CS2040S
Data Structures and Algorithms
(e-learning edition)

All about minimum spanning trees…

Roadmap

Last time: Minimum Spanning Trees

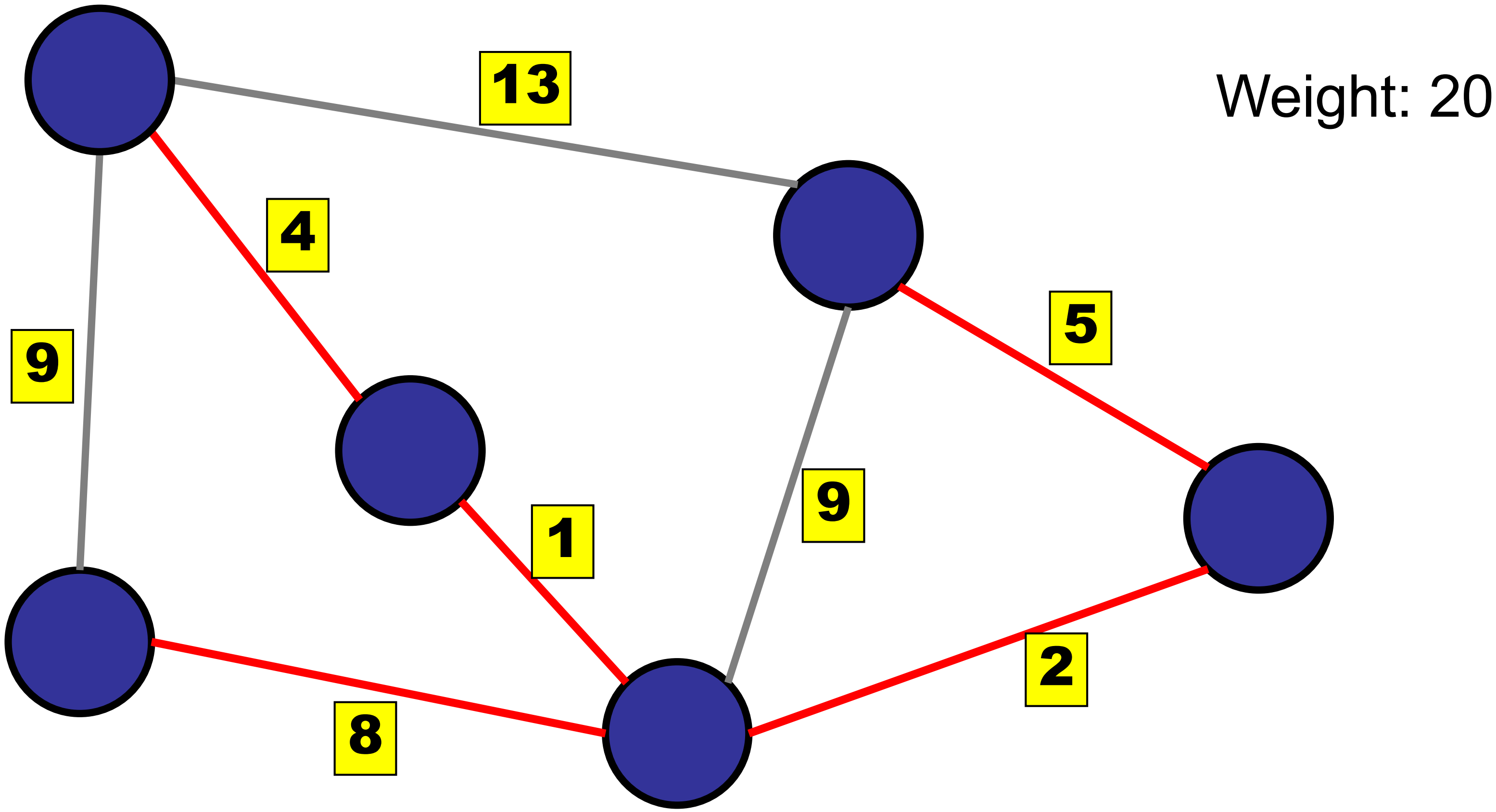
- Prim's Algorithm
- Kruskal's Algorithm
- Boruvka's Algorithm

Today: Variations

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

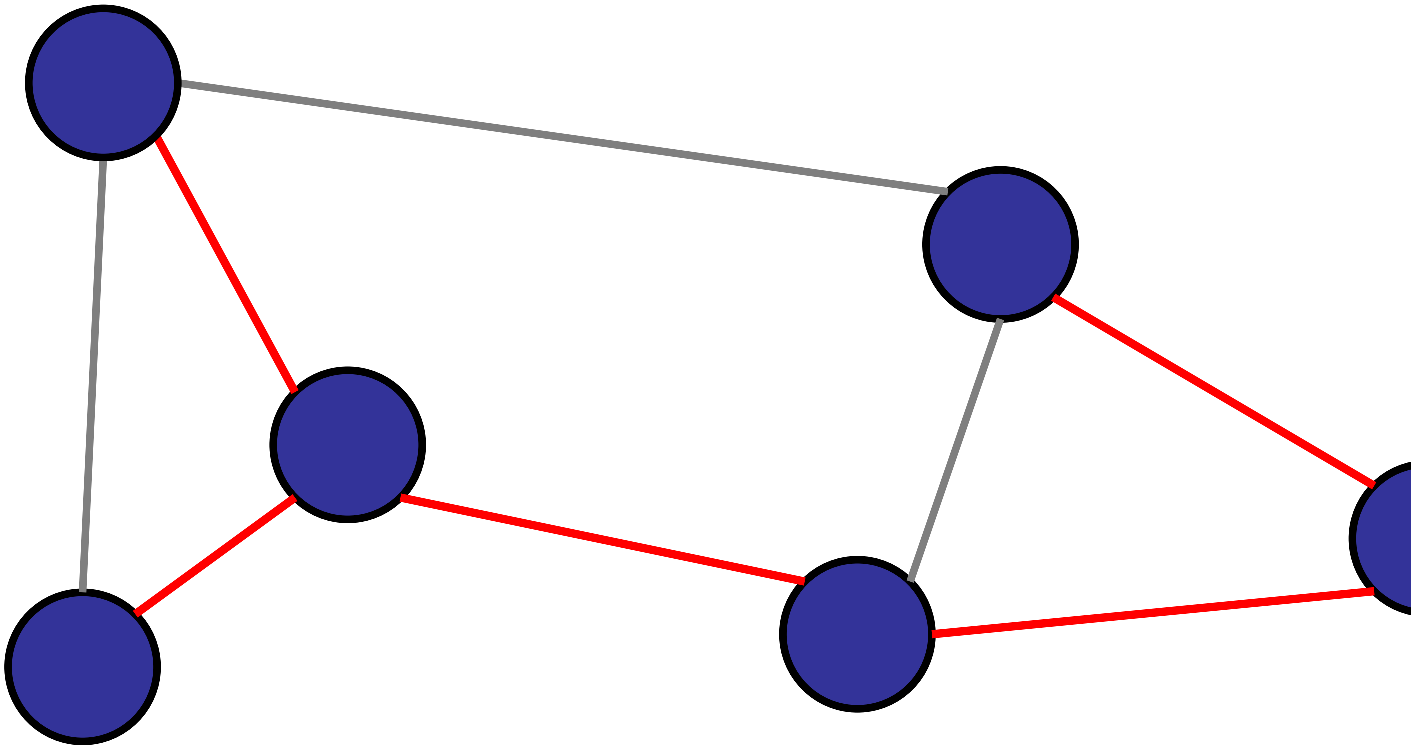
Minimum Spanning Tree

Definition: a spanning tree with minimum weight



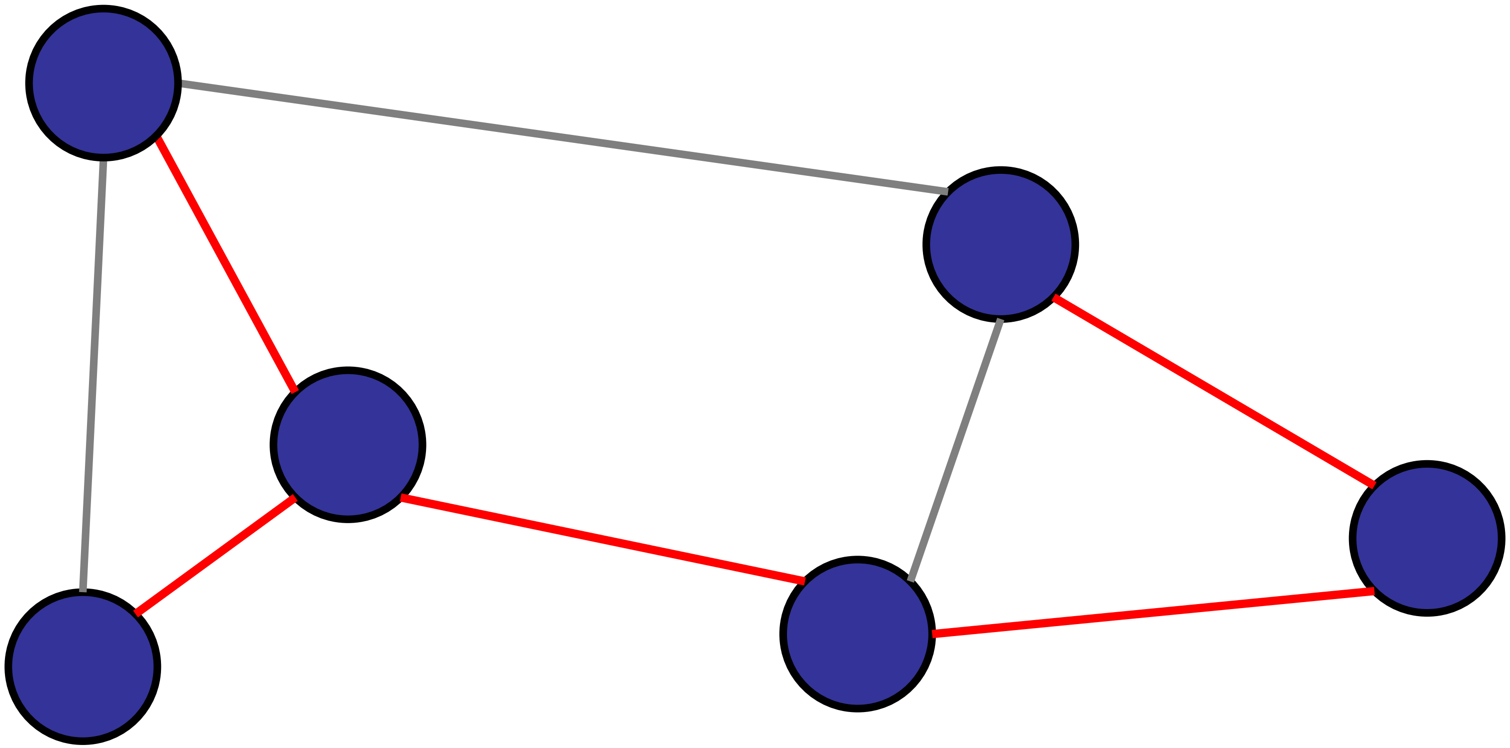
Properties of MST

Property 1: No cycles



Properties of MST

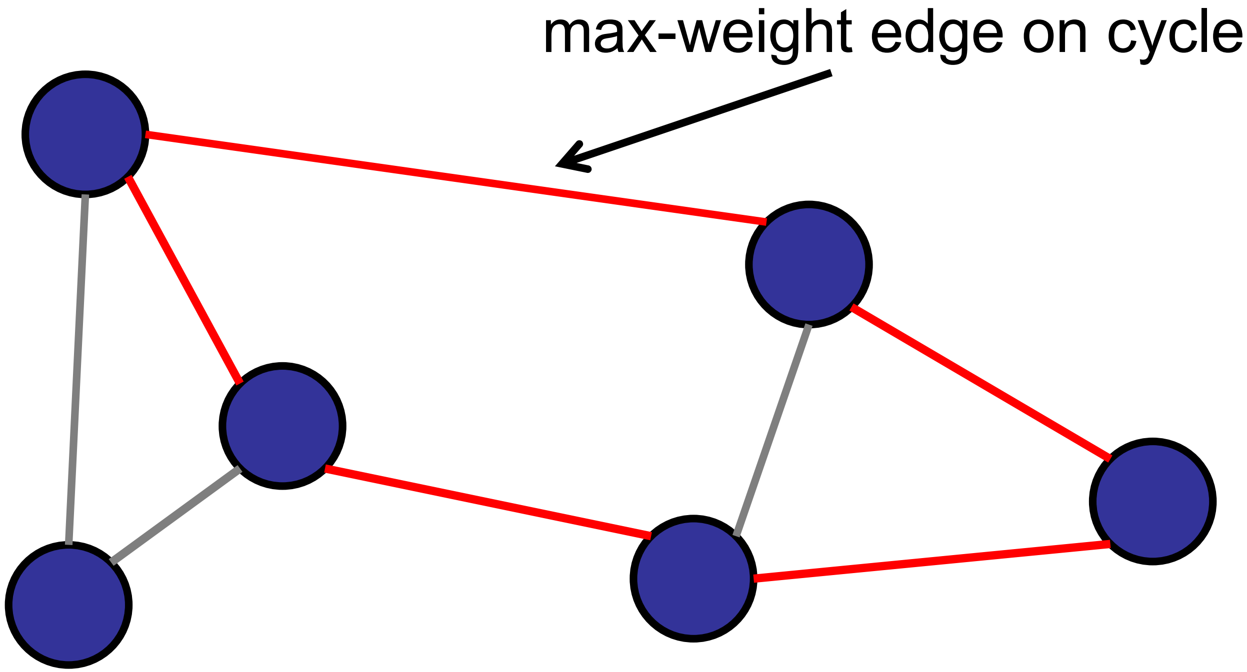
Property 2: If you cut an MST, the two pieces are both MSTs.



Properties of MST

Property 3: Cycle property

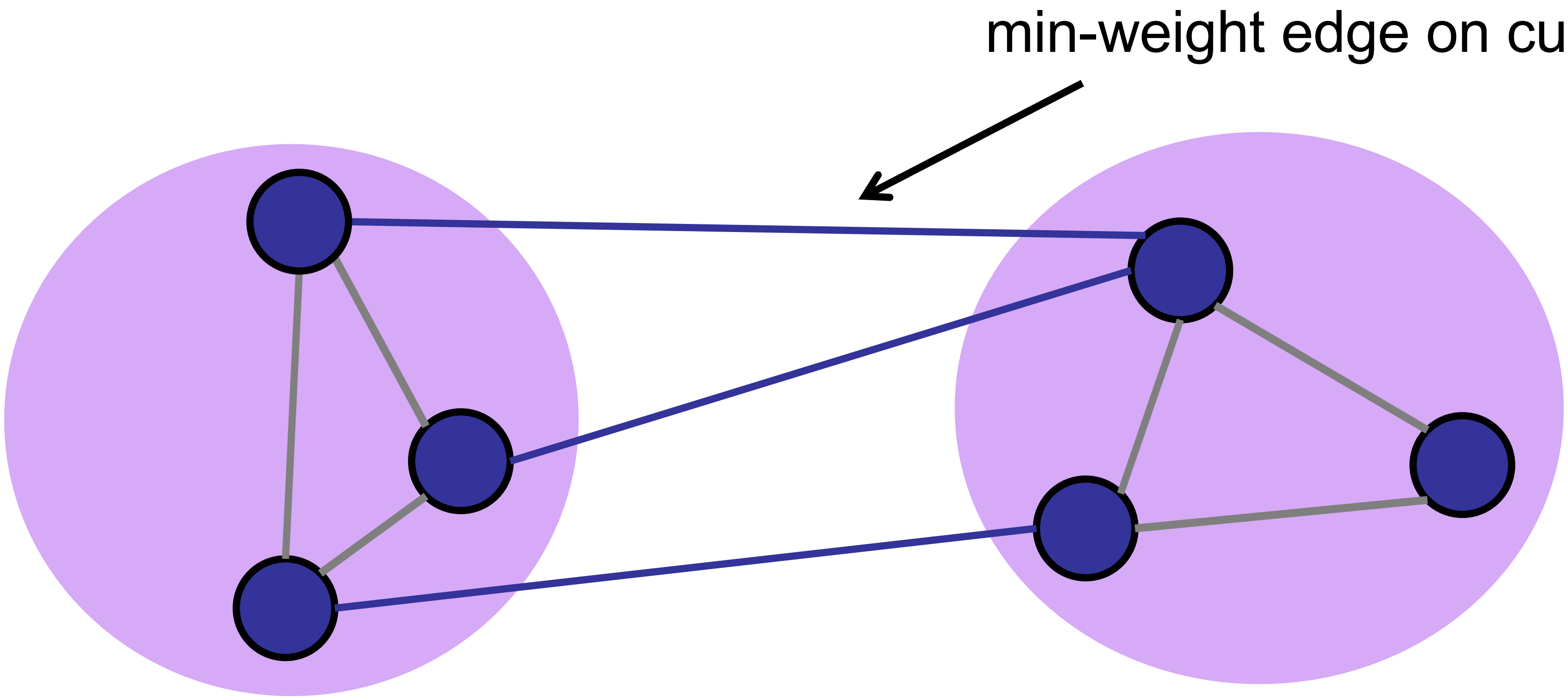
For every cycle, the maximum weight edge is *not* in the MST.



Properties of MST

Property 4: Cut property

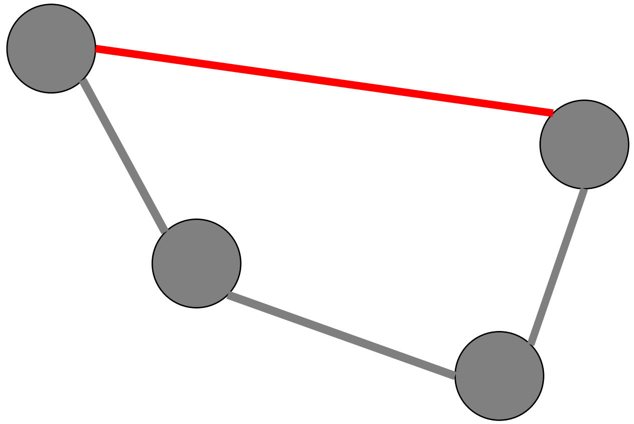
For every cut D, the minimum weight edge that crosses the cut *is* in the MST.



Generic MST Algorithm

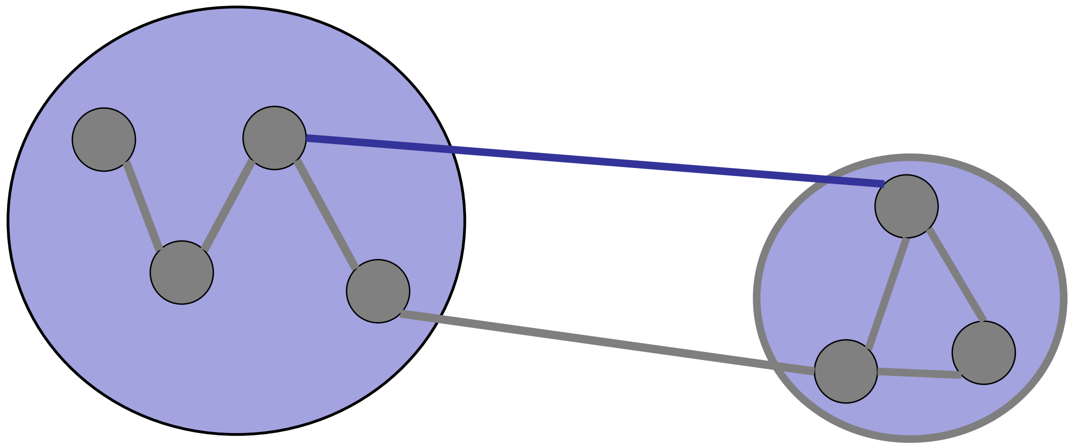
Red rule:

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.



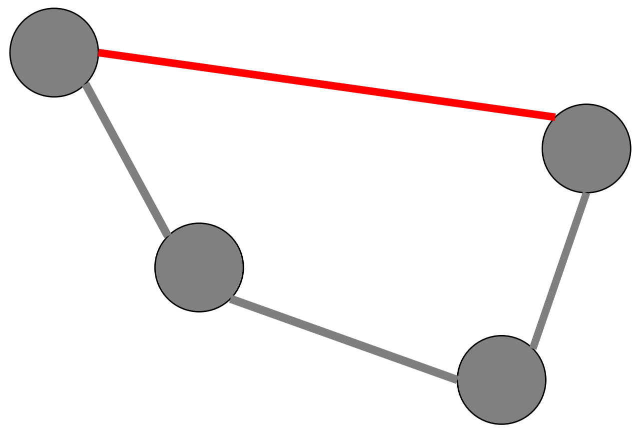
Generic MST Algorithm

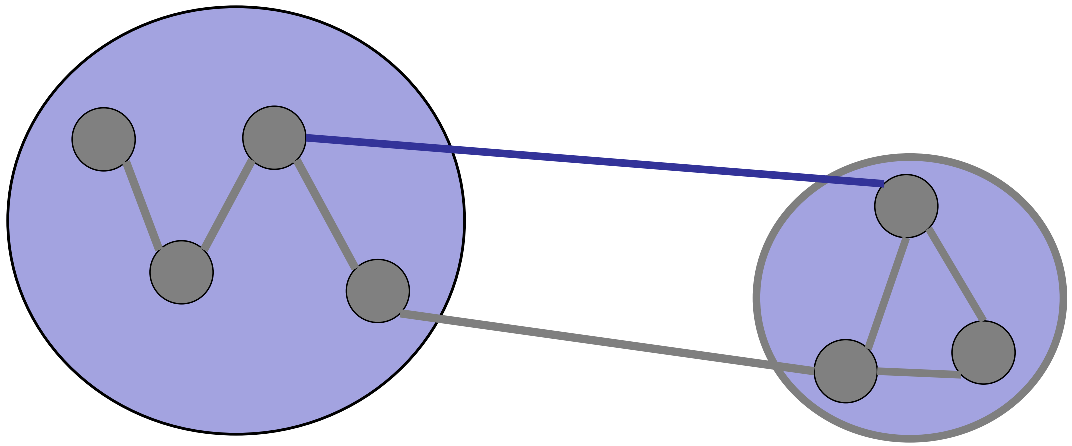
Greedy Algorithm:

Repeat:

Apply red rule or blue rule to an arbitrary edge.

until no more edges can be colored.



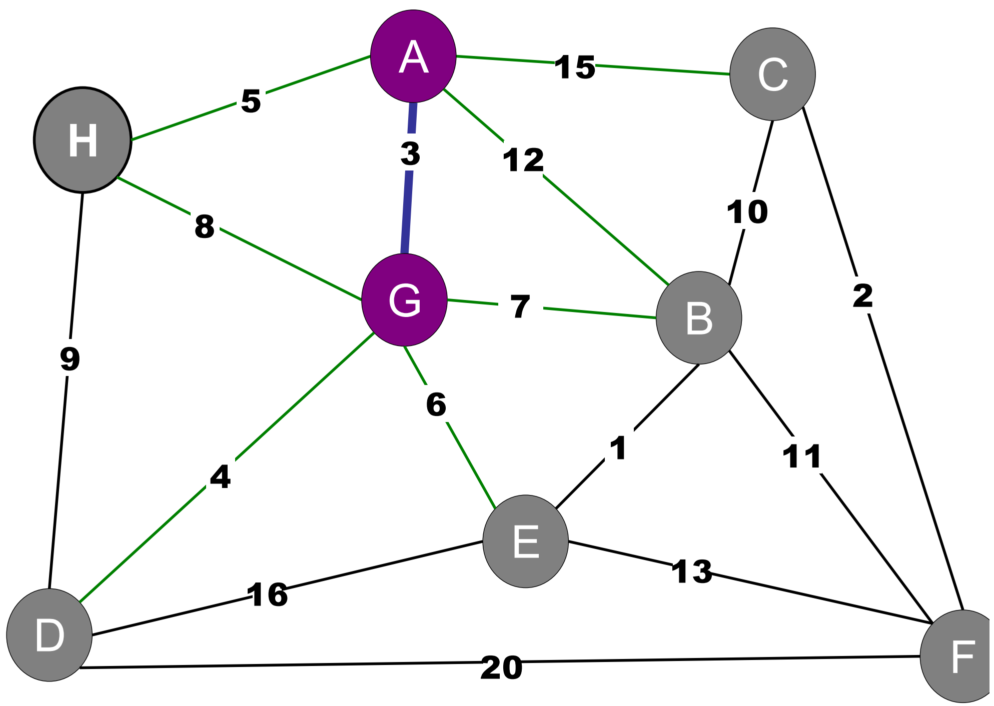


Prim's Algorithm

Prim's Algorithm. (Jarnik 1930, Dijkstra 1957, Prim 1959)

Basic idea:

- S : set of nodes connected by blue edges.
- Initially: S = {A}
- Repeat:
  - Identify cut: {S, V–S}
  - Find minimum weight edge on cut.
  - Add new node to S.



Analysis:

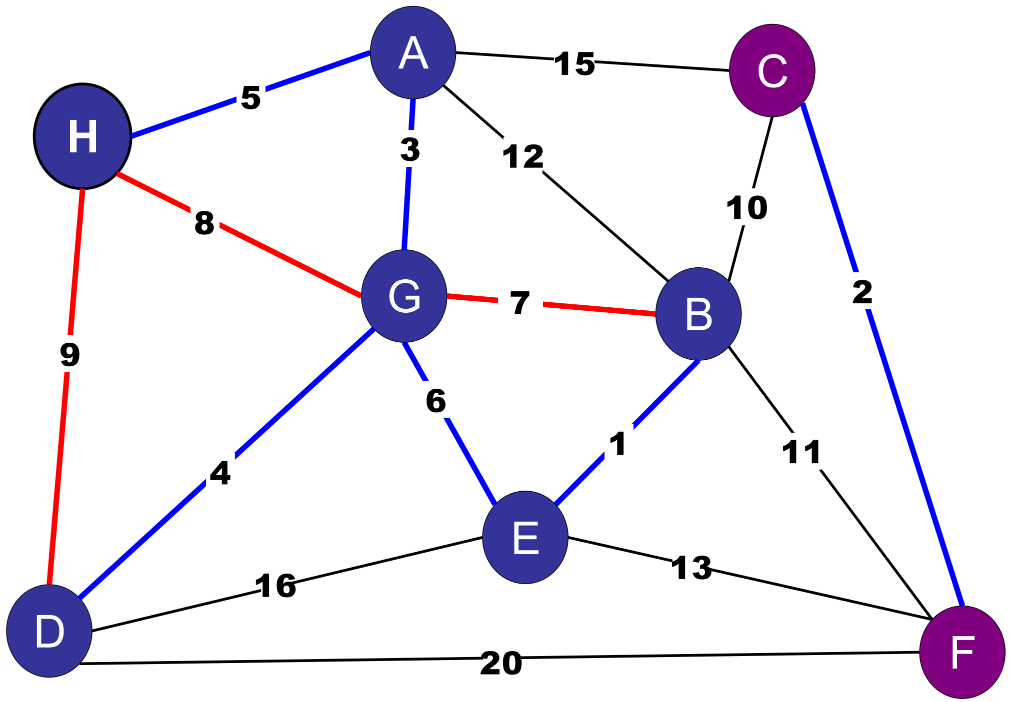
- Running time: O(E log V).

Kruskal's Algorithm

Kruskal's Algorithm. (Kruskal 1956)

Basic idea:

- Sort edges by weight.
- Consider edges in ascending order:
  - If both endpoints are in the **same** blue tree, then color the edge red.
  - Otherwise, color the edge blue.



Performance:

- Sorting: $O(E \log E) = O(E \log V)$
- For E edges:
  - Find: $O(\alpha(n))$ or $O(\log V)$
  - Union: $O(\alpha(n))$ or $O(\log V)$

Boruvka's Algorithm

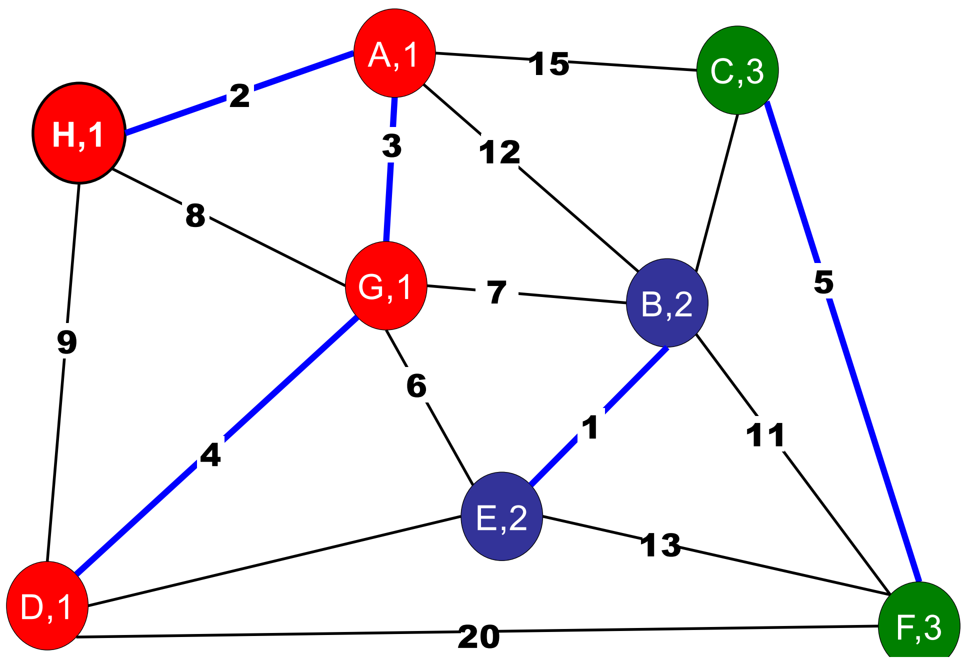
Boruvka's Algorithm

Initially:

- Create n connected components, one for each node in the graph.

Repeat "Boruvka" Steps:

- For each connected component, search for the minimum weight outgoing edge.
- Add selected edges.
- Merge connected components.



Roadmap

Last time: Minimum Spanning Trees

- Prim's Algorithm
- Kruskal's Algorithm
- Boruvka's Algorithm

Today: Variations

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

MST Variants

What if all the edges have the same weight?

How fast can you find an MST?

1. O(V)
2. O(E)
3. O(E log V)
4. O(V log E)
5. O(VE)

MST Variants

What if all the edges have the same weight?

– Depth-First-Search or Breadth-First-Search

If all edge-weights are 2, what is the **cost** of a MST?

1. V-1
2. V
3. 2(V-1)
4. 2V
5. E-V
6. E

MST Variants

What if all the edges have the same weight?

- Depth-First-Search or Breadth-First-Search
- An MST contains exactly (V-1) edges.
- Every spanning tree contains (V-1) edges!
- Thus, any spanning tree you find with DFS/BFS is a minimum spanning tree.

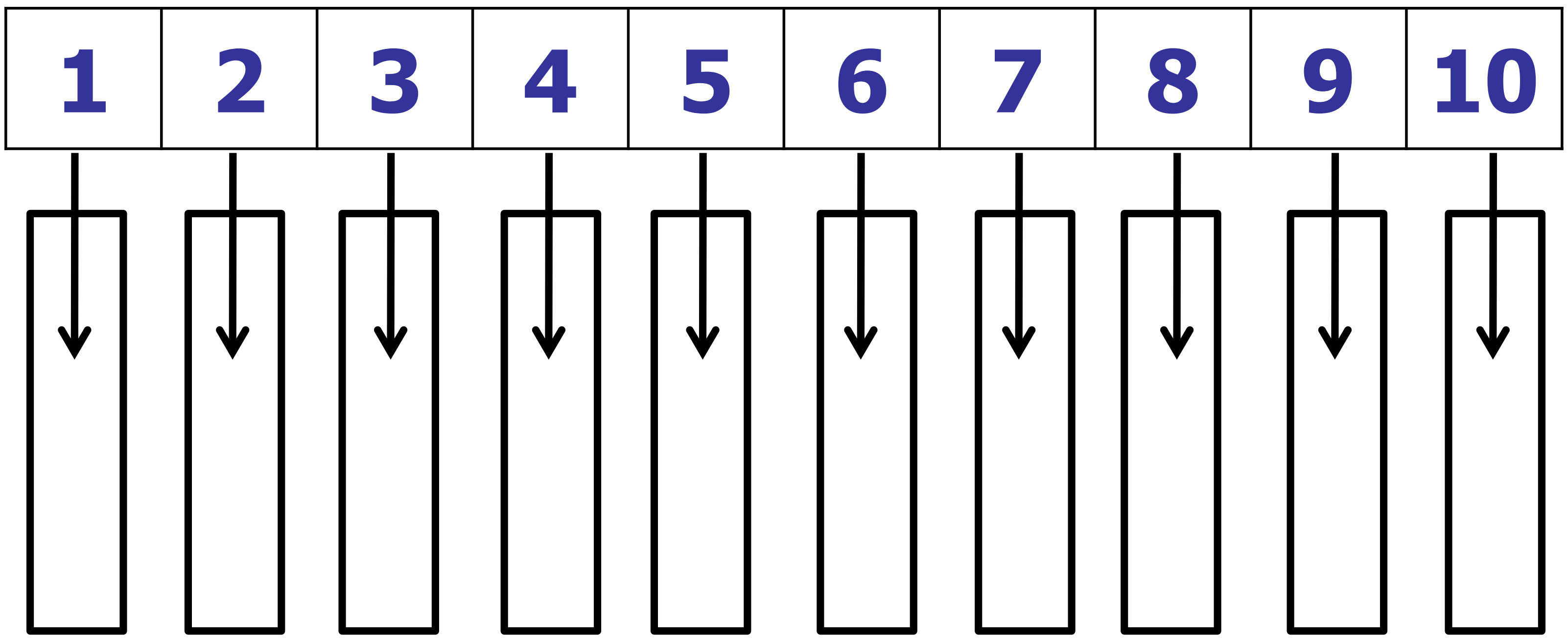
Kruskal’s Variants

What if all the edges have weights from {1..10}?

Kruskal's Variants

What if all the edges have weights from {1..10}?

Idea: Use an array of size 10



slot A[j] holds a linked list of edges of weight j

Kruskal's Variants

What if all the edges have weights from {1..10}?

Idea: Use an array of size 10

- Putting edges in array of linked lists: O(E)
- Iterating over all edges in ascending order: O(E)
- Checking whether to add an edge: O(α)
- Union two components: O(α)

Total: O(αE)

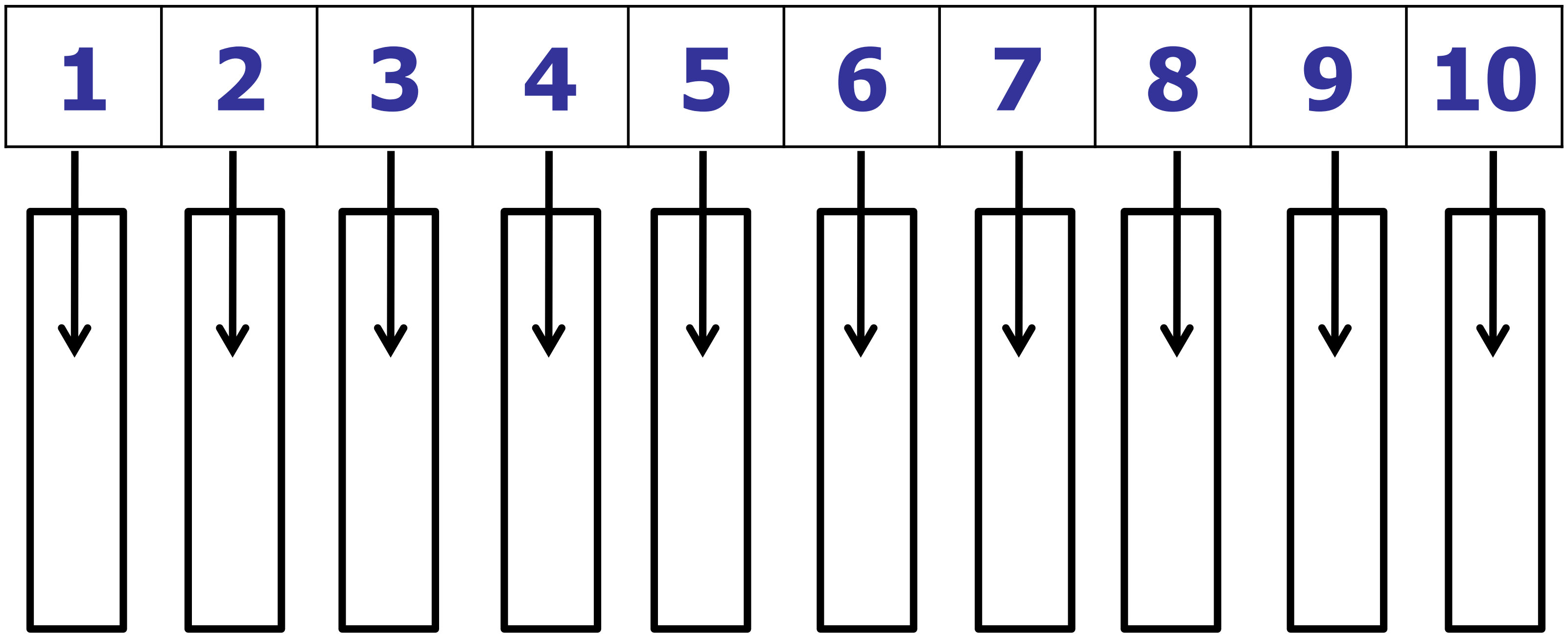
What is the running time of (modified) Prim's if all the edge weights are in {1..10}?

1. O(V)
2. O(E)
3. O(E log V)
4. O(V log E)
5. O(EV)

Prim's Variants

What if all the edges have weights from {1..10}?

Idea: Use an array of size 10 as a Priority Queue

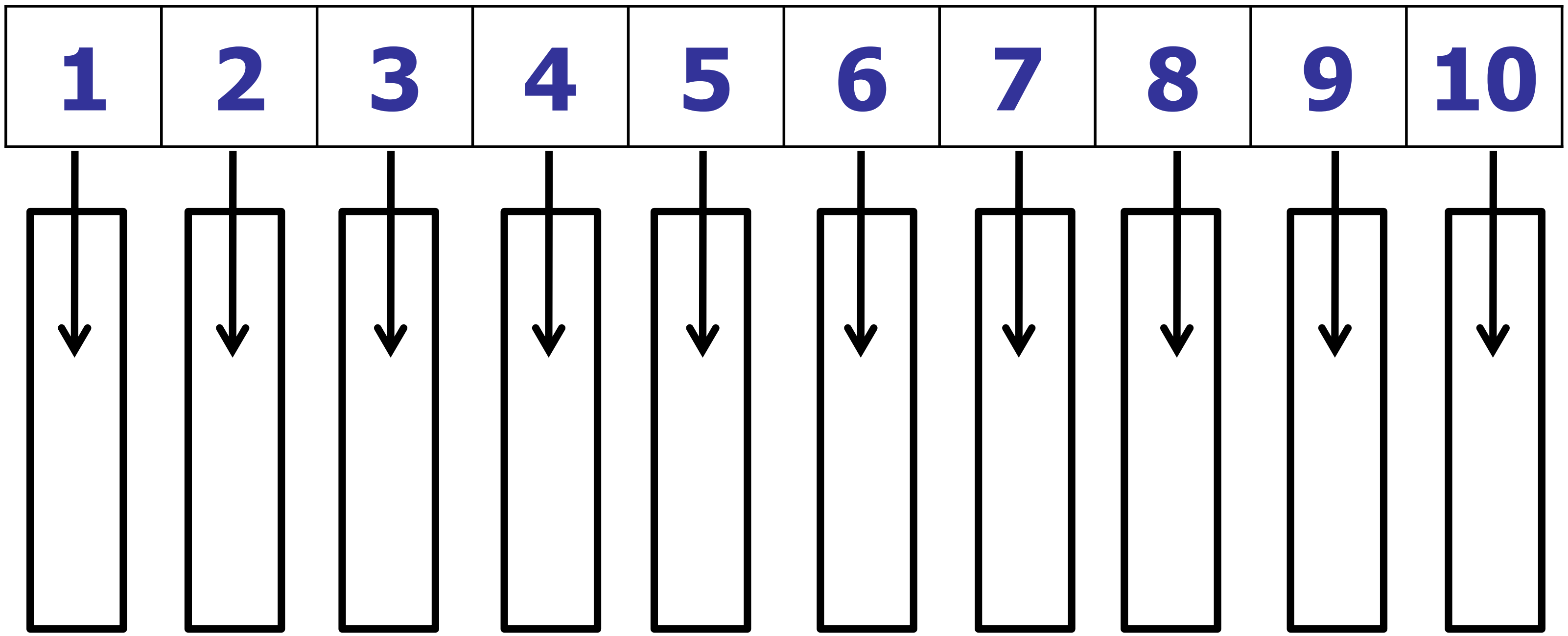


slot A[j] holds a linked list of **nodes** of weight j

Prim’s Variants

What if all the edges have weights from {1..10}?

Idea: Use an array of size 10 as a Priority Queue



decreaseKey: move node to new linked list

Prim’s Variants

What if all the edges have weights from {1..10}?

Implement Priority Queue:

- Use an array of size 10 to implement
- Insert: put node in correct list
- Remove: lookup node (e.g., in hash table) and remove from liked list.
- ExtractMin: Remove from the minimum bucket.
- DecreaseKey: lookup node (e.g., in hash table) and move to correct liked list.

Prim's Variants

What if all the edges have weights from {1..10}?

Idea: Use an array of size 10

- Inserting/Removing nodes from PQ: $O(V)$
- decreaseKey: $O(E)$

Total: $O(V + E) = O(E)$

Prim's Variants

What if all the edges have weights from {1..10}?

Implement Priority Queue….

Why does this fail for Dijkstra's Algorithm?

Roadmap

Today: Minimum Spanning Trees

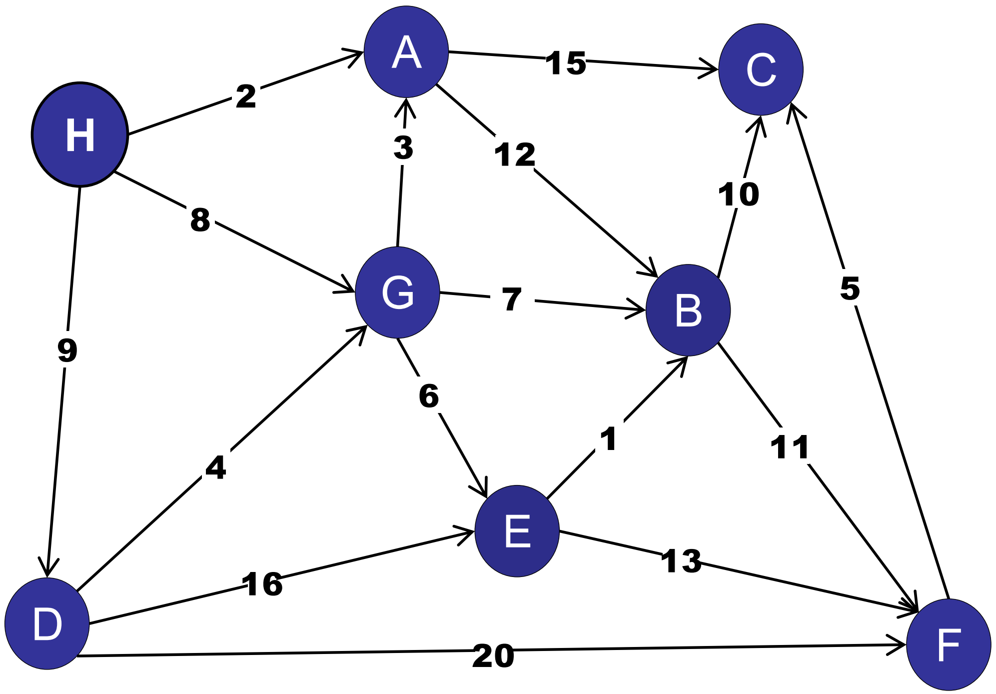
- Prim's Algorithm
- Kruskal's Algorithm
- Boruvka's Algorithm

Variations:

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

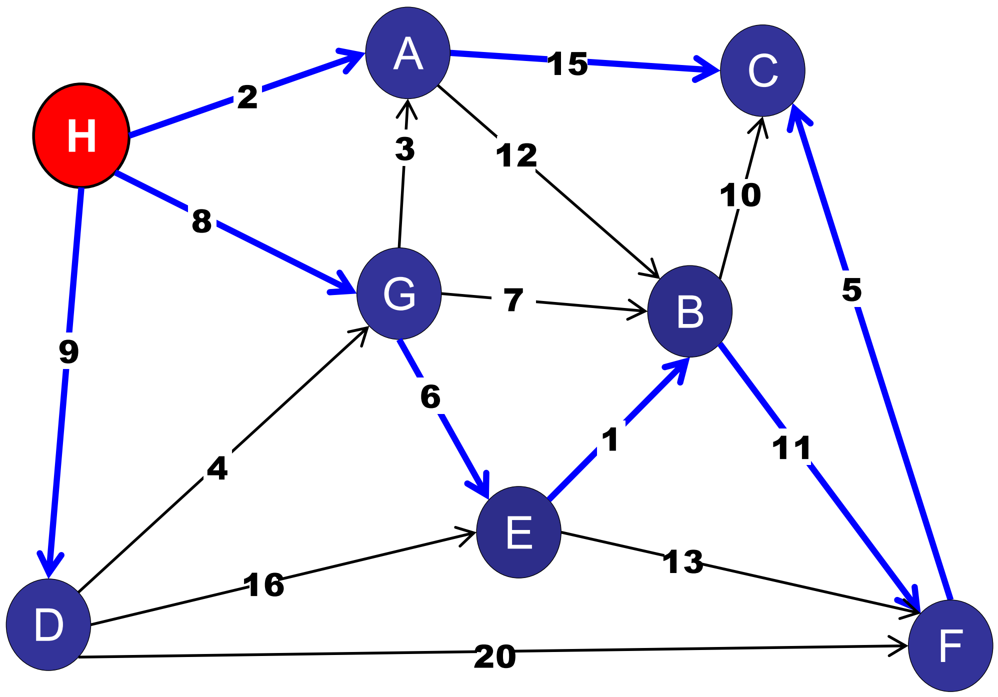
Directed Minimum Spanning Tree

What if the edges are directed?



Directed Minimum Spanning Tree

A rooted spanning tree:



Every node is reachable on a path from the root.

No cycles.

Directed Minimum Spanning Tree

Harder problem:

- Cut property does not hold.
- Cycle property does not hold.
- Generic MST algorithm does not work.

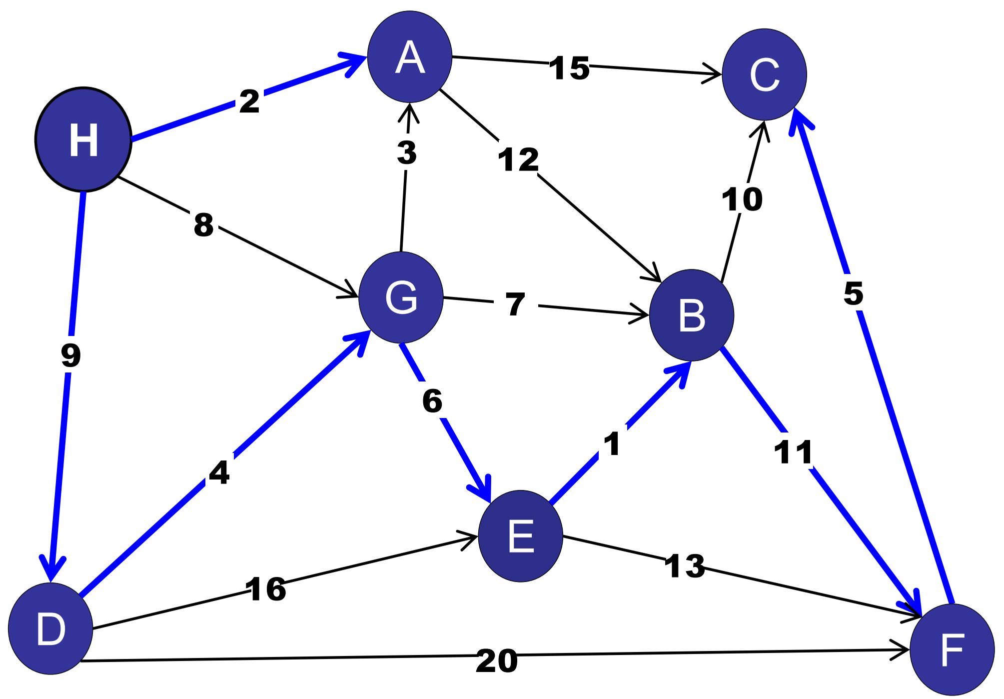
Prim's, Kruskal's, Boruvka's do not work.

See CS3230 / CS4234 for more details…

Directed Minimum Spanning Tree

For a directed acyclic graph with one "root":

For every node except the root: add minimum weight incoming edge.



Directed Minimum Spanning Tree

For a directed acyclic graph with one "root":

For every node except the root: add minimum weight incoming edge.

Observations:

- No cycles (since acyclic graph).
- Each edge is chosen only once.

Tree:
V nodes
V – 1 edges
No cycles

Directed Minimum Spanning Tree

For a directed acyclic graph with one "root":

For every node except the root: add minimum weight incoming edge.

Observations:

- No cycles (since acyclic graph).
- Each edge is chosen only once.

Tree:
V nodes
V – 1 edges
No cycles

- Every node has to have at least one incoming edge in the MST, so this is the minimum spanning tree.

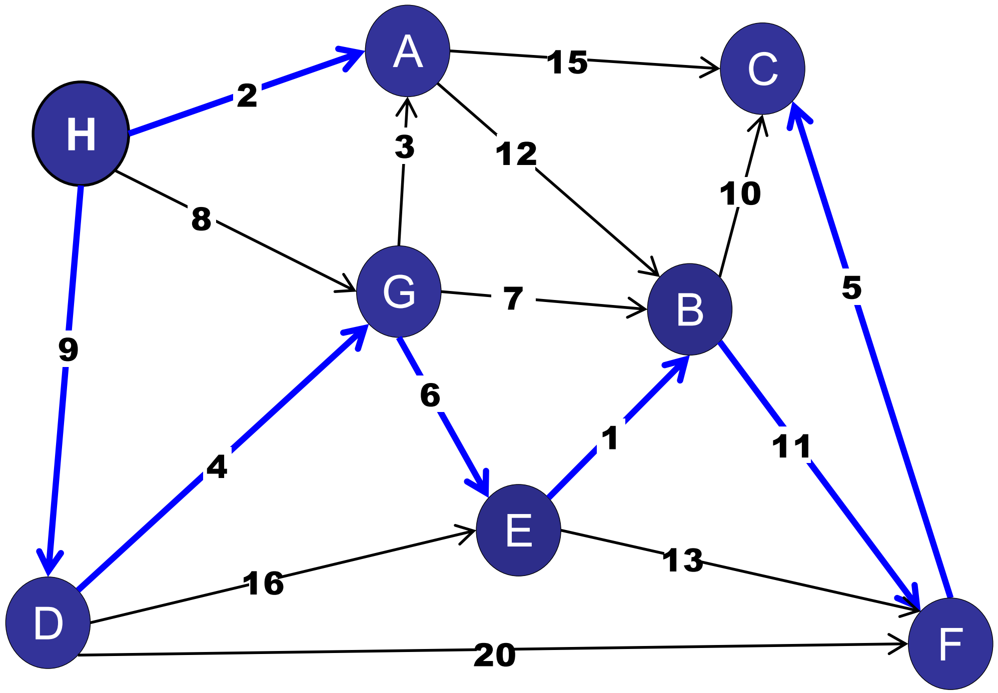
Directed Minimum Spanning Tree

For a directed acyclic graph with one "root":

For every node except the root: add minimum weight incoming edge.

Conclusion: Minimum Spanning Tree

O(E) time



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

Maximum Spanning Tree

A MaxST is a spanning tree of maximum weight.

How do you find a MaxST?

Maximum Spanning Tree

Reweighting a spanning tree:

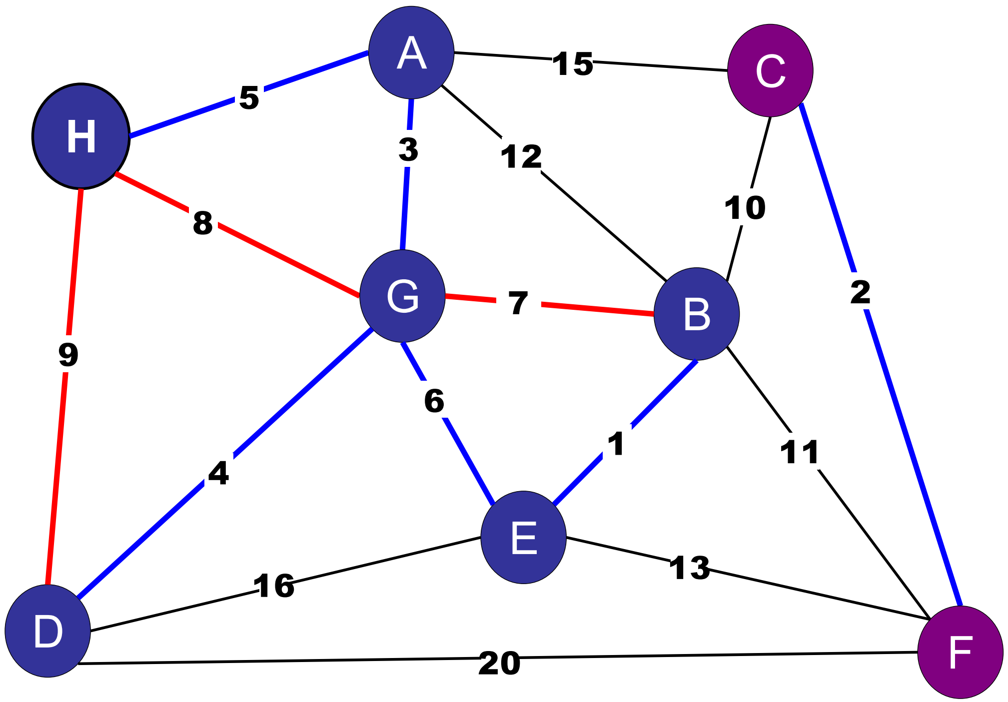
- What happens if you add a constant k to the weight of every edge?

Kruskal's Algorithm

Kruskal's Algorithm. (Kruskal 1956)

Basic idea:

- Sort edges by weight.
- Consider edges in ascending order:
  - If both endpoints are in the **same** blue tree, then color the edge red.
  - Otherwise, color the edge blue.



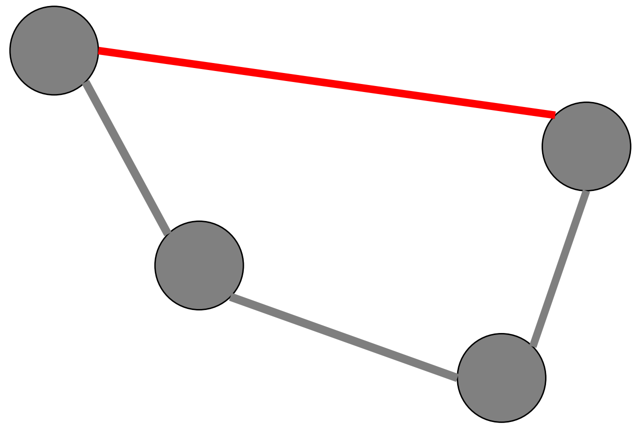
What matters?

- Relative edge weights.
- Absolute edge weights have no impact.

Generic MST Algorithm

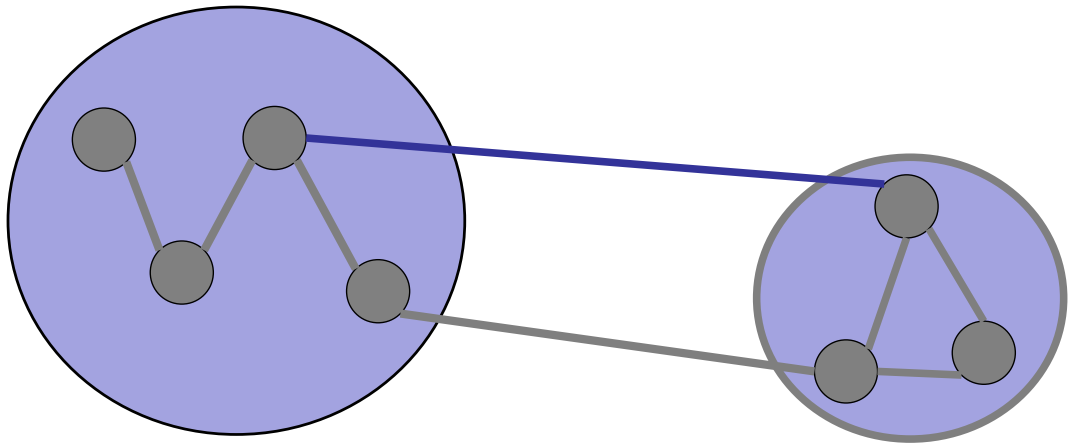
Red rule:

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.



Maximum Spanning Tree

Reweighting a spanning tree:

- What happens if you add a constant k to the weight of every edge?

No change!

We can add or subtract weights without effecting the MST.

(Very different from shortest paths…)

Maximum Spanning Tree

MST with negative weights?

Maximum Spanning Tree

MST with negative weights?

No problem!

1. Reweight MST by adding a big enough value to each edge so that it is positive.

2. Actually, no need to reweight. Only relative edge weights matter, so negative weights have no bad impact.

Maximum Spanning Tree

A MaxST is a spanning tree of maximum weight.

How do you find a MaxST?

Easy!

1. Multiply each edge weight by -1.
2. Run MST algorithm.
3. MST that is “most negative” is the maximum.

Roadmap

Last time: Minimum Spanning Trees

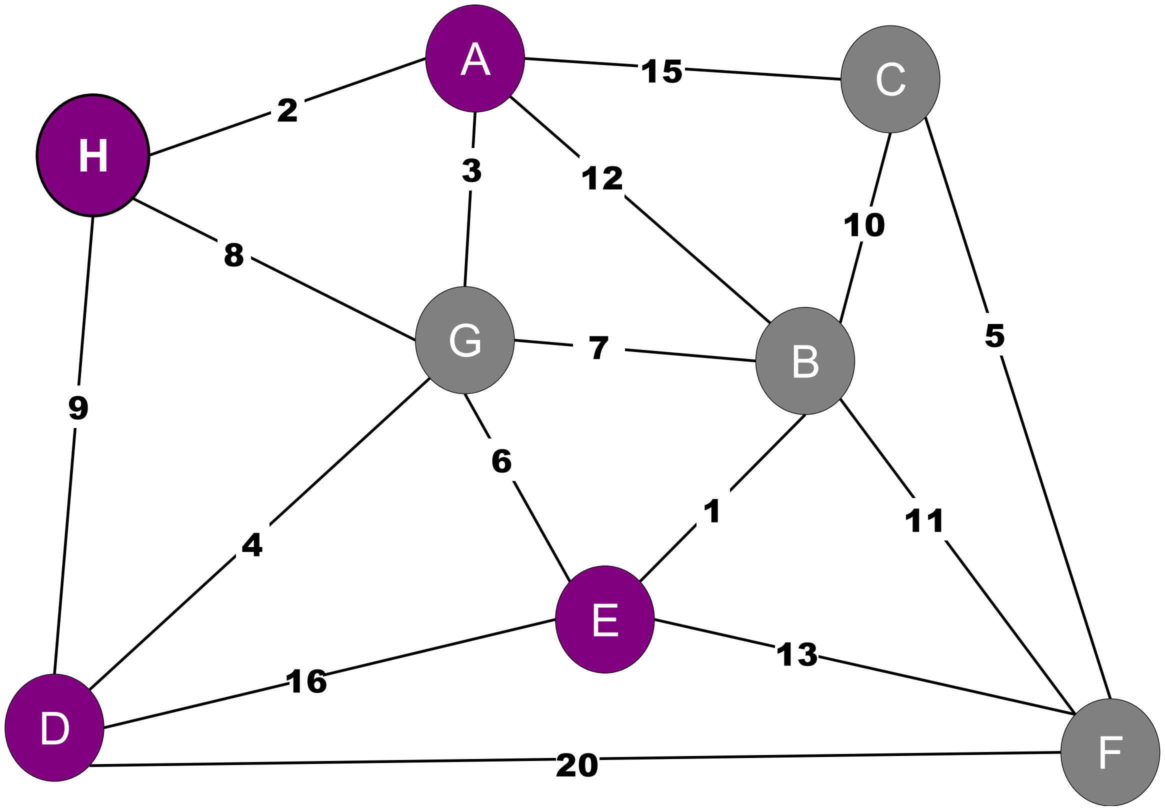
- Prim's Algorithm
- Kruskal's Algorithm
- Boruvka's Algorithm

Today: Variations

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

Steiner Tree Problem

What if I want a minimum spanning tree of a subset of the vertices?

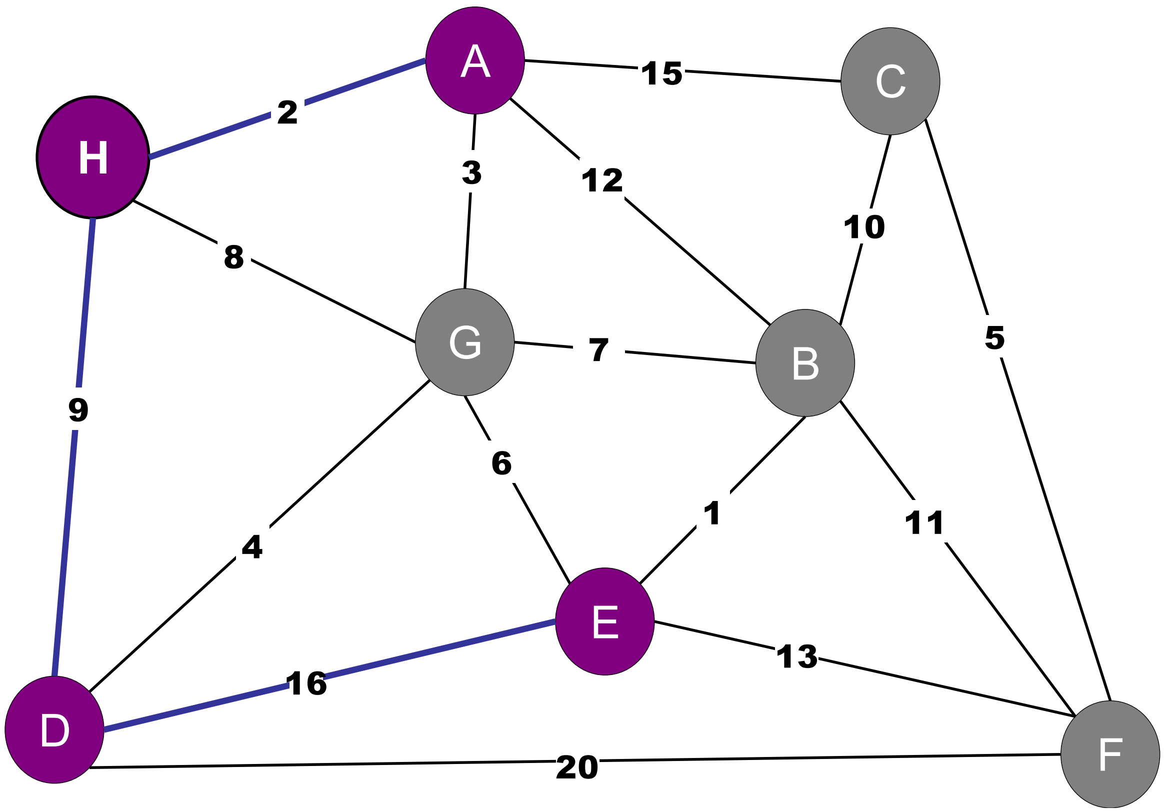


Steiner Tree Problem

What if I want a minimum spanning tree of a subset of the vertices?

1. Just use the sub-graph.

weight = 27

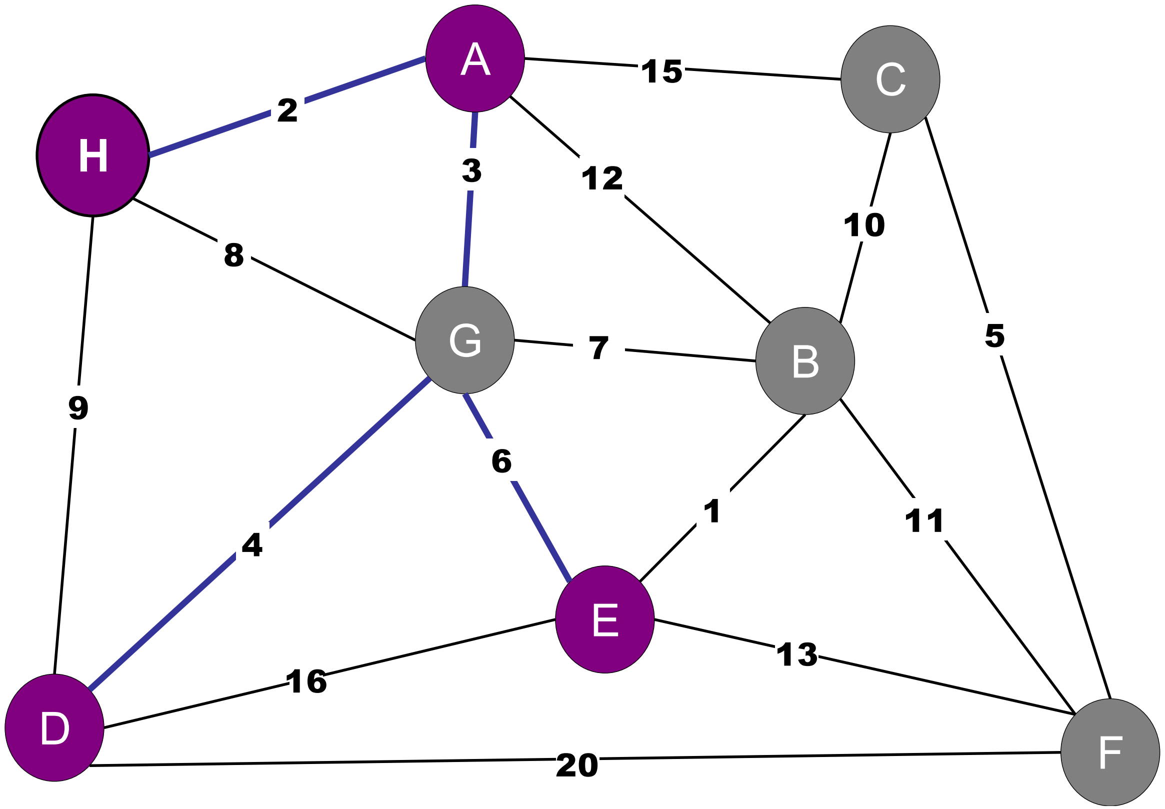


Steiner Tree Problem

What if I want a minimum spanning tree of a subset of the vertices?

1. Just use the sub-graph.
2. Use other nodes.

weight = 15

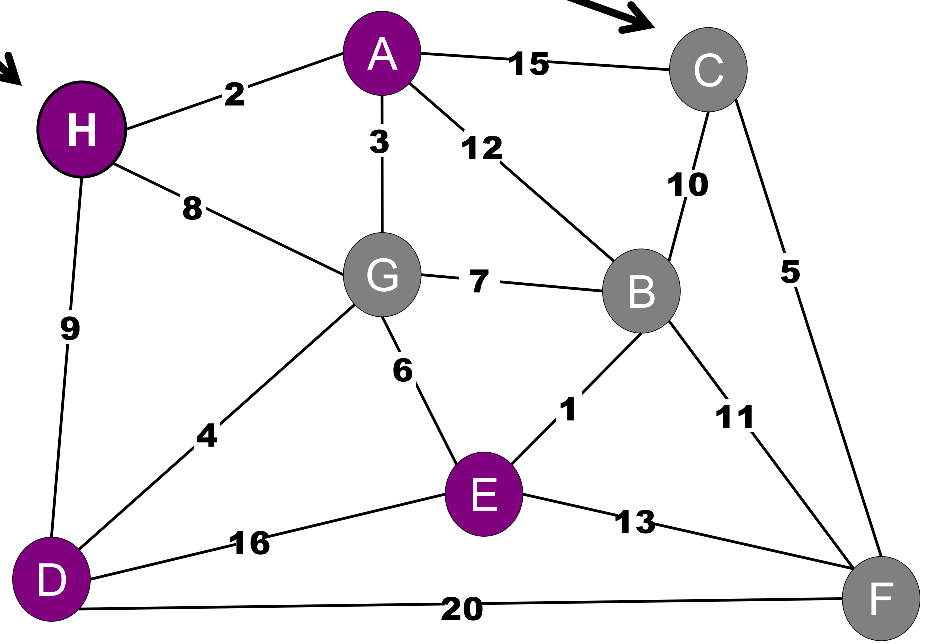


Steiner Tree Problem

What is the minimum spanning tree of a subset of the vertices?

- Steiner nodes
- Required nodes

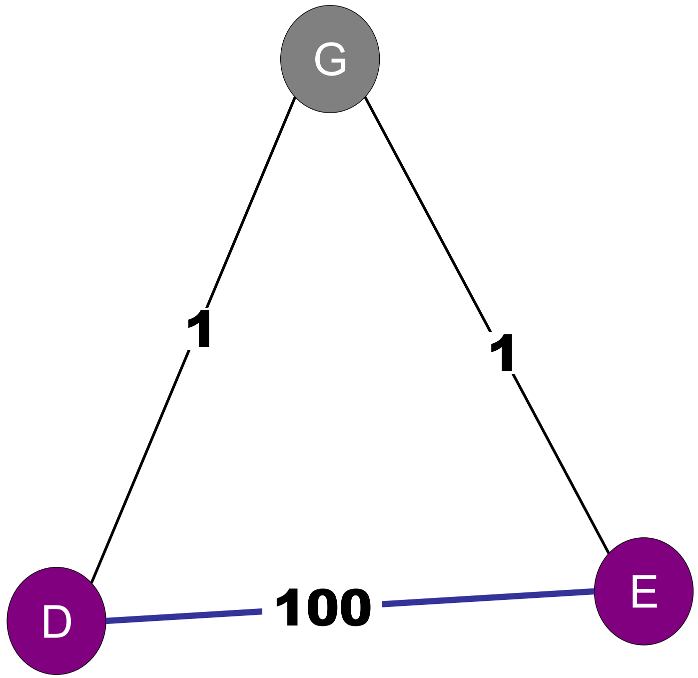
Find spanning tree of required vertices. You may include Steiner vertices, optionally.



Steiner Tree Problem

Just calculate MST doesn't work:

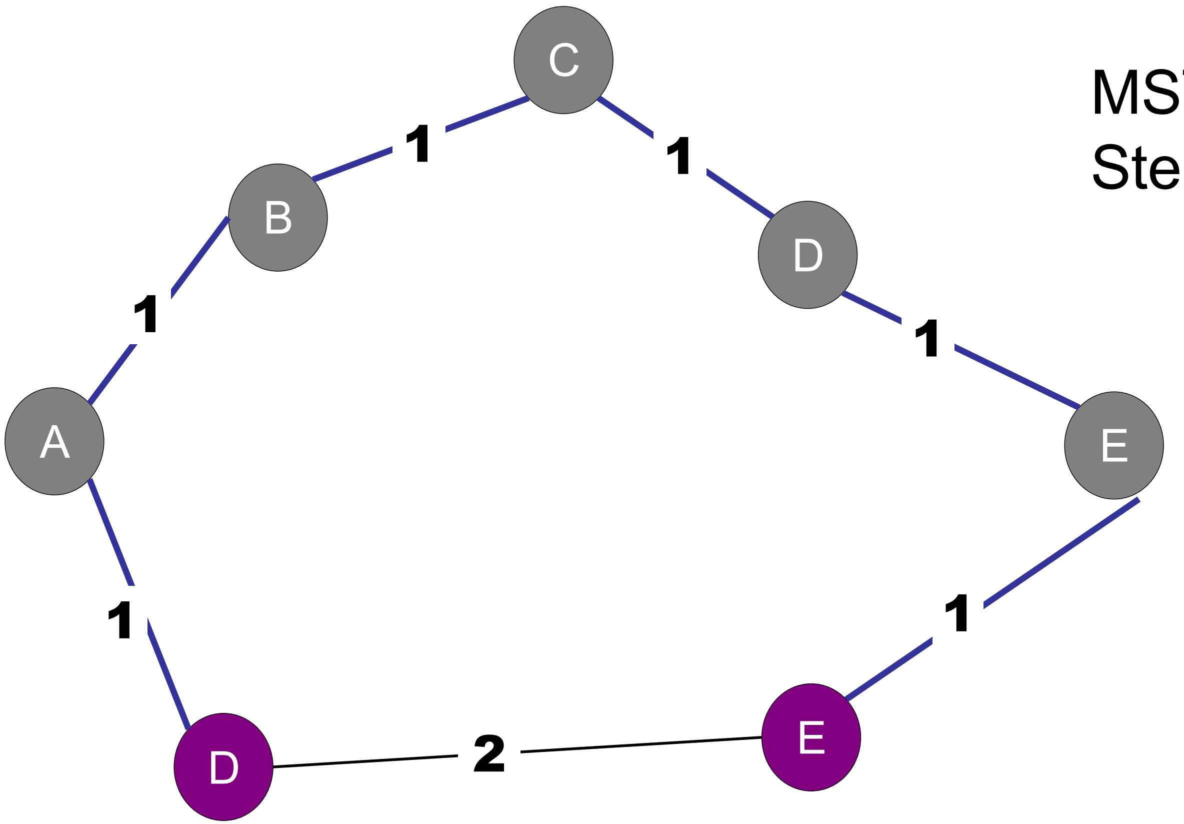
1. Calculate MST with no Steiner nodes.



Steiner Tree Problem

Just calculate MST doesn't work:

2. Calculate MST with all Steiner nodes.



MST = 6

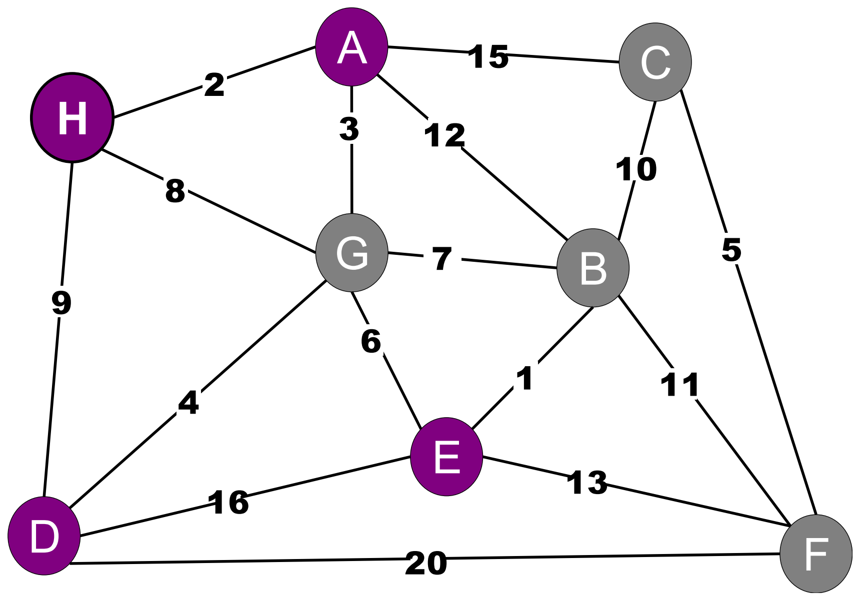
Steiner Tree = 2

Steiner Tree Problem

What is the minimum spanning tree of a subset of the vertices?

Bad News: NP-Hard

No efficient (polynomial) time algorithm (unless P = NP).



Steiner Tree Problem

What is the minimum spanning tree of a subset of the vertices?

Good News: Efficient approximation algorithms

Algorithm SteinerMST guarantees:

- OPT(G) = minimum cost Steiner Tree
- T = output of SteinerMST
- T < 2*OPT(G)

Steiner Tree Problem

Algorithm SteinerMST guarantees:

- OPT(G) = minimum cost Steiner Tree
- T = output of SteinerMST
- T < 2*OPT(G)

Example:

- Optimal Steiner Tree has cost 50.
- Our algorithm always outputs a solution with cost < 100.

Steiner Tree Problem

Algorithm SteinerMST:

1. For every pair of required vertices (v,w), calculate the shortest path from (v to w).
   - Use Dijkstra V times.
   - Or wait until we cover All-Pairs-Shortest-Paths next time.

Steiner Tree Problem

Example: Step 1

Shortest Paths:

(A,H) = 2

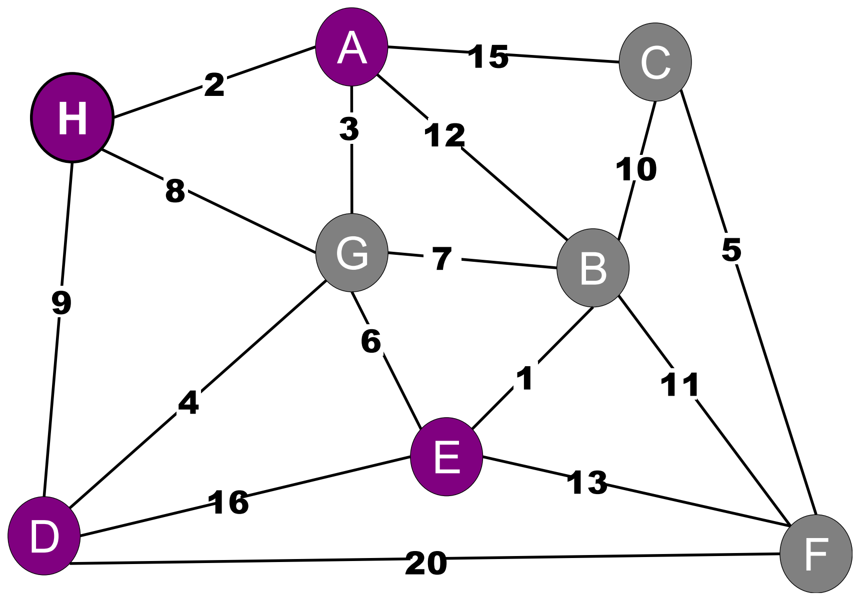
(A,D) = 7

(A,E) = 9

(H,D) = 9

(H,E) = 11

(D,E) = 10



Steiner Tree Problem

Algorithm SteinerMST:

1. For every required vertex (v,w), calculate the shortest path from (v to w).
2. Construct new graph on required nodes.
   - V = required nodes
   - E = shortest path distances

Steiner Tree Problem

Example: Step 2

Shortest Paths:

(A,H) = 2

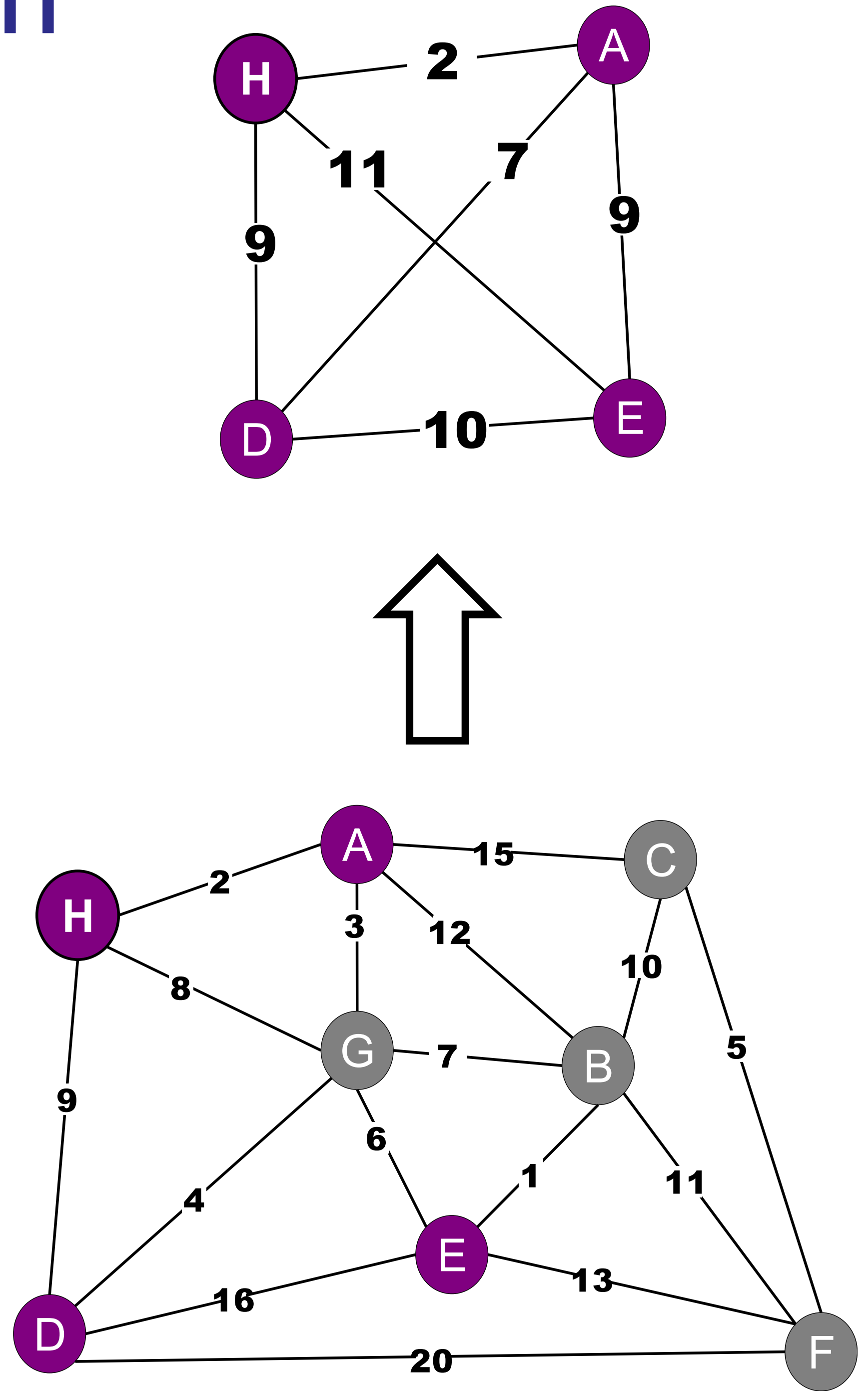
(A,D) = 7

(A,E) = 9

(H,D) = 9

(H,E) = 11

(D,E) = 10



Steiner Tree Problem

Algorithm SteinerMST:

1. For every required vertex (v,w), calculate the shortest path from (v to w).
2. Construct new graph on required nodes.
3. Run MST on new graph.
   - Use Prim’s or Kruskal’s or Boruvka’s
   - MST gives edges on new graph

Steiner Tree Problem

Example: Step 3

Shortest Paths:

(A,H) = 2

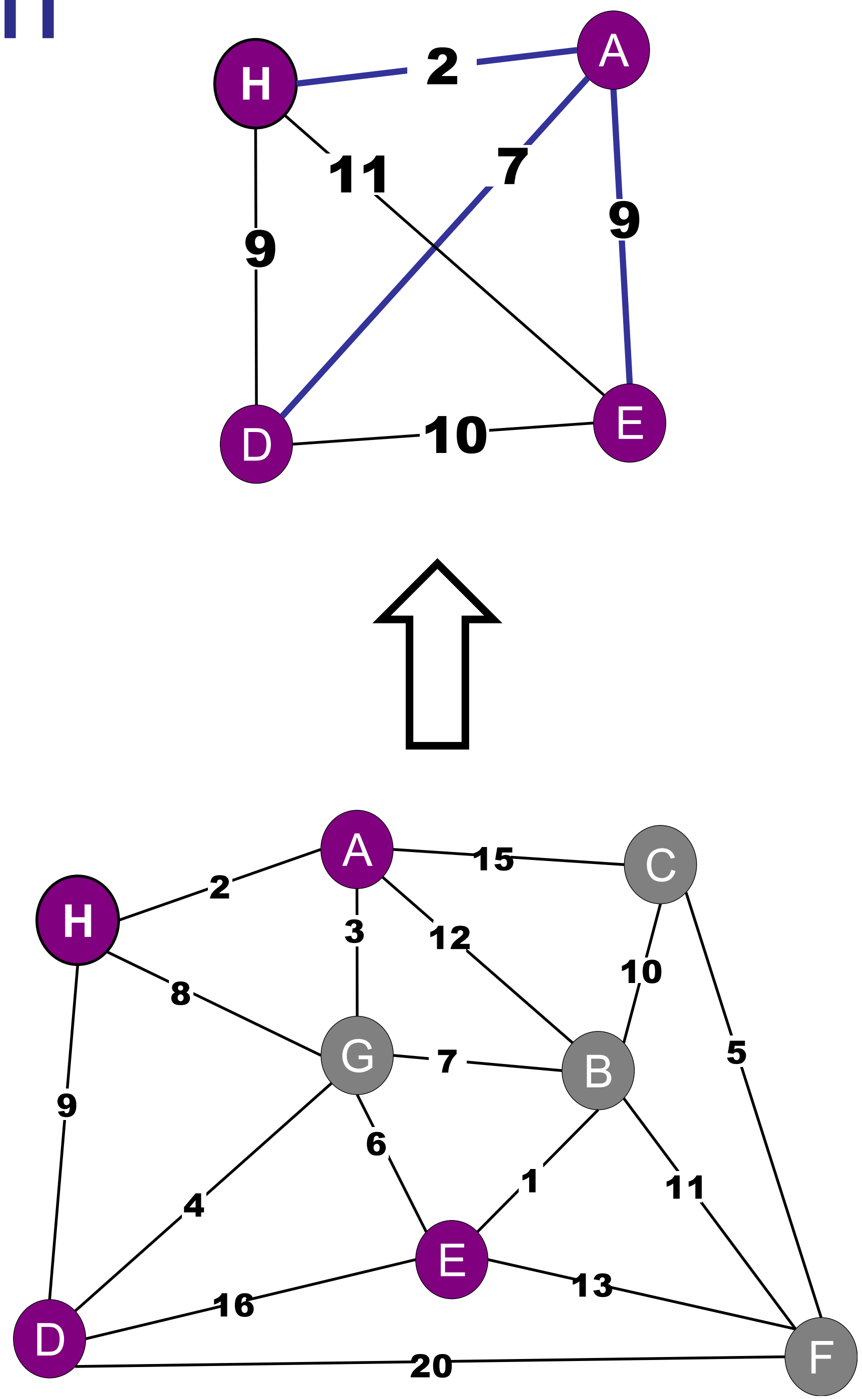
(A,D) = 7

(A,E) = 9

(H,D) = 9

(H,E) = 11

(D,E) = 10



Steiner Tree Problem

Algorithm SteinerMST:

1. For every required vertex (v,w), calculate the shortest path from (v to w).
2. Construct new graph on required nodes.
3. Run MST on new graph.
4. Map new edges back to original graph.
   - Use shortest path discovered in Step 1.
   - Add these edges to Steiner MST.
   - Remove duplicates.

Steiner Tree Problem

Example: Step 4

Shortest Paths:

(A,H) = 2

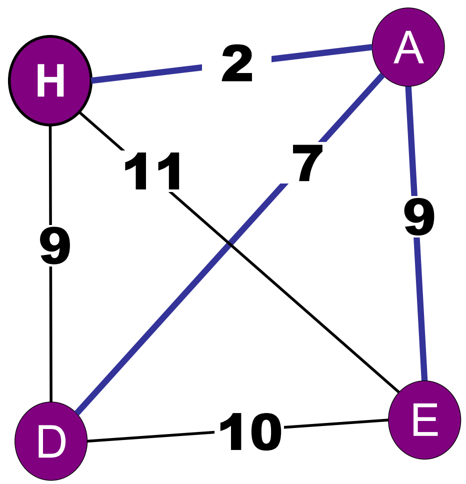
(A,D) = 7

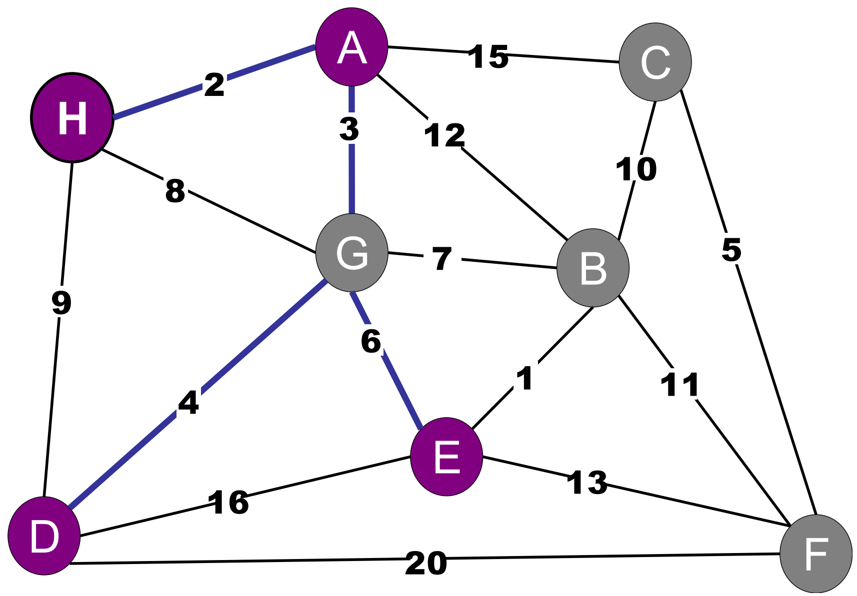
(A,E) = 9

(H,D) = 9

(H,E) = 11

(D,E) = 10





Steiner Tree Problem

Algorithm SteinerMST:

1. For every required vertex (v,w), calculate the shortest path from (v to w).
2. Construct new graph on required nodes.
3. Run MST on new graph.
4. Map new edges back to original graph.

Note: Does NOT guarantee optimal Steiner tree.

Steiner Tree Problem

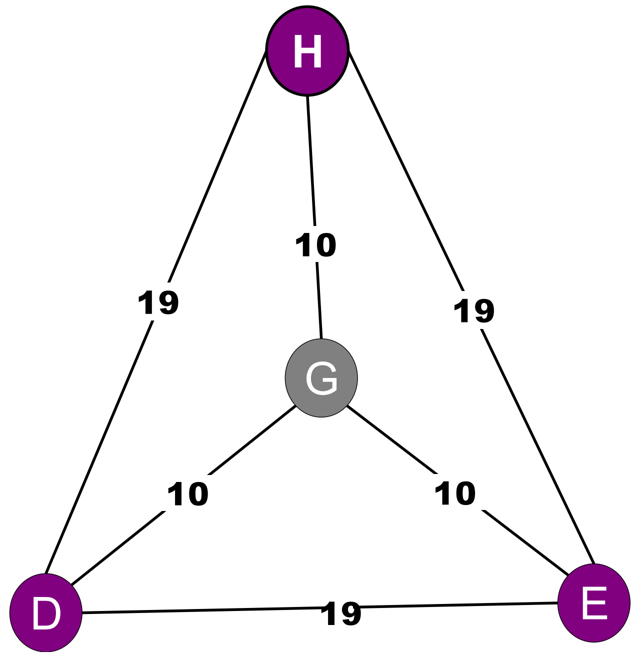
Example:

Shortest Paths:

(D,H) = 19

(D,E) = 19

(E,H) = 19



Steiner Tree Problem

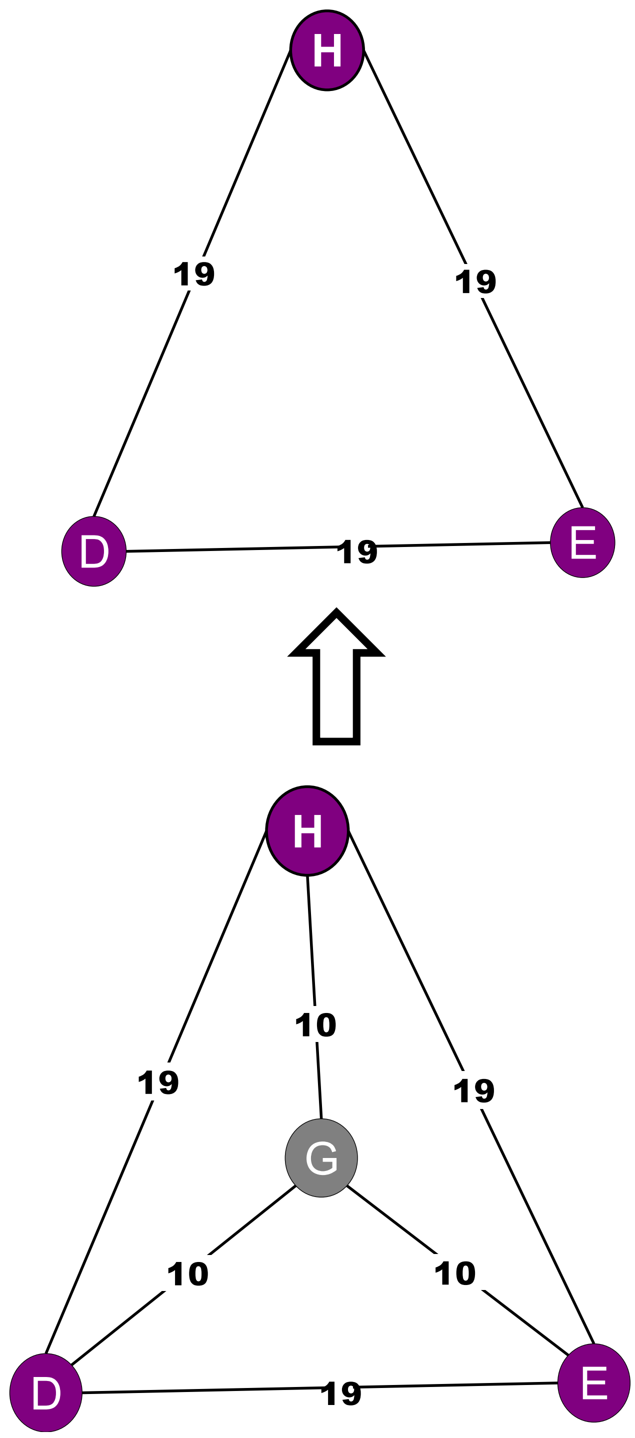
Example:

Shortest Paths:

(D,H) = 19

(D,E) = 19

(E,H) = 19



Steiner Tree Problem

Example:

Shortest Paths:

(D,H) = 19

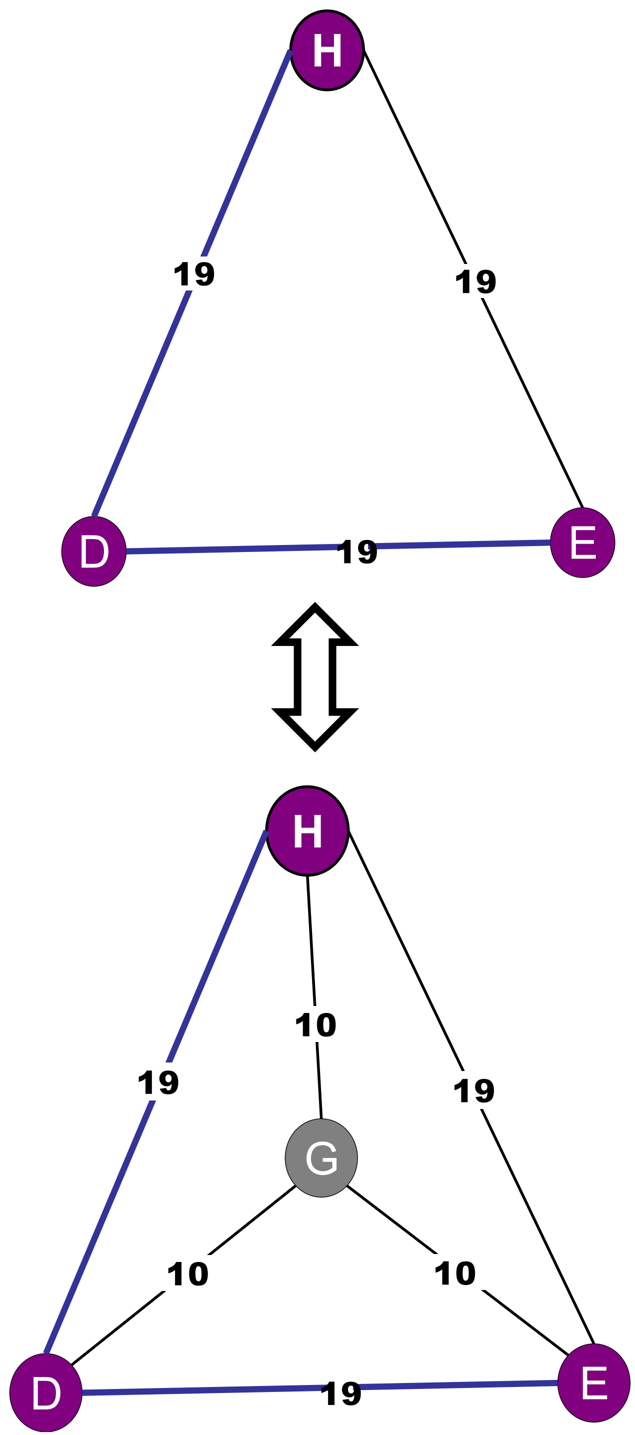
(D,E) = 19

(E,H) = 19

Cost = 38:

OPT Steiner = 30

Challenge: bigger gap!



Steiner Tree Problem

Algorithm SteinerMST:

1. For every required vertex (v,w), calculate the shortest path from (v to w).
2. Construct new graph on required nodes.
3. Run MST on new graph.
4. Map new edges back to original graph.

Note: Does NOT guarantee optimal Steiner tree.

Steiner Tree Problem

Algorithm SteinerMST:

1. Let O be OPT tree.

   Let T be SteinerMST tree.

Steiner Tree Problem

Algorithm SteinerMST:

1. Let O be OPT (Steiner) tree.

   Let T be SteinerMST tree.

2. Let D = DFS on O.

   cost(D) = 2*OPT.

Traverse each edge exactly twice!

Steiner Tree Problem

Example: Step 3

Shortest Paths:

(A,H) = 2

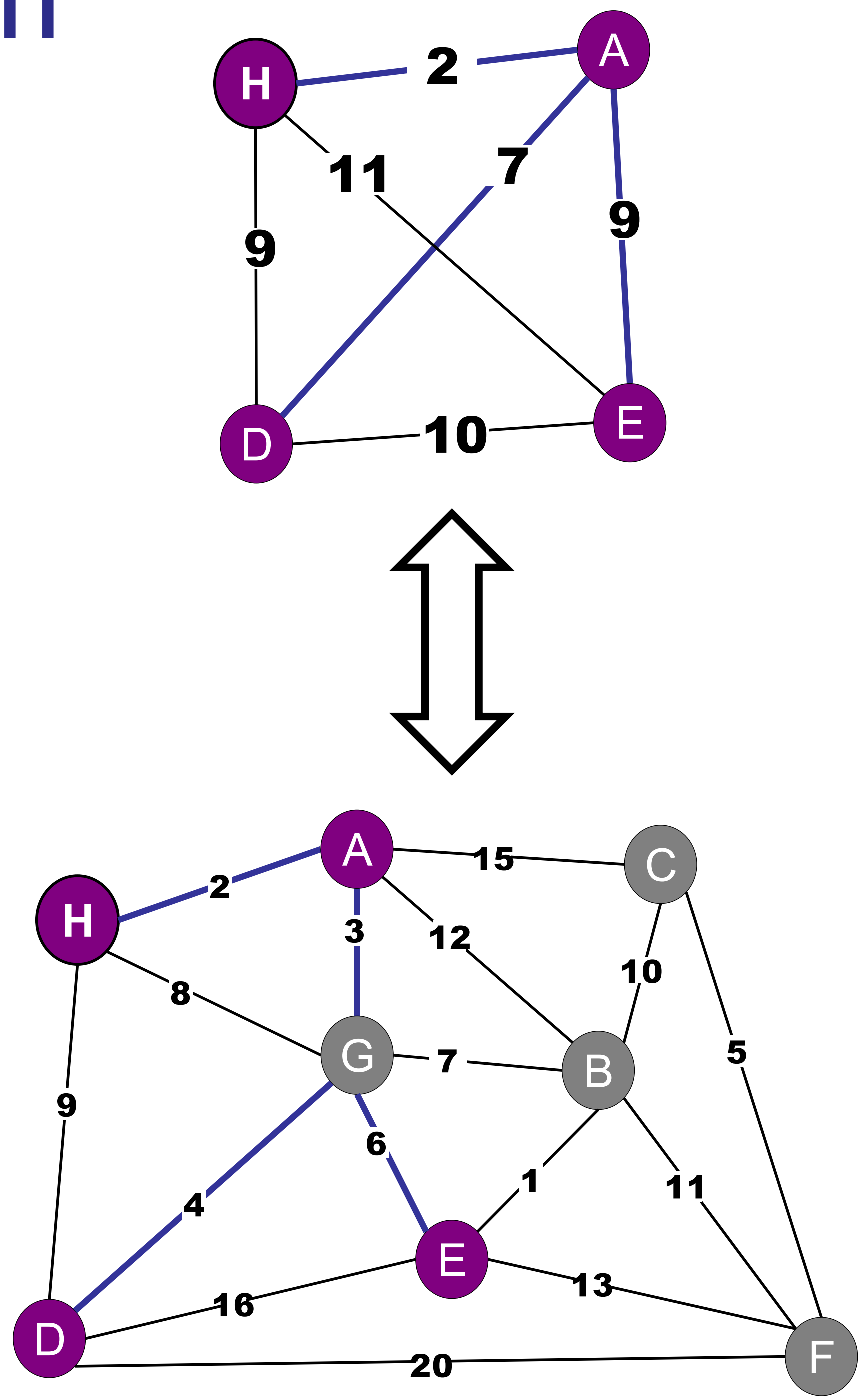
(A,D) = 7

(A,E) = 9

(H,D) = 9

(H,E) = 11

(D,E) = 10



Steiner Tree Problem

Algorithm SteinerMST:

1. Let O be OPT tree.

   Let T be SteinerMST tree.

2. Let D = DFS on O.

   cost(D) = 2*OPT.

3. D = {H, A, G, D, G, E, G, A, H}

Steiner Tree Problem

Algorithm SteinerMST:

1. Let O be OPT tree.

   Let T be SteinerMST tree.

2. Let D = DFS on O.

   cost(D) = 2*OPT.

3. D = {H, A, G, D, G, E, G, A, H}

4. cost(D) = w(H,A) + w(A,G) + ... + w(A,H)

Steiner Tree Problem

Algorithm SteinerMST:

1. Let O be OPT tree.

   Let T be SteinerMST tree.
2. Let D = DFS on O.

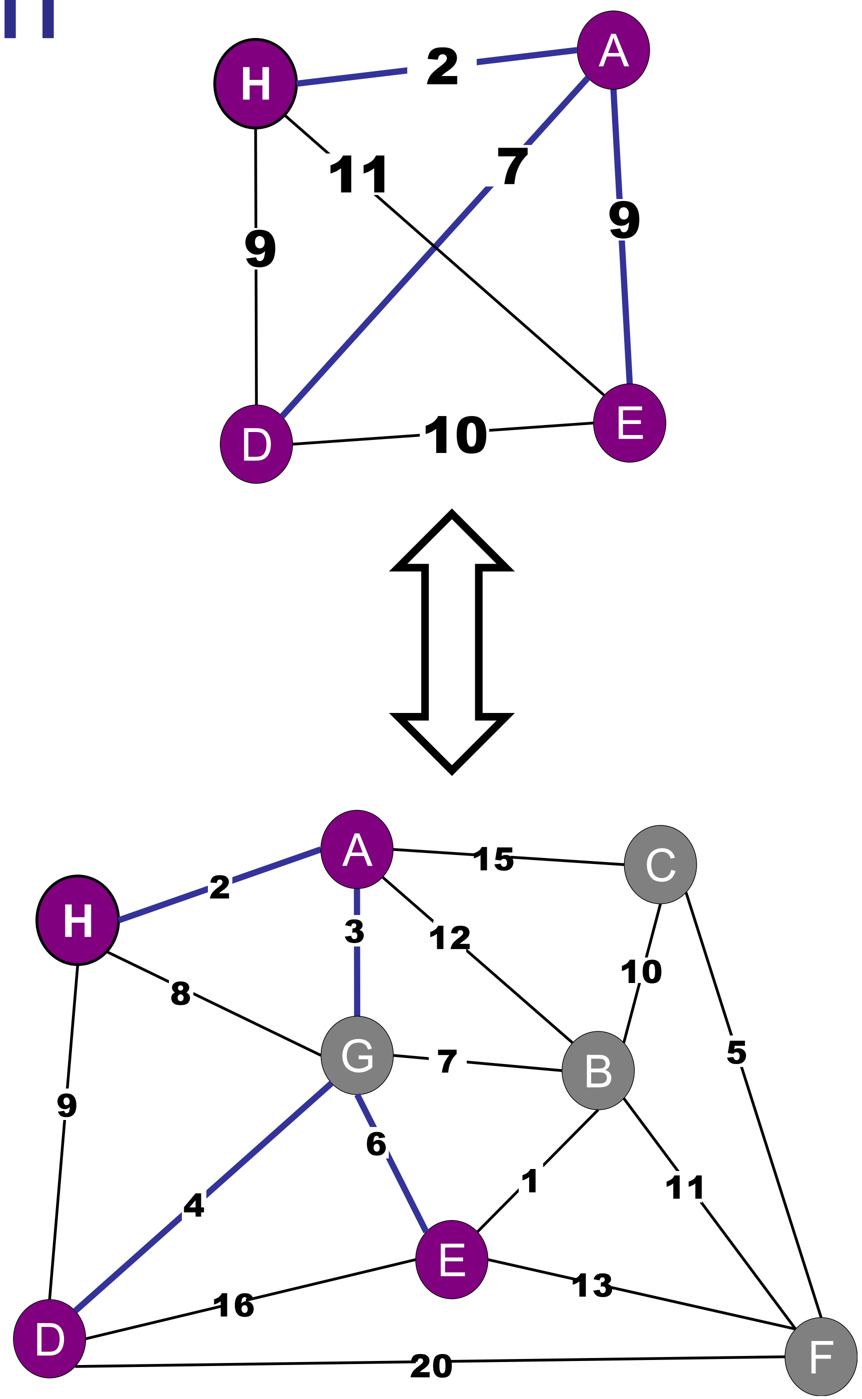
   cost(D) = 2*OPT.
3. D = {H, A, G, D, G, E, G, A, H}
4. cost(D) = w(H,A) + w(A,G) + … + w(A,H)
5. Skip Steiner Nodes: D' = {H, A, D, E, A, H}

Steiner Tree Problem

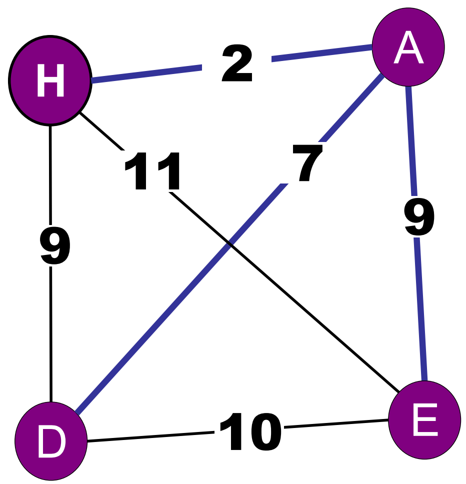
D′ = {H, A, D, E, A, H}

D′ = set of edges on
shortest path graph

D′ = spanning subgraph
of shortest path graph



Steiner Tree Problem



D′ = {H, A, D, E, A, H}

D′ = set of edges on shortest path graph

D′ = spanning subgraph of shortest path graph

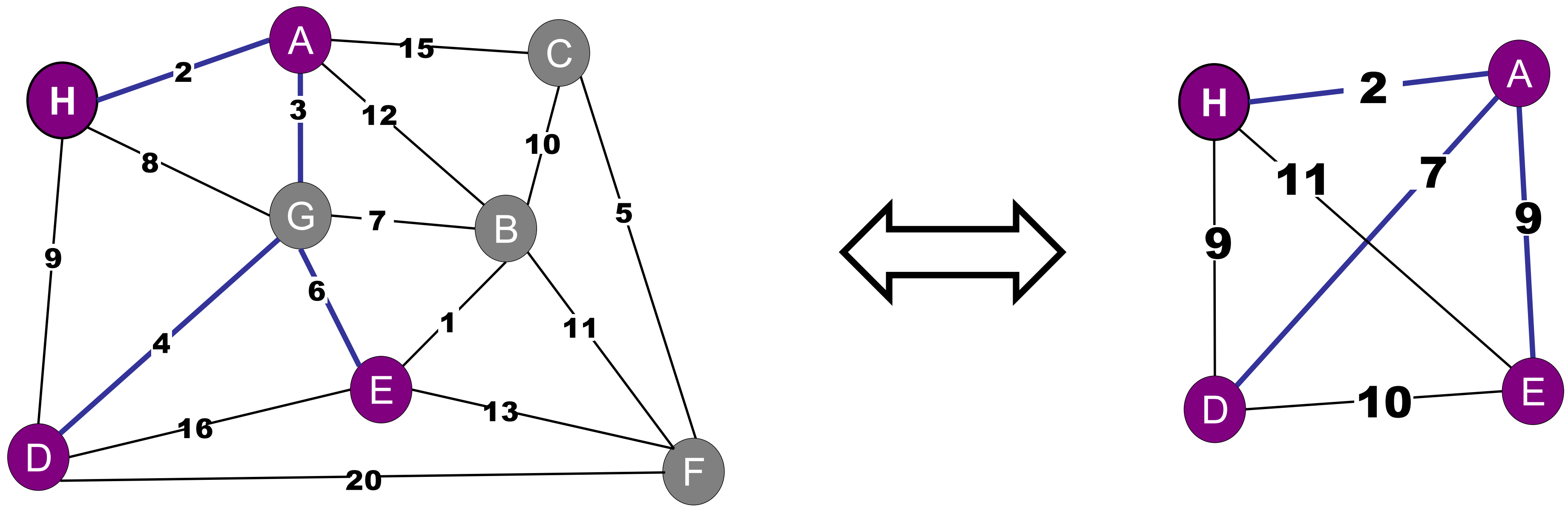
cost(D′) = cost of traversing shortest paths

<= cost(D) <= 2*OPT

Steiner Tree Problem

Algorithm SteinerMST:

6. cost(D’) = cost of traversing shortest paths
   <= cost(D) <= 2*OPT.
7. D’ spans shortest path graph
8. cost(T) = cost(MST) < cost(D’) <= 2*OPT



Steiner Tree Problem

Algorithm SteinerMST:

1. For every required vertex (v,w), calculate the shortest path from (v to w).
2. Construct new graph on required nodes.
3. Run MST on new graph.
4. Map new edges back to original graph.

Note: Does NOT guarantee optimal Steiner tree.

Best known approximation: 1.55

Roadmap

Last time: Minimum Spanning Trees

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

Today: Variations:

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