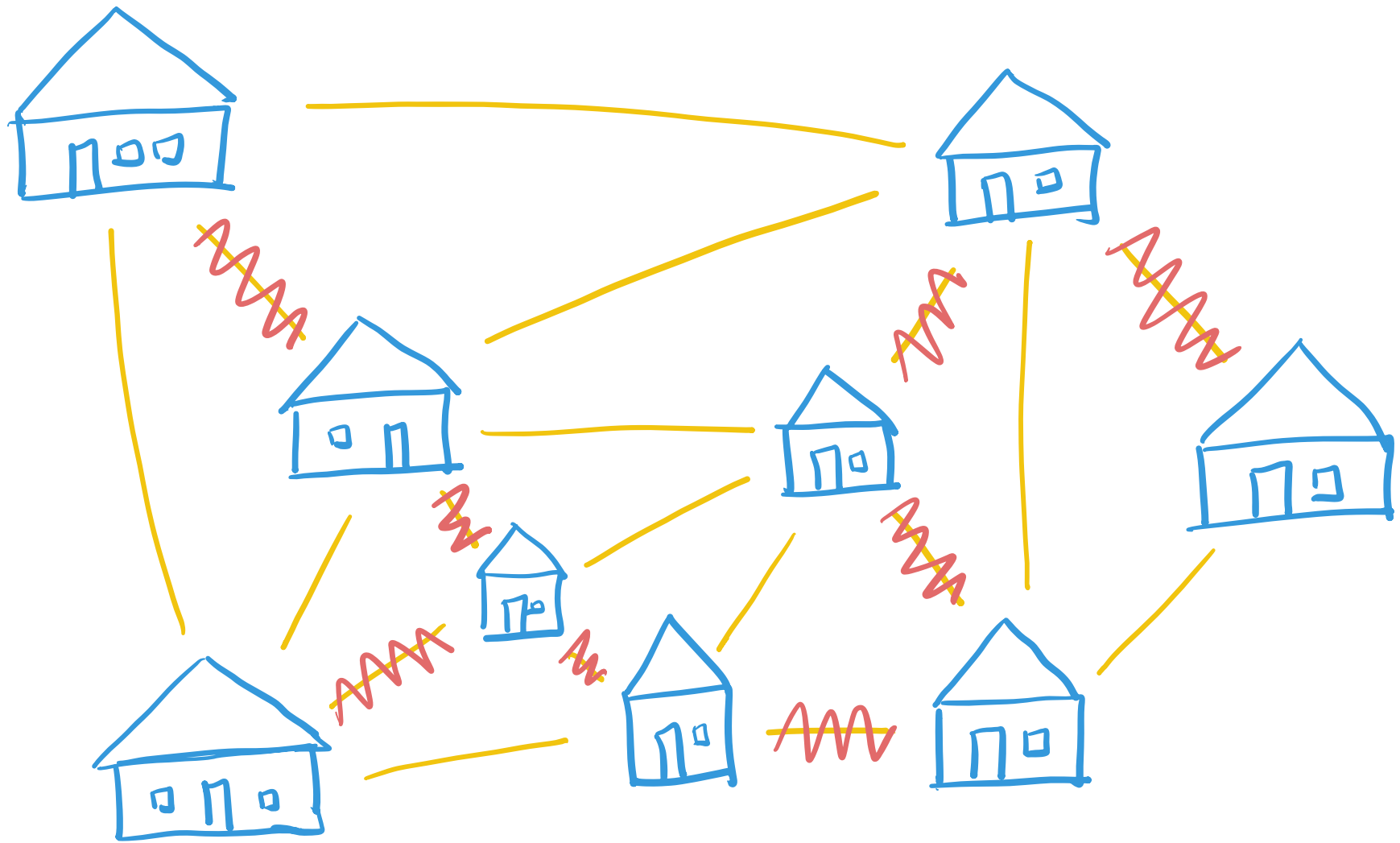


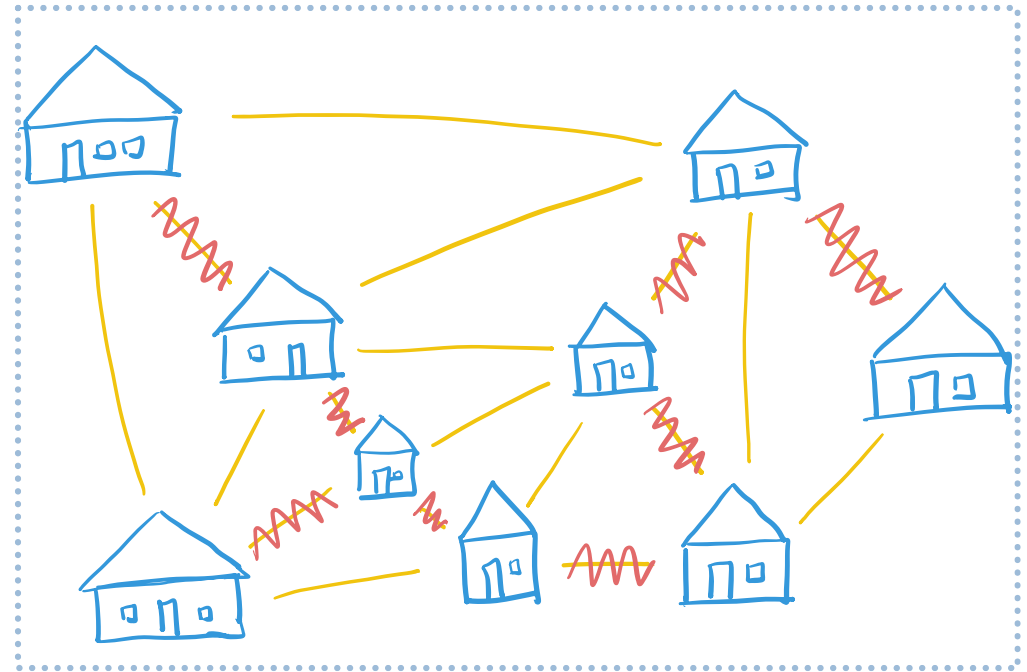
Spanning Trees



GOAL: connect town w/ minimum amount
of electrical wire

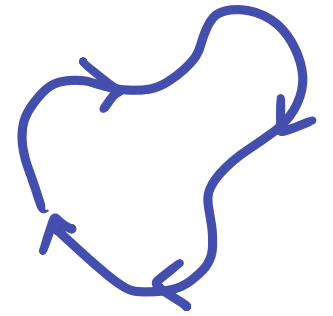
Quick observations

- no reason to have cycles
- biased towards short wires
- unused wires have alternative paths (which may be longer)

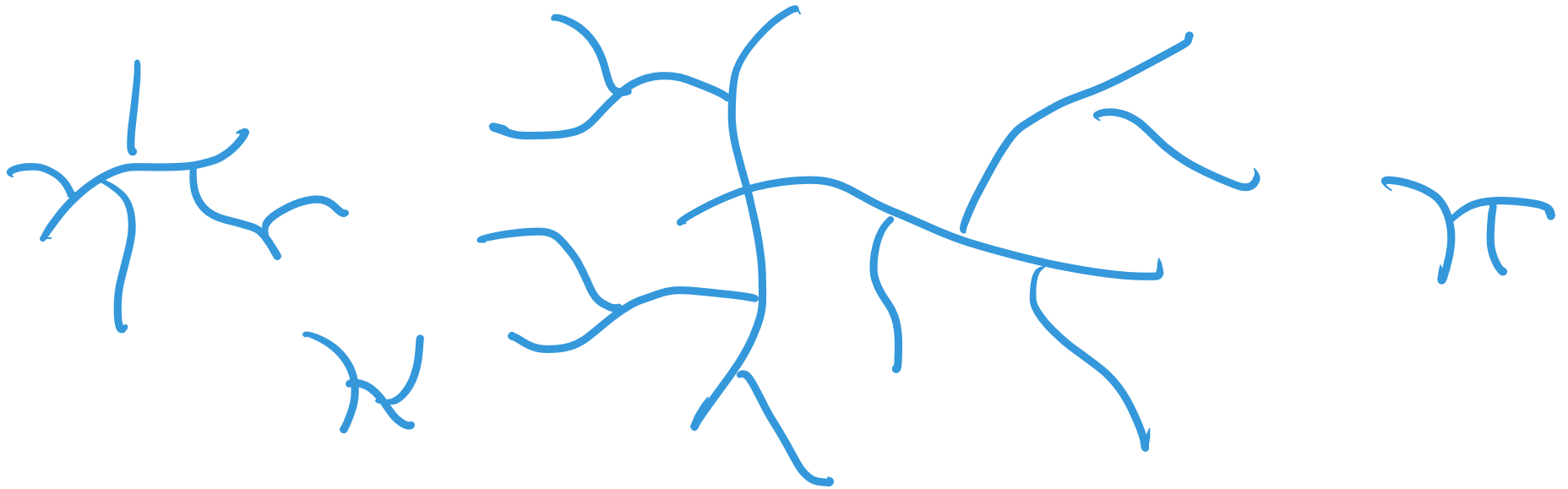


$G=(V,E)$ undirected, $F \subseteq E$ subset of edges.

"circuit" ^(nonempty) walk that starts/ends at same vertex



"forest": set of edges w/ no circuits



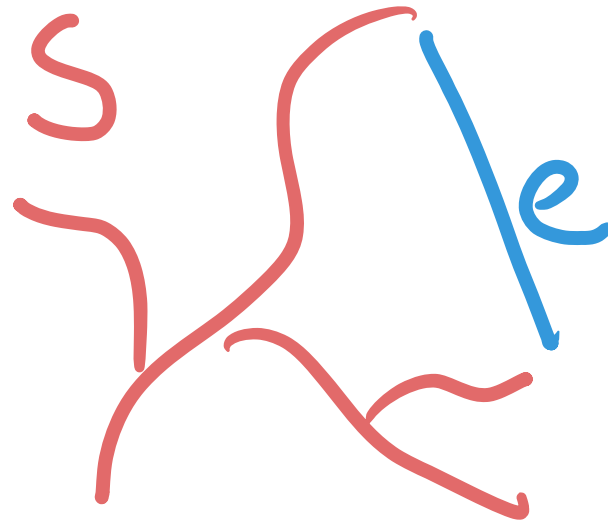
"Tree": forest w/ one connected component

$G=(V, E)$ undirected, $S \subseteq E$

" S spans e " if endpoints of e are connected by S

$$\text{span}(S) \stackrel{\text{def}}{=} \{e \in E: S \text{ spans } e\}$$

"S is spanning" if S spans every $e \in E$



2 problems

1. compute a forest $F \subseteq E$ of maximum cardinality (or weight)

2. compute a spanning set $S \subseteq E$ of minimum cardinality (or weight)

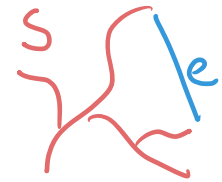
"circuit": ^(nonempty) walk that starts/ends at same vertex

"forest": set of edges w/ no circuits



"Tree": forest w/ 1 conn. component

"S spans e" if endpoints of e are connected by S



$\text{span}(S) \stackrel{\text{def}}{=} \{e \in E : S \text{ spans } e\}$

"S is spanning" if S spans every $e \in E$

let $G=(V,E)$ be connected

Lemma. $F \subseteq E$ forest, $S \subseteq E$ spanning. then

$$|F| \leq n-1 \leq |S|$$

2 problems

1. compute a forest $F \subseteq E$ of maximum cardinality (or weight)
2. compute a spanning set $S \subseteq E$ of minimum cardinality (or weight)

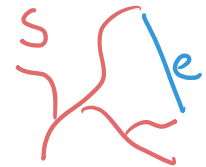
"circuit": ^(nonempty) walk that starts/ends at same vertex

"forest": set of edges w/ no circuits



"Tree": forest w/ 1 conn. component

"S spans e" if endpoints of e are connected by S



$\text{span}(S) \stackrel{\text{def}}{=} \{e \in E: S \text{ spans } e\}$

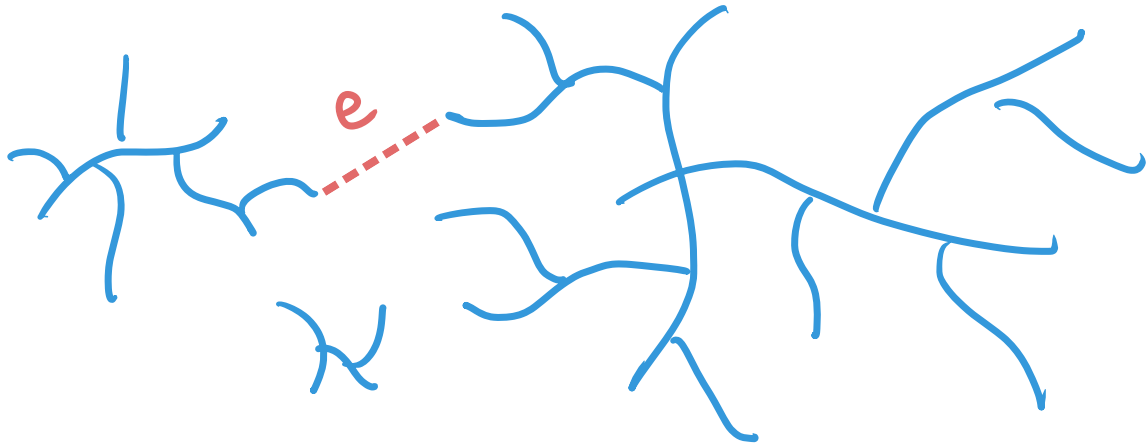
"S is spanning" if S spans every $e \in E$

$G=(V,E)$ connected

Lemma. $F \subseteq E$ forest, $S \subseteq E$ spanning. Then

$$|F| \leq n-1 \leq |S|$$

Proof. let $F = \emptyset$ be an empty forest
add edges to F one at a time



each edge decreases # connected components
by 1, from n down to 1

2 problems

1. compute a forest $F \subseteq E$ of maximum cardinality (or weight)
2. compute a spanning set $S \subseteq E$ of minimum cardinality (or weight)

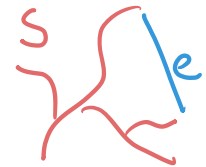
"circuit": ^(nonempty) walk that starts/ends at same vertex

"forest": set of edges w/ no circuits



"Tree": forest w/ 1 conn. component

" S spans e " if endpoints of e are connected by S



$\text{span}(S) \stackrel{\text{def}}{=} \{e \in E : S \text{ spans } e\}$

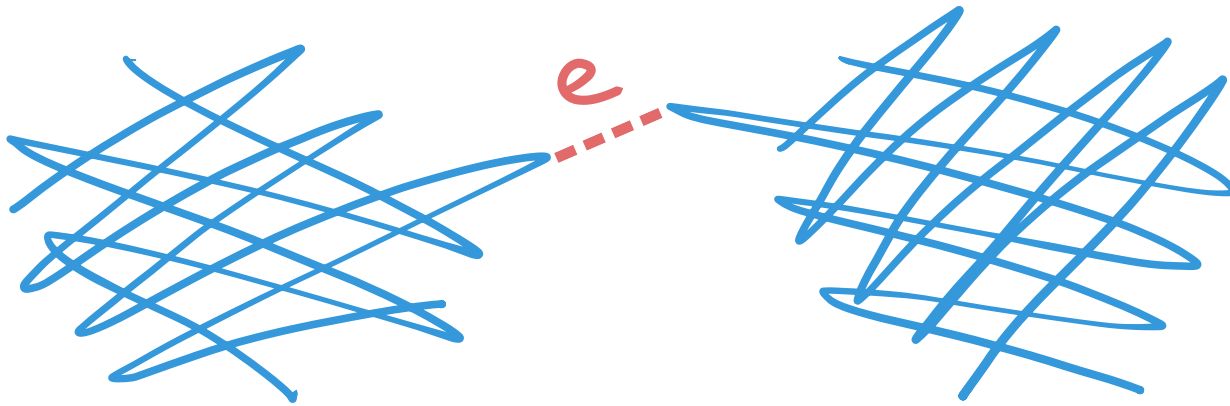
" S is spanning" if S spans every $e \in E$

$G=(V,E)$ connected

Lemma. $F \subseteq E$ forest, $S \subseteq E$ spanning. Then
 $|F| \leq n-1 \leq |S|$

Proof. let S be a spanning set.

there is 1 connected component w/r/t S



if we delete an edge, # conn. components increase by at most 1.

deleting all edges $\Rightarrow$ n conn. components

2 problems

1. compute a forest $F \subseteq E$ of maximum cardinality (or weight)
2. compute a spanning set $S \subseteq E$ of minimum cardinality (or weight)

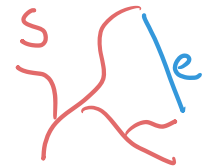
"circuit": ^(nonempty) walk that starts/ends at same vertex

"forest": set of edges w/ no circuits



"Tree": forest w/ 1 conn. component

" S spans e " if endpoints of e are connected by S

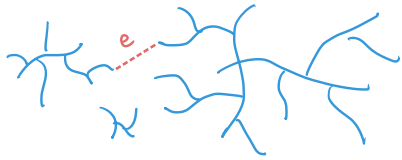


$\text{span}(S) \stackrel{\text{def}}{=} \{e \in E : S \text{ spans } e\}$

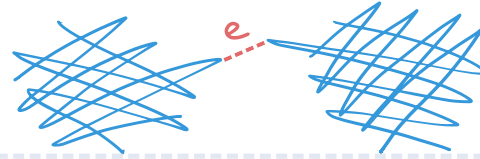
" S is spanning" if S spans every $e \in E$

$G=(V,E)$ connected

Lemma. $F \subseteq E$ forest, $S \subseteq E$ spanning. Then



$$|F| \leq n-1 \leq |S|$$



Now suppose G has k connected components.

claim: $|F| \leq n-k \leq |S|$

why?

2 problems

1. compute a forest $F \subseteq E$ of maximum cardinality (or weight)
2. compute a spanning set $S \subseteq E$ of minimum cardinality (or weight)

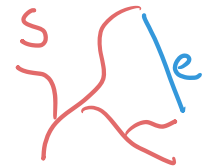
"circuit": ^(nonempty) walk that starts/ends at same vertex

"forest": set of edges w/ no circuits



"Tree": forest w/ 1 conn. component

" S spans e " if endpoints of e are connected by S



$\text{span}(S) \stackrel{\text{def}}{=} \{e \in E : S \text{ spans } e\}$

" S is spanning" if S spans every $e \in E$

$F \subseteq E$ forest

F is a maximal forest if for all $e \in E \setminus F$,
 $F + e$ is not a forest.

F is a maximum forest if it has maximum size over all forests.

$S \subseteq E$ spanning.

S is a minimal spanning set if for all $e \in S$,
 $S - e$ is not a spanning set

S is a minimum spanning set if it has minimum size over all spanning sets.

2 problems

1. compute a forest $F \subseteq E$ of maximum cardinality (or weight)
2. compute a spanning set $S \subseteq E$ of minimum cardinality (or weight)

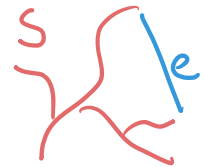
"circuit": ^(nonempty) walk that starts/ends at same vertex

"forest": set of edges w/ no circuits



"Tree": forest w/ 1 conn. component

" S spans e " if endpoints of e are connected by S



$\text{span}(S) \stackrel{\text{def}}{=} \{e \in E : S \text{ spans } e\}$

" S is spanning" if S spans every $e \in E$

G has k conn. components.

$F \subseteq E$ forest, $S \subseteq E$ spanning.

then $|F| \leq n - k \leq |S|$

Theorem. $F \subseteq E$ forest, $S \subseteq E$ spanning.

1. If F is maximal forest, then F is spanning.

2. If S is minimal spanning set, then S is a forest.

Corollary.

- Maximal forests are ^(globally) maximum forests
- Minimal spanning sets are ^(globally) minimum spanning sets.

In both cases they are spanning forest w/
 $n-k$ edges

2 problems

1. compute a forest $F \subseteq E$ of maximum cardinality (or weight)
2. compute a spanning set $S \subseteq E$ of minimum cardinality (or weight)

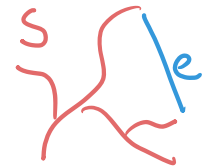
"circuit": ^(nonempty) walk that starts/ends at same vertex

"forest": set of edges w/ no circuits



"Tree": forest w/ 1 conn. component

" S spans e " if endpoints of e are connected by S



$\text{span}(S) \stackrel{\text{def}}{=} \{e \in E : S \text{ spans } e\}$

" S is spanning" if S spans every $e \in E$

G has k conn. components.

$F \subseteq E$ forest, $S \subseteq E$ spanning.

then $|F| \leq n-k \leq |S|$

Theorem. $F \subseteq E$ forest, $S \subseteq E$ spanning.

1. If F is maximal forest, then F is spanning.

2. If S is minimal spanning set, then S is a forest.

2 problems

1. compute a forest $F \subseteq E$ of maximum cardinality (or weight)
2. compute a spanning set $S \subseteq E$ of minimum cardinality (or weight)

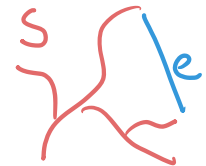
"circuit": ^(nonempty) walk that starts/ends at same vertex

"forest": set of edges w/ no circuits



"Tree": forest w/ 1 conn. component

" S spans e " if endpoints of e are connected by S



$\text{span}(S) \stackrel{\text{def}}{=} \{e \in E : S \text{ spans } e\}$

" S is spanning" if S spans every $e \in E$

G has k conn. components.

$F \subseteq E$ forest, $S \subseteq E$ spanning.

then $|F| \leq n - k \leq |S|$

Theorem. $F \subseteq E$ forest, $S \subseteq E$ spanning.

1. If F is maximal forest, then F is spanning.

2. If S is minimal spanning set, then S is a forest.

2 problems

1. compute a forest $F \subseteq E$ of maximum cardinality (or weight)
2. compute a spanning set $S \subseteq E$ of minimum cardinality (or weight)

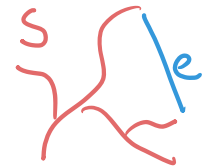
"circuit": ^(nonempty) walk that starts/ends at same vertex

"forest": set of edges w/ no circuits



"Tree": forest w/ 1 conn. component

" S spans e " if endpoints of e are connected by S



$\text{span}(S) \stackrel{\text{def}}{=} \{e \in E : S \text{ spans } e\}$

" S is spanning" if S spans every $e \in E$

G has k conn. components.

$F \subseteq E$ forest, $S \subseteq E$ spanning.

then $|F| \leq n - k \leq |S|$

Theorem. $F \subseteq E$ forest, $S \subseteq E$ spanning.

1. If F is maximal forest, then F is spanning.
2. If S is minimal spanning set, then S is a forest.

Corollary.

- Maximal forests are ^(globally) maximum forests
- Minimal spanning sets are ^(globally) minimum spanning sets.

In both cases they are spanning forest w/
 $n-k$ edges

Corollary. If G is connected, there is
a spanning tree w/ $n-1$ edges.

2 problems

1. compute a forest $F \subseteq E$ of maximum cardinality (or weight)
2. compute a spanning set $S \subseteq E$ of minimum cardinality (or weight)

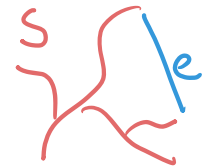
"circuit": ^(nonempty) walk that starts/ends at same vertex

"forest": set of edges w/ no circuits



"Tree": forest w/ 1 conn. component

" S spans e " if endpoints of e are connected by S



$\text{span}(S) \stackrel{\text{def}}{=} \{e \in E : S \text{ spans } e\}$

" S is spanning" if S spans every $e \in E$

G has k conn. components.

$F \subseteq E$ forest, $S \subseteq E$ spanning.

then $|F| \leq n-k \leq |S|$

2 problems

1. compute a forest $F \subseteq E$ of
maximum cardinality (or weight)

2. compute a spanning set $S \subseteq E$ of
minimum cardinality (or weight)

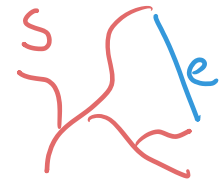
"circuit": ^(nonempty) walk that starts/ends at same vertex

"forest": set of edges w/ no circuits



"Tree": forest w/ 1 conn. component

"S spans e" if endpoints of e are connected by S



$\text{span}(S) \stackrel{\text{def}}{=} \{e \in E : S \text{ spans } e\}$

"S is spanning" if S spans every $e \in E$

Assume that all edge weights are distinct

2 problems

Good edges

$e \in E$ is "good" if

1. compute a forest $F \subseteq E$ of maximum cardinality (or weight)
2. compute a spanning set $S \subseteq E$ of minimum cardinality (or weight)

For every set of edges $F \subseteq E - e$ that spans e (and doesn't include e),

$$w(e) < \max_{f \in F} w(f).$$

Good edges

2 problems

1. compute a forest $F \subseteq E$ of maximum cardinality (or weight)
2. compute a spanning set $S \subseteq E$ of minimum cardinality (or weight)

$e \in E$ is "good" if

For every set of edges $F \subseteq E - e$ that spans e (and doesn't include e),

$$w(e) < \max_{f \in F} w(f).$$

Lemma 13.5. *Let $e \in E$. Then the following conditions are equivalent.*

1. *Every minimum weight spanning tree contains e .*
2. *For every set of edges $F \subseteq E - e$ that spans e (and doesn't include e),*
(e is good)

$$w(e) < \max_{f \in F} w(f).$$

2 problems

1. compute a forest $F \subseteq E$ of maximum cardinality (or weight)
2. compute a spanning set $S \subseteq E$ of minimum cardinality (or weight)

$e \in E$ is "good" if

For every set of edges $F \subseteq E - e$ that spans e (and doesn't include e),

$$w(e) < \max_{f \in F} w(f).$$

Lemma 13.5. Let $e \in E$. Then the following conditions are equivalent.

1. Every minimum weight spanning tree contains e .
2. For every set of edges $F \subseteq E - e$ that spans e (and doesn't include e),
(e is good)

$$w(e) < \max_{f \in F} w(f).$$

Proof. Assume #1.

2 problems

1. compute a forest $F \subseteq E$ of maximum cardinality (or weight)
2. compute a spanning set $S \subseteq E$ of minimum cardinality (or weight)

$e \in E$ is "good" if

For every set of edges $F \subseteq E - e$ that spans e (and doesn't include e),

$$w(e) < \max_{f \in F} w(f).$$

Lemma 13.5. Let $e \in E$. Then the following conditions are equivalent.

1. Every minimum weight spanning tree contains e .

2. For every set of edges $F \subseteq E - e$ that spans e (and doesn't include e),

(e is good)

$$w(e) < \max_{f \in F} w(f).$$

Proof. Assume #2.

2 problems

1. compute a forest $F \subseteq E$ of maximum cardinality (or weight)
2. compute a spanning set $S \subseteq E$ of minimum cardinality (or weight)

$e \in E$ is "good" if

For every set of edges $F \subseteq E - e$ that spans e (and doesn't include e),

$$w(e) < \max_{f \in F} w(f).$$

Lemma 13.5. Let $e \in E$. Then the following conditions are equivalent.

1. Every minimum weight spanning tree contains e .
2. For every set of edges $F \subseteq E - e$ that spans e (and doesn't include e),

(e is good)

$$w(e) < \max_{f \in F} w(f).$$

generic-MST($G = (V, E), w : E \rightarrow \mathbb{R}$)

1. $T \leftarrow \emptyset$

2. Until T is a spanning tree.

A. Add a good edge $e \in E \setminus \text{span}(T)$ to T

// Or throw an error if no such edge exists.

3. Return T .

2 problems

1. compute a forest $F \subseteq E$ of maximum cardinality (or weight)
2. compute a spanning set $S \subseteq E$ of minimum cardinality (or weight)

$e \in E$ is "good" if

For every set of edges $F \subseteq E - e$ that spans e (and doesn't include e),

$$w(e) < \max_{f \in F} w(f).$$

Lemma 13.5. Let $e \in E$. Then the following conditions are equivalent.

1. Every minimum weight spanning tree contains e .
2. For every set of edges $F \subseteq E - e$ that spans e (and doesn't include e),
(e is good)

$$w(e) < \max_{f \in F} w(f).$$

generic-MST($G = (V, E), w : E \rightarrow \mathbb{R}$)

1. $T \leftarrow \emptyset$
2. Until T is a spanning tree.
 - A. Add a good edge $e \in E \setminus \text{span}(T)$ to T
// Or throw an error if no such edge exists.
3. Return T .

2 problems

1. compute a forest $F \subseteq E$ of maximum cardinality (or weight)
2. compute a spanning set $S \subseteq E$ of minimum cardinality (or weight)

$e \in E$ is "good" if

For every set of edges $F \subseteq E - e$ that spans e (and doesn't include e),

$$w(e) < \max_{f \in F} w(f).$$

Lemma 13.5. Let $e \in E$. Then the following conditions are equivalent.

1. Every minimum weight spanning tree contains e .
2. For every set of edges $F \subseteq E - e$ that spans e (and doesn't include e),
(e is good)

$$w(e) < \max_{f \in F} w(f).$$

generic-MST($G = (V, E), w : E \rightarrow \mathbb{R}$)

1. $T \leftarrow \emptyset$
2. Until T is a spanning tree.
 - A. Add a good edge $e \in E \setminus \text{span}(T)$ to T
// Or throw an error if no such edge exists.
3. Return T .

Lemma 13.7. Let $H \subset E$ be any non-spanning set of edges. Let $e \in E \setminus \text{span}(H)$ be the minimum weight edge not spanned by H . Then e is a good edge.

Greedy algorithm

greedy-MST($G = (V, E), w : E \rightarrow \mathbb{R}$)

/ Also known as Kruskal's algorithm. */*

1. Sort and index $E = \{e_1, \dots, e_n\}$ in increasing order of weight.
2. Set $T \leftarrow \emptyset$.
3. For $i = 1, \dots, n$:

/ Below we discuss how to implement the following predicate efficiently. */*

A. If $T + e_i$ is a forest:

/ e_i is the minimum weight edge not spanned by T */*

1. Set $T \leftarrow T + e_i$.

4. Return T .

2 problems

1. compute a forest $F \subseteq E$ of maximum cardinality (or weight)
2. compute a spanning set $S \subseteq E$ of minimum cardinality (or weight)

$e \in E$ is "good" if

For every set of edges $F \subseteq E - e$ that spans e (and doesn't include e),

$$w(e) < \max_{f \in F} w(f).$$

Lemma 13.5. Let $e \in E$. Then the following conditions are equivalent.

1. Every minimum weight spanning tree contains e .
2. For every set of edges $F \subseteq E - e$ that spans e (and doesn't include e),
(e is good)

$$w(e) < \max_{f \in F} w(f).$$

generic-MST($G = (V, E), w : E \rightarrow \mathbb{R}$)

1. $T \leftarrow \emptyset$
2. Until T is a spanning tree.
 - A. Add a good edge $e \in E \setminus \text{span}(T)$ to T
// Or throw an error if no such edge exists.
3. Return T .

Lemma 13.7. Let $H \subset E$ be any non-spanning set of edges. Let $e \in E \setminus \text{span}(H)$ be the minimum weight edge not spanned by H . Then e is a good edge.

Greedy algorithm

greedy-MST($G = (V, E), w : E \rightarrow \mathbb{R}$)

/ Also known as Kruskal's algorithm. */*

1. Sort and index $E = \{e_1, \dots, e_n\}$ in increasing order of weight.
2. Set $T \leftarrow \emptyset$.
3. For $i = 1, \dots, n$:

/ Below we discuss how to implement the following predicate efficiently. */*

A. If $T + e_i$ is a forest:

/ e_i is the minimum weight edge not spanned by T */*

1. Set $T \leftarrow T + e_i$.

4. Return T .

↑
why is e_i good?

2 problems

1. compute a forest $F \subseteq E$ of maximum cardinality (or weight)
2. compute a spanning set $S \subseteq E$ of minimum cardinality (or weight)

$e \in E$ is "good" if

For every set of edges $F \subseteq E - e$ that spans e (and doesn't include e),

$$w(e) < \max_{f \in F} w(f).$$

Lemma 13.5. Let $e \in E$. Then the following conditions are equivalent.

1. Every minimum weight spanning tree contains e .
2. For every set of edges $F \subseteq E - e$ that spans e (and doesn't include e),
(e is good)

$$w(e) < \max_{f \in F} w(f).$$

generic-MST($G = (V, E), w : E \rightarrow \mathbb{R}$)

1. $T \leftarrow \emptyset$
2. Until T is a spanning tree.
 - A. Add a good edge $e \in E \setminus \text{span}(T)$ to T
// Or throw an error if no such edge exists.
3. Return T .

Lemma 13.7. Let $H \subset E$ be any non-spanning set of edges. Let $e \in E \setminus \text{span}(H)$ be the minimum weight edge not spanned by H . Then e is a good edge.

"Parallel algorithm"

parallel-MST($G = (V, E), w : E \rightarrow \mathbb{R}$)

/* Also known as Borůvka's algorithm. */

1. $T \leftarrow \emptyset$
2. While T more than 1 connected component:
 - A. For each component S of T , identify the smallest weight edge in $\partial(S)$.
 - B. Add all of these edges to T .
3. Return T

2 problems

1. compute a forest $F \subseteq E$ of maximum cardinality (or weight)
2. compute a spanning set $S \subseteq E$ of minimum cardinality (or weight)

$e \in E$ is "good" if

For every set of edges $F \subseteq E - e$ that spans e (and doesn't include e),

$$w(e) < \max_{f \in F} w(f).$$

Lemma 13.5. Let $e \in E$. Then the following conditions are equivalent.

1. Every minimum weight spanning tree contains e .
2. For every set of edges $F \subseteq E - e$ that spans e (and doesn't include e),
(e is good)

$$w(e) < \max_{f \in F} w(f).$$

generic-MST($G = (V, E), w : E \rightarrow \mathbb{R}$)

1. $T \leftarrow \emptyset$
2. Until T is a spanning tree.
 - A. Add a good edge $e \in E \setminus \text{span}(T)$ to T
// Or throw an error if no such edge exists.
3. Return T .

Lemma 13.7. Let $H \subset E$ be any non-spanning set of edges. Let $e \in E \setminus \text{span}(H)$ be the minimum weight edge not spanned by H . Then e is a good edge.

"Search algorithm"

search-MST($G = (V, E), w : E \rightarrow \mathbb{R}$)

/ Also known as Prim's algorithm. */*

1. Set $T \leftarrow \emptyset$ and $S \leftarrow \{v\}$ for an arbitrary vertex v .

/ We maintain the invariant that S is the connected component of v in T . */*

2. While T is not a spanning tree:

/ We can accelerate the following steps with a Fibonacci heap. */*

A. Let e be the minimum weight edge in $\partial(S)$.

B. Set $T \leftarrow T + e, S \leftarrow S \cup e$.

3. Return T .

Running times

greedy-MST($G = (V, E), w : E \rightarrow \mathbb{R}$)

/ Also known as Kruskal's algorithm. */*

1. Sort and index $E = \{e_1, \dots, e_n\}$ in increasing order of weight.
2. Set $T \leftarrow \emptyset$.
3. For $i = 1, \dots, n$:

/ Below we discuss how to implement the following predicate efficiently. */*

A. If $T + e_i$ is a forest:

/ e_i is the minimum weight edge not spanned by T */*

1. Set $T \leftarrow T + e_i$.

4. Return T .

parallel-MST($G = (V, E), w : E \rightarrow \mathbb{R}$)

/ Also known as Borůvka's algorithm. */*

1. $T \leftarrow \emptyset$
2. While T more than 1 connected component:
 - A. For each component S of T , identify the smallest weight edge in $\partial(S)$.
 - B. Add all of these edges to T .
3. Return T

search-MST($G = (V, E), w : E \rightarrow \mathbb{R}$)

/ Also known as Prim's algorithm. */*

1. Set $T \leftarrow \emptyset$ and $S \leftarrow \{v\}$ for an arbitrary vertex v .

/ We maintain the invariant that S is the connected component of v in T . */*

2. While T is not a spanning tree:

/ We can accelerate the following steps with a Fibonacci heap. */*

A. Let e be the minimum weight edge in $\partial(S)$.

B. Set $T \leftarrow T + e, S \leftarrow S \cup e$.

3. Return T .

Greedy algorithm

greedy-MST($G = (V, E), w : E \rightarrow \mathbb{R}$)

/ Also known as Kruskal's algorithm. */*

1. Sort and index $E = \{e_1, \dots, e_n\}$ in increasing order of weight.

2. Set $T \leftarrow \emptyset$.

3. For $i = 1, \dots, n$:

/ Below we discuss how to implement the following predicate efficiently. */*

A. If $T + e_i$ is a forest:

/ e_i is the minimum weight edge not spanned by T */*

1. Set $T \leftarrow T + e_i$.

4. Return T .

2 problems

1. compute a forest $F \subseteq E$ of maximum cardinality (or weight)

2. compute a spanning set $S \subseteq E$ of minimum cardinality (or weight)

$e \in E$ is "good" if

For every set of edges $F \subseteq E - e$ that spans e (and doesn't include e),

$$w(e) < \max_{f \in F} w(f).$$

Lemma 13.5. Let $e \in E$. Then the following conditions are equivalent.

1. Every minimum weight spanning tree contains e .

2. For every set of edges $F \subseteq E - e$ that spans e (and doesn't include e),
(e is good)

$$w(e) < \max_{f \in F} w(f).$$

generic-MST($G = (V, E), w : E \rightarrow \mathbb{R}$)

1. $T \leftarrow \emptyset$

2. Until T is a spanning tree.

A. Add a good edge $e \in E \setminus \text{span}(T)$ to T

// Or throw an error if no such edge exists.

3. Return T .

Disjoint union data structure

1. $\text{union}(u, v)$: Given two elements u and v , take the union of the their sets.

2. $\text{same-set}(u, v)$: Returns whether or not x and y are in the same set.

3. $\text{new-set}(x)$: Given a new element x , creates the singleton set $\{x\}$.

Theorem 13.8. There is a data structure that can implement the union, same-set, and new-set operations listed above that, for any m operations over n total elements, takes $O(m + n\alpha(n))$ time, where $\alpha(n)$ is the inverse Ackerman function.

"Parallel algorithm"

parallel-MST($G = (V, E), w : E \rightarrow \mathbb{R}$)

/ Also known as Borůvka's algorithm. */*

1. $T \leftarrow \emptyset$
2. While T more than 1 connected component:
 - A. For each component S of T , identify the smallest weight edge in $\partial(S)$.
 - B. Add all of these edges to T .
3. Return T

2 problems

1. compute a forest $F \subseteq E$ of maximum cardinality (or weight)
2. compute a spanning set $S \subseteq E$ of minimum cardinality (or weight)

$e \in E$ is "good" if

For every set of edges $F \subseteq E - e$ that spans e (and doesn't include e),

$$w(e) < \max_{f \in F} w(f).$$

Lemma 13.5. Let $e \in E$. Then the following conditions are equivalent.

1. Every minimum weight spanning tree contains e .
2. For every set of edges $F \subseteq E - e$ that spans e (and doesn't include e),
(e is good)

$$w(e) < \max_{f \in F} w(f).$$

generic-MST($G = (V, E), w : E \rightarrow \mathbb{R}$)

1. $T \leftarrow \emptyset$
2. Until T is a spanning tree.
 - A. Add a good edge $e \in E \setminus \text{span}(T)$ to T
// Or throw an error if no such edge exists.
3. Return T .

"Search algorithm"

search-MST($G = (V, E), w : E \rightarrow \mathbb{R}$)

/ Also known as Prim's algorithm. */*

1. Set $T \leftarrow \emptyset$ and $S \leftarrow \{v\}$ for an arbitrary vertex v .

/ We maintain the invariant that S is the connected component of v in T . */*

2. While T is not a spanning tree:

/ We can accelerate the following steps with a Fibonacci heap. */*

A. Let e be the minimum weight edge in $\partial(S)$.

B. Set $T \leftarrow T + e, S \leftarrow S \cup e$.

3. Return T .

2 problems

1. compute a forest $F \subseteq E$ of maximum cardinality (or weight)
2. compute a spanning set $S \subseteq E$ of minimum cardinality (or weight)

$e \in E$ is "good" if

For every set of edges $F \subseteq E - e$ that spans e (and doesn't include e),

$$w(e) < \max_{f \in F} w(f).$$

Lemma 13.5. Let $e \in E$. Then the following conditions are equivalent.

1. Every minimum weight spanning tree contains e .
2. For every set of edges $F \subseteq E - e$ that spans e (and doesn't include e),
(e is good)

$$w(e) < \max_{f \in F} w(f).$$

generic-MST($G = (V, E), w : E \rightarrow \mathbb{R}$)

1. $T \leftarrow \emptyset$

2. Until T is a spanning tree.

A. Add a good edge $e \in E \setminus \text{span}(T)$ to T
// Or throw an error if no such edge exists.

3. Return T .

Priority queues / heaps

1. $\text{insert}(k, p)$: inserts an key k with priority p .
2. $\text{decrease}(k, p)$: Let k be a key in the heap. If p is less than the current priority of k , decrease the priority to p .
3. $\text{remove-min}()$: removes and returns the key value pair (k, p) with the minimum priority p .

Fact 8.1. *There exists a data structure, called a **Fibonacci heap**, that has the following guarantee. Over any sequence of I calls to insert-key, D calls to decrease-key, and R calls to remove-min, the Fibonacci heap takes*

$$O(I + D + R \log n)$$

time in total.

"Search algorithm"

search-MST($G = (V, E), w : E \rightarrow \mathbb{R}$)

/ Also known as Prim's algorithm. */*

1. Set $T \leftarrow \emptyset$ and $S \leftarrow \{v\}$ for an arbitrary vertex v .

/ We maintain the invariant that S is the connected component of v in T . */*

2. While T is not a spanning tree:

/ We can accelerate the following steps with a Fibonacci heap. */*

A. Let e be the minimum weight edge in $\partial(S)$.

B. Set $T \leftarrow T + e, S \leftarrow S \cup e$.

3. Return T .

2 problems

1. compute a forest $F \subseteq E$ of maximum cardinality (or weight)
2. compute a spanning set $S \subseteq E$ of minimum cardinality (or weight)

$e \in E$ is "good" if

For every set of edges $F \subseteq E - e$ that spans e (and doesn't include e),

$$w(e) < \max_{f \in F} w(f).$$

Lemma 13.5. Let $e \in E$. Then the following conditions are equivalent.

1. Every minimum weight spanning tree contains e .
2. For every set of edges $F \subseteq E - e$ that spans e (and doesn't include e),
(e is good)

$$w(e) < \max_{f \in F} w(f).$$

generic-MST($G = (V, E), w : E \rightarrow \mathbb{R}$)

1. $T \leftarrow \emptyset$

2. Until T is a spanning tree.

A. Add a good edge $e \in E \setminus \text{span}(T)$ to T
// Or throw an error if no such edge exists.

3. Return T .

$O(1)$

1. insert(k, p): inserts a key k with priority p .

$O(1)$

2. decrease(k, p): Let k be a key in the heap. If p is less than the current priority of k , decrease the priority to p .

$O(\log n)$

3. remove-min(): removes and returns the key value pair (k, p) with the minimum priority p .

Running times

greedy-MST($G = (V, E), w : E \rightarrow \mathbb{R}$)

/ Also known as Kruskal's algorithm. */*

1. Sort and index $E = \{e_1, \dots, e_n\}$ in increasing order of weight.
2. Set $T \leftarrow \emptyset$.
3. For $i = 1, \dots, n$:

/ Below we discuss how to implement the following predicate efficiently. */*

A. If $T + e_i$ is a forest:

/ e_i is the minimum weight edge not spanned by T */*

1. Set $T \leftarrow T + e_i$.

4. Return T .

parallel-MST($G = (V, E), w : E \rightarrow \mathbb{R}$)

/ Also known as Borůvka's algorithm. */*

1. $T \leftarrow \emptyset$
2. While T more than 1 connected component:
 - A. For each component S of T , identify the smallest weight edge in $\partial(S)$.
 - B. Add all of these edges to T .
3. Return T

search-MST($G = (V, E), w : E \rightarrow \mathbb{R}$)

/ Also known as Prim's algorithm. */*

1. Set $T \leftarrow \emptyset$ and $S \leftarrow \{v\}$ for an arbitrary vertex v .

/ We maintain the invariant that S is the connected component of v in T . */*

2. While T is not a spanning tree:

/ We can accelerate the following steps with a Fibonacci heap. */*

A. Let e be the minimum weight edge in $\partial(S)$.

B. Set $T \leftarrow T + e, S \leftarrow S \cup e$.

3. Return T .