L13 5800

mar 4/7 2022

shelat

connecting houses

A village begins with just a single home.

image:www.princegeorgeva.org, thefranciscofamily.org, www.rightdriveacademy.co.uk, www.ccscambridge.org, www.drawingcoach.com, www.pastoral.org.uk, www.daasgallery.com

connecting houses

At some point, a neighbor moves in.

image:www.princegeorgeva.org, thefranciscofamily.org, www.rightdriveacademy.co.uk, www.ccscambridge.org, www.drawingcoach.com, www.pastoral.org.uk, www.daasgallery.com

connecting houses

And with human nature, they build a road to connect each other.

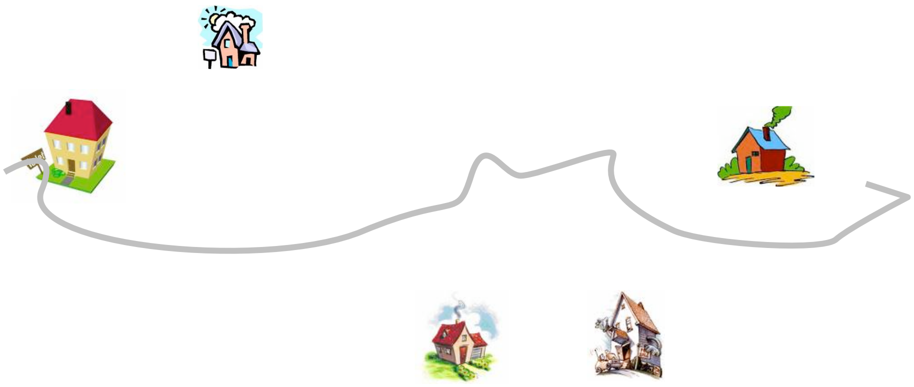
image:www.princegeorgeva.org, thefranciscofamily.org, www.rightdriveacademy.co.uk, www.ccscambridge.org, www.drawingcoach.com, www.pastoral.org.uk, www.daasgallery.com

connecting houses

Soon others follow, and each wants a way to reach their neighbors.

What is the best way to build such a network?

connecting houses

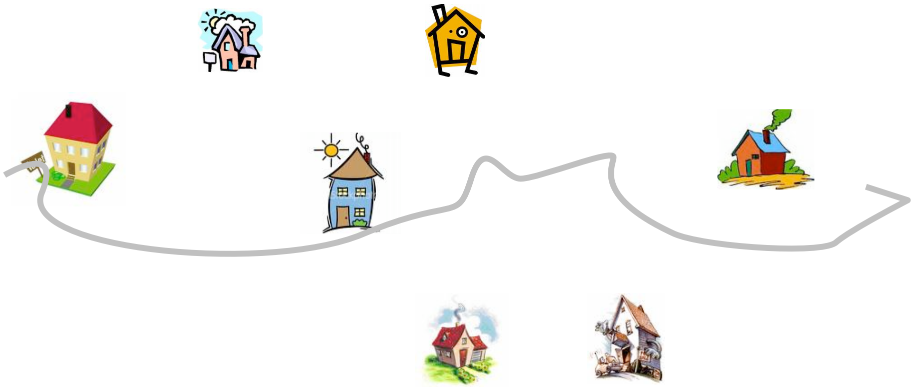


Soon others follow, and each wants a way to reach their neighbors.

image:www.princegeorgeva.org, thefranciscofamily.org, www.rightdriveacademy.co.uk, www.ccscambridge.org, www.drawingcoach.com, www.pastoral.org.uk, www.daasgallery.com

What is the best way to build such a network?

connecting houses



Soon others follow, and each wants a way to reach their neighbors.

image:www.princegeorgeva.org, thefranciscofamily.org, www.rightdriveacademy.co.uk, www.ccscambridge.org, www.drawingcoach.com, www.pastoral.org.uk, www.daasgallery.com

What is the best way to build such a network?

connecting houses

Soon others follow, and each wants a way to reach their neighbors.

image:www.princegeorgeva.org, thefranciscofamily.org, www.rightdriveacademy.co.uk, www.ccscambridge.org, www.drawingcoach.com, www.pastoral.org.uk, www.daasgallery.com

What is the best way to build such a network?

connecting houses

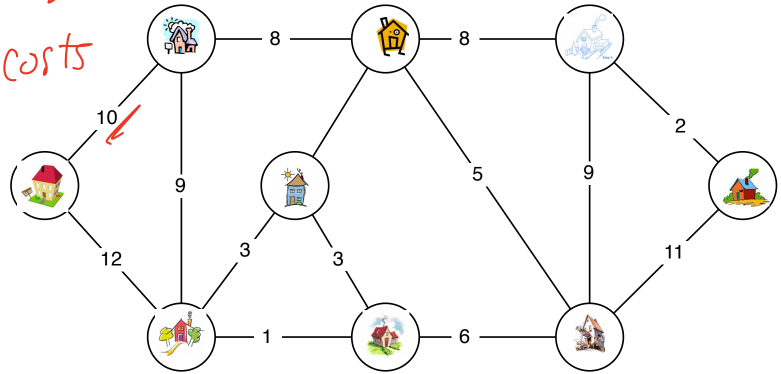
Soon others follow, and each wants a way to reach their neighbors.

image:www.princegeorgeva.org, thefranciscofamily.org, www.rightdriveacademy.co.uk, www.ccscambridge.org, www.drawingcoach.com, www.pastoral.org.uk, www.daasgallery.com

What is the best way to build such a network?

weights

costs

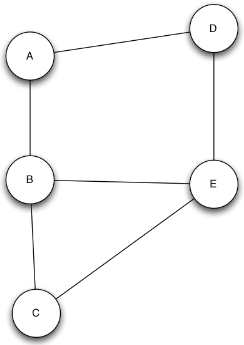


The best way to represent the input to this problem is a graph.

graphs

clrs [ch 22]

$$G = (V, E)$$



A graph is a pair of two sets,
A set of vertices, and a set of edges.

Edges may have annotations, such as weights, w(e).

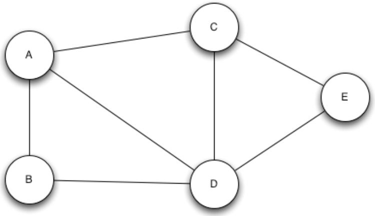
$e \in E$

$c(e) : E \to N$

representation

$$G = (V, E)$$

adjacency list



space:

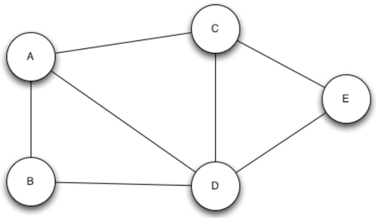
time list neighbors:

time check an edge:

The first way to represent a graph is via its adjacency list. For the edges, each vertex maintains a list of its neighbors.

representation

$$G = (V, E)$$



adjacency list

V $\rightarrow$

a {b,c,d}

b {d}

c {d,e}

d {e}

e {}

space:

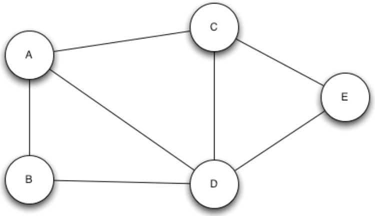
time list neighbors:

time check an edge:

The first way to represent a graph is via its adjacency list.
For the edges, each vertex maintains a list of its neighbors.

representation

$$G = (V, E)$$



adjacency list

a {b,c,d}

b {d}

c {d,e}

d {e}

e {}

space: O(|E|)

time list neighbors: O(degree(v))

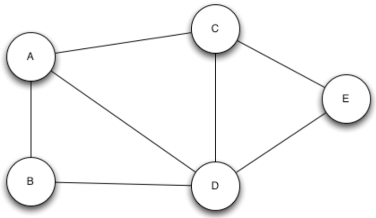
time check an edge: O(V)

The first way to represent a graph is via its adjacency list.
For the edges, each vertex maintains a list of its neighbors.

representation

$$G = (V, E)$$

adjacency matrix



space:

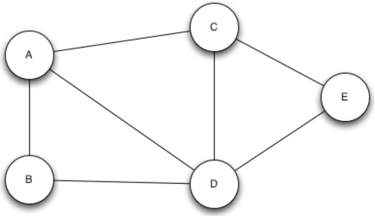
time list neighbors:

time check an edge:

The second way to represent a graph is via its adjacency matrix.

representation

$$G = (V, E)$$



adjacency matrix

	a	b	c	d	e
a	0	1	1	1	0
b	1	0	0	1	0
c	1	0	0	1	1
d	1	1	1		1
e			1	1	

space:

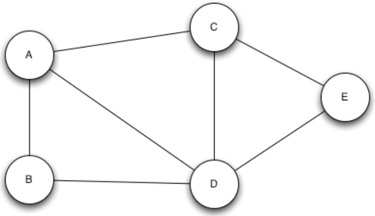
time list neighbors:

time check an edge:

The second way to represent a graph is via its adjacency matrix.

representation

$$G = (V, E)$$



adjacency matrix

V

$$\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline a & 0 & 1 & 1 & 1 & 0 \\ b & 1 & 0 & 0 & 1 & 0 \\ c & 1 & 0 & 0 & 1 & 1 \\ d & 1 & 1 & 1 & & 1 \\ e & & & 1 & 1 & \end{array}$$

V

space: $O(|V^2|)$

time list neighbors: $O(V)$

time check an edge: $O(1)$

The second way to represent a graph is via its adjacency matrix.

definition: path

a sequence of nodes $v_1, v_2, \ldots, v_k$

with the property that $(v_i, v_{i+1}) \in E$

simple path:

cycle:

definition: path

a sequence of nodes $v_1, v_2, \ldots, v_k$

with the property that $(v_i, v_{i+1}) \in E$

simple path: Path in which each vertex appears at most once.

cycle:

definition: path

a sequence of nodes $v_1, v_2, \ldots, v_k$
with the property that $(v_i, v_{i+1}) \in E$

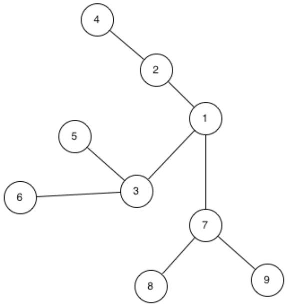
simple path: Path in which each vertex appears at most once.

cycle: Path with the same start and end vertex.

definition:tree

connected graph:

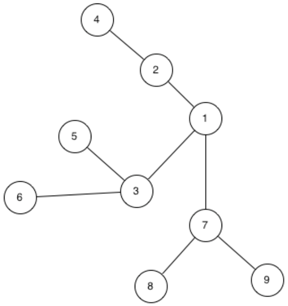
a tree is



definition:tree

connected graph: A graph G in which for each pair of vertices, (u,v), there exists a path from u to v.

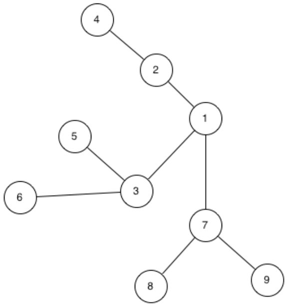
a tree is



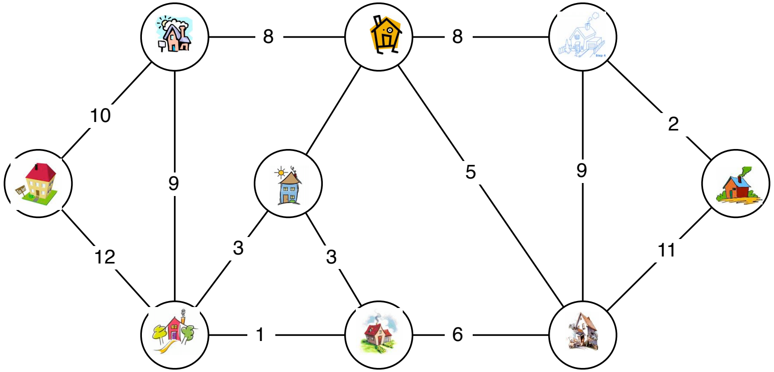
definition:tree

connected graph: A graph G in which for each pair of vertices, (u,v), there exists a path from u to v.

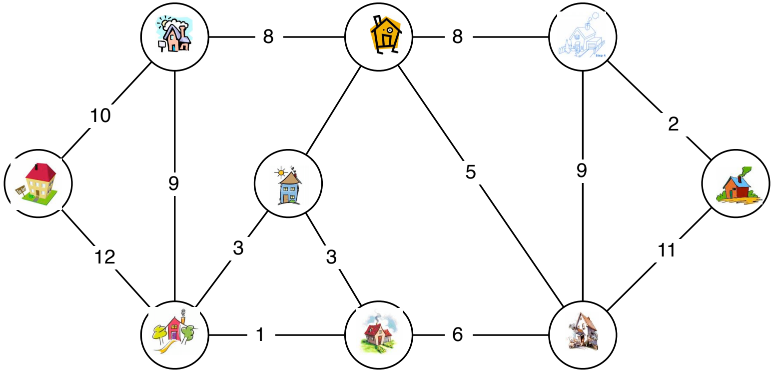
a tree is A connected graph with no cycles



what we want:



what we want:



We want to connect all nodes in G in the cheapest way.
We want a tree in G with the minimum sum of edge costs.

minimum spanning tree

looking for a set of edges that $T \subseteq E$
(a) connects all vertices
(b) has the least cost $\min \sum_{(u,v) \in T} w(u, v)$

① How many edges does the solution have?? $V-1$

② Do not want any cycles.
Because cycles add extra cost
without adding extra connected nodes.
Removing a cycle ⇒ lower cost solution

minimum spanning tree

looking for a set of edges that $T \subseteq E$

(a) connects all vertices

(b) has the least cost $\min \sum_{(u,v) \in T} w(u,v)$

This object is called a minimum spanning tree.

facts

looking for a set of edges that $T \subseteq E$

(a) connects all vertices

(b) has the least cost $\min \sum_{(u,v) \in T} w(u,v)$

how many edges does solution have ?

does solution have a cycle?

facts

looking for a set of edges that $T \subseteq E$

(a) connects all vertices

(b) has the least cost $\min \sum_{(u,v) \in T} w(u,v)$

how many edges does solution have ? V-1

does solution have a cycle?

facts

looking for a set of edges that $T \subseteq E$

(a) connects all vertices

(b) has the least cost $\min \sum_{(u,v) \in T} w(u,v)$

how many edges does solution have ?

V-1

does solution have a cycle?

No. Because removing the cycle leads to a cheaper solution.

Greedy strategy

start with an empty set of edges A

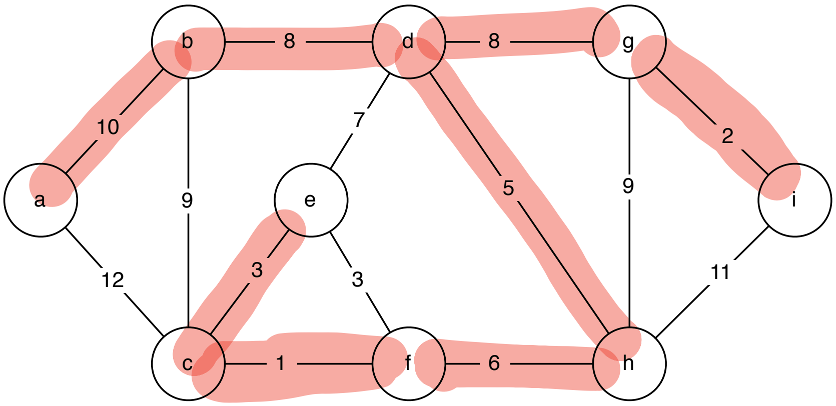
repeat for v-1 times:

    add lightest edge that does not create a cycle

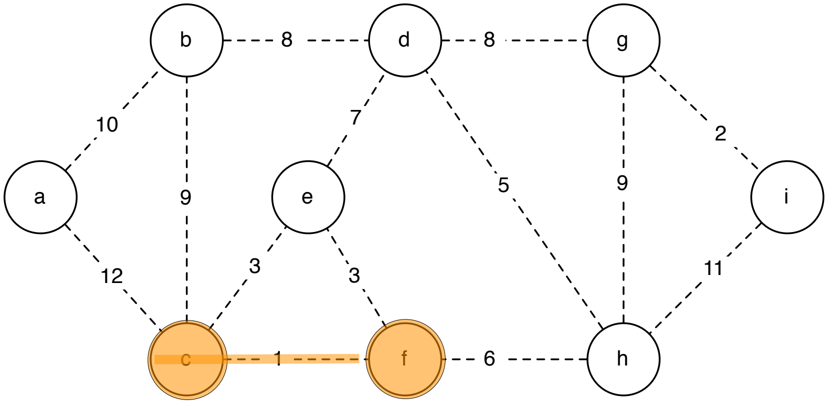
example

8 edge

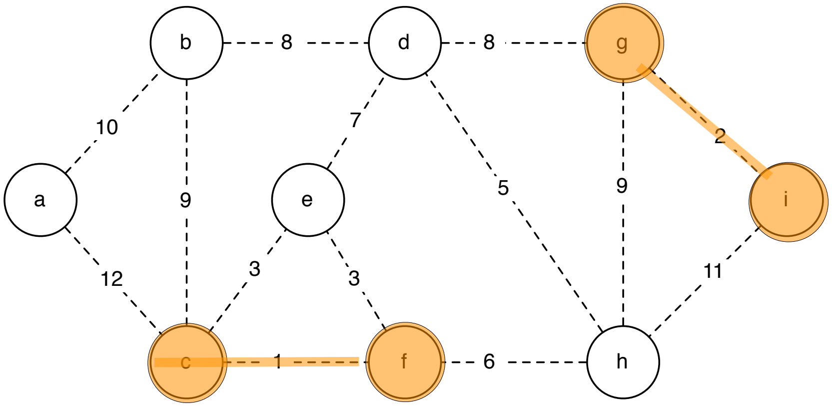
Solution.



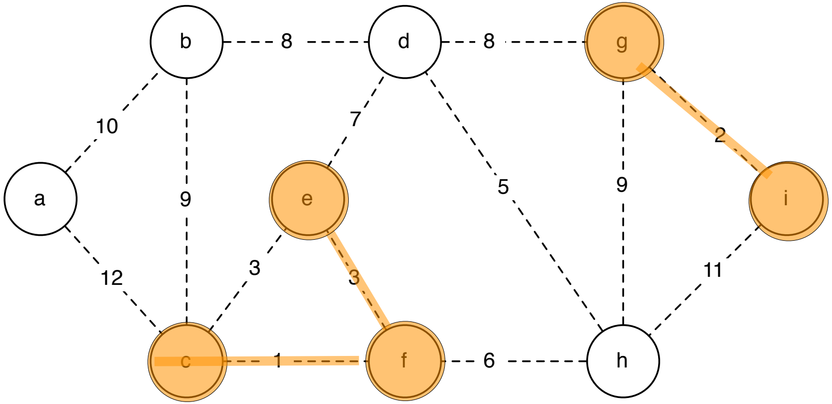
Kruskal



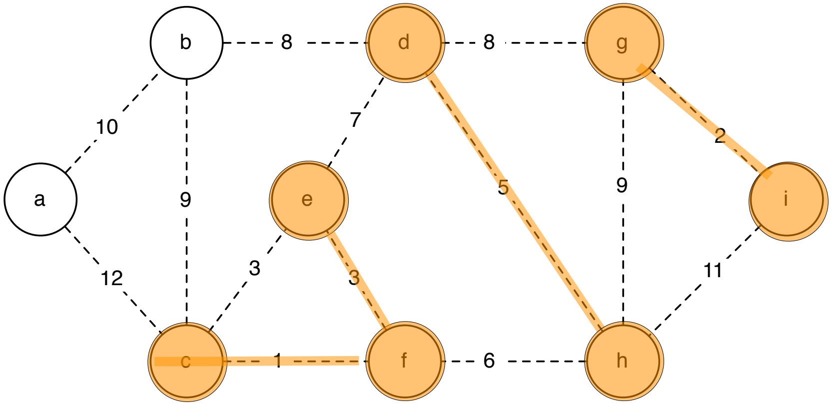
Kruskal



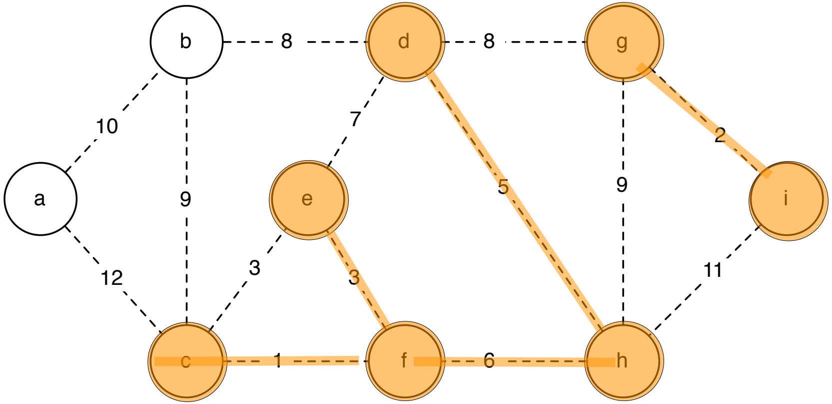
Kruskal



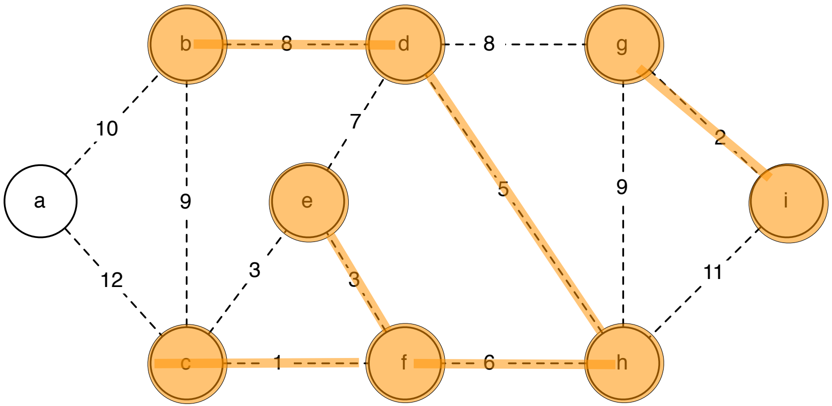
Kruskal



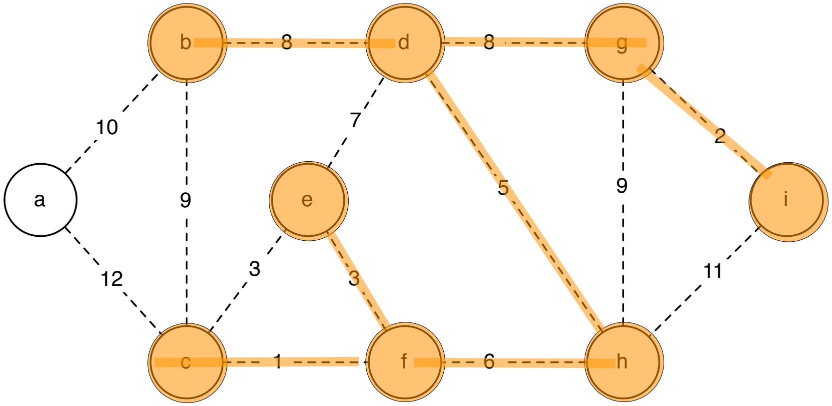
Kruskal



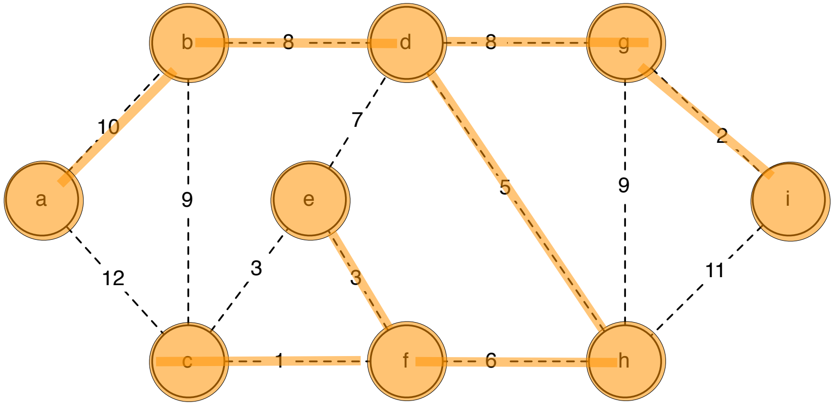
Kruskal



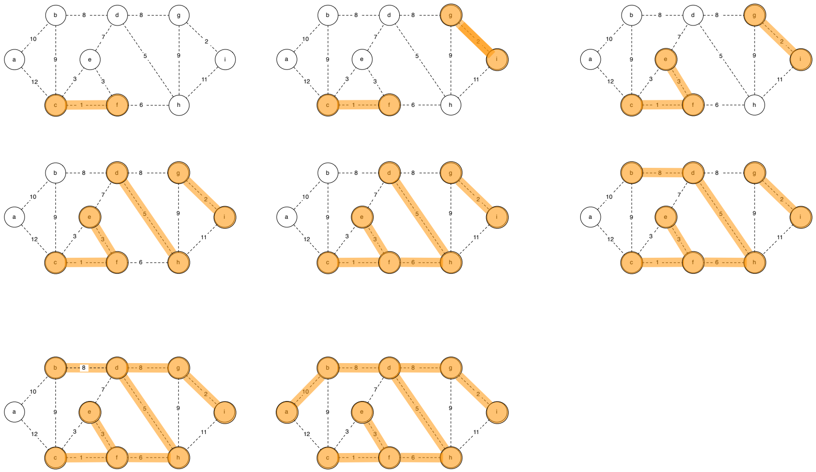
Kruskal



Kruskal



Kruskal



why does this work?

$T \leftarrow \emptyset$
repeat $V - 1$ times:
  add to T the lightest edge $e \in E$ that does not create a cycle

definition: cut (graph cut)

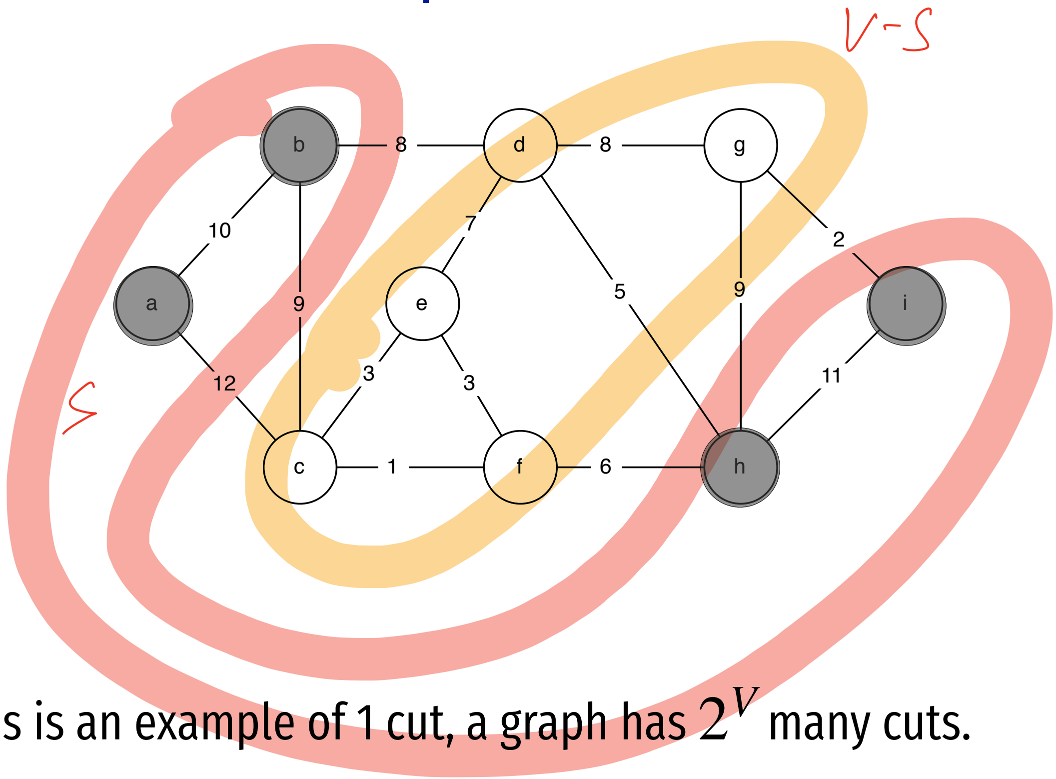
A cut is a partition of the vertices into 2 sets.

e.g.: $(S, V-S)$

definition: cut

A cut is a partition of V into two sets.

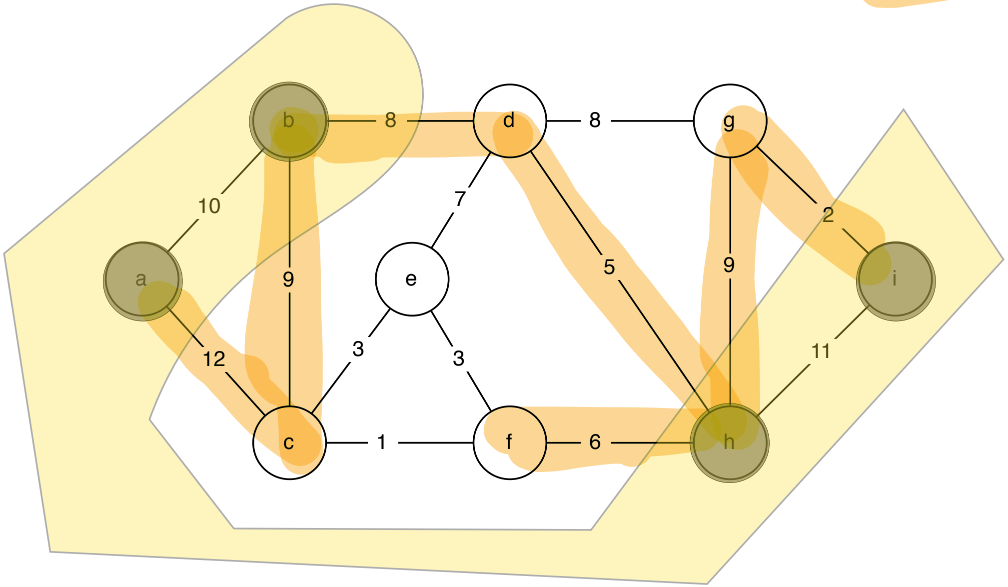
example of a cut



This is an example of 1 cut, a graph has 2^V many cuts.

example of a cut

edges which cross this cut.



This is an example of 1 cut, a graph has 2^V many cuts.

definition: crossing a cut

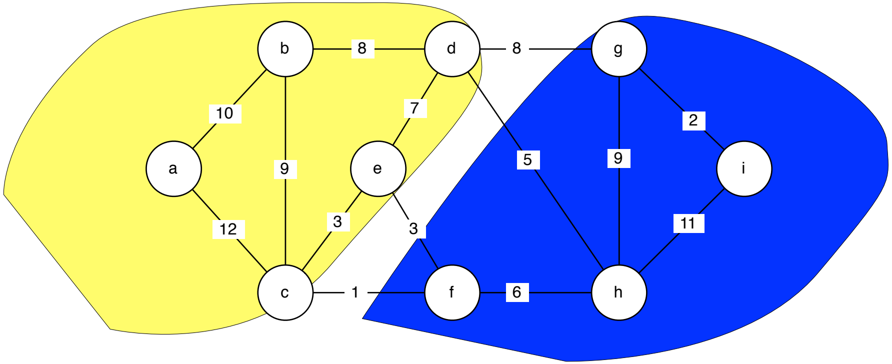
$x, y \in V$

A edge $e = (x, y)$ crosses a cut $(S, V - S)$ if $x \in S$ and $y \in V - S$.

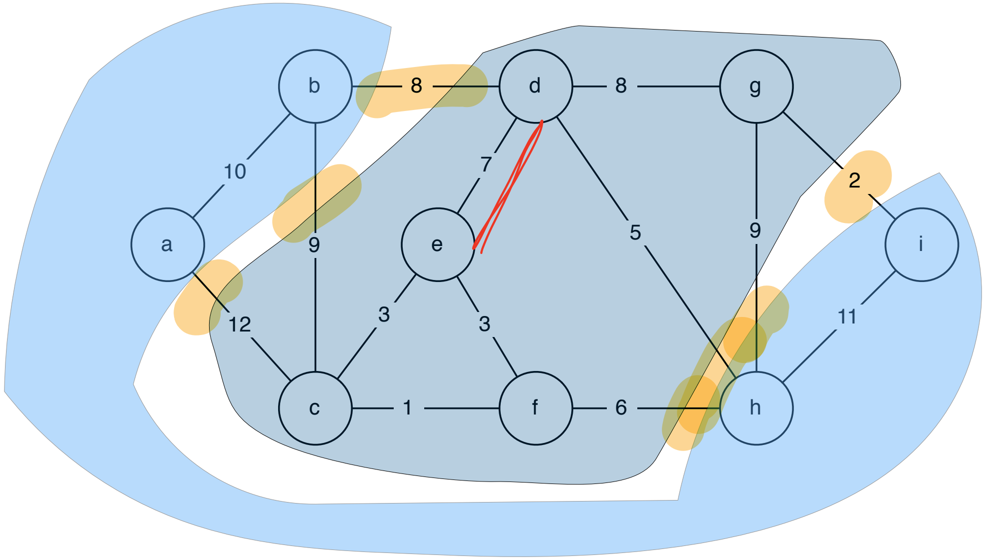
definition: crossing a cut

an edge $e = (u, v)$ crosses a graph cut (S,V-S) if

$u \in S \qquad v \in V - S$



example of a crossing



(a, b)

(h, i)

Respect this cut.

Edge (b, d) crosses the cut $\{a, b, h, i\}, \{c, d, e, f, g\}$.

definition: respect

A set of edges A respects a cut S if no edge in A crosses the cut.

Cut theorem

Thm: Suppose that the set of edges A is part of some MST of $G=(V,E)$.

Let $(S, V-S)$ be any cut that A respects.

Let e be the min. cost edge that crosses $(S, V-S)$.

Then: $A \cup \{e\}$ is then part of some MST.

Cut theorem

Suppose the set of edges A is part of an m.s.t.

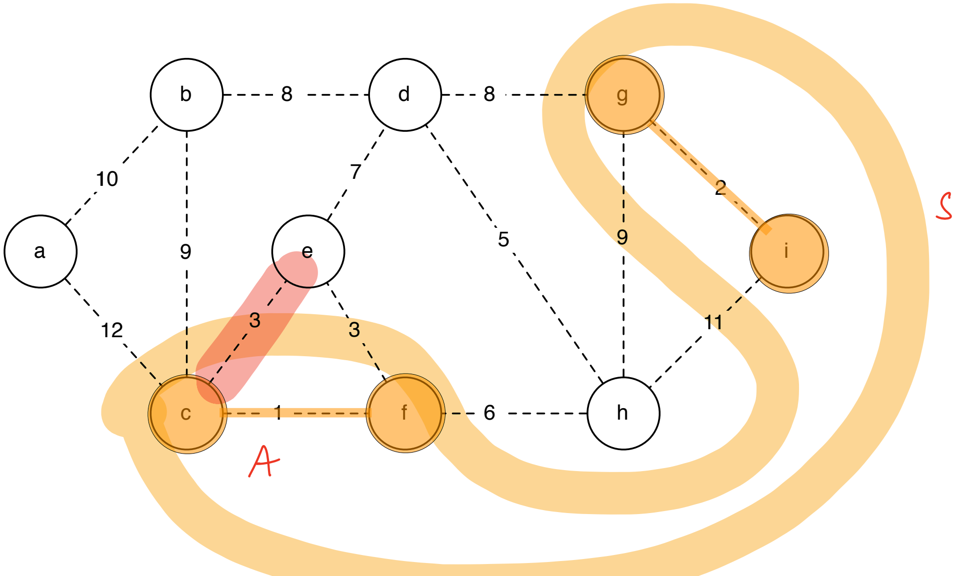
Let $(S, V-S)$ be any cut that A respects .

Let edge e be the min-weight edge across $(S, V-S)$

Then: $A \cup \{e\}$ is part of an m.s.t.

example of theorem

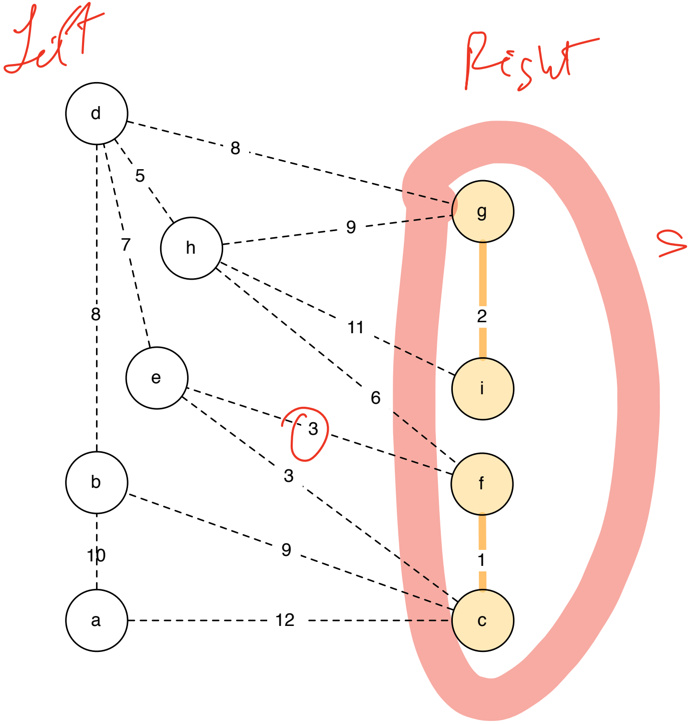
$A = \{(c,f)$
$(g,i)\}$



Consider these two edges as part of some MST.

We can redraw the graph to identify this cut.

The min cost edge that crosses this cut is part of some MST.



proof of cut theorem

Theorem 2 *Suppose the set of edges A is part of a minimum spanning tree of $G = (V, E)$. Let $(S, V - S)$ be any cut that respects A and let e be the edge with the minimum weight that crosses $(S, V - S)$. Then the set $A \cup \{e\}$ is part of a minimum spanning tree.*

Proof: Let $e = (u, v)$. Let T be some MST of G.

If $A \cup \{e\}$ is already part of T, then our thm holds.

$T \not\ni \{e\}$

If $A \cup \{e\}$ is not part of T, we will construct a new T' which contains $A \cup \{e\}$ and is also an MST.

Step 1: Add e to T. What is the result?? This creates a cycle from $u \to v \to \cdots \to u$.

$w(e') \geq w(e)$

Let $e' \neq e$ be the edge w/ highest cost on this cycle.

Now consider $T' = (T - \{e'\}) \cup \{e\}$. $w(T') \leq w(T)$ $\square$.

proof of cut theorem

Theorem 2 *Suppose the set of edges* A *is part of a minimum spanning tree of* $G = (V, E)$*. Let* $(S, V - S)$ *be any cut that respects* A *and let* e *be the edge with the minimum weight that crosses* $(S, V - S)$*. Then the set* $A \cup \{e\}$ *is part of a minimum spanning tree.*

Let $e = (u, v)$.

If $A \cup \{e\}$ is already in T then theorem follows.

Suppose that $A \cup \{e\}$ is not part of T.

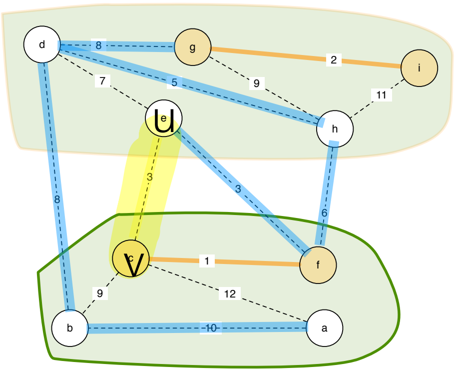
Add e to the tree T.

This creates a cycle. Let e' be another edge on this cycle.

Now consider $T' = T - \{e'\} \cup \{e\}$.

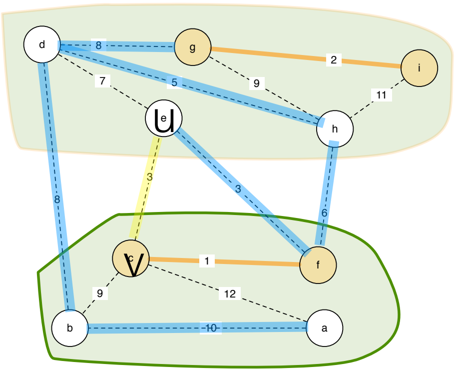
The weight $w(T') \le w(T)$ since e has min weight.

proof of cut thm



As an example, the set A is in orange.
The edge e is yellow and T is blue.
We will construct a T' which includes A+e.

proof of cut thm



As an example, the set A is in orange.
The edge e is yellow and T is blue.
We will construct a T’ which includes A+e.

Add e to T, which creates a cycle.

Let e’ be the first edge on the cycle that crosses the cut.

correctness

Kruskal-pseudocode(G)

```
1  A ← ∅
2  repeat  V − 1 times:
3          add to A the lightest edge e ∈ E that does not create a cycle
```

Thm: Kruskal's algorithm terminates with A as a MST of G.

Proof: By INDUCTION. In step ①, A is part of some MST.

Our argument will maintain the invariant that A is part of an MST.

IH: Suppose that after k steps of loop 2, A is part of an MST.

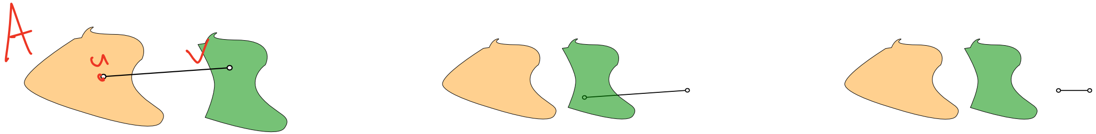
We plan to argue that at the end of the $k+1$st iteration of loop 2, A is part of some MST.

correctness

```
KRUSKAL-PSEUDOCODE(G)
1  A ← ∅
2  repeat  V − 1 times:
3          add to A the lightest edge e ∈ E that does not create a cycle
```

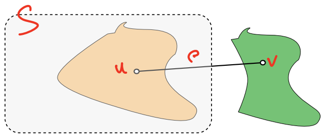
Proof: By induction. in step 1, A is part of some MST.
Suppose that after k steps, A is part of some MST (line 2).
In line 3, we add an edge e=(u,v).

Because e does not create a cycle, there are 3 cases to consider:



3 cases for edge e.

Case 1: e=(u,v) and both u,v are in A. Recall, e is the lightest edge that doesn't create a cycle.



We need to identify a cut (S, V-S) that A respects and that e crosses

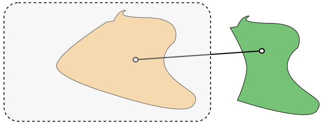
S is the "component" of A that contains just u

all of the nodes one can reach by starting at u and only using edges in A

$\Rightarrow$ A respects S because No edge in A crosses S.

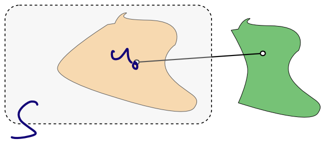
$\Rightarrow$ e crosses S, & e is the lightest edge that does so.

3 cases for edge e.
Case 1: e=(u,v) and both u,v are in A.



In this case, set S to be the component that contains {u}

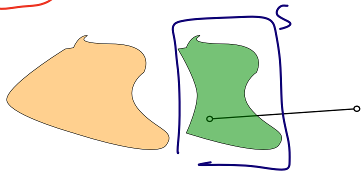
3 cases for edge e.
Case 1: e=(u,v) and both u,v are in A.



In this case, set S to be the component that contains {u}
The edge e crosses this cut and A respects S.
By the cut theorem, A+e belongs to an MST.

3 cases for edge e.

Case 2: e=(u,v) and only u is in A.

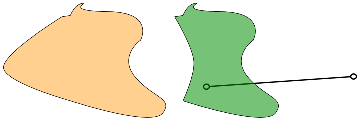


How do you set ?? Set S to be the component that contains {u}.

The edge e crosses this cut and A respects S.

Theorem 2 *Suppose the set of edges A is part of a minimum spanning tree of $G = (V, E)$. Let $(S, V - S)$ be any cut that respects A and let e be the edge with the minimum weight that crosses $(S, V - S)$. Then the set $A \cup \{e\}$ is part of a minimum spanning tree.*

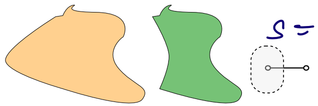
3 cases for edge e.
Case 2: e=(u,v) and only u is in A.



In this case, set S to be the component that contains {u}

The edge e crosses this cut and A respects S.

3 cases for edge e.
Case 3: e=(u,v) and neither u nor v are in A.



$S = \{u\}$

In this case, set S to be the component that contains {u}

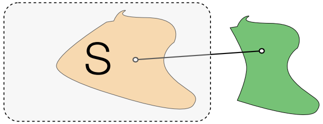
Notice A respects S. e is the lightest edge that crosses S. By the exchange lemma,

$A \cup \{e\}$ is part of an MST.

```
KRUSKAL-PSEUDOCODE(G)
1  A ← ∅
2  repeat  V − 1 times:
3          add to A the lightest edge e ∈ E that does not create a cycle
```

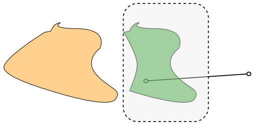
Theorem 3 *The Kruskal algorithm outputs a minimum spanning tree.*

Proof. By Induction. At the first step, A is a set of edges that is part of a minimum spanning tree of G. Suppose this is true by induction for the first i loops of the algorithm.

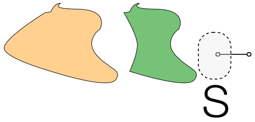


Consider the $i+1^{\text{th}}$ iteration and let $e = (u, v)$ be the edge added to A in line 2. By construction, e is the lightest edge in E that does not create a cycle in A.

Since e does not create a cycle in A, e must either connect two connected components of A, extend one connected component of A or connect two nodes that are not covered by A. In the first two cases, let A_1 be the connected component in A that covers u. In the third case, let $A_1 = \{u\}$.



Consider the graph cut $(A_1, V - A_1)$. By selection, e is the lightest edge that crosses this cut: all other edges are either heavier, or they create a cycle in A and therefore do not cross the cut since they connect nodes in A_1 or in $V - A_1$. Thus, by the previous theorem, $A \cup \{e\}$ must be part of a minimum spanning tree.



During each iteration, line 3 always succeeds. This follows because A is part of some MST by hypothesis. At the end of the loop, $|A| = V - 1$. Therefore, A must be the full spanning tree since it has the correct size. □

14 sentences.

analysis?

KRUSKAL-PSEUDOCODE(G)

```
1  A ← ∅
2  repeat V − 1 times:
3      add to A the lightest edge e ∈ E that does not create a cycle
```

Running time

how to identify this property

UNION-FIND data structure.

GENERAL-MST-STRATEGY$(G = (V, E))$

1 $A \leftarrow \emptyset$
2 **repeat** $V - 1$ times:
3 Pick a cut $(S, V - S)$ that respects A
4 Let e be min-weight edge over cut $(S, V - S)$
5 $A \leftarrow A \cup \{e\}$

In line w/ the cut theorem.

In fact, this approach can be generalized into a family of algorithms.

Keep A a tree throughout the algorithm.

Keep A = S.

Prim's algorithm

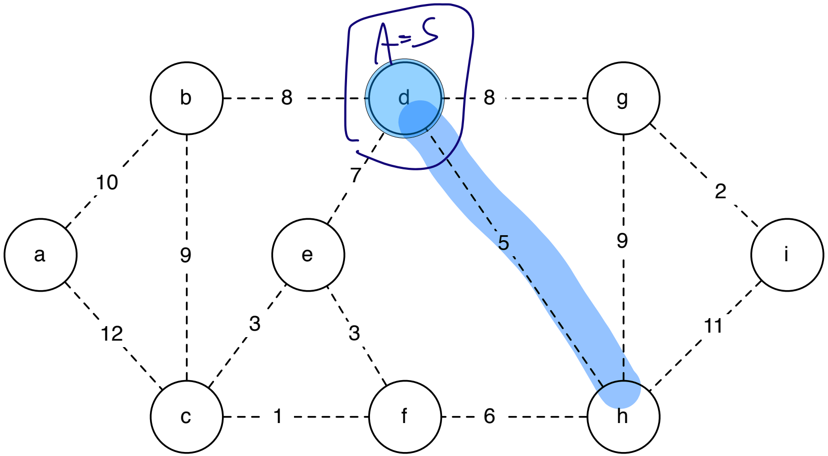
GENERAL-MST-STRATEGY$(G = (V, E))$

$A \leftarrow \emptyset$
repeat $V - 1$ times:
  Pick a cut $(S, V - S)$ that respects A
  Let e be min-weight edge over cut $(S, V - S)$
  $A \leftarrow A \cup \{e\}$

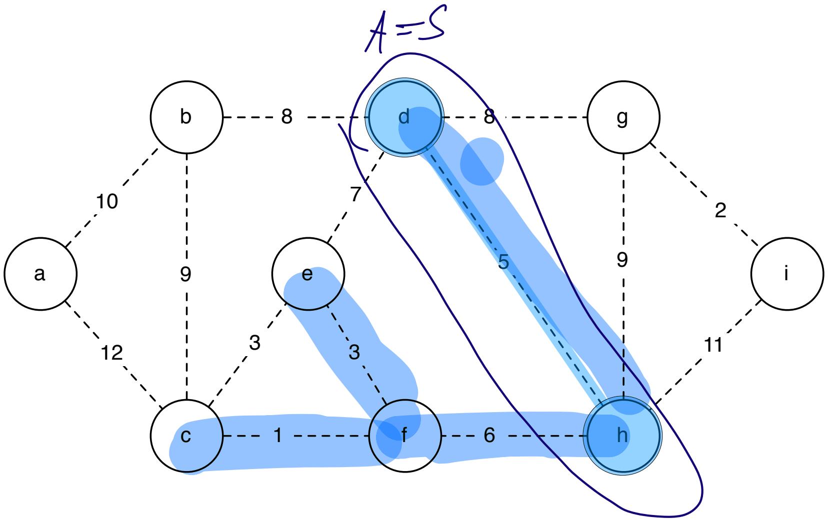
A is a subtree

edge e is lightest edge that grows the subtree

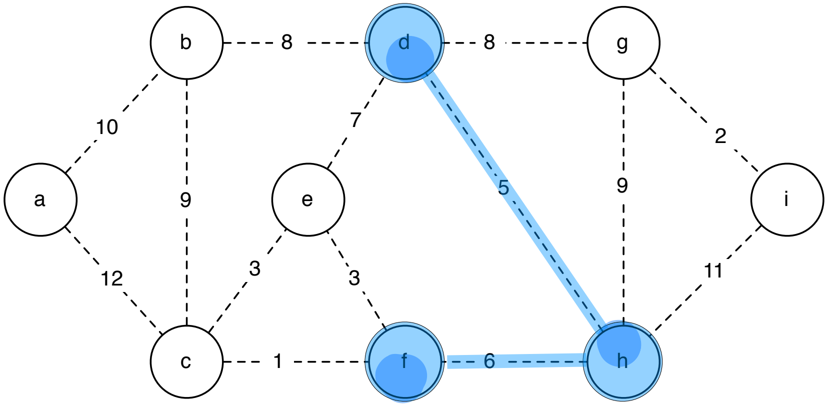
prim



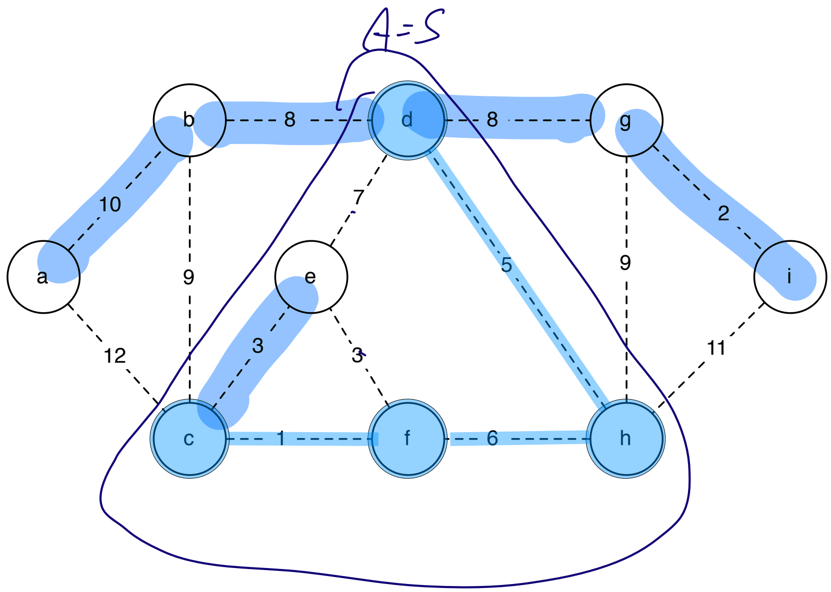
prim



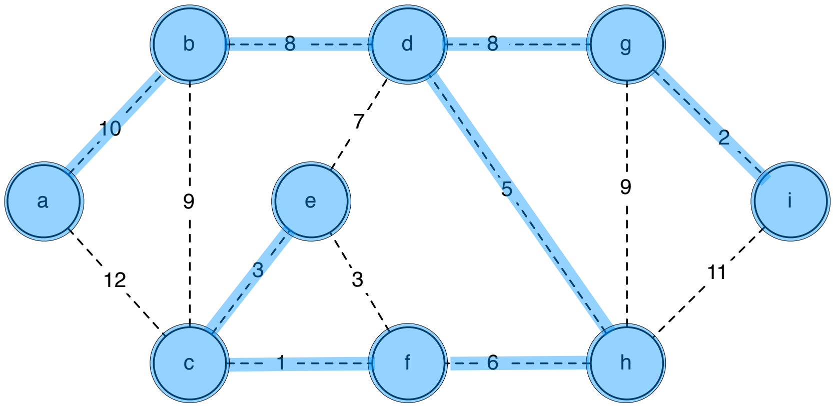
prim



prim



prim



implementation

idea: Maintain the set A. Update neighbors of A with weights.

Use a new data structure, priority queue to track light edges.

new data structure

A priority queue is a data structure with 3 operations:

Make: Initializes a queue with some elements

ExtractMin: Returns the lightest entry in the queue.

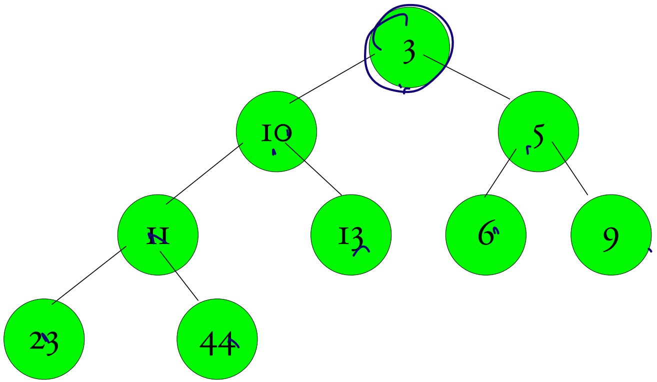
DecreaseKey: decreases the value of some entry in the queue.

binary heap

full tree, key value <= to key of children

binary heap

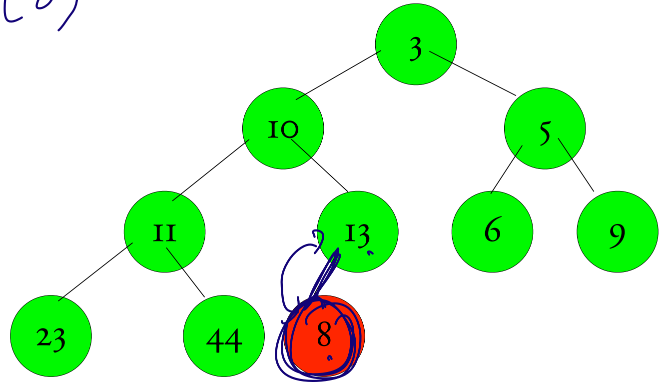
full tree, key value <= to key of children



binary heap

full tree, key value <= to key of children

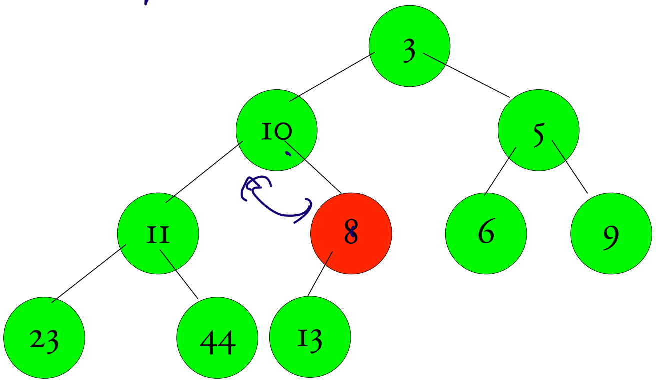
Add an entry. (8)



binary heap

full tree, key value <= to key of children

Add (log n) time (swaps)

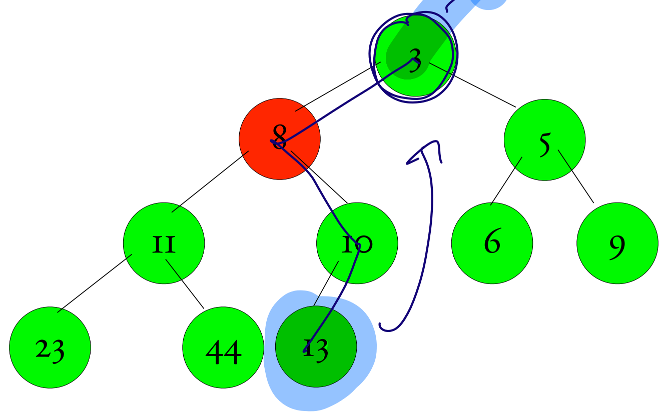


binary heap

full tree, key value <= to key of children

how to extractmin?

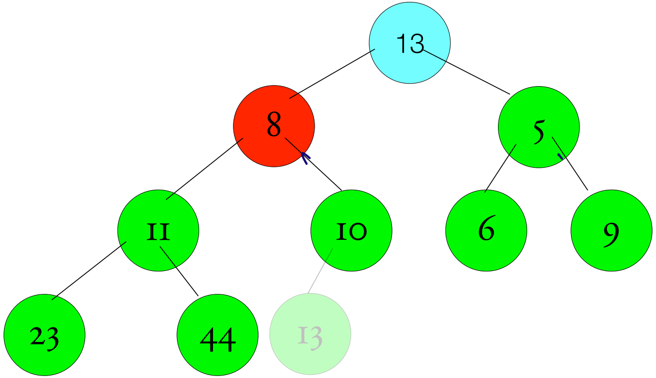
root is always the min.



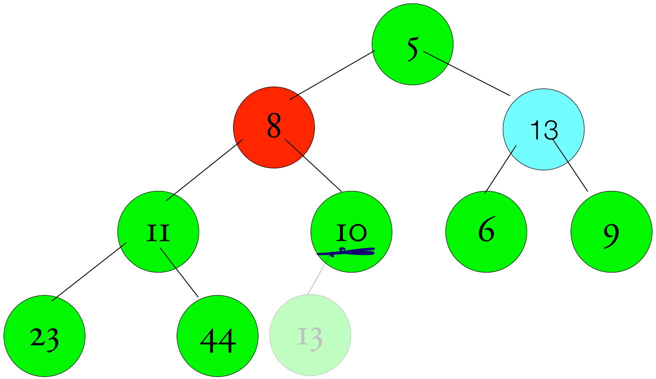
binary heap

full tree, key value <= to key of children

how to extractmin?



binary heap

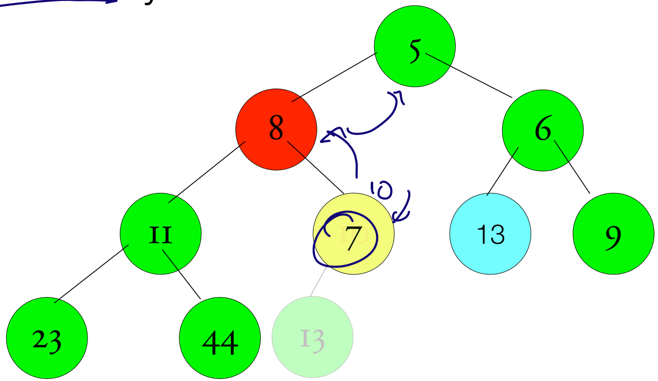


binary heap

full tree, key value <= to key of children

O(log n) how to extractmin? remove root. replace w/ the last child. Swap down.

how to decreasekey?

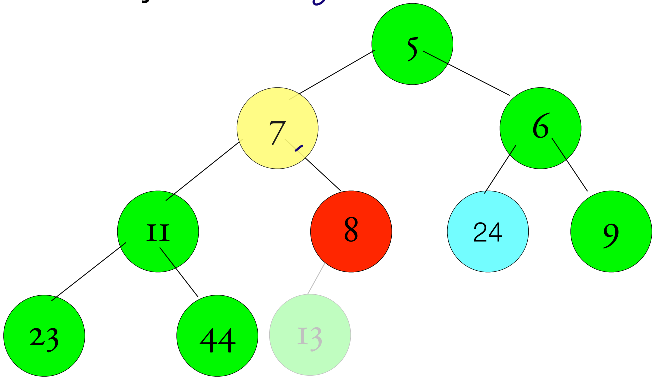


binary heap

full tree, key value <= to key of children

how to extractmin?

O(log n) how to decreasekey? decrease & swap up.



EACH operation on a priority queue/heap take O(log n)

implementation

use a priority queue to keep track of light edges

insert:

makequeue: $O(\log n)$

extractmin:

decreasekey:

Prim’s algorithm

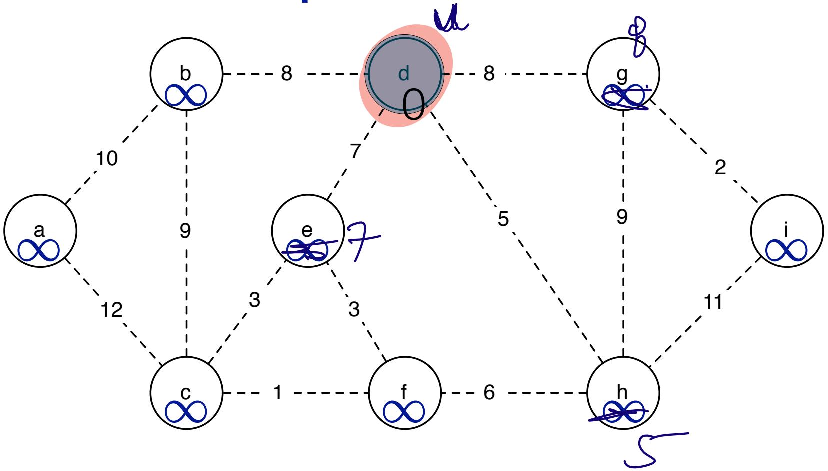
implementation

PRIM$(G = (V, E))$

$Q \leftarrow \emptyset$ $\quad \triangleright$ Q is a Priority Queue
Initialize each $v \in V$ with key $k_v \leftarrow \infty$, $\pi_v \leftarrow$ NIL
Pick a starting node r and set $k_r \leftarrow 0$
Insert all nodes into Q with key k_v.
while $Q \neq \emptyset$
    do $u \leftarrow$ EXTRACT-MIN(Q)
        for each $v \in Adj(u)$
            do if $v \in Q$ and $w(u, v) < k_v$
                then $\pi_v \leftarrow u$
                    DECREASE-KEY$(Q, v, w(u, v))$ $\quad \triangleright$ Sets $k_v \leftarrow w(u, v)$

$V \cdot \log(V)$ steps.

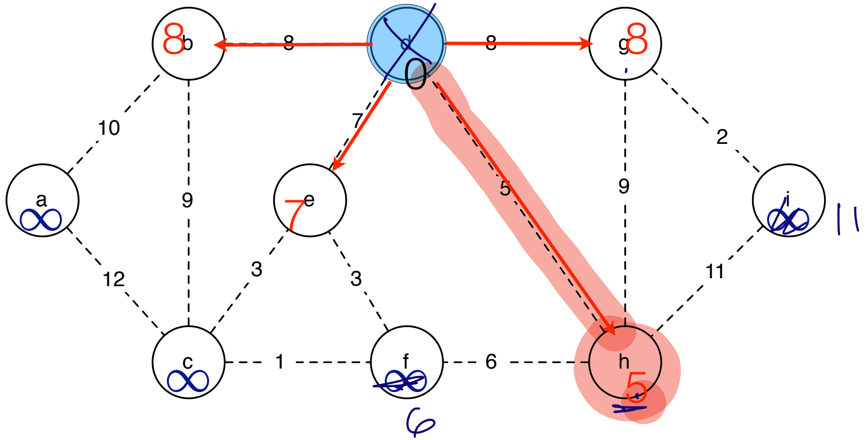
prim



PRIM$(G = (V, E))$

```
Q ← ∅        ▷ Q is a Priority Queue
Initialize each v ∈ V with key k_v ← ∞, π_v ← NIL
Pick a starting node r and set k_r ← 0
Insert all nodes into Q with key k_v.
while Q ≠ ∅
      do u ← EXTRACT-MIN(Q)
         for each v ∈ Adj(u)
             do if v ∈ Q and w(u, v) < k_v
                  then π_v ← u
                       DECREASE-KEY(Q, v, w(u, v))    ▷ Sets k_v ← w(u, v)
```

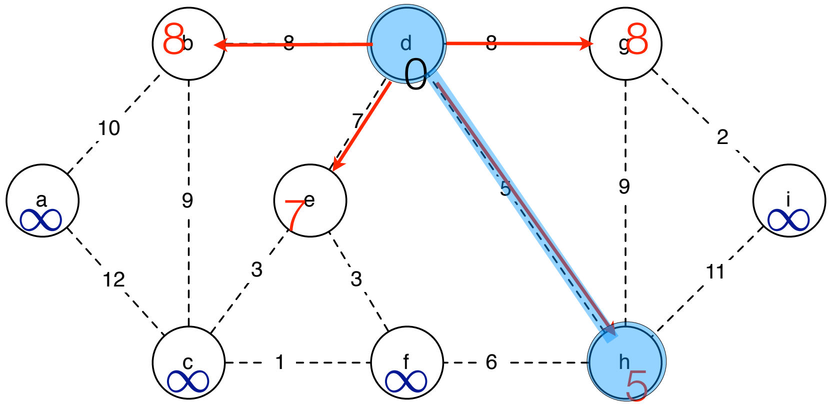
prim



PRIM$(G = (V, E))$

$Q \leftarrow \emptyset$ $\triangleright$ Q is a Priority Queue
Initialize each $v \in V$ with key $k_v \leftarrow \infty$, $\pi_v \leftarrow$ NIL
Pick a starting node r and set $k_r \leftarrow 0$
Insert all nodes into Q with key k_v.
while $Q \neq \emptyset$
  do $u \leftarrow$ EXTRACT-MIN(Q)
    for each $v \in Adj(u)$
      do if $v \in Q$ and $w(u, v) < k_v$
        then $\boxed{\pi_v \leftarrow u}$
          DECREASE-KEY$(Q, v, w(u, v))$ $\triangleright$ Sets $k_v \leftarrow w(u, v)$

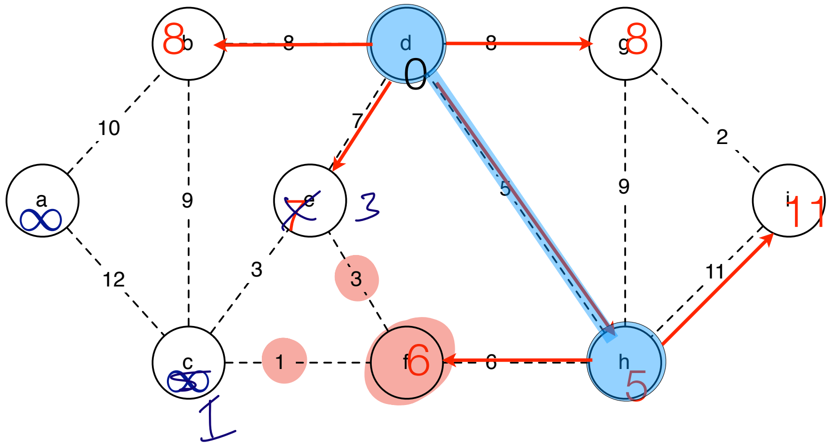
prim



PRIM$(G = (V, E))$

$Q \leftarrow \emptyset$ $\triangleright$ Q is a Priority Queue
Initialize each $v \in V$ with key $k_v \leftarrow \infty$, $\pi_v \leftarrow$ NIL
Pick a starting node r and set $k_r \leftarrow 0$
Insert all nodes into Q with key k_v.
while $Q \neq \emptyset$
  do $u \leftarrow$ EXTRACT-MIN(Q)
    for each $v \in Adj(u)$
      do if $v \in Q$ and $w(u, v) < k_v$
        then $\pi_v \leftarrow u$
          DECREASE-KEY$(Q, v, w(u, v))$ $\triangleright$ Sets $k_v \leftarrow w(u, v)$

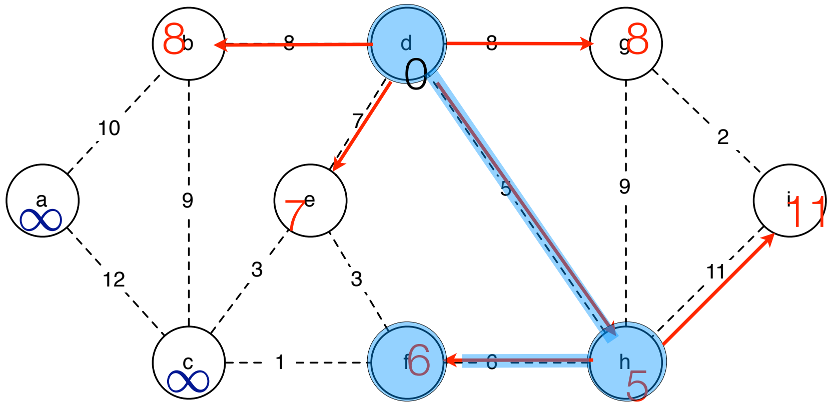
prim



PRIM$(G = (V, E))$

$Q \leftarrow \emptyset$ $\quad \triangleright$ Q is a Priority Queue
Initialize each $v \in V$ with key $k_v \leftarrow \infty$, $\pi_v \leftarrow$ NIL
Pick a starting node r and set $k_r \leftarrow 0$
Insert all nodes into Q with key k_v.
while $Q \neq \emptyset$
  do $u \leftarrow$ EXTRACT-MIN(Q)
    for each $v \in Adj(u)$
      do if $v \in Q$ and $w(u, v) < k_v$
        then $\pi_v \leftarrow u$
          DECREASE-KEY$(Q, v, w(u, v))$ $\quad \triangleright$ Sets $k_v \leftarrow w(u, v)$

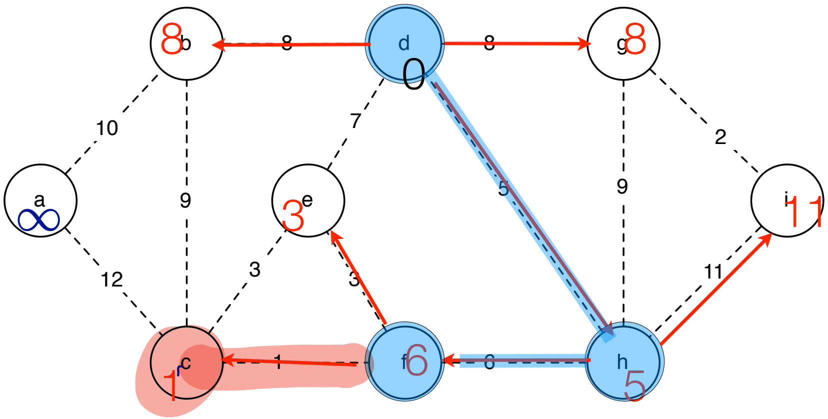
prim



$\textsc{Prim}(G = (V, E))$

$Q \leftarrow \emptyset$ $\quad \triangleright$ Q is a Priority Queue
Initialize each $v \in V$ with key $k_v \leftarrow \infty$, $\pi_v \leftarrow \textsc{nil}$
Pick a starting node r and set $k_r \leftarrow 0$
Insert all nodes into Q with key k_v.
while $Q \neq \emptyset$
  do $u \leftarrow \textsc{Extract-Min}(Q)$
    for each $v \in Adj(u)$
      do if $v \in Q$ and $w(u, v) < k_v$
        then $\pi_v \leftarrow u$
          $\textsc{Decrease-Key}(Q, v, w(u, v))$ $\quad \triangleright$ Sets $k_v \leftarrow w(u, v)$

prim



$\text{PRIM}(G = (V, E))$

$Q \leftarrow \emptyset$ $\quad \triangleright$ Q is a Priority Queue
Initialize each $v \in V$ with key $k_v \leftarrow \infty$, $\pi_v \leftarrow$ NIL
Pick a starting node r and set $k_r \leftarrow 0$
Insert all nodes into Q with key k_v.
while $Q \neq \emptyset$
  do $u \leftarrow$ EXTRACT-MIN(Q)
    for each $v \in Adj(u)$
      do if $v \in Q$ and $w(u, v) < k_v$
        then $\pi_v \leftarrow u$
          DECREASE-KEY$(Q, v, w(u, v))$ $\quad \triangleright$ Sets $k_v \leftarrow w(u, v)$

running time

$\Theta(E \cdot \log V)$

PRIM$(G = (V, E))$

$Q \leftarrow \emptyset$ $\quad \triangleright$ Q is a Priority Queue
Initialize each $v \in V$ with key $k_v \leftarrow \infty$, $\pi_v \leftarrow$ NIL
Pick a starting node r and set $k_r \leftarrow 0$
Insert all nodes into Q with key k_v.
while $Q \neq \emptyset$
  do $u \leftarrow$ EXTRACT-MIN(Q)
    for each $v \in Adj(u)$
      do if $v \in Q$ and $w(u, v) < k_v$
        then $\pi_v \leftarrow u$
          DECREASE-KEY$(Q, v, w(u, v))$ $\quad \triangleright$ Sets $k_v \leftarrow w(u, v)$

V log V

V·log V time in total
b/c V iterations (call
per iteration

overall, it will result in E log V

why?? b/c decrease-key can be called
once per edge at most

implementation

```
PRIM(G = (V, E))
Q ← ∅        ▷ Q is a Priority Queue
Initialize each v ∈ V with key k_v ← ∞, π_v ← NIL
Pick a starting node r and set k_r ← 0
Insert all nodes into Q with key k_v.
while Q ≠ ∅
    do u ← EXTRACT-MIN(Q)
        for each v ∈ Adj(u)
            do if v ∈ Q and w(u, v) < k_v
                then π_v ← u
                    DECREASE-KEY(Q, v, w(u, v))    ▷ Sets k_v ← w(u, v)
```

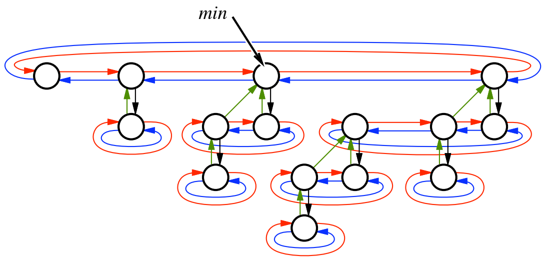
$$O(V \log V + E \log V) = O(E \log V)$$

implementation

use a priority queue to keep track of light edges

	priority queue	fibonacci heap	
insert:	O(log n)	log n	
makequeue:	n	n	
extractmin:	O(log n)	log n	amortized
decreasekey:	O(log n)	O(1)	amortized

fibonacci heap



faster implementation

PRIM($G = (V, E)$)

$Q \leftarrow \emptyset$ $\qquad \triangleright$ Q is a Priority Queue
Initialize each $v \in V$ with key $k_v \leftarrow \infty$, $\pi_v \leftarrow$ NIL
Pick a starting node r and set $k_r \leftarrow 0$
Insert all nodes into Q with key k_v.
while $Q \neq \emptyset$
  do $u \leftarrow$ EXTRACT-MIN(Q)
    for each $v \in Adj(u)$
      do if $v \in Q$ and $w(u, v) < k_v$
        then $\pi_v \leftarrow u$
          DECREASE-KEY($Q, v, w(u, v)$) $\qquad \triangleright$ Sets $k_v \leftarrow w(u, v)$

$$O(E + V \log V)$$

Research in mst

FREDMAN-TARJAN 84:	$E + V \log V$
GABOW-GALIL-SPENCER-TARJAN 86:	$E \log(\log^* V)$
CHAZELLE 97	$E\alpha(V) \log \alpha(V)$
CHAZELLE 00	$E\alpha(V)$
PETTIE-RAMACHANDRAN 02:	(optimal)
KARGER-KLEIN-TARJAN 95: (randomized)	E
Euclidean mst:	$V \log V$

Ackerman function

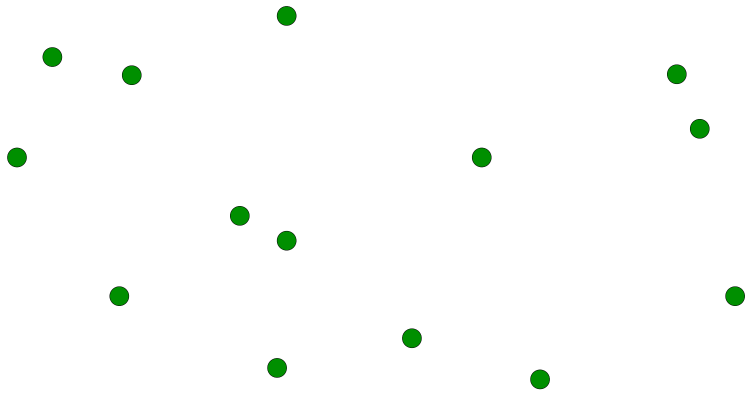
$$A(m,n) = \begin{cases} n+1 & m = 0 \\ A(m-1,1) & m > 0,\ n = 0 \\ A(m-1, A(m,n-1)) & m,n > 0 \end{cases}$$

$$A(4,2) =$$

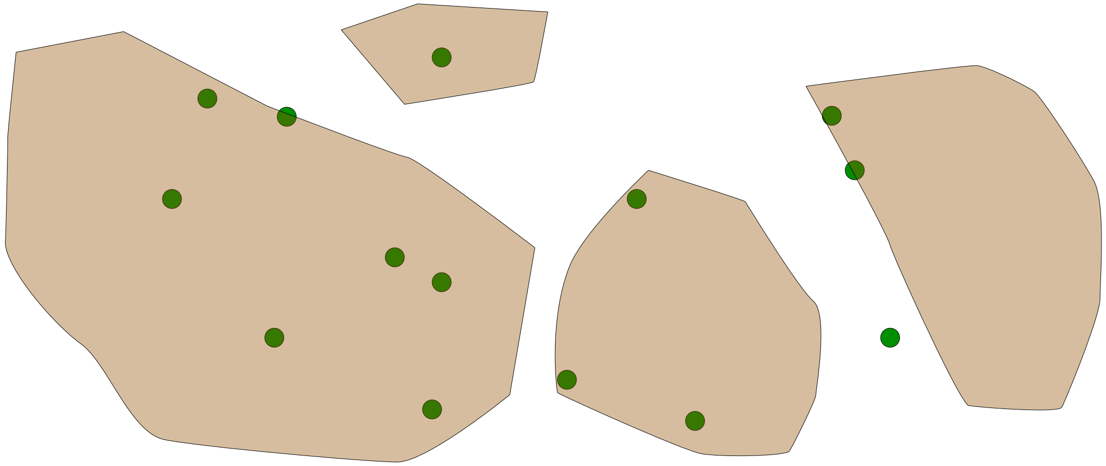
inverse ackerman

$$\alpha(n) =$$

application of mst



application of mst



Use Kruskal's algorithm to perform k-clustering.

application of mst