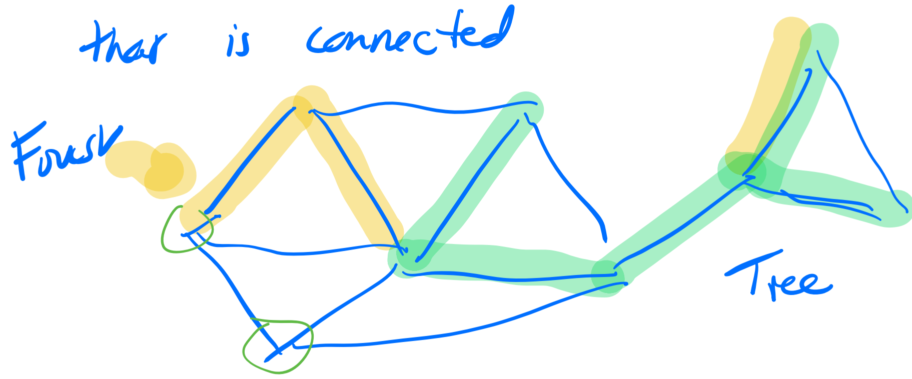
Minimum Spanning Trees

"Greedy algorithm"

Given $G=(V,E)$ undirected

Def: a tree $T \subseteq E$ without a cycle that is connected



A forest $F \subseteq E$ is a subset of edges with a cycle that may or may not be connected
(A forest is a collection of trees)

Def: A spanning tree is a tree that connects all of the vertices.

Claim: A spanning tree:

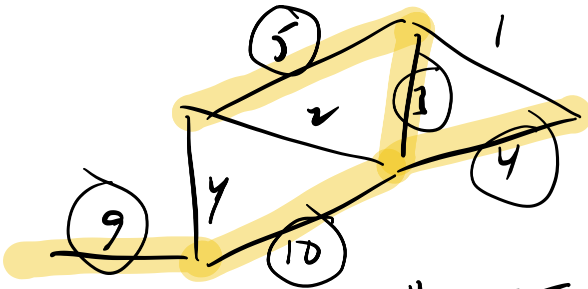
- Is ~~connected~~ acyclic
- Spans (all vertices included)
- has $n-1$ edges

Any 2 of these implies the third!

Think about this

Given weights on edges, the weight of a spanning tree $T = \sum w(e)$

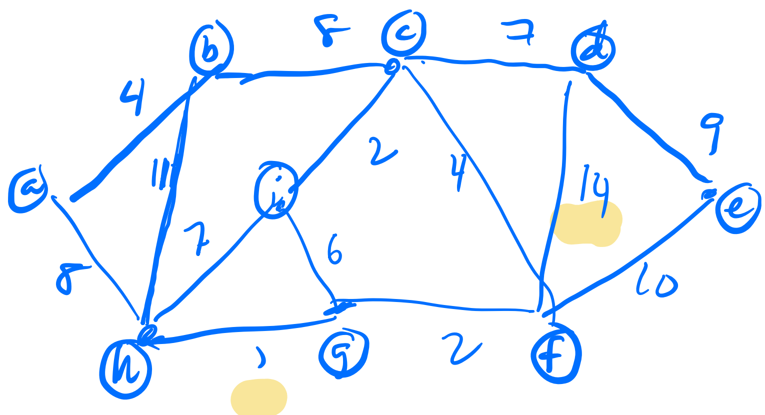
$\sum_{e \in T}$

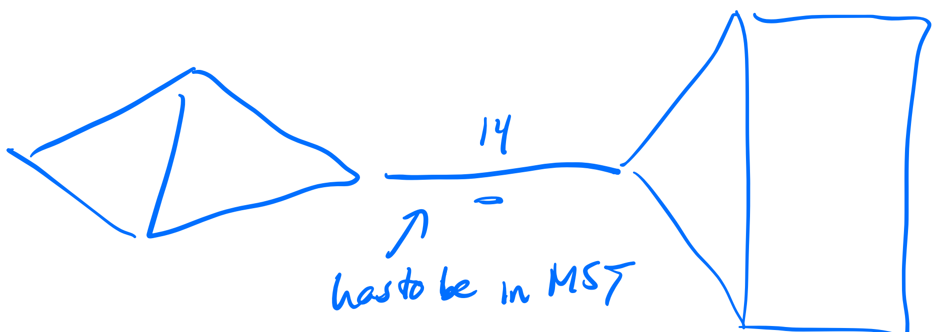


← wt ST = 29

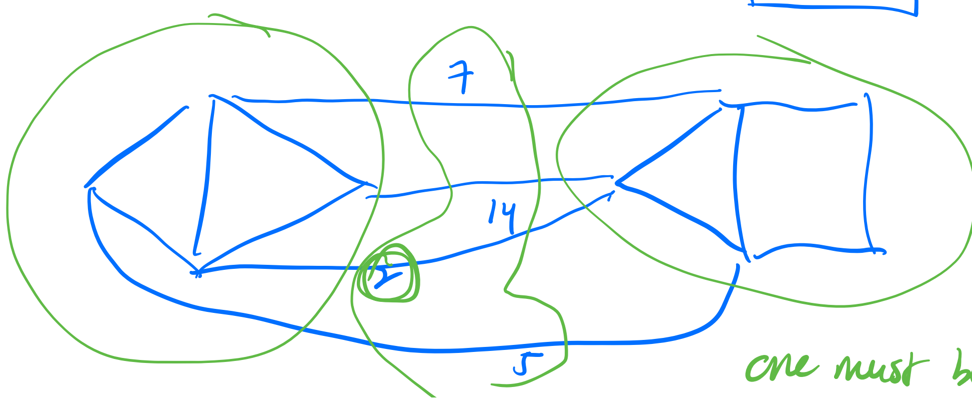
"MST"

Defⁿ A <u>minimum Spanning Tree</u> is a spanning tree with smallest weight.





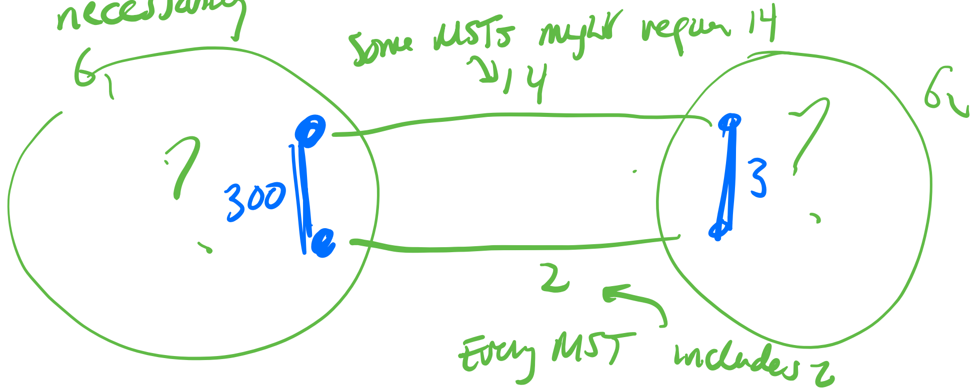
has to be in MST



one must be in MST

Guarantee → Want to include 2

not this necessarily → Hope to not include other.



Build a ST incrementally, 1 edge a time,

Invariant: so that ($X \subseteq E$) X is always part of some MST.

Def: If $X \subseteq E$ is part of some MST, we say an edge $e=(u,v)$ is safe if $X \cup e$ is also part of some MST.

Approach: Start empty

Add safe edges

Stop when you get a ST = MST

$\emptyset$ satisfies invariant

Find $e \in E$ a safe edge

$\Rightarrow e_1$ satisfies invariant

Find e a safe edge

$\Rightarrow \{e_1, e_2\}$ satisfies invariant

⋮

$\{e_1, e_2, \ldots, e_{n-1}\}$ satisfies invariant

$\underbrace{\{e_1, e_2, \ldots, e_{n-1}\}}_{\text{spanning tree}}$, so it is a MST.

Lemma: (cut lemma)

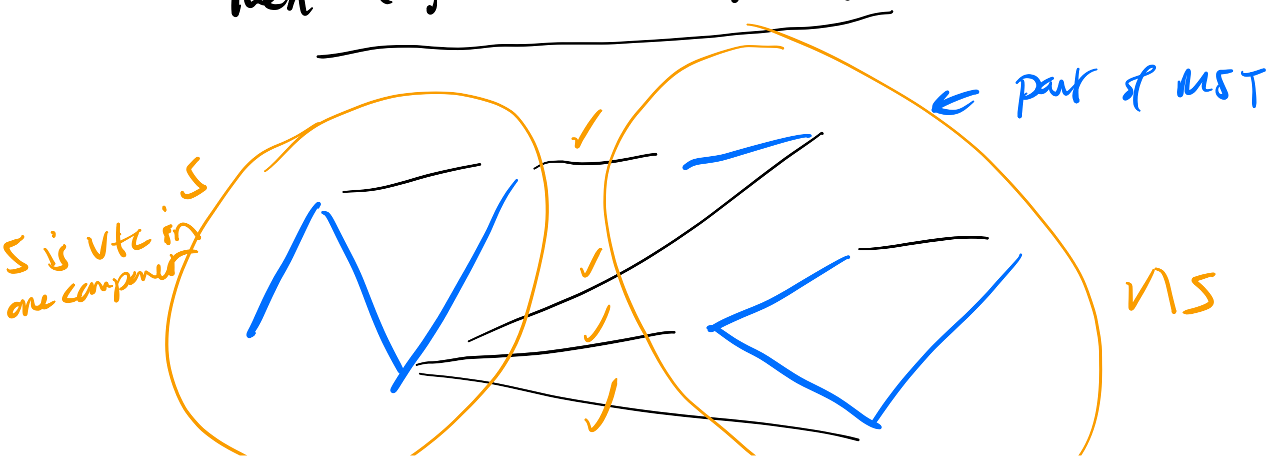
Let $G=(V,E)$ be a connected, undirected graph with weights w on E.

Let $A \subseteq E$ be a set that is part of some MST.

Let $(S, V \setminus S)$ be any cut that "respects A" (no edges of A cross cut from S to $V \setminus S$)

+ let (u,v) be any lightest edge connecting S to $V \setminus S$.

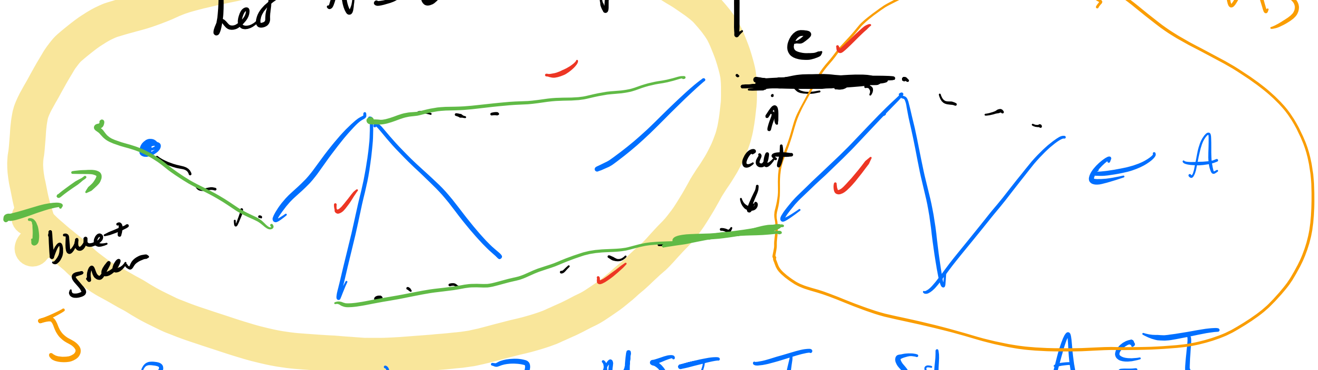
Then (u,v) is a safe edge.



cut

Proof of cut lemma

Let $A \subseteq E$ be part of a MST T



By assumption $\exists$ MST T s.t. $A \subseteq T$

Let S contain 1 or more connected components of A.

Let e be the lightest edge connecting a vertex in S to a vertex in $V \setminus S$

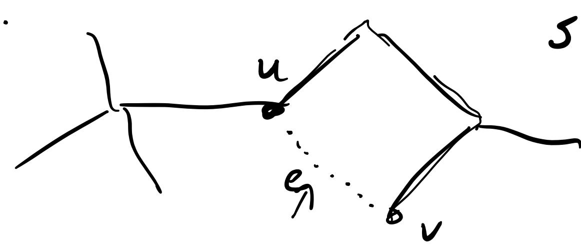
- If $e \in T$ then we're done

  T must include some edge crossing cut

  If $A \subseteq T$ and $e \in T$ then $A \cup \{e\} \subseteq T$ — Invariant is satisfied

- If $e = (u,v) \notin T$, then $T \cup \{e\}$ contains a cycle.

spans



Let e' be the lightest edge in that cycle

Let e be the lightest edge on that cycle

$e' = e$

Claim $T \cup \{e\} \setminus \{e'\}$ is the same wt ~~or lighter~~ $\leftarrow$ T is a MST

We know $w(e) \leq w(e')$ because e is the lightest edge crossing the cut

So $T \cup \{e\} \setminus \{e'\}$ is lighter (not possible) or the same wt.

Kruskal's algorithm (G, w)

IDea: Sort edges by wt.

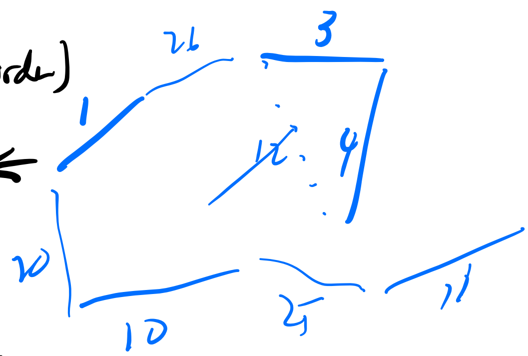
Maintain invariant

add 1 by 1 wo making cycles

Stop when it spans

```
A ← ∅
for each v ∈ V(G)
    Makeset (v)   ←

Sort edges
for each edge (u,v) ∈ E (in order)
do if Find(u) ≠ Find(v)   ←
   then A ← add [(u,v)]
        union (u,v)   ←
return A
```



? How can we tell if we make a cycle?

Running time depends on data structures (Thursday)

Prim's algorith. (G, w, v)

for each $u \in V(G)$

do $key(u) \leftarrow \infty$ (unseen)

$\pi(u) = \emptyset$

$key(v) \leftarrow 0$

while $Q \neq \emptyset$

do $u \leftarrow$ del min (Q) (cheapest)

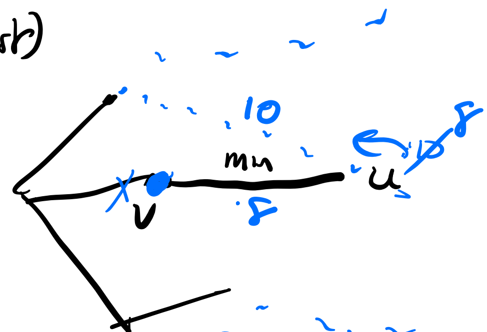
for each $v \in G$ adj to u do:

if $w(u,v) < key(v)$

then $\pi(v) \leftarrow u$

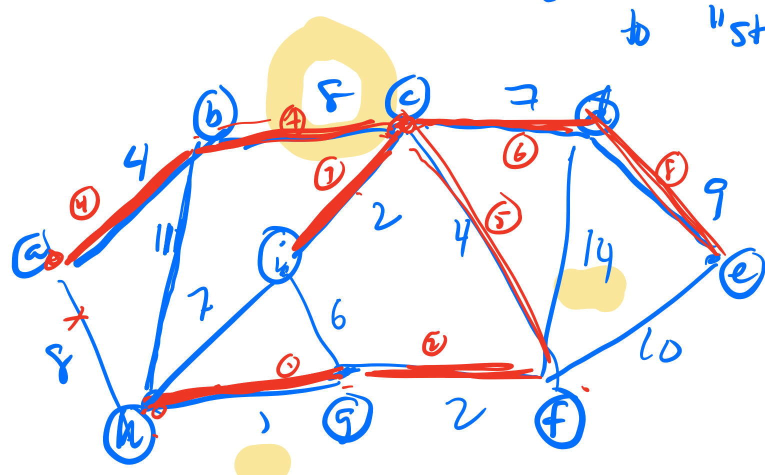
$key(v) \leftarrow w(u,v)$

cheapest edges that keep v connected



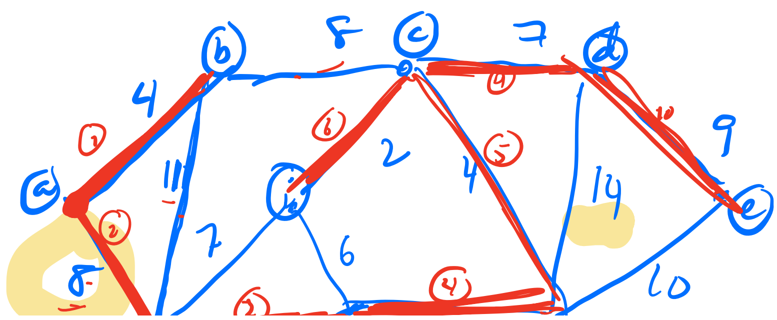
key min dist from vtx to "structure"

Kruskal



MST

Prim



abcgfh
abcgfh i
abcgfhid
abcgfhide.