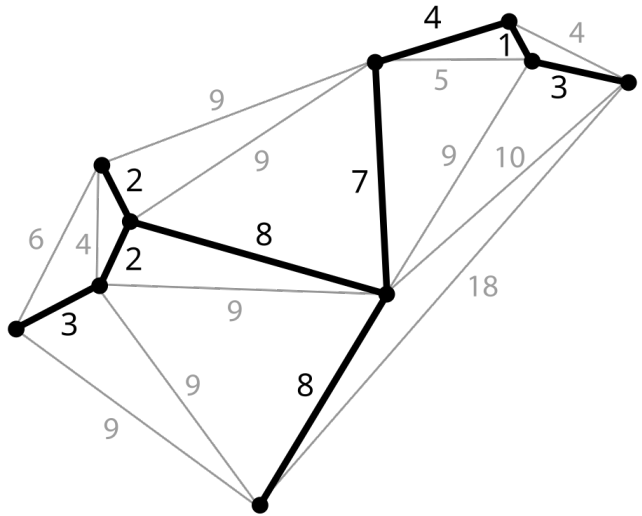
Lecture 21

Minimum Spanning Trees



Announcements

- Office hours are up on course website
  - T: 12pm-4pm; F: 11am-1pm
  - Schedule an appointment individually or as a group

- Project II deliverable 1 feedback out today

Last time

Finished up Divide and Conquer algorithms

- Looked at nearest neighbors problem
- Lab: two other structural deviations

Today

Revisit graphs

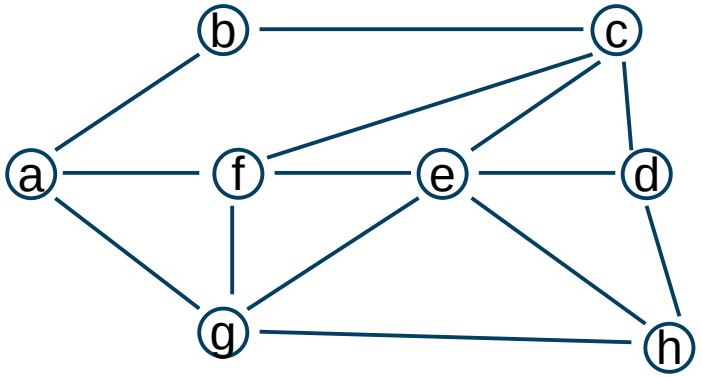
Algorithms for finding MSTs

- Prim's
- Kruskal's

Cuts and Cycles

Recall some graph properties:

- `abcdef` is a **simple path** from a to `f`
- `abcdefa` is a **cycle**



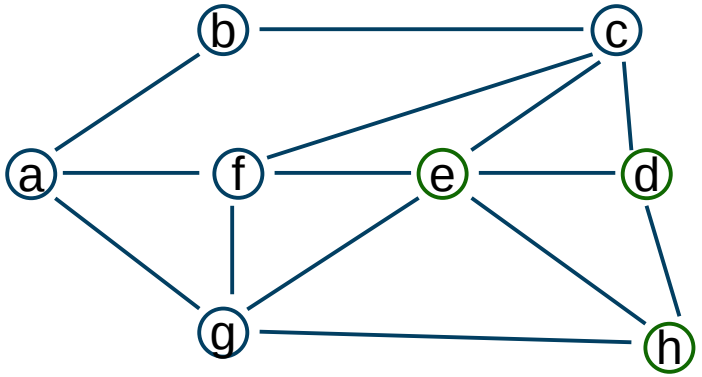
Cuts and Cycles

Recall some graph properties:

- `abcdef` is a **simple path** from a to `f`
- `abcdefa` is a **cycle**

Here are some new properties:

- A **cut** is a partition of the vertices into two non-empty subsets `S` and `V\S`



`S = {e, d, h}`

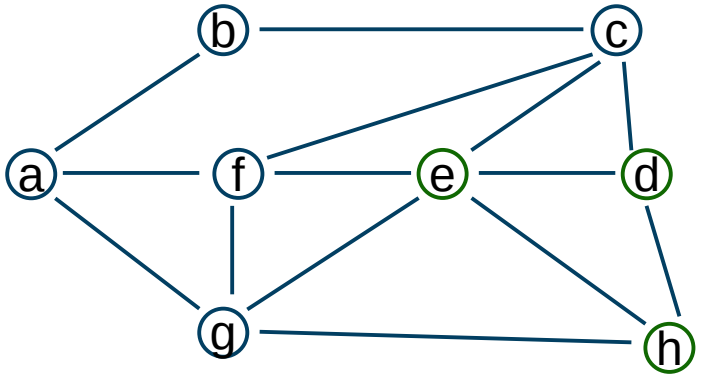
Cuts and Cycles

Recall some graph properties:

- `abcdef` is a **simple path** from a to `f`
- `abcdefa` is a **cycle**

Here are some new properties:

- A **cut** is a partition of the vertices into two non-empty subsets `S` and `V\S`
- A **cutset** for a cut `S` is the set of edges with exactly one end in `S`



`S = {e, d, h}`

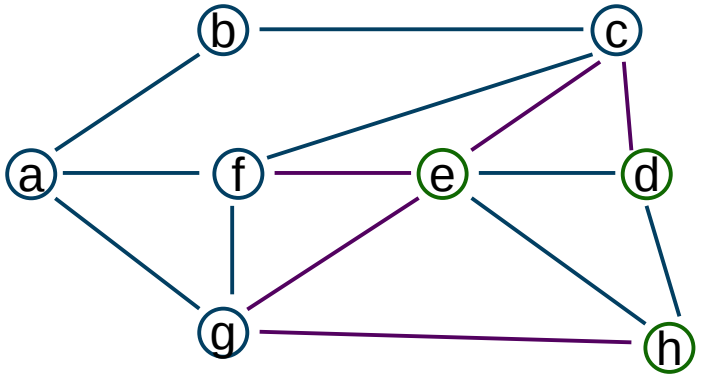
Cuts and Cycles

Recall some graph properties:

- `abcdef` is a **simple path** from a to `f`
- `abcdefa` is a **cycle**

Here are some new properties:

- A **cut** is a partition of the vertices into two non-empty subsets `S` and `V\S`
- A **cutset** for a cut `S` is the set of edges with exactly one end in `S`
- Deleting a cutset `D` of edges disconnects the subsets of a cut `S`



`S = {e, d, h}`

`D = {eg, ef, ce, cd, gh}`

Exercise I

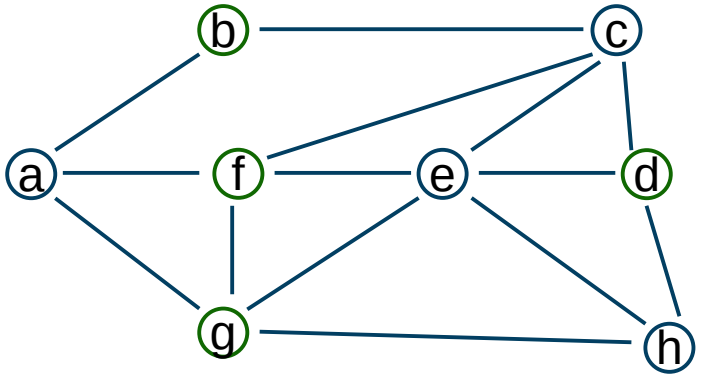
Consider the cut `S = {b,f,g,d}`.
Which edge is in the cutset of S?

a) `(b,f)`

b) `(g,f)`

c) `(f,e)`

d) `(c,e)`



Exercise I

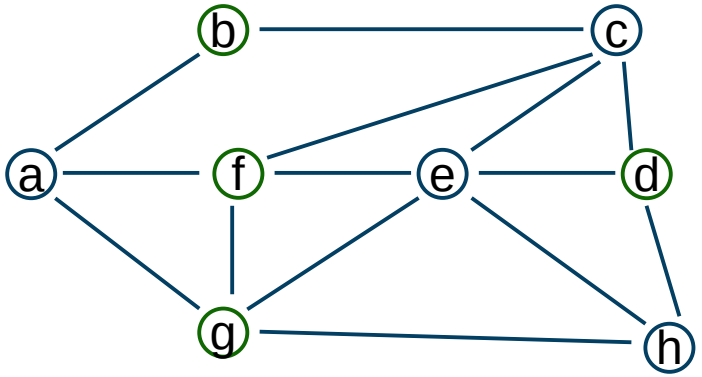
Consider the cut `S = {b,f,g,d}`.
Which edge is in the cutset of S?

a) `(b,f)`

b) `(g,f)`

c) `(f,e)`

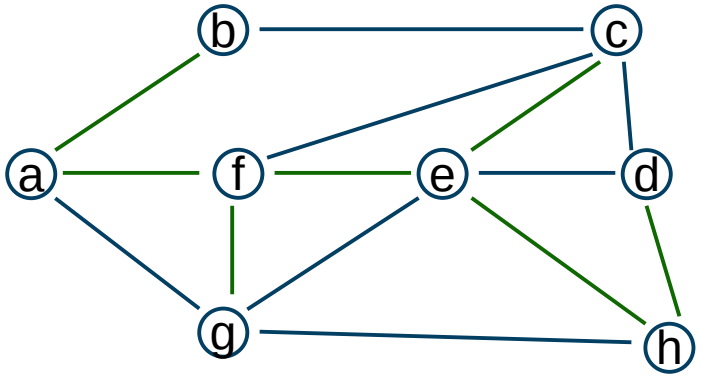
d) `(c,e)`



Spanning Trees

Let `H = (V,T)` be a subgraph of an undirected graph `G = (V,E)`

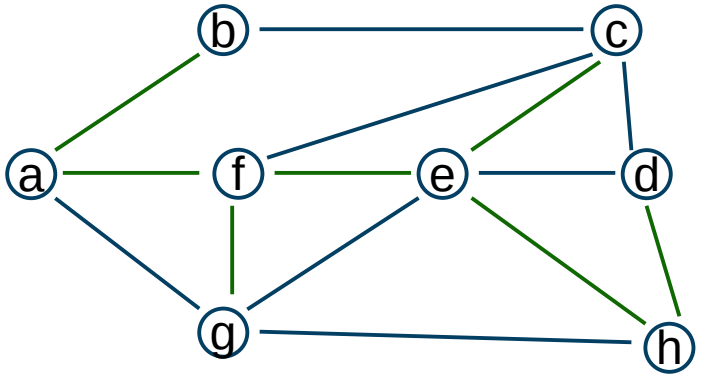
- `H` is a **spanning tree** of `G` if `H` is both acyclic and connected
- A spanning tree is a subgraph on all the vertices that is a tree



Spanning Trees

Let `H = (V,T)` be a subgraph of an undirected graph `G = (V,E)`

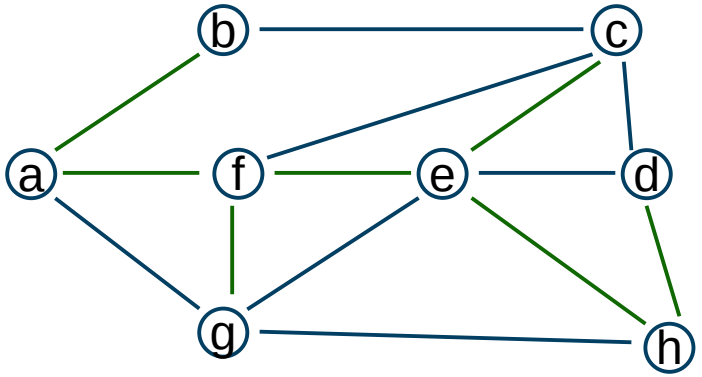
- `H` is a **spanning tree** of `G` if `H` is both acyclic and connected
- A spanning tree is a subgraph on all the vertices that is a tree
- Where have we seen spanning trees before?



Spanning Trees

Let `H = (V,T)` be a subgraph of an undirected graph `G = (V,E)`

- `H` is a **spanning tree** of `G` if `H` is both acyclic and connected
- A spanning tree is a subgraph on all the vertices that is a tree
- Where have we seen spanning trees before?
  - BFS and DFS trees



Exercise II

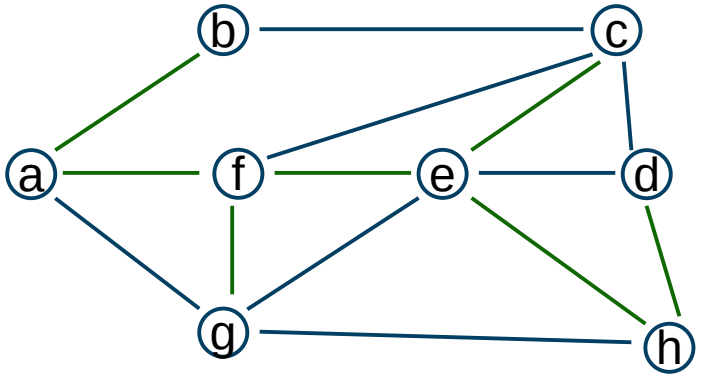
Which of the following properties is true for all spanning trees H?

a) H contains exactly |V| - 1 edges

b) Removing any edge from H disconnects it

c) Adding any edge to H creates a cycle

d) All of the above



Exercise II

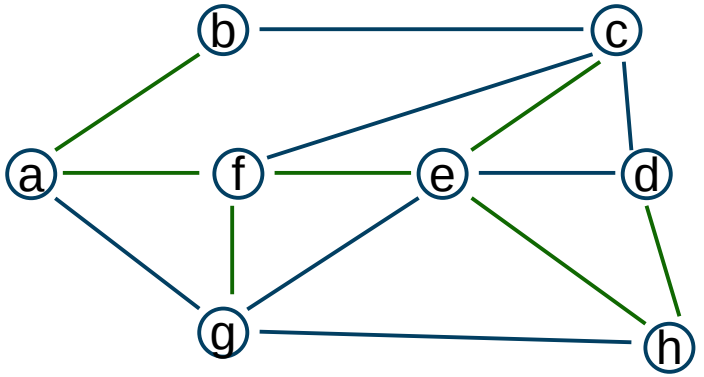
Which of the following properties is true for all spanning trees `H`?

a) `H` contains exactly `|V|-1` edges

b) Removing any edge from `H` disconnects it

c) Adding any edge to `H` creates a cycle

d) All of the above

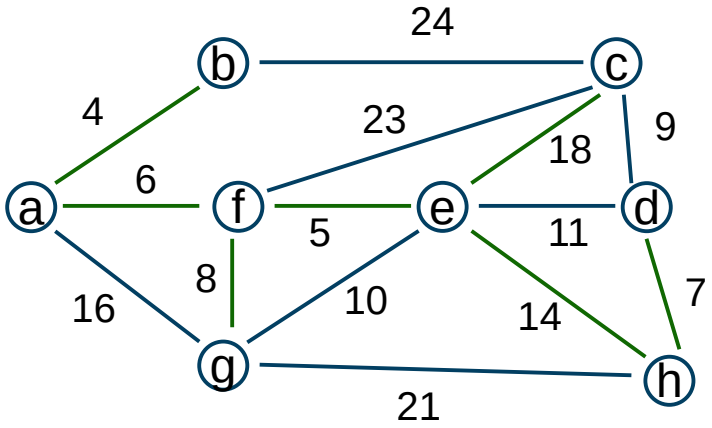


Spanning Trees

Let $G = (V, E)$ be a connected, undirected graph with edge weights c_e

A **minimum spanning tree** (V, T) is a spanning tree of G such that the sum of the edge costs in T is minimized

- An MST is a spanning tree with the lowest edge costs

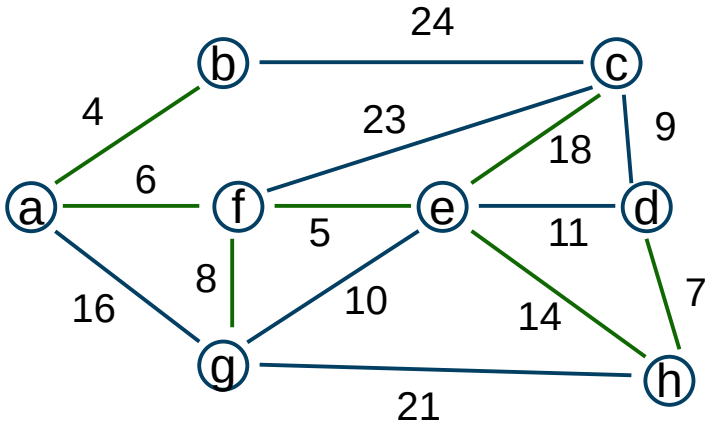


Spanning Trees

Let $G = (V, E)$ be a connected, undirected graph with edge weights c_e

A **minimum spanning tree** (V, T) is a spanning tree of G such that the sum of the edge costs in T is minimized

- An MST is a spanning tree with the lowest edge costs
- Finding an MST is a fundamental problem in CS with diverse applications



Cut Property

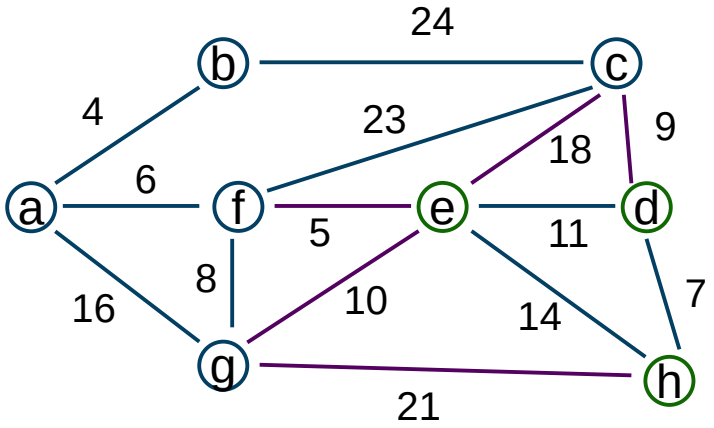
Th: Let `D` be the cutset of a cut `S` in graph `G`. Then, the minimum weight edge `e` in `D` belongs to every MST of `G`.

- `e` is the cheapest or lightest edge in the cut
- It is safe to add edge `e` to an MST

Cut Property

Th: Let `D` be the cutset of a cut `S` in graph `G`. Then, the minimum weight edge `e` in `D` belongs to every MST of `G`.

- `e` is the cheapest or lightest edge in the cut
- It is safe to add edge `e` to an MST



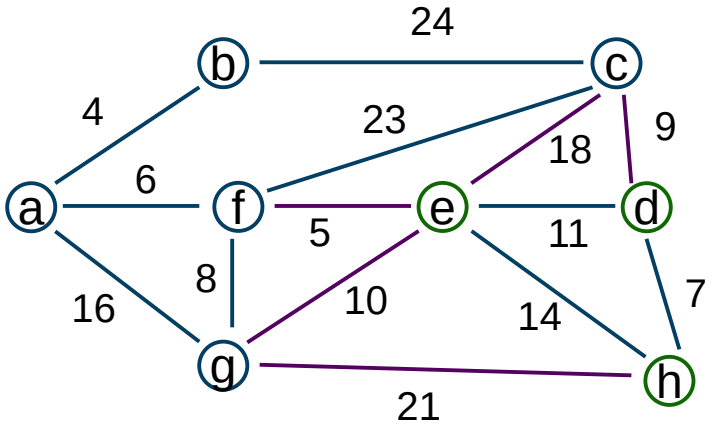
`S = {e, d, h}`

`D = {eg, ef, ce, cd, gh}`

Cut Property

Th: Let `D` be the cutset of a cut `S` in graph `G`. Then, the minimum weight edge `e` in `D` belongs to every MST of `G`.

- `e` is the cheapest or lightest edge in the cut
- It is safe to add edge `e` to an MST
- `ef` belongs to every MST of `G`



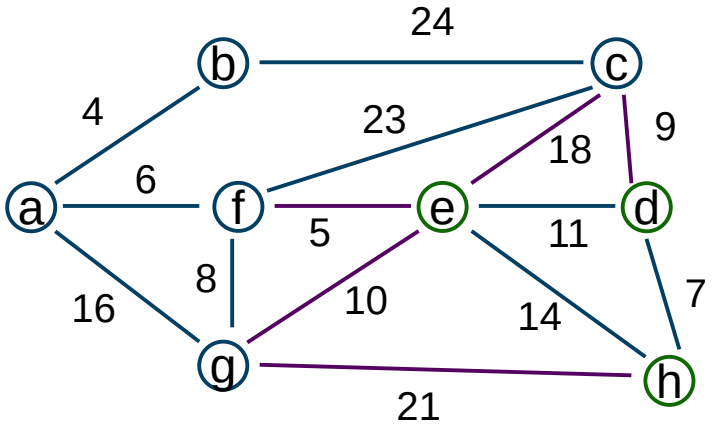
`S = {e, d, h}`

`D = {eg, ef, ce, cd, gh}`

Cut Property

Th: Let `D` be the cutset of a cut `S` in graph `G`. Then, the minimum weight edge `e` in `D` belongs to every MST of `G`.

- `e` is the cheapest or lightest edge in the cut
- It is safe to add edge `e` to an MST
- `ef` belongs to every MST of `G`
- Prove via contradiction by supposing `e` is a the minimum weight edge in the cutset for `S` but `e` is not in an MST for `G`

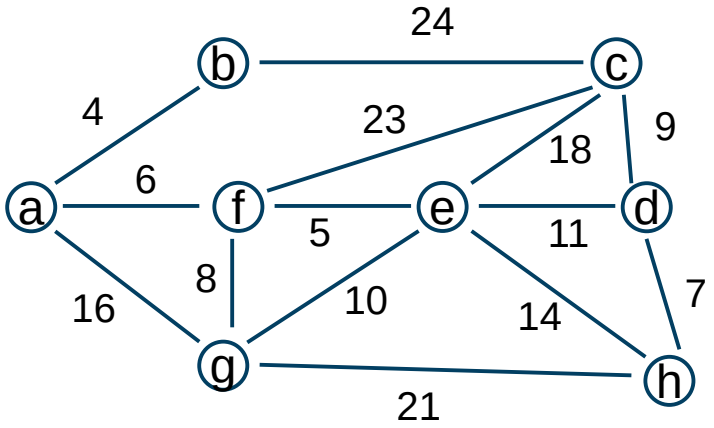


`S = {e, d, h}`

`D = {eg, ef, ce, cd, gh}`

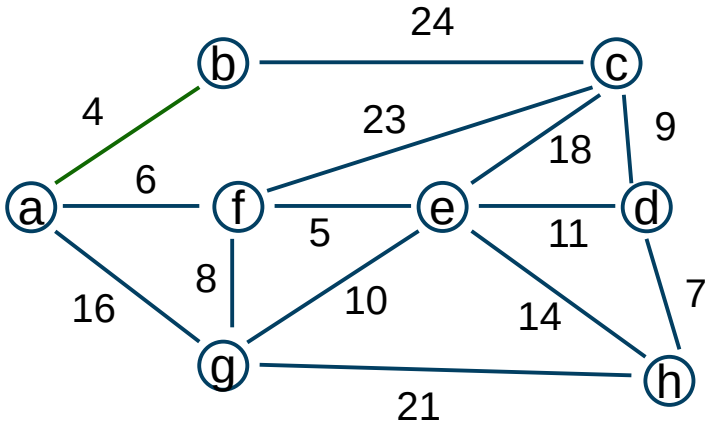
Kruskal's Algorithm

Idea: add edges in order of weight so long as it does not create a cycle



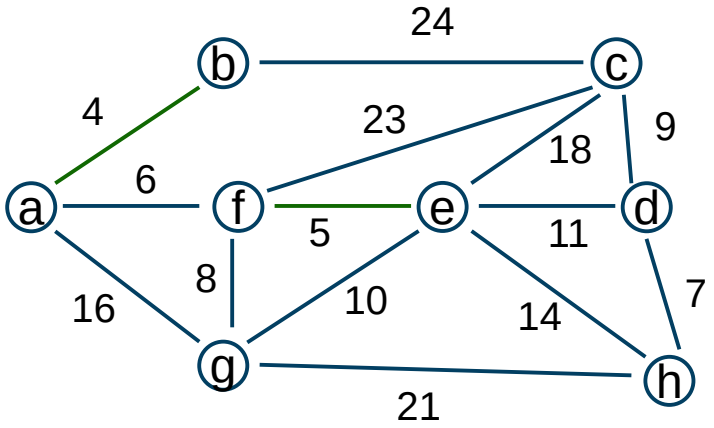
Kruskal's Algorithm

Idea: add edges in order of weight so long as it does not create a cycle



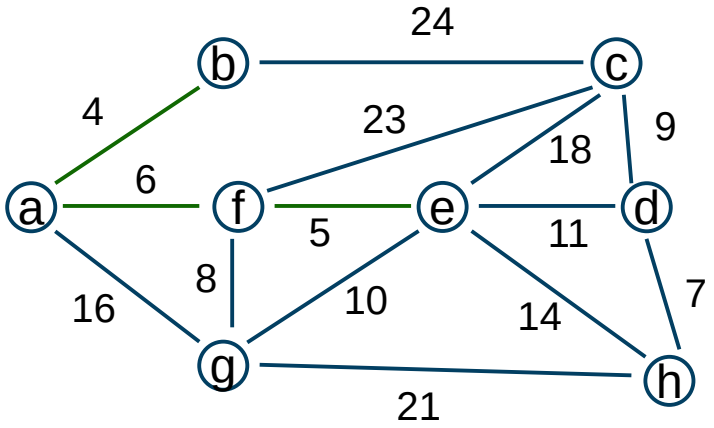
Kruskal's Algorithm

Idea: add edges in order of weight so long as it does not create a cycle



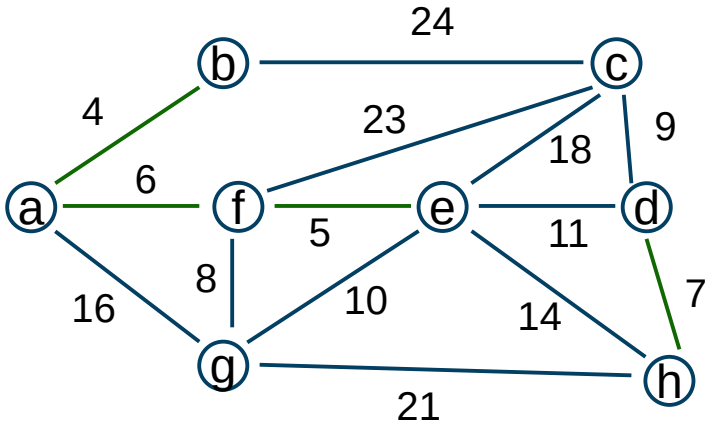
Kruskal's Algorithm

Idea: add edges in order of weight so long as it does not create a cycle



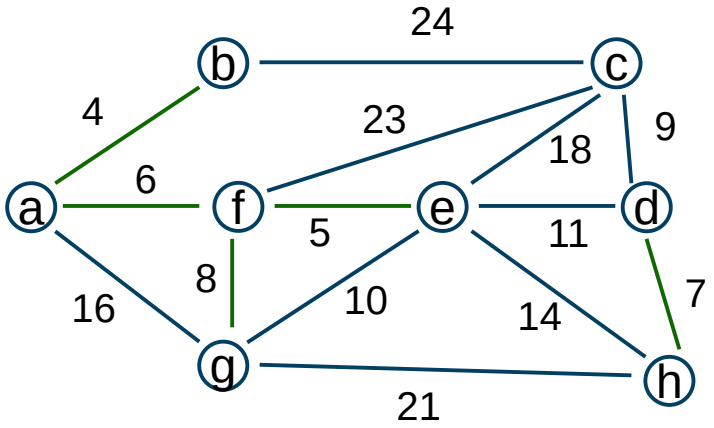
Kruskal's Algorithm

Idea: add edges in order of weight so long as it does not create a cycle



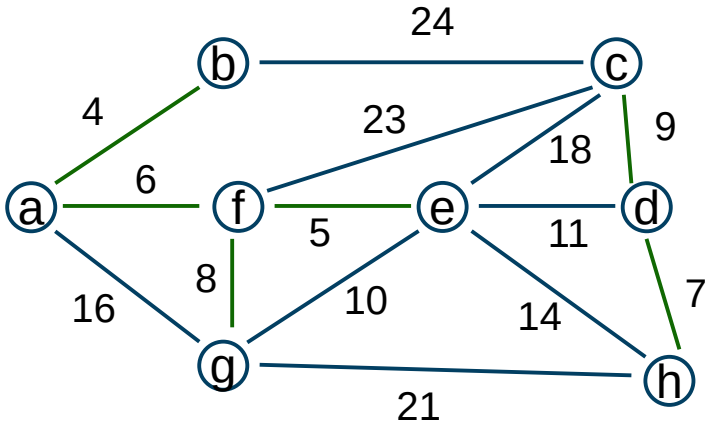
Kruskal's Algorithm

Idea: add edges in order of weight so long as it does not create a cycle



Kruskal's Algorithm

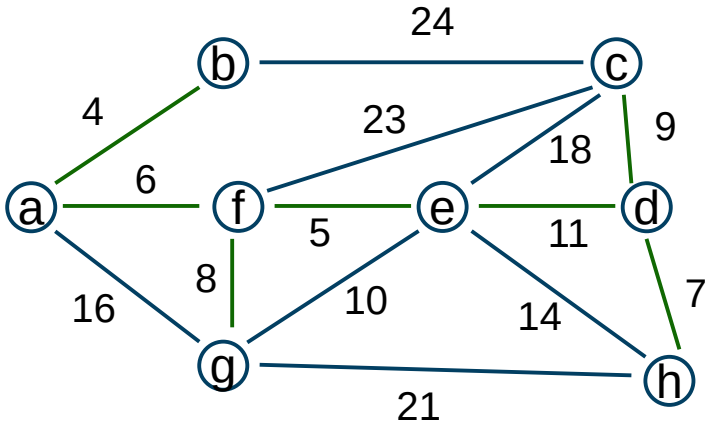
Idea: add edges in order of weight so long as it does not create a cycle



Kruskal's Algorithm

Idea: add edges in order of weight so long as it does not create a cycle

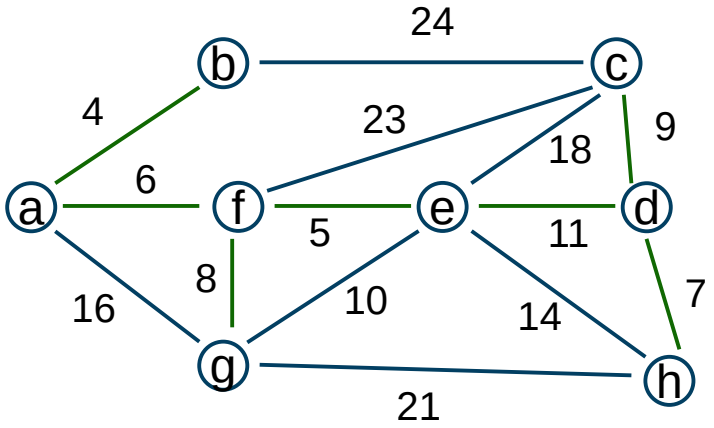
Are we done?



Kruskal's Algorithm

Idea: add edges in order of weight so long as it does not create a cycle

Are we done? Yes, we have an MST!



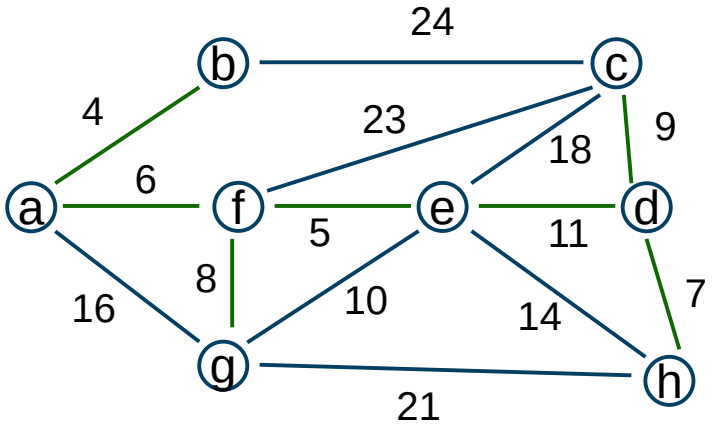
Kruskal's Algorithm

Idea: add edges in order of weight so long as it does not create a cycle

Are we done? Yes, we have an MST!

Running time: `O(m log m)`

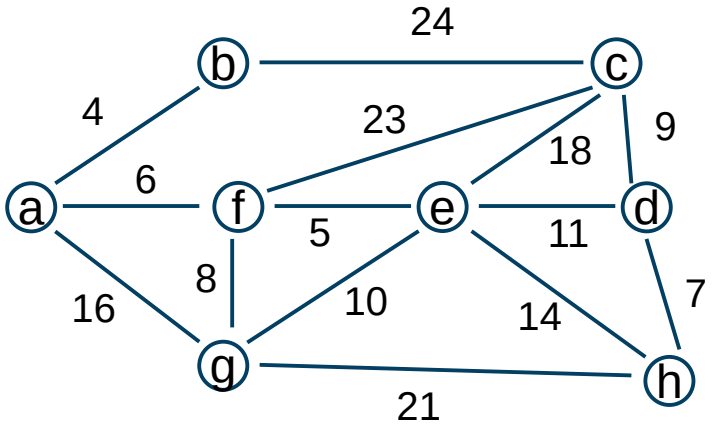
- `m = |E|` is the number of edges



Prim's Algorithm

Idea: grow a tree as a single connected component from a vertex s

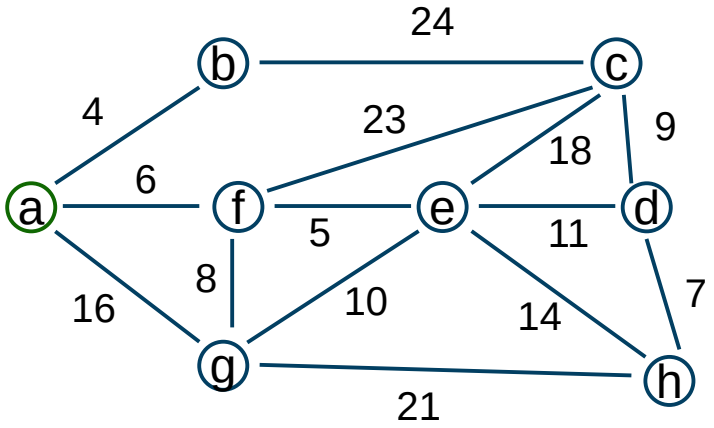
- Let S be the component containing s
- Add the cheapest edge from S to V\S



Prim's Algorithm

Idea: grow a tree as a single connected component from a vertex s

- Let S be the component containing s
- Add the cheapest edge from S to V\S

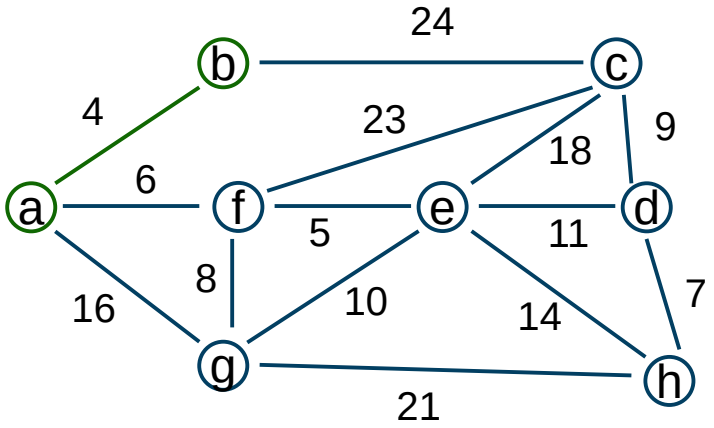


S = {a}

Prim's Algorithm

Idea: grow a tree as a single connected component from a vertex s

- Let S be the component containing s
- Add the cheapest edge from S to V\S

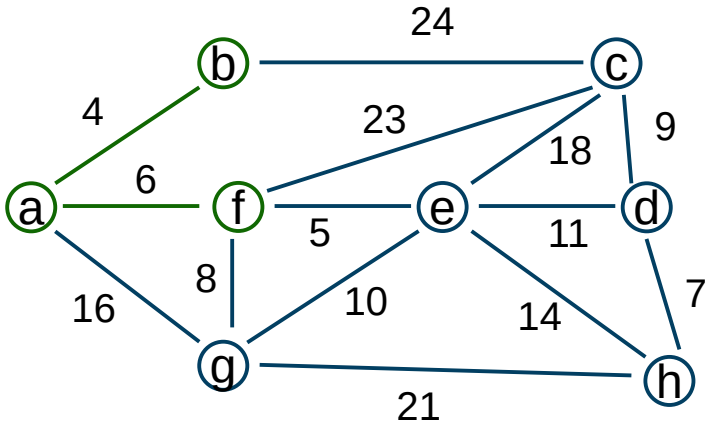


`S = {a,b}`

Prim's Algorithm

Idea: grow a tree as a single connected component from a vertex s

- Let S be the component containing s
- Add the cheapest edge from S to V\S

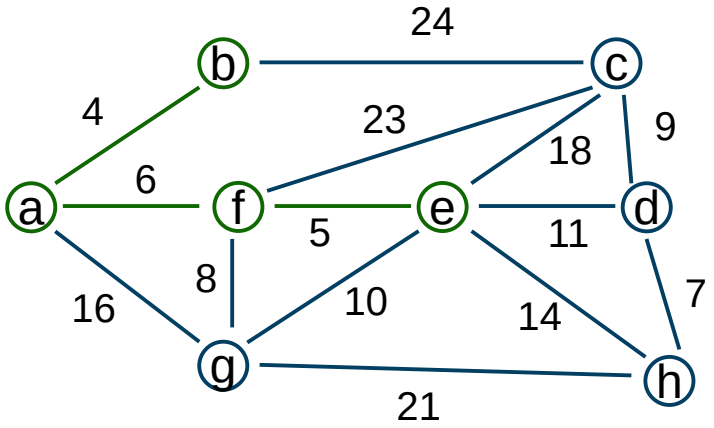


`S = {a,b,f}`

Prim's Algorithm

Idea: grow a tree as a single connected component from a vertex s

- Let S be the component containing s
- Add the cheapest edge from S to V\S

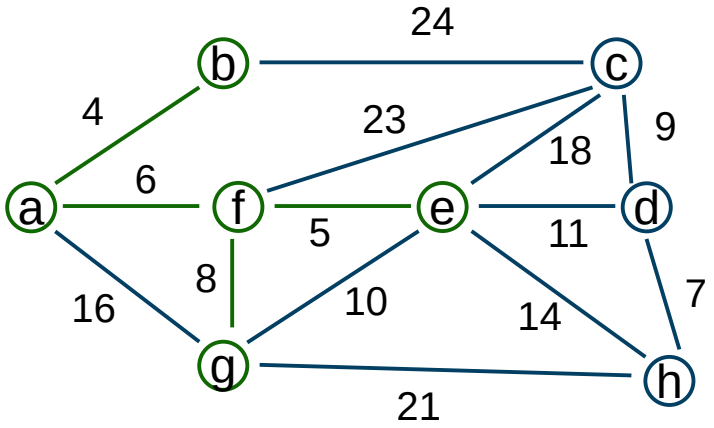


`S = {a,b,f,e}`

Prim's Algorithm

Idea: grow a tree as a single connected component from a vertex s

- Let S be the component containing s
- Add the cheapest edge from S to V\S

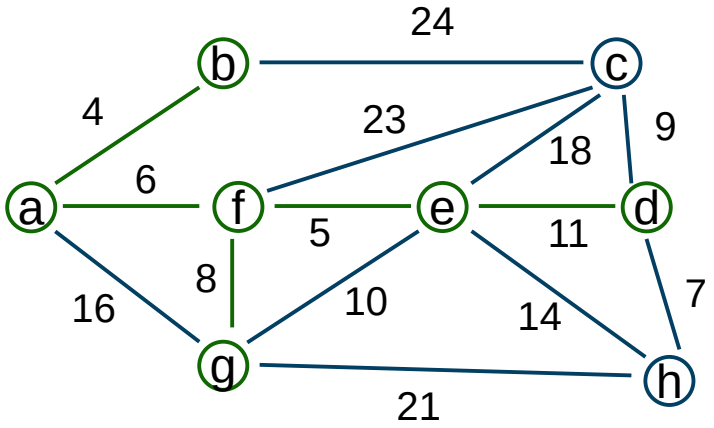


`S = {a,b,f,e,g}`

Prim's Algorithm

Idea: grow a tree as a single connected component from a vertex s

- Let S be the component containing s
- Add the cheapest edge from S to V\S

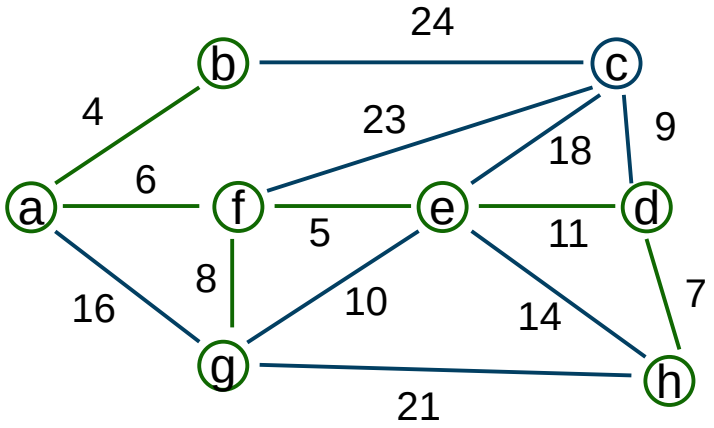


`S = {a,b,f,e,g,d}`

Prim's Algorithm

Idea: grow a tree as a single connected component from a vertex s

- Let S be the component containing s
- Add the cheapest edge from S to V\S

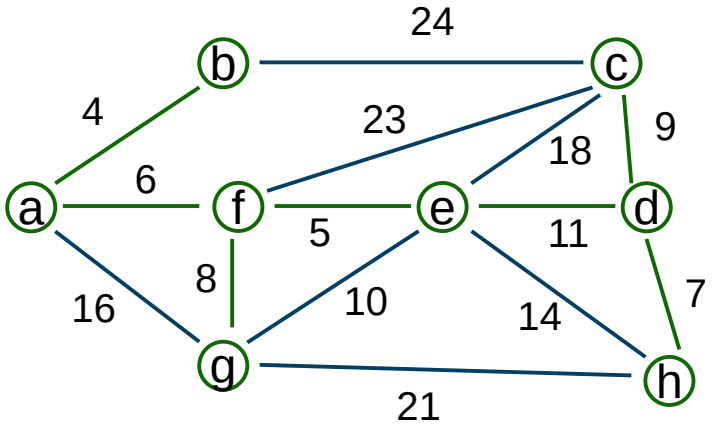


`S = {a,b,f,e,g,d,h}`

Prim's Algorithm

Idea: grow a tree as a single connected component from a vertex s

- Let S be the component containing s
- Add the cheapest edge from S to V\S



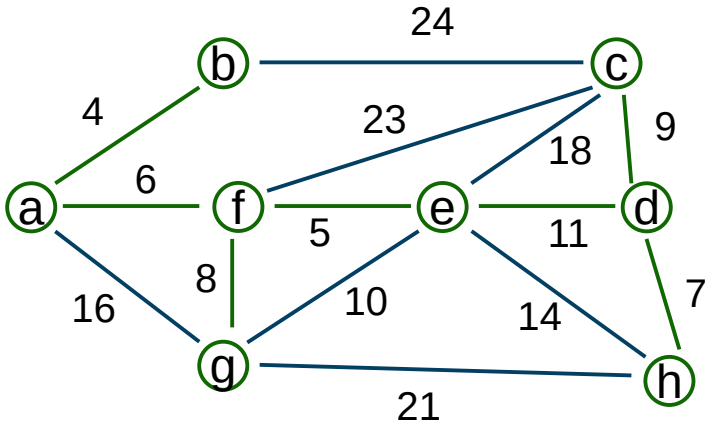
Why does Prim's Algo find an MST?

`S = {a,b,f,e,g,d,h,c}`

Prim's Algorithm

Idea: grow a tree as a single connected component from a vertex s

- Let S be the component containing s
- Add the cheapest edge from S to V\S



Why does Prim's Algo find an MST?

- Applies the cut property at every step

`S = {a,b,f,e,g,d,h,c}`

Next Time

- Heaps