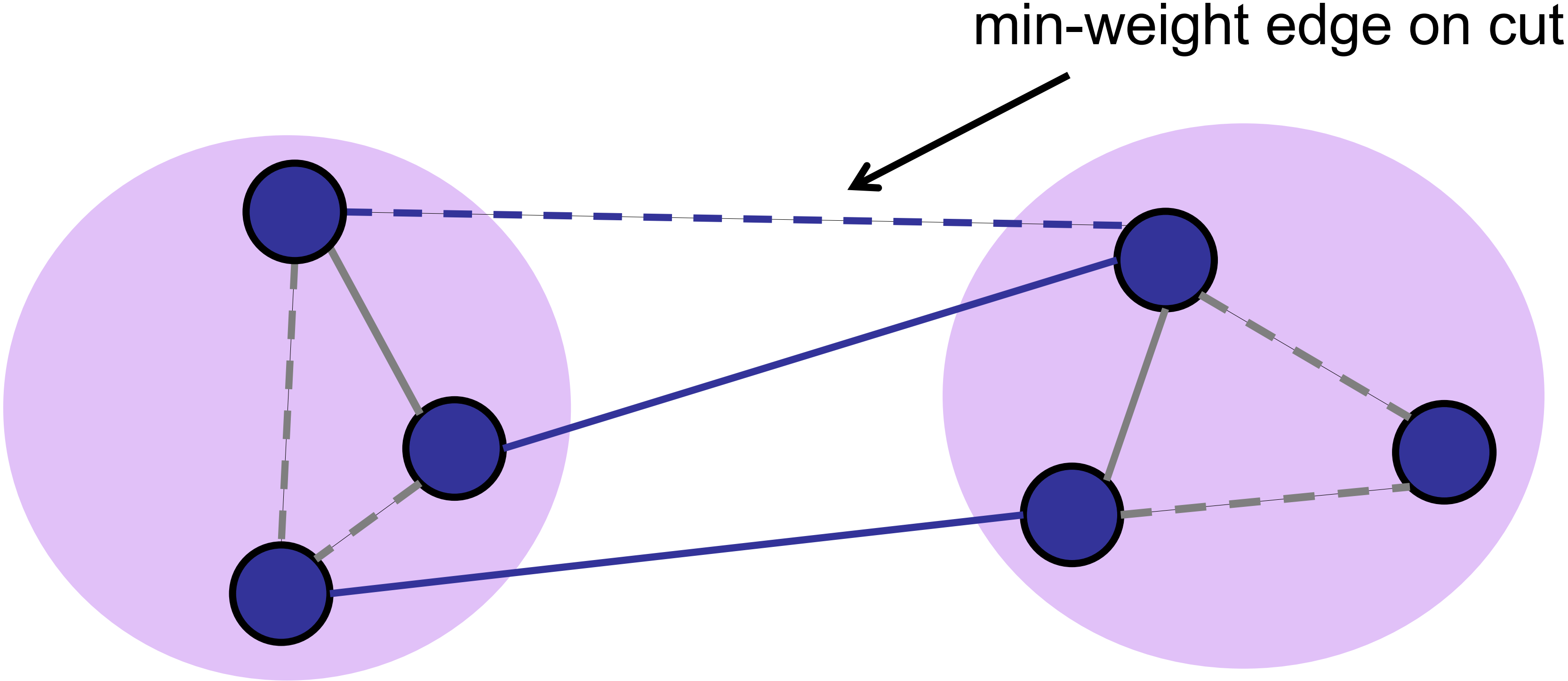
Properties of MST

Proof: Cut-and-paste

Assume not.

Add min-weight edge on cut.

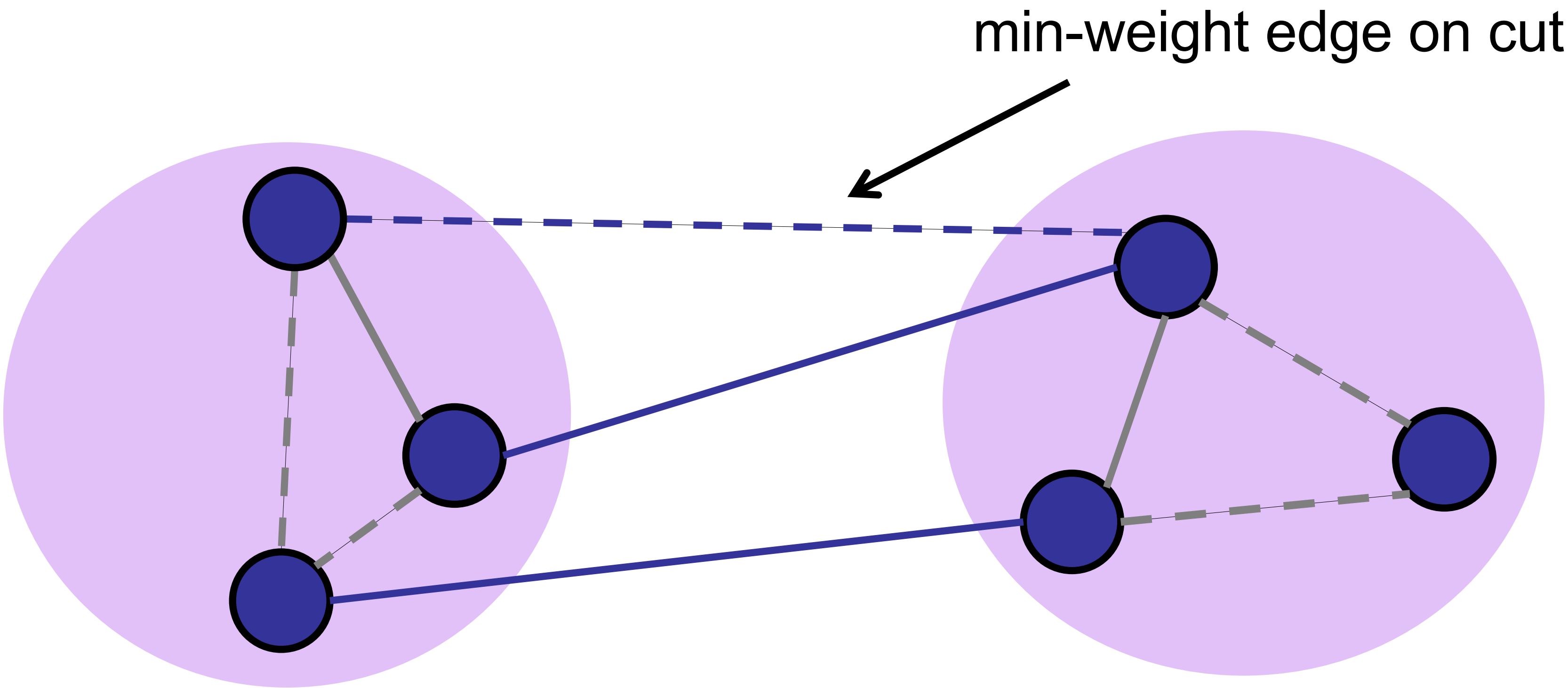
Remove heaviest edge on cycle.



Properties of MST

Proof: Cut-and-paste

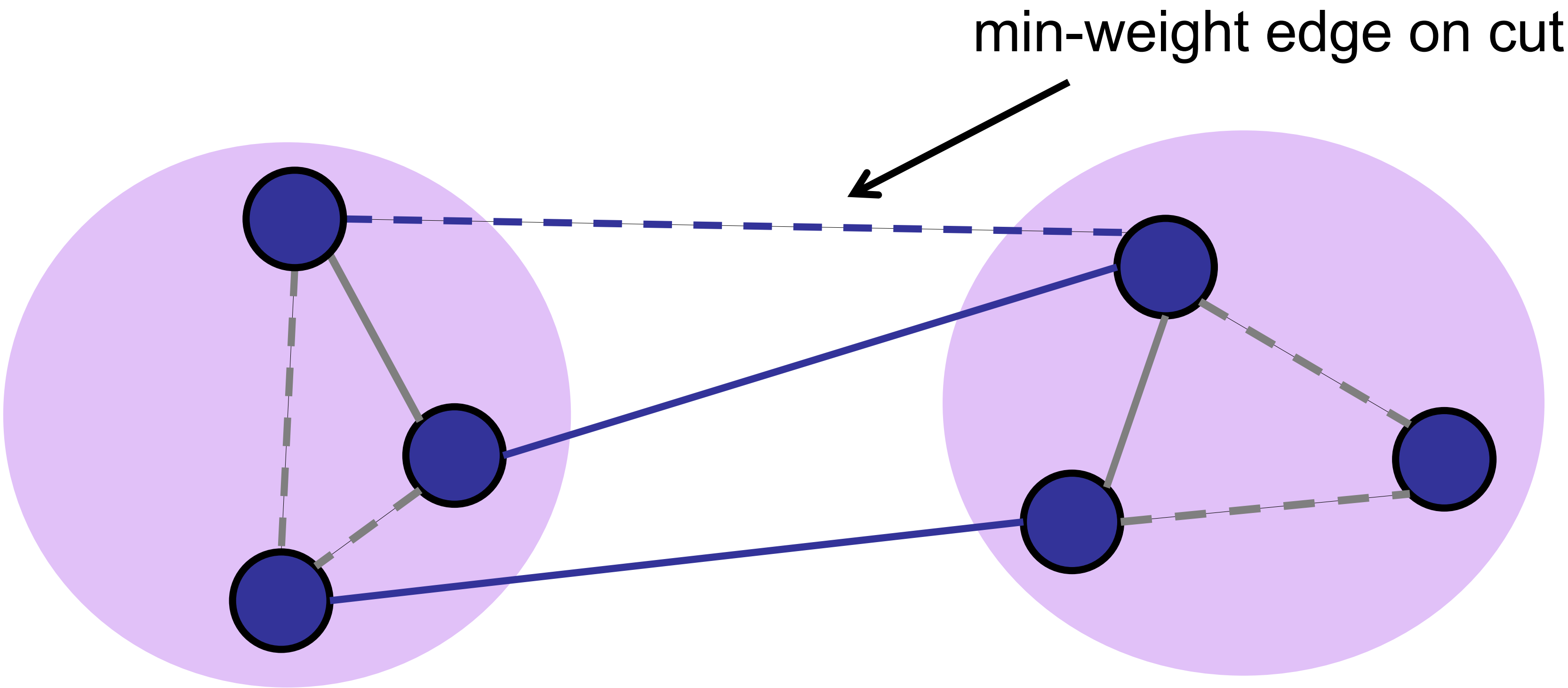
Result: a new spanning tree.



Properties of MST

Proof: Cut-and-paste

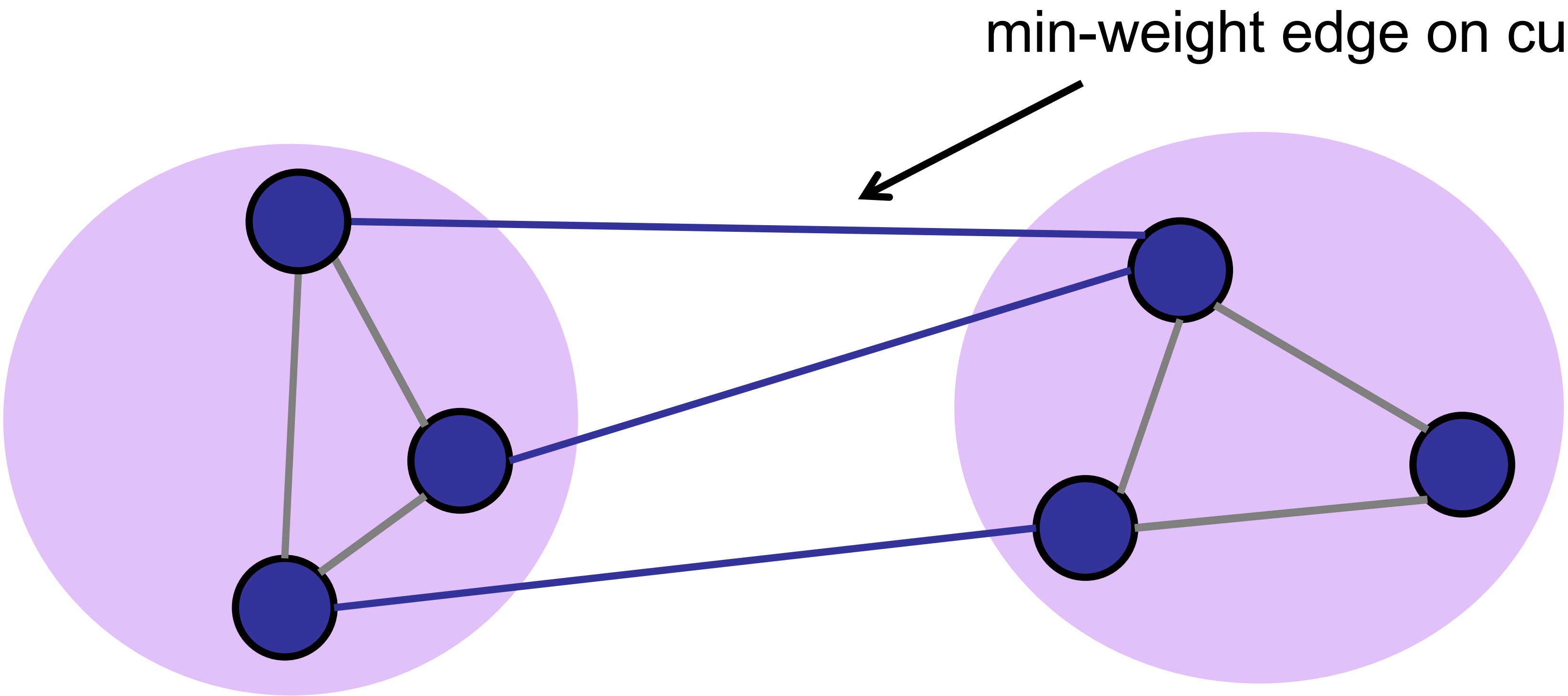
Less weight: replaced heavier edge with lighter edge.



Properties of MST

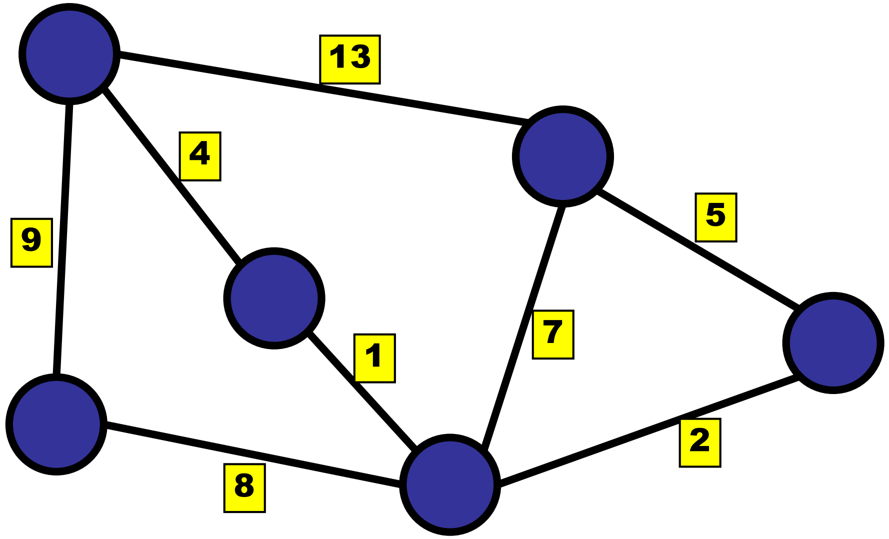
Property 4: Cut property

For every partition of the nodes, the minimum weight edge across the cut *is* in the MST.



True or False:
For every vertex, the minimum outgoing edge is always part of the MST.

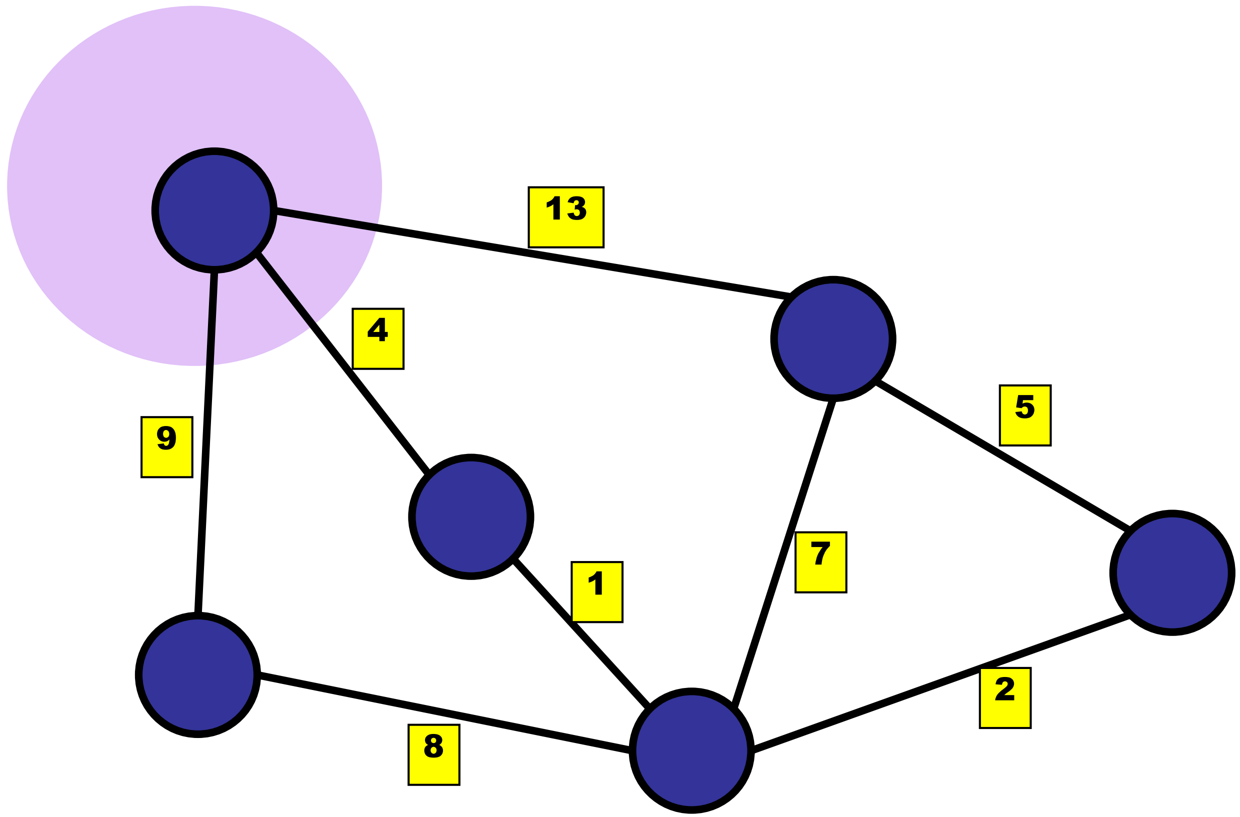
✓ 1. True
2. False
3. I don't know.



Properties of MST

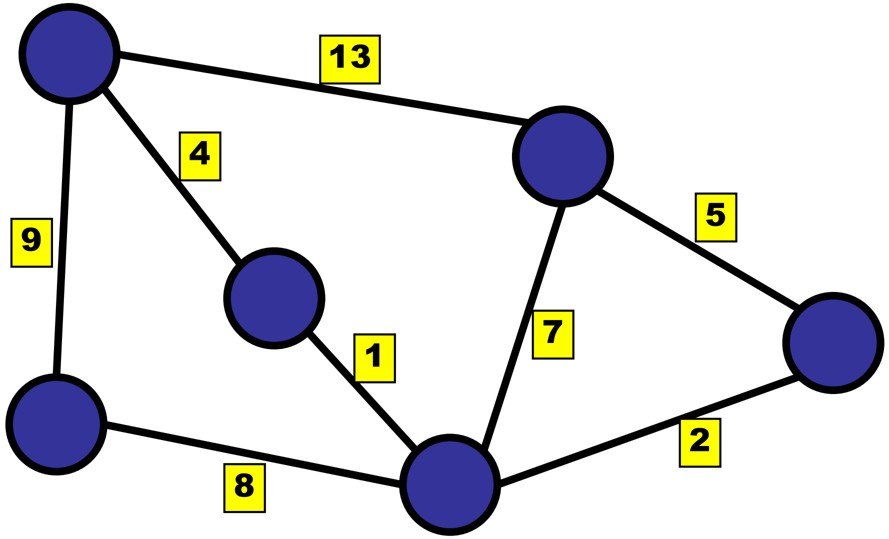
Property 4: Cut property

For every partition of the nodes, the minimum weight edge across the cut *is* in the MST.



True or False:
For every vertex, the maximum outgoing edge is never part of the MST.

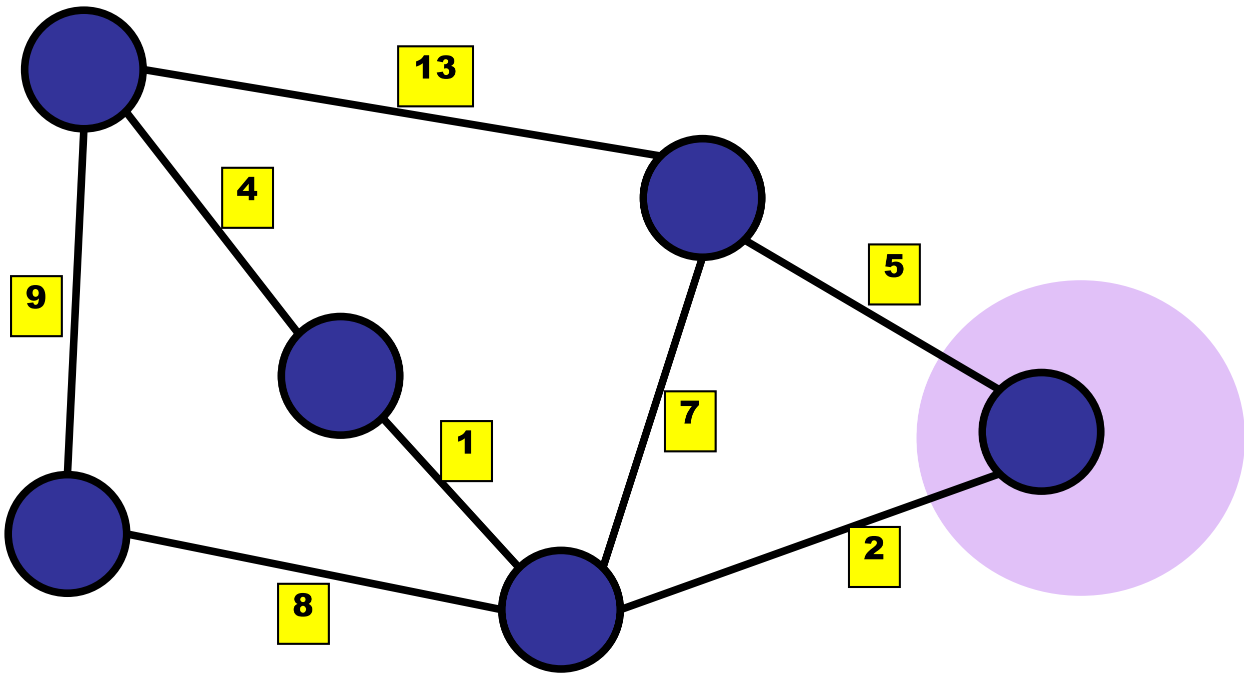
1. True
2. False
3. I don't know.



Properties of MST

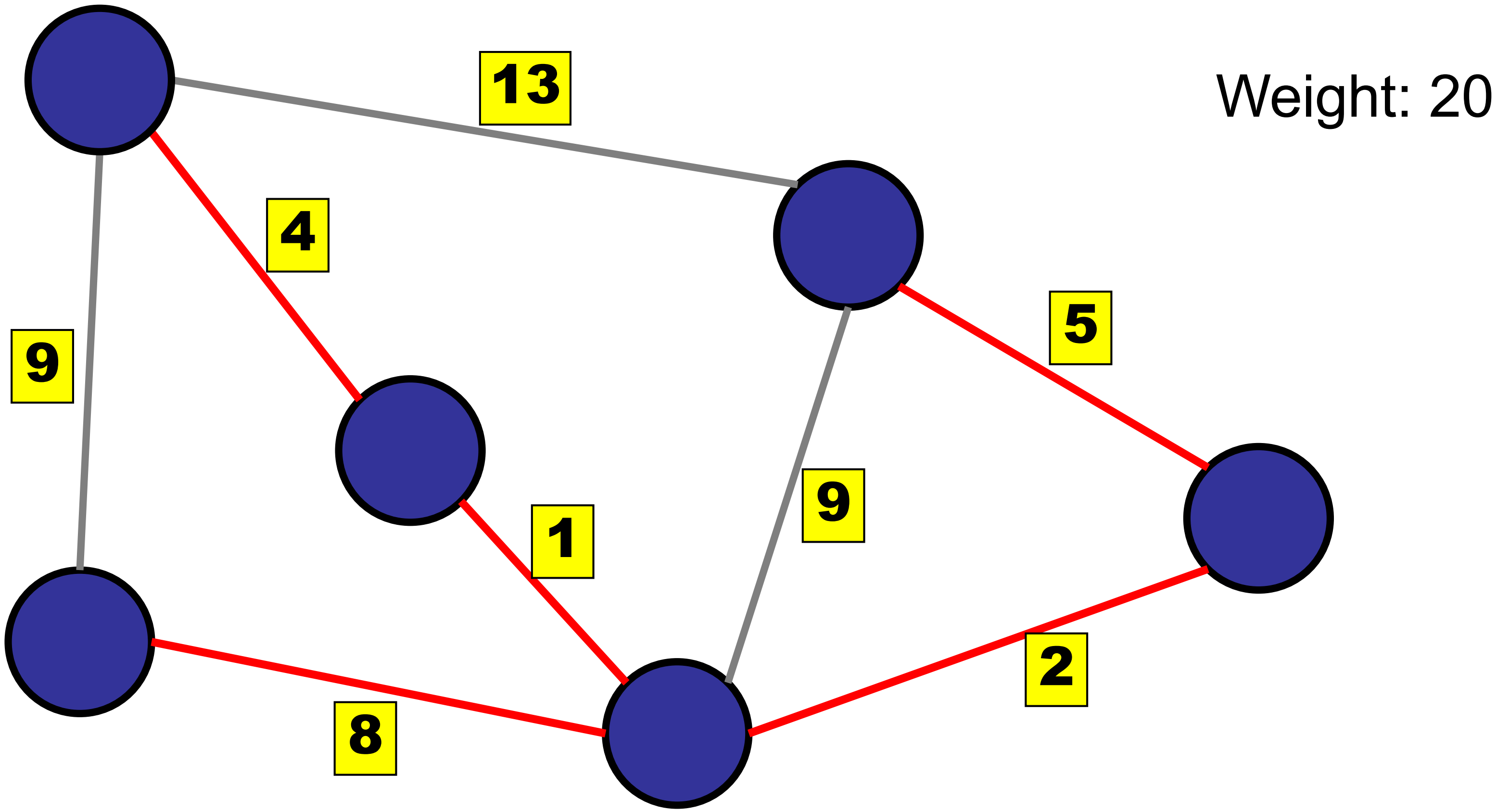
Property 4: Cut property

For every partition of the nodes, the minimum weight edge across the cut *is* in the MST.



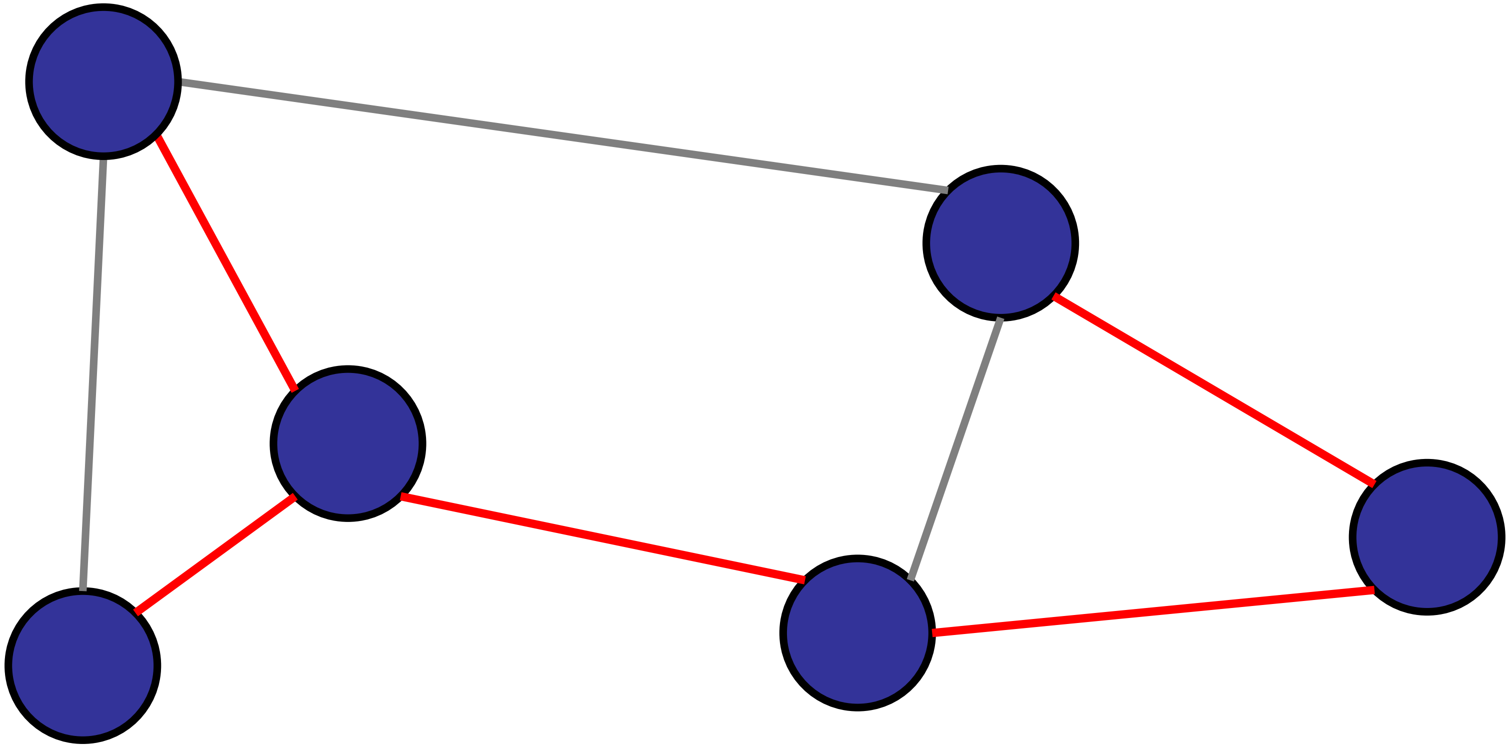
Minimum Spanning Tree

Definition: a spanning tree with minimum weight



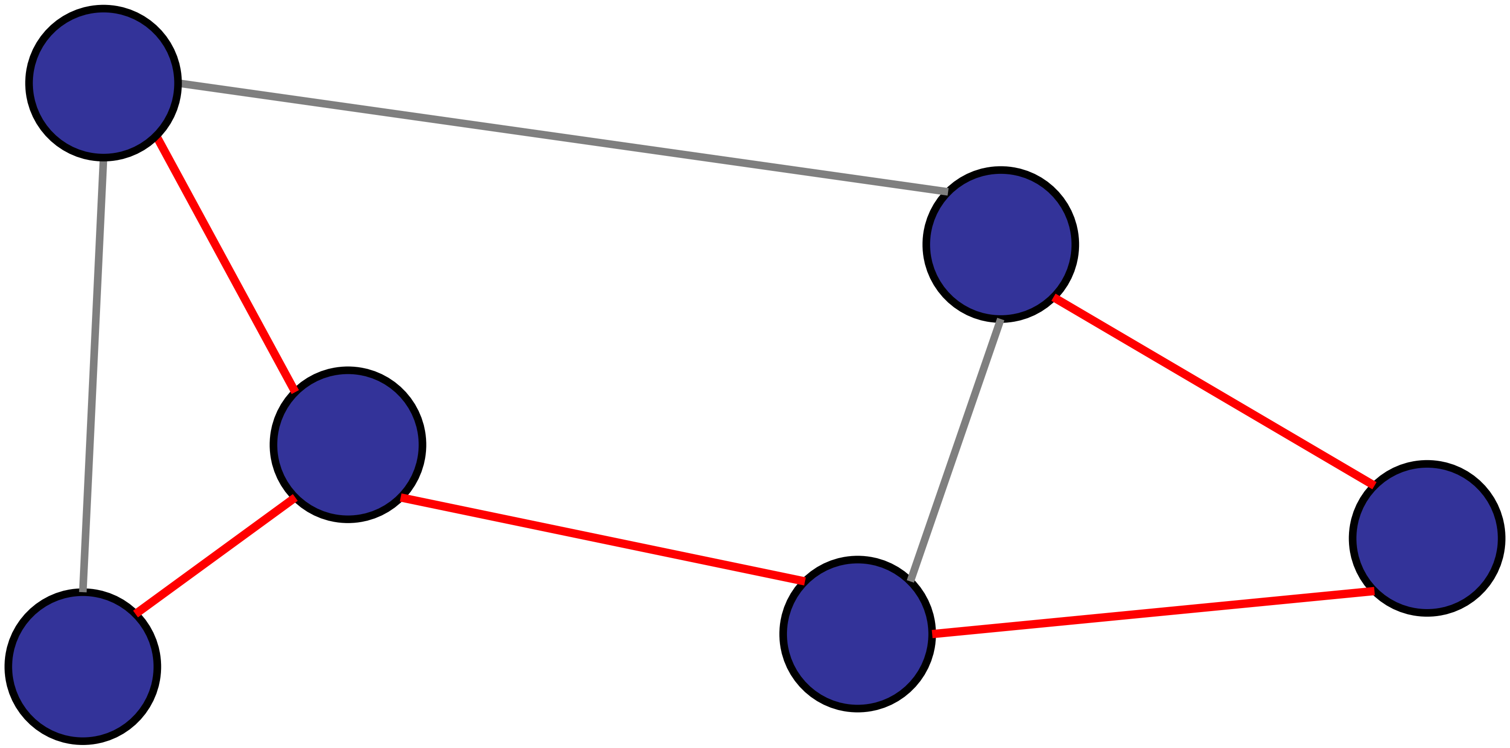
Properties of MST

Property 1: No cycles



Properties of MST

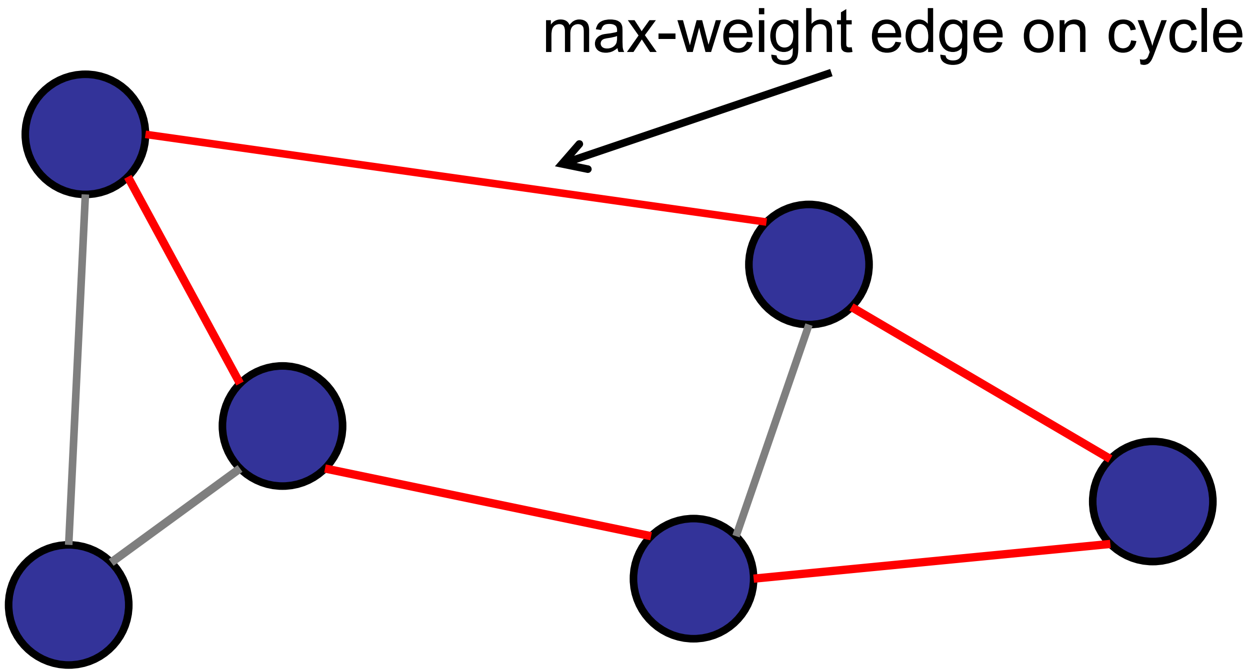
Property 2: If you cut an MST, the two pieces are both MSTs.



Properties of MST

Property 3: Cycle property

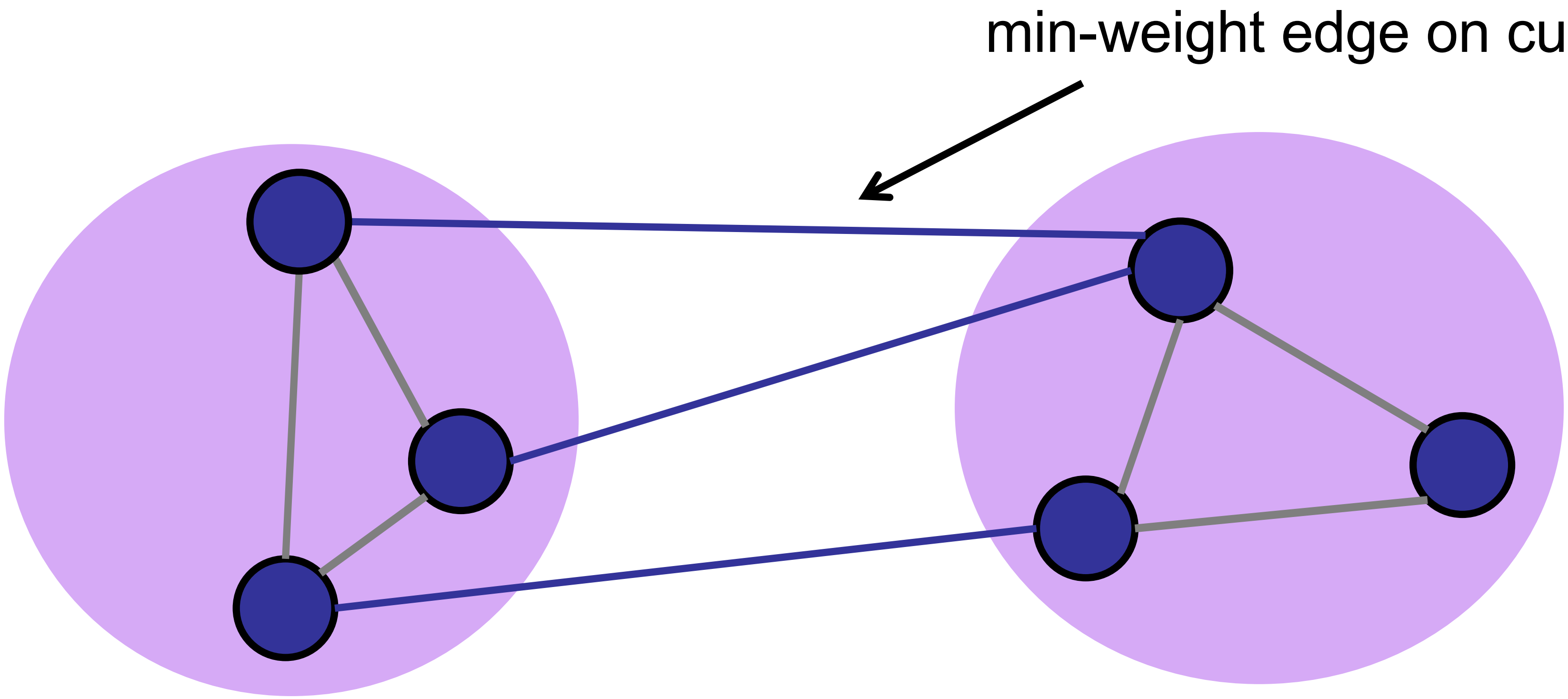
For every cycle, the maximum weight edge is *not* in the MST.



Properties of MST

Property 4: Cut property

For every cut D, the minimum weight edge that crosses the cut *is* in the MST.



Property of MST

- No cycles
- If you cut an MST, the two pieces are both MSTs.
- Cycle property
  - For every cycle, the maximum weight edge is not in the MST.
- Cut property
  - For every cut D, the minimum weight edge that crosses the cut is in the MST.

Roadmap

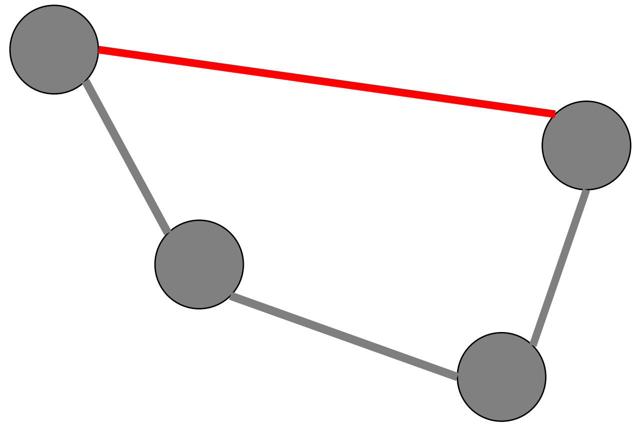
Minimum Spanning Trees

- The MST Problem
- Basic Properties of an MST
- **Generic MST Algorithm**
- Prim's Algorithm
- Kruskal's Algorithm
- Boruvka's Algorithm
- Variations

Generic MST Algorithm

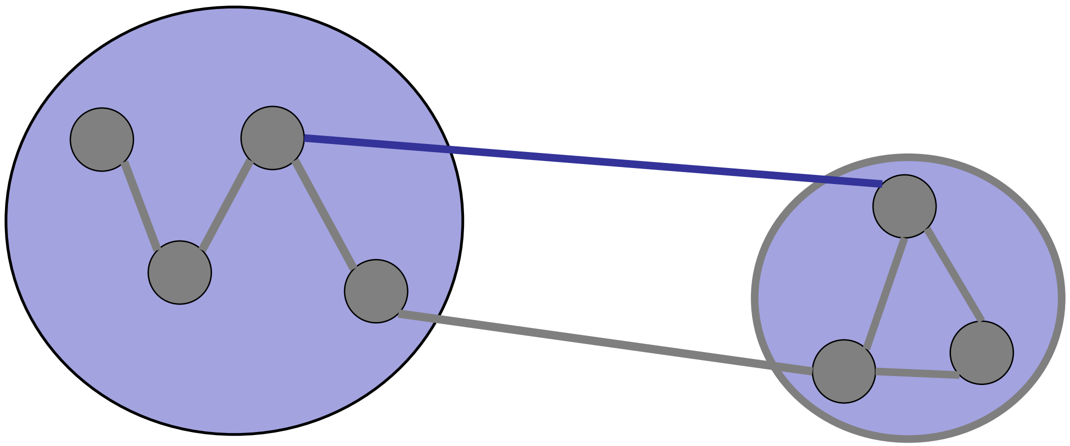
Red rule:

If C is a cycle with no red arcs, then color the max-weight edge in C red.



Blue rule:

If D is a cut with no blue arcs, then color the min-weight edge in D blue.



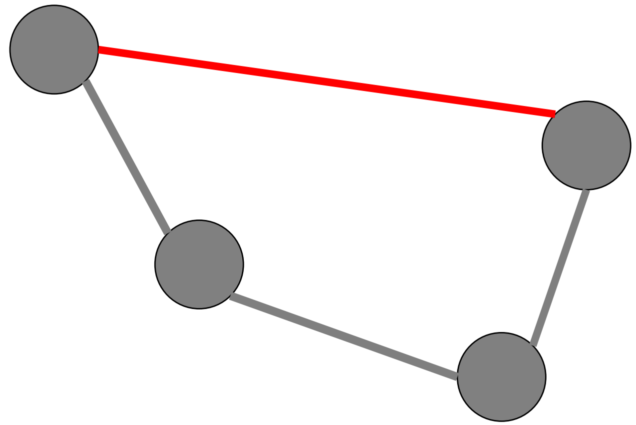
Generic MST Algorithm

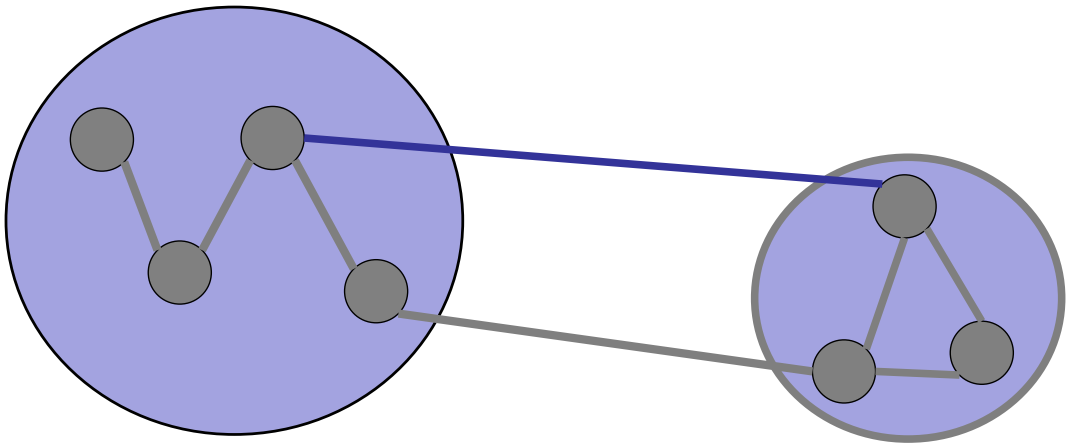
Greedy Algorithm:

Repeat:

Apply red rule or blue rule to an arbitrary edge.

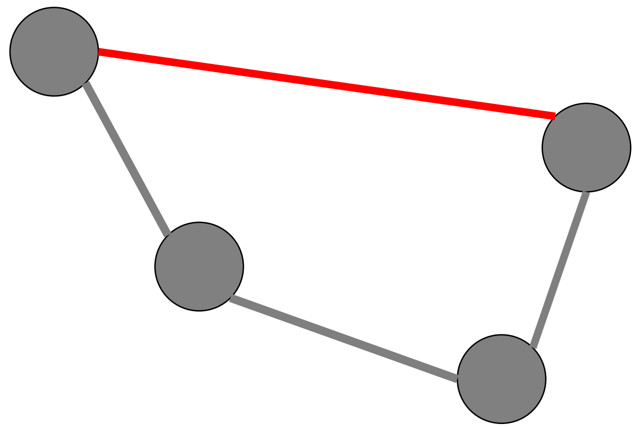
until no more edges can be colored.

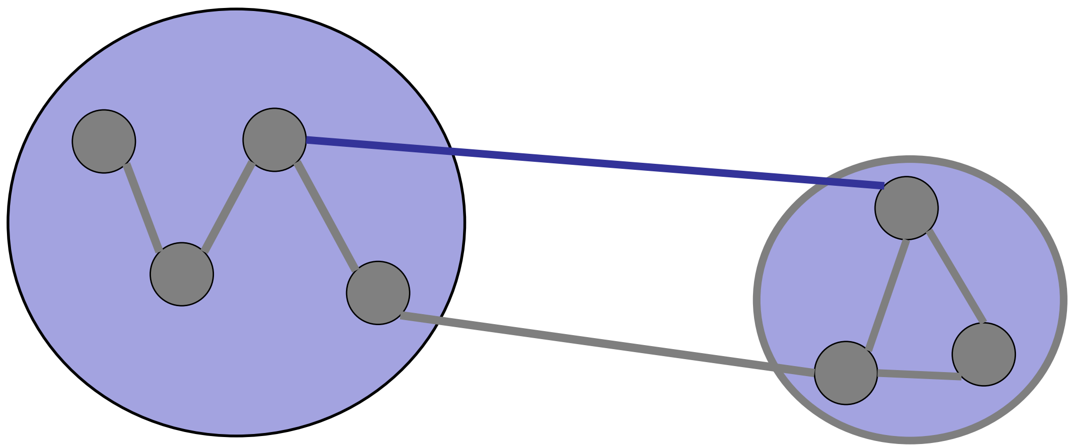




Generic MST Algorithm

Claim: On termination, the blue edges are an MST.



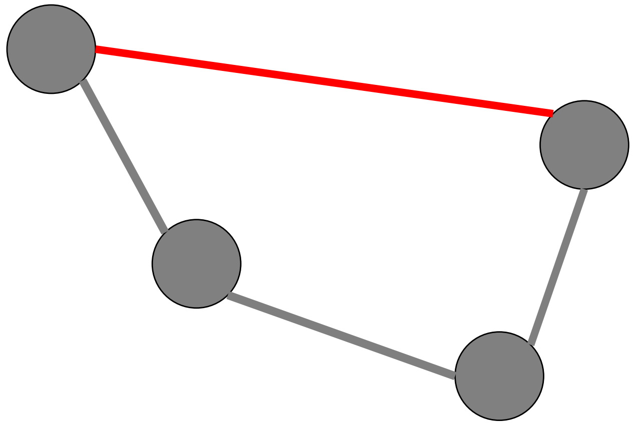


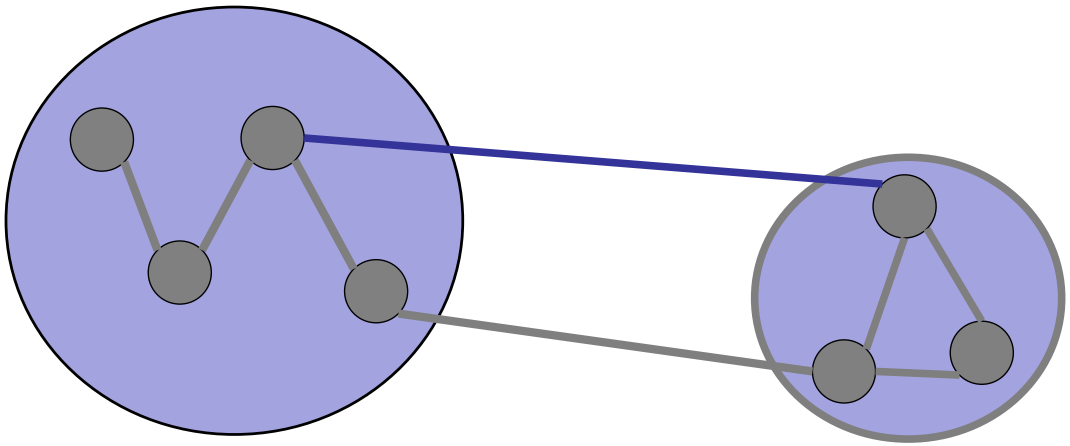
Generic MST Algorithm

Claim: On termination, the blue edges are an MST.

On termination:

1. Every cycle has a red edge.
   No blue cycles.



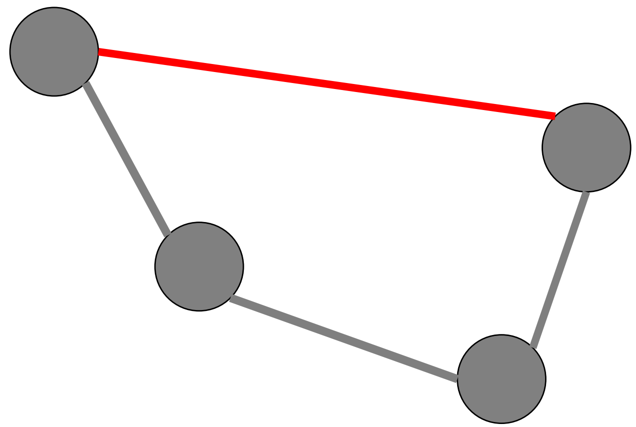


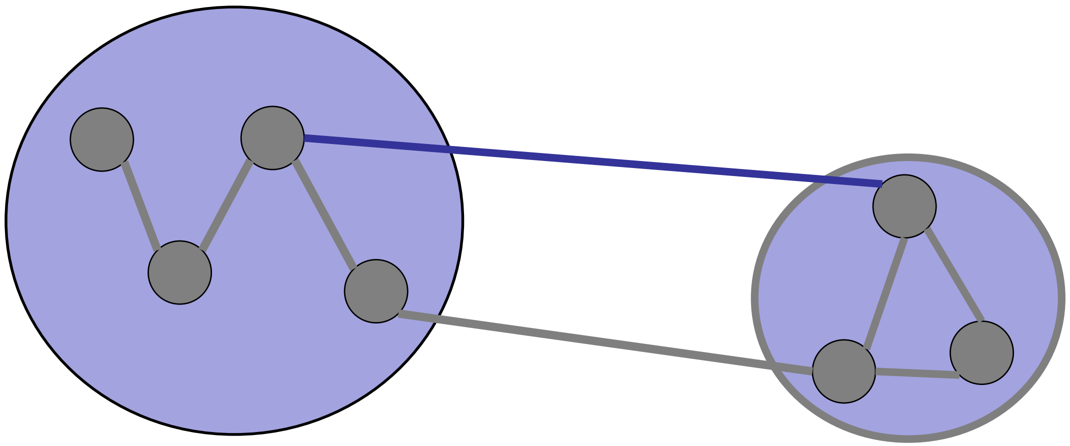
Generic MST Algorithm

Claim: On termination, the blue edges are an MST.

On termination:

1. Every cycle has a red edge.
   No blue cycles.
2. Blue edges form a forest.



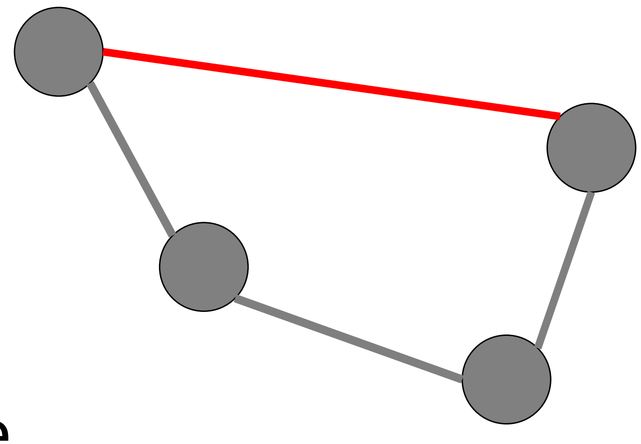


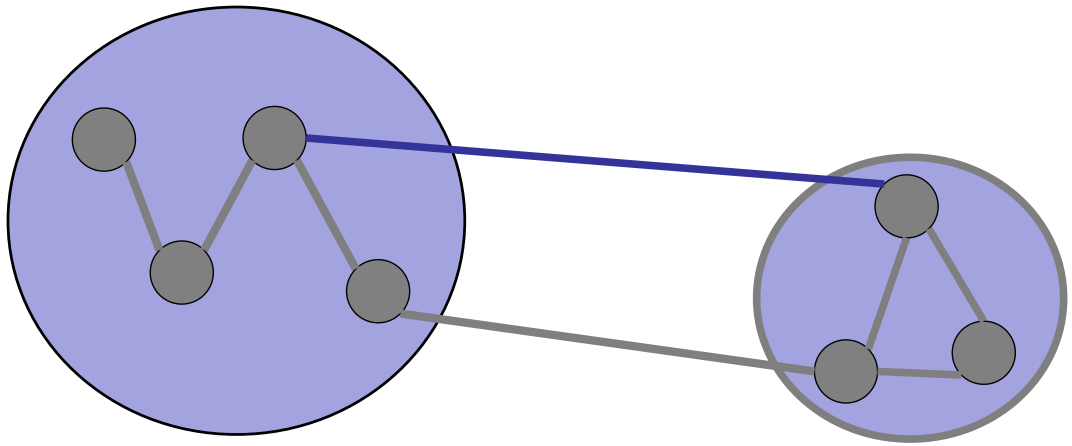
Generic MST Algorithm

Claim: On termination, the blue edges are an MST.

On termination:

1. Every cycle has a red edge.
   No blue cycles.
2. Blue edges form a forest.
3. Blue edges form a spanning tree.
   (Otherwise, there is a cut with no blue edge.)

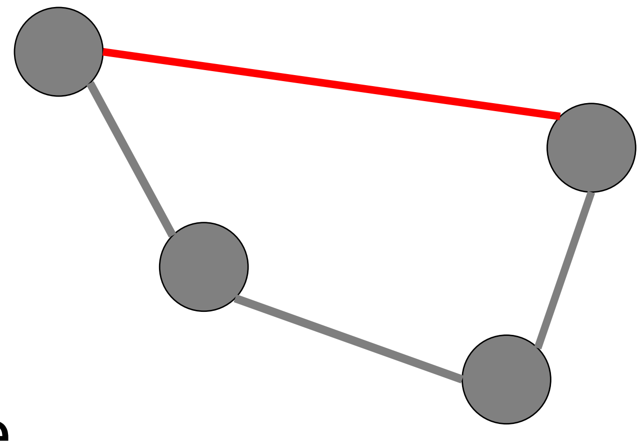




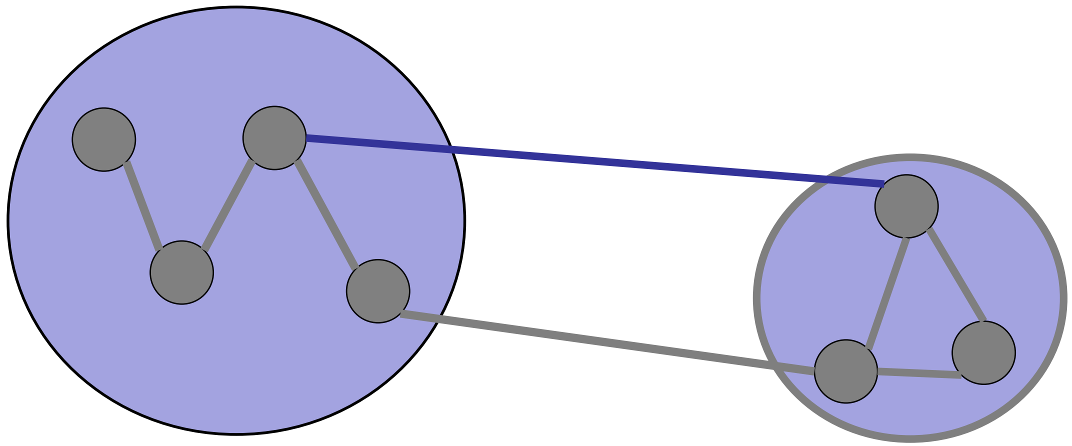
Generic MST Algorithm

Claim: On termination, the blue edges are an MST.

On termination:

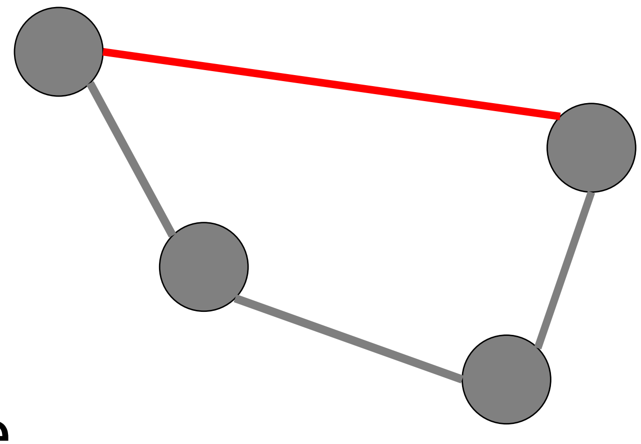


1. Every cycle has a red edge.
   No blue cycles.
2. Blue edges form a forest.
3. Blue edges form a spanning tree.
   (Otherwise, there is a cut with no blue edge.)
4. Every edge is colored.



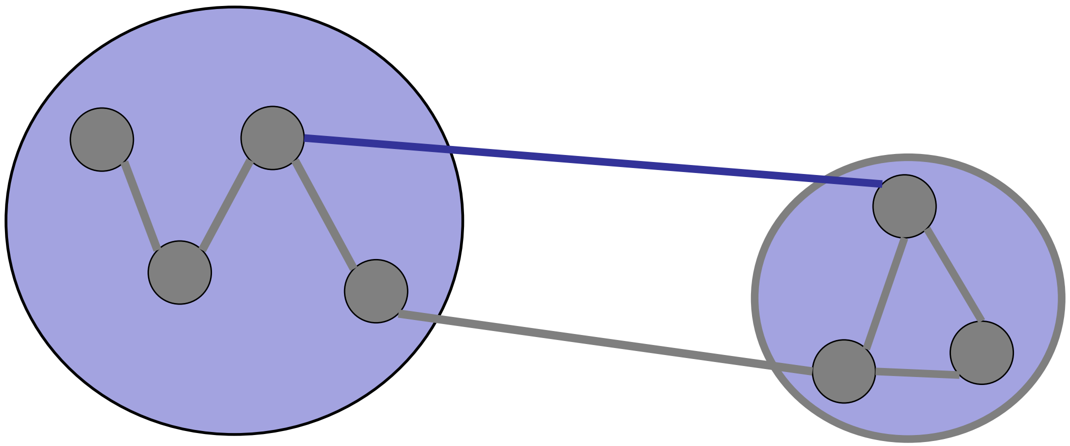
Generic MST Algorithm

Claim: On termination, the blue edges are an MST.



On termination:

1. Every cycle has a red edge.
   No blue cycles.
2. Blue edges form a forest.
3. Blue edges form a spanning tree.
   (Otherwise, there is a cut with no blue edge.)
4. Every edge is colored.
5. Every blue edge is in
   the MST (Property 4).



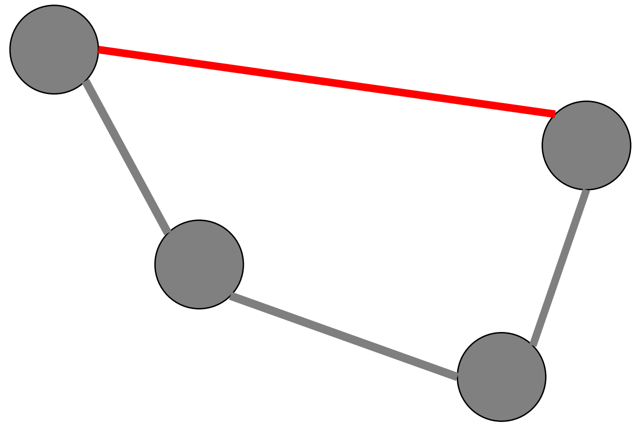
Generic MST Algorithm

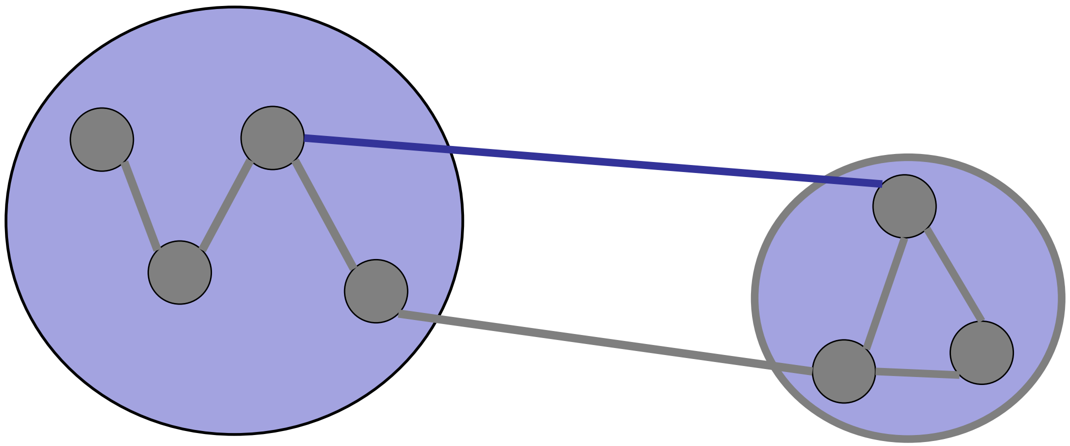
Greedy Algorithm:

Repeat:

Apply red rule or blue rule to an arbitrary edge.

until no more edges can be colored.





Candidate MST Algorithm

Divide-and-Conquer:

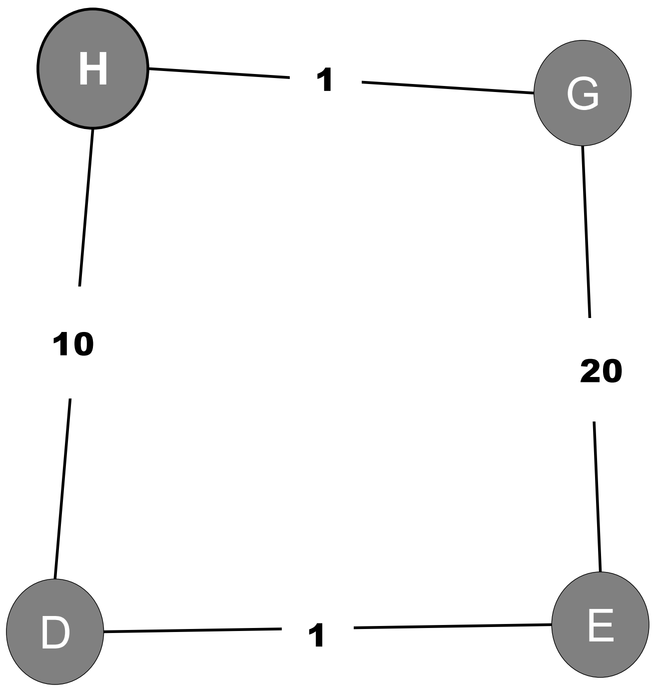
1. If the number of vertices is 1, then return.
2. Divide the nodes into two sets.
3. Recursively calculate the MST of each set.
4. Find the lightest edge the connects the two sets and add it to the MST.
5. Return.

The problem with this algorithm is?

1. Nothing. It efficiently implements the red-blue strategy.
2. It is too expensive to implement because finding the lightest edge is hard.
3. It is too expensive to implement because partitioning the nodes is expensive.
4. It returns the wrong answer.

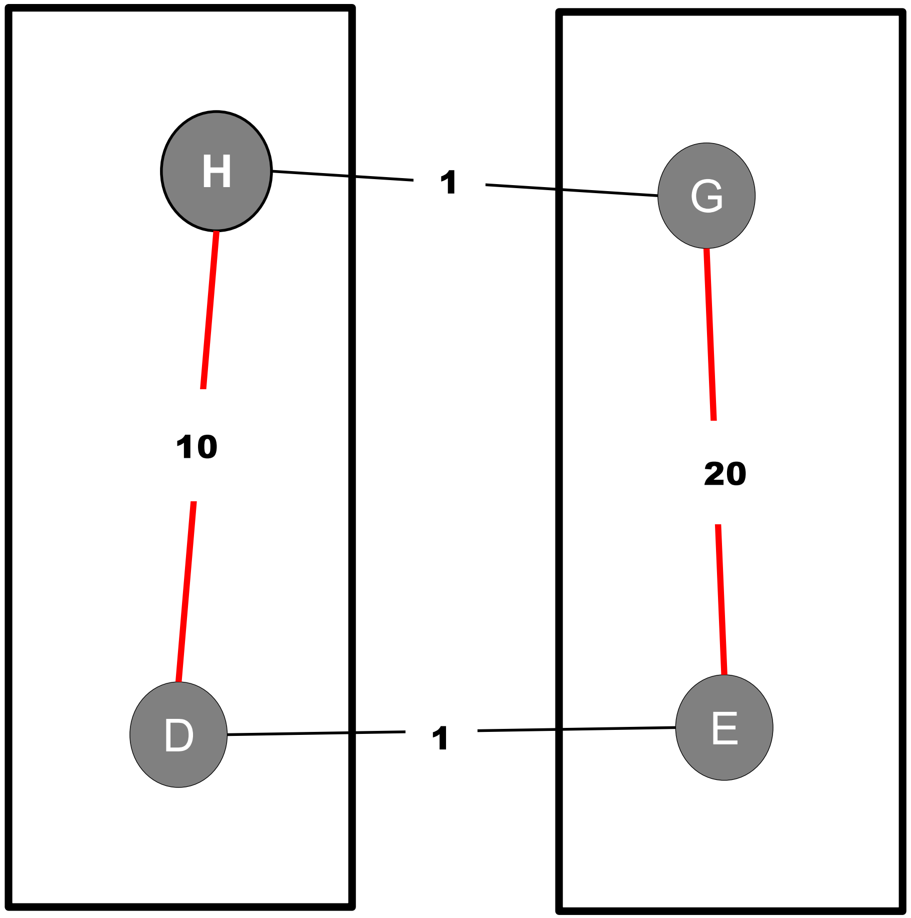
Candidate MST Algorithm

Example:



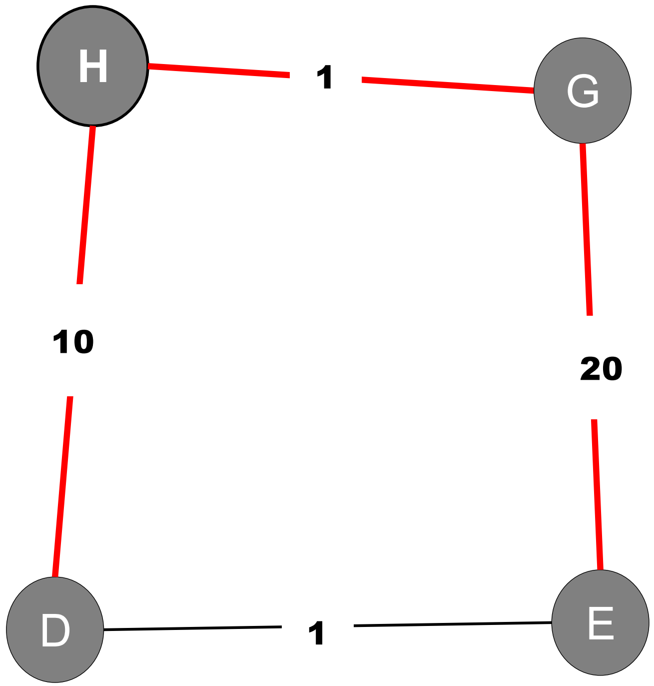
Candidate MST Algorithm

Example: Divide-and-Conquer



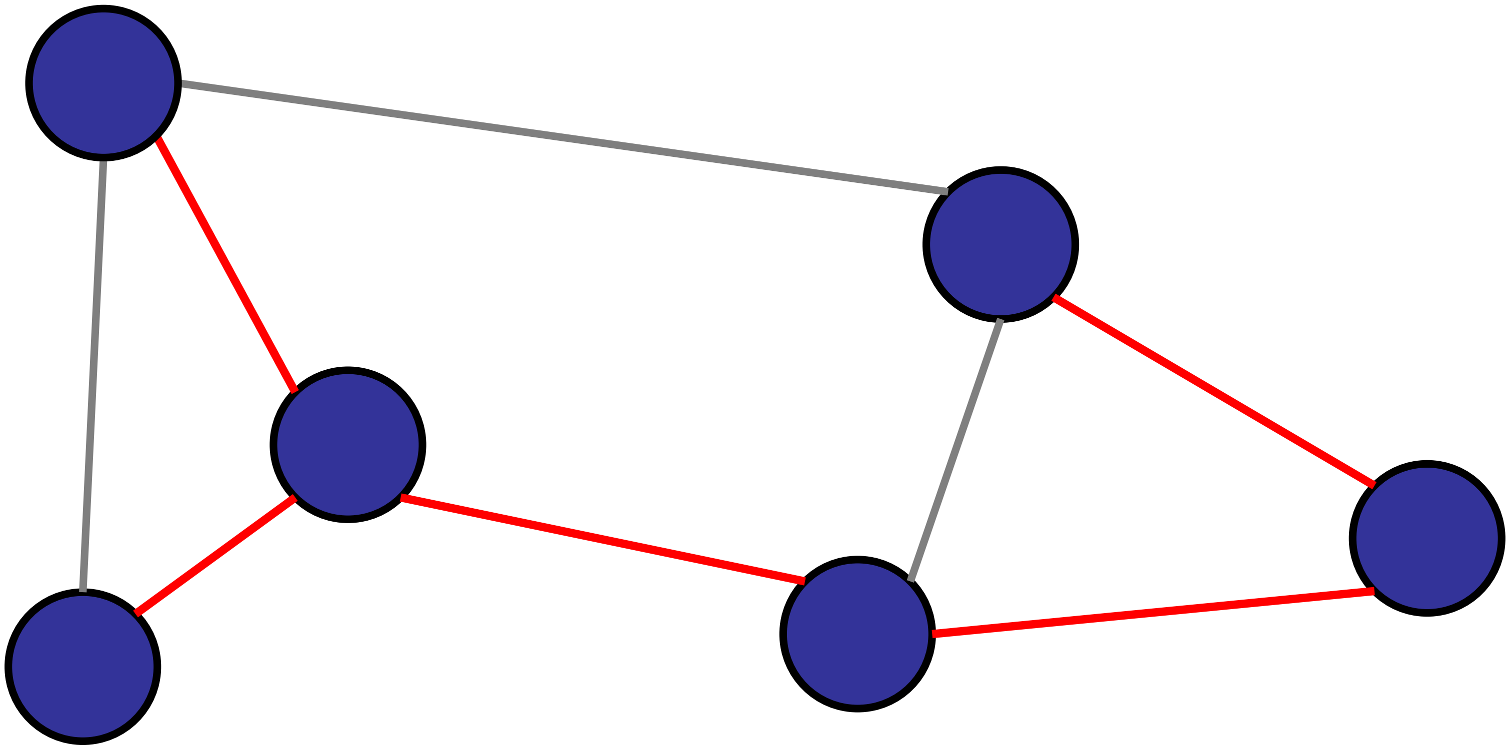
Candidate MST Algorithm

Example: Divide-and-Conquer



Properties of MST

Property 2: If you cut an MST, the two pieces are both MSTs.



BAD MST Algorithm

Divide-and-Conquer:

1. If the number of vertices is 1, then return.
2. Divide the nodes into two sets.
3. Recursively calculate the MST of each set.
4. Find the lightest edge the connects the two sets and add it to the MST.
5. Return.

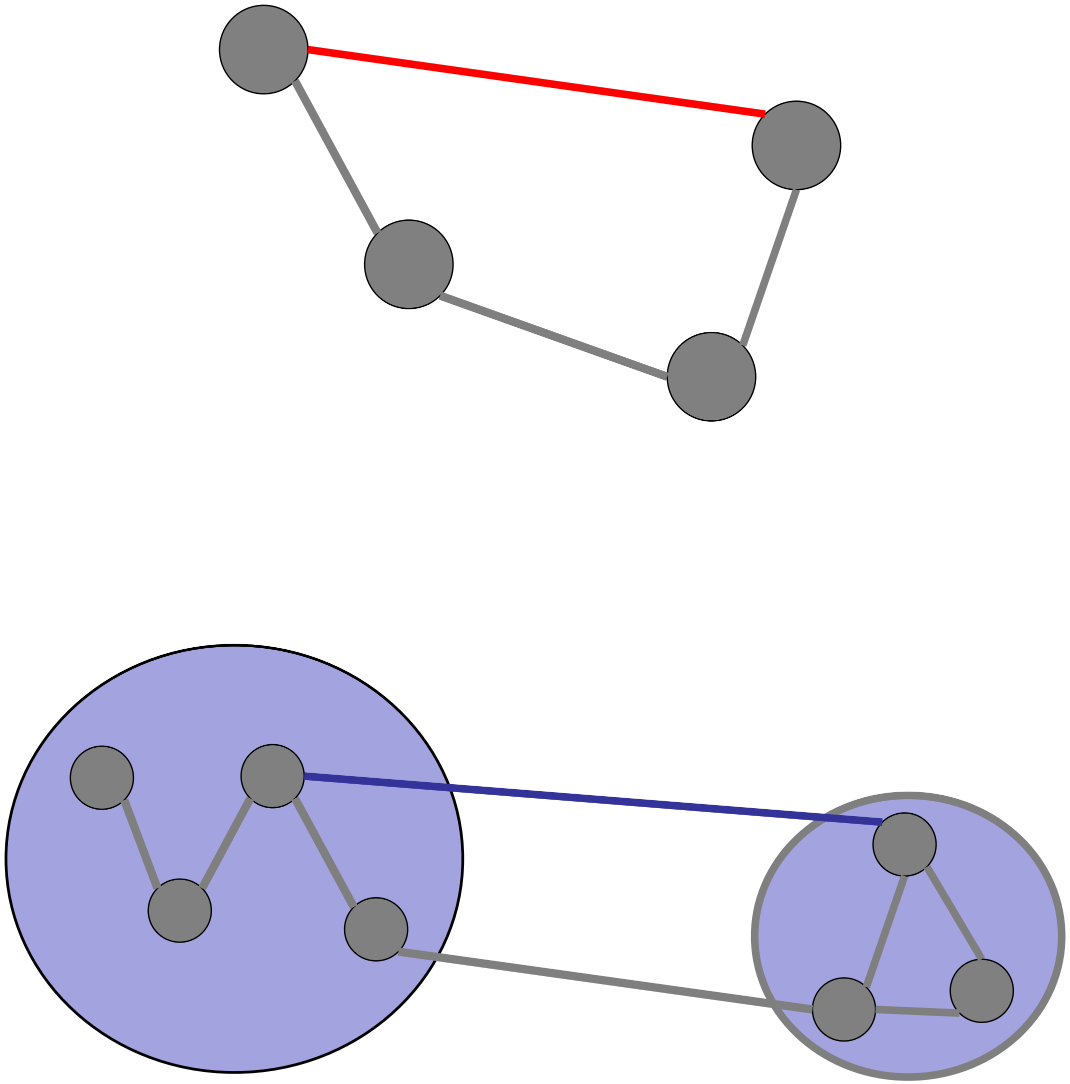
Generic MST Algorithm

Greedy Algorithm:

Repeat:

Apply red rule or blue rule to an arbitrary edge.

until no more edges can be colored.



Roadmap

Minimum Spanning Trees

- The MST Problem
- Basic Properties of an MST
- Generic MST Algorithm
- **Prim’s Algorithm**
- Kruskal’s Algorithm
- Boruvka’s Algorithm
- Variations