L17

4102

3.22.2016

abhi shelat

MST question

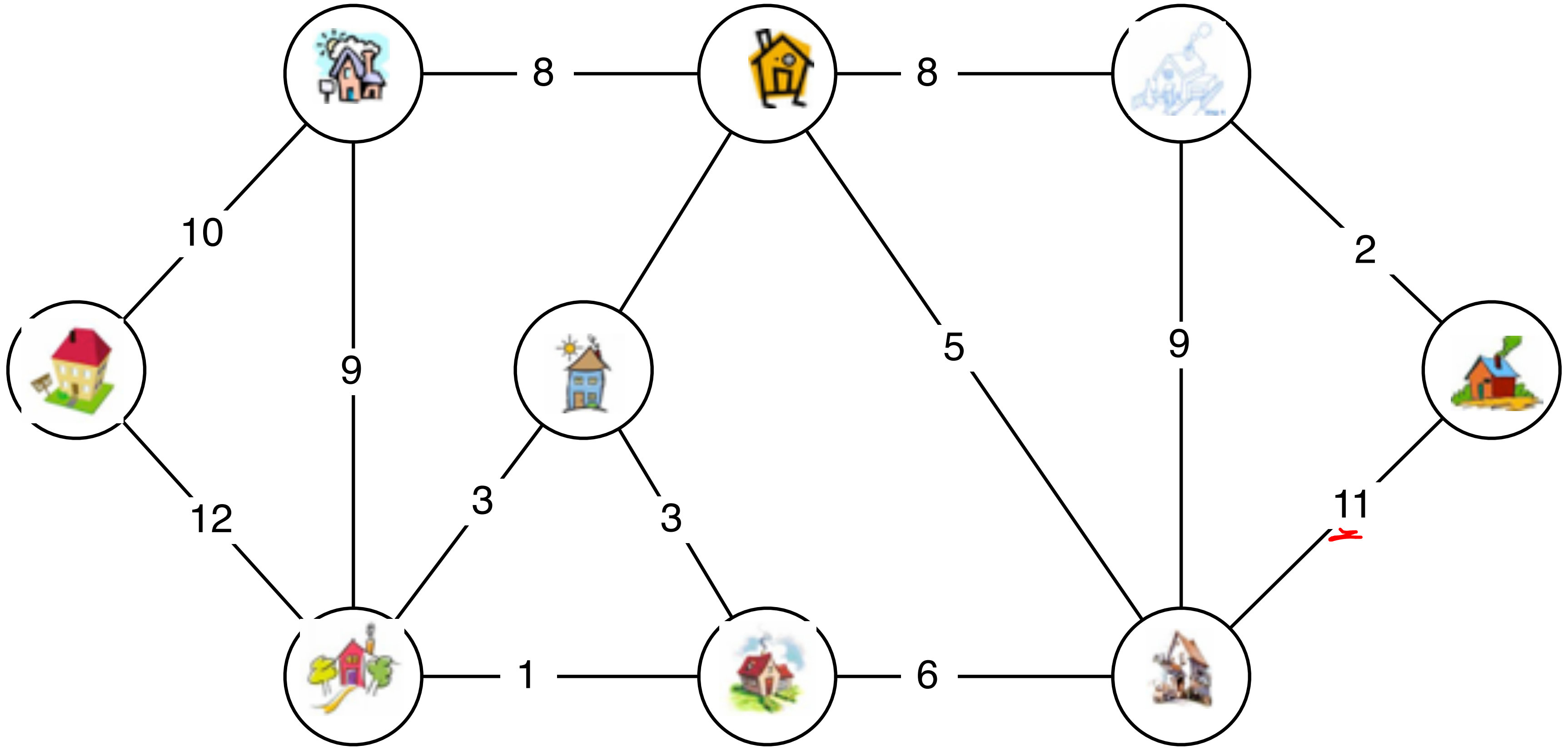
what is a graph cut?

what does it mean for a set A to respect a cut S?

what does the cut theorem say?

userid:

G=(V,E,w)



we want:

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

(a) connects all vertices

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

minimum spanning tree

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

(a) connects all vertices

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

minimum spanning tree

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

(a) connects all vertices

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

how many edges does solution have ? $V-1$

does solution have a cycle? No cycle

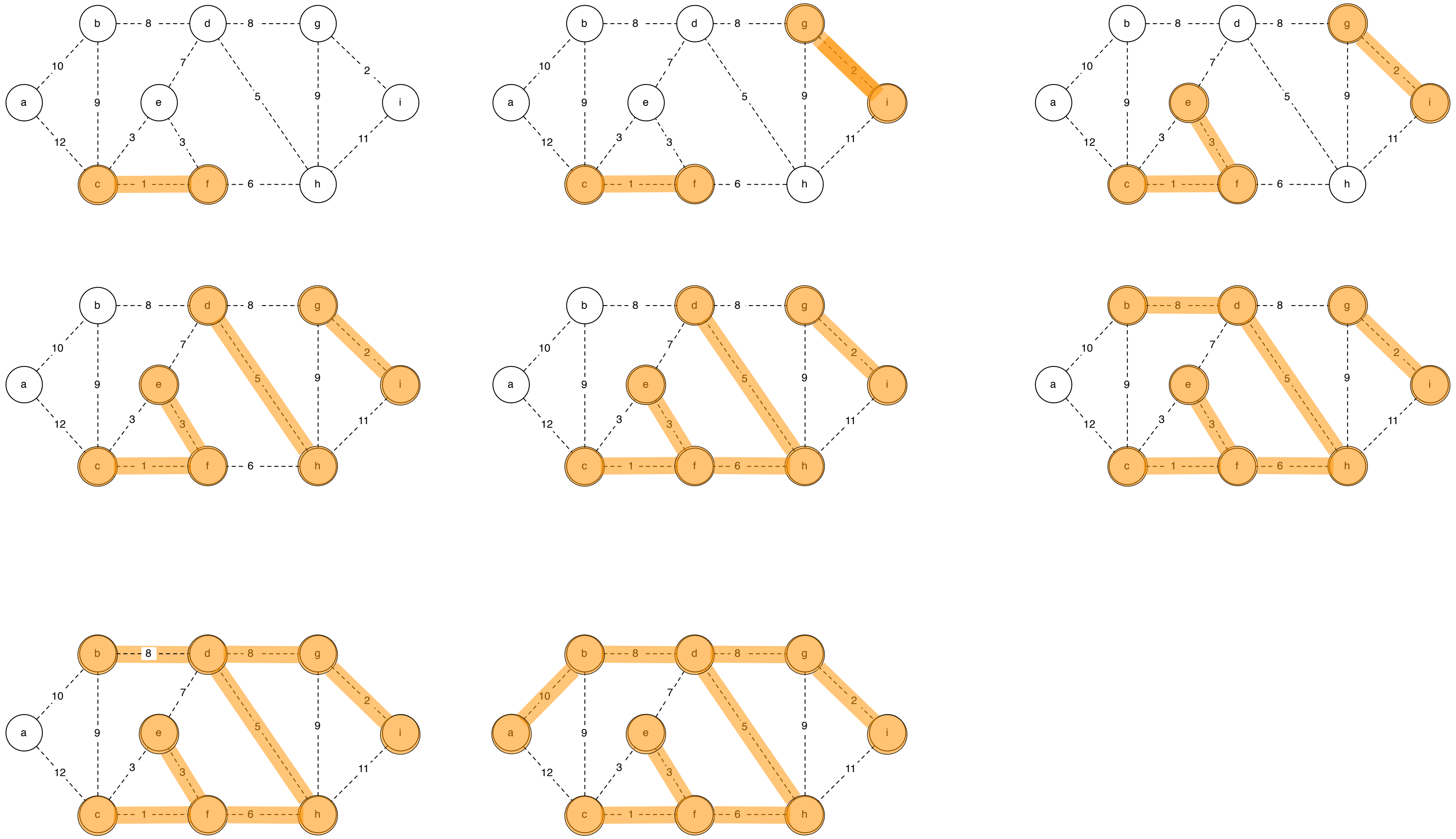
strategy

start with an empty set of edges A

repeat for v-1 times:

    add lightest edge that does not create a cycle

Kruskal’s algorithm



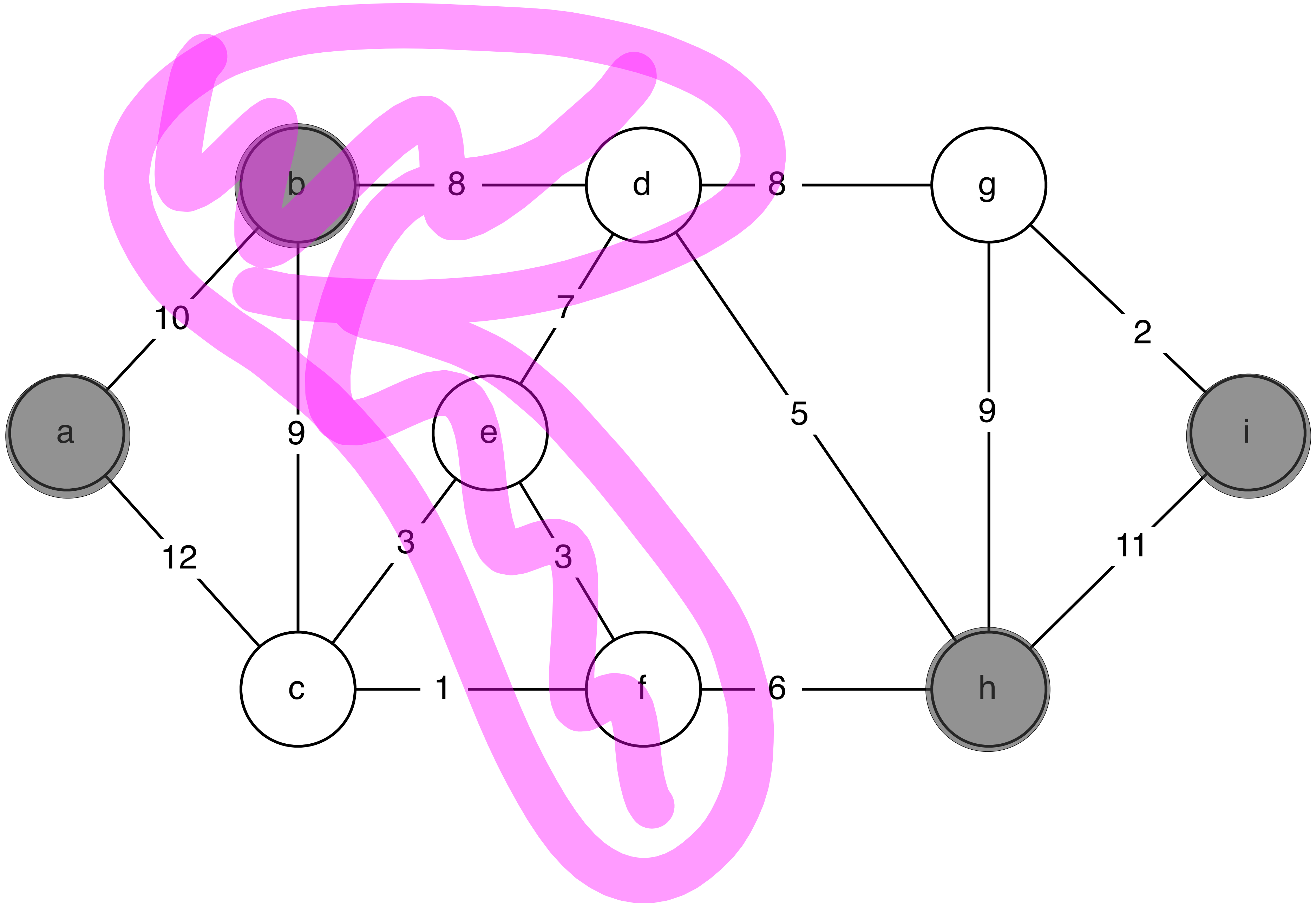
why does this work?

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

definition: cut

cut is a partition of graph $G = (V, E)$ into two sets of vertices $(S, V-S)$.

example of a cut



definition: crossing a cut

If $(S, V-S)$ is a cut, then edge $e=(u,v)$

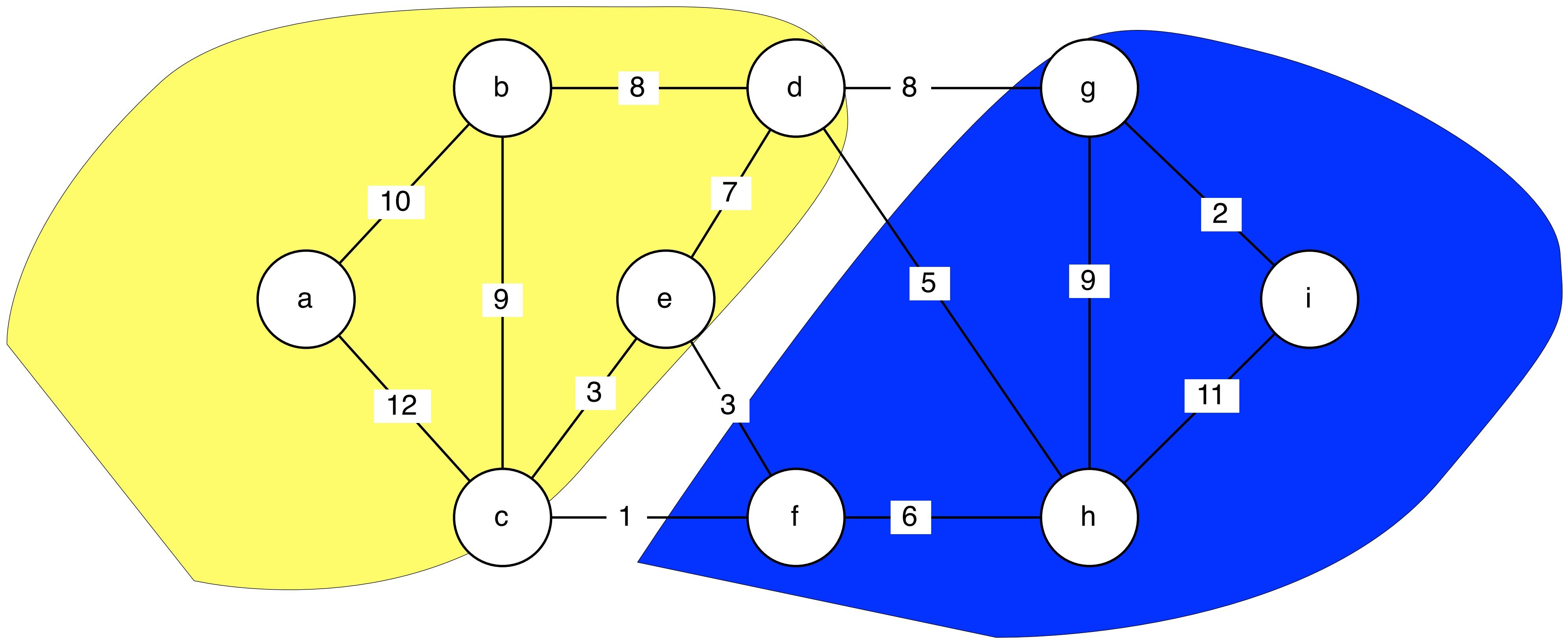
crosses the cut if

$u \in S$ and $v \in V-S$.

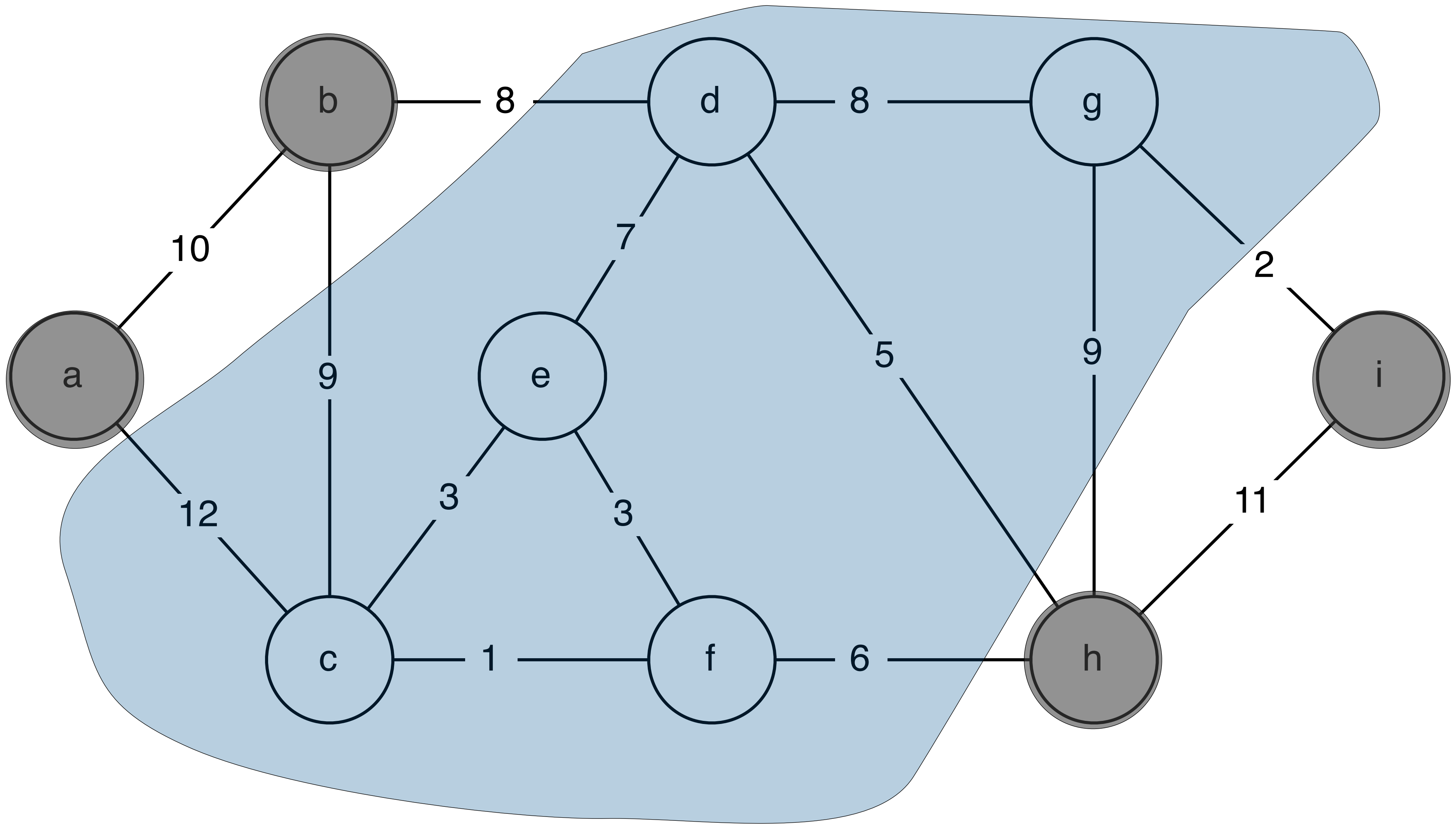
definition: crossing a cut

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



definition: respect

A set B of edges respects cut $(S, V-S)$ if

$\forall e \in B$, e does not cross $(S, V-S)$.

Cut theorem

Cut theorem

partial solution

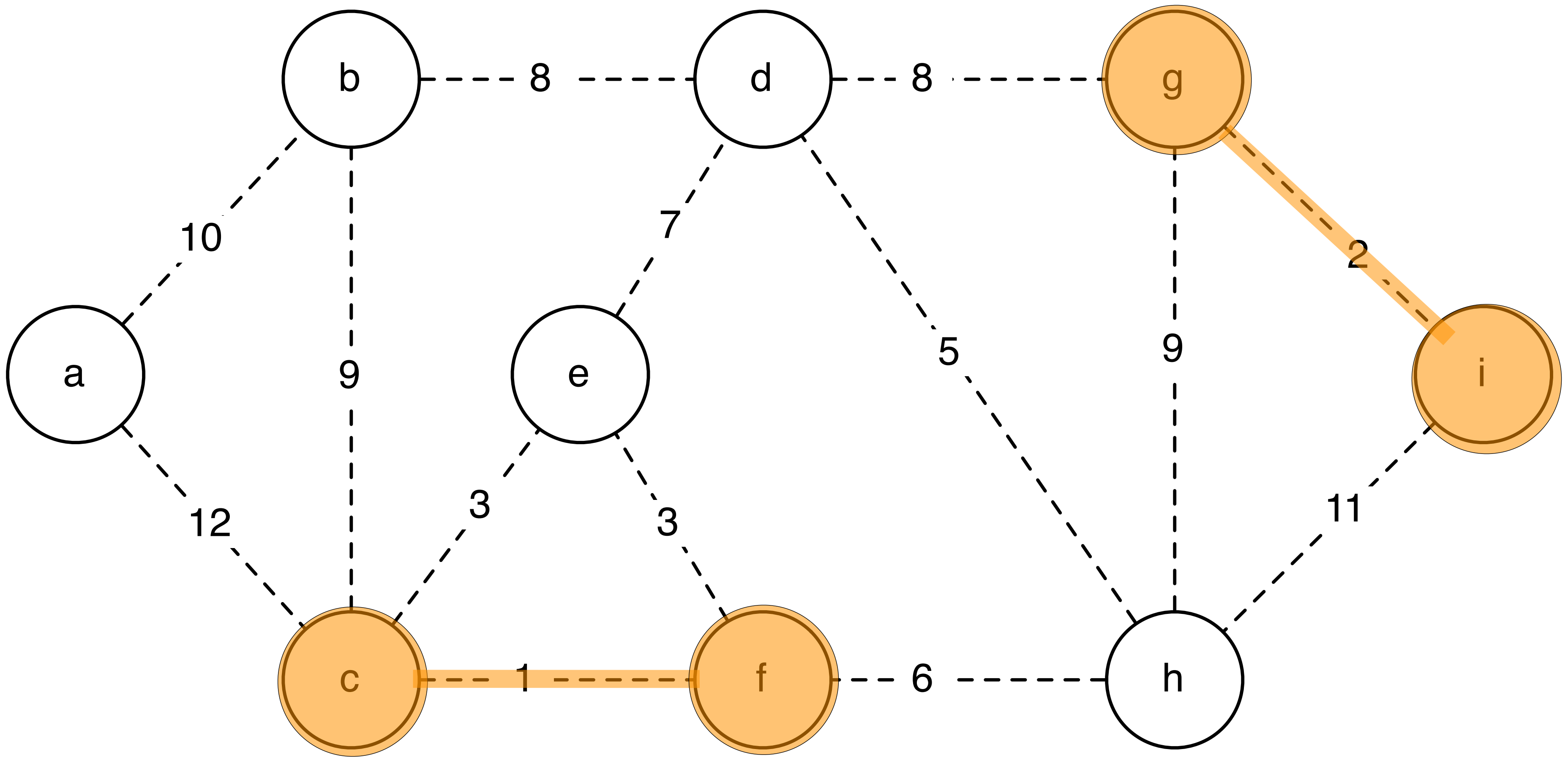
Suppose the set of edges A is part of an m.s.t.

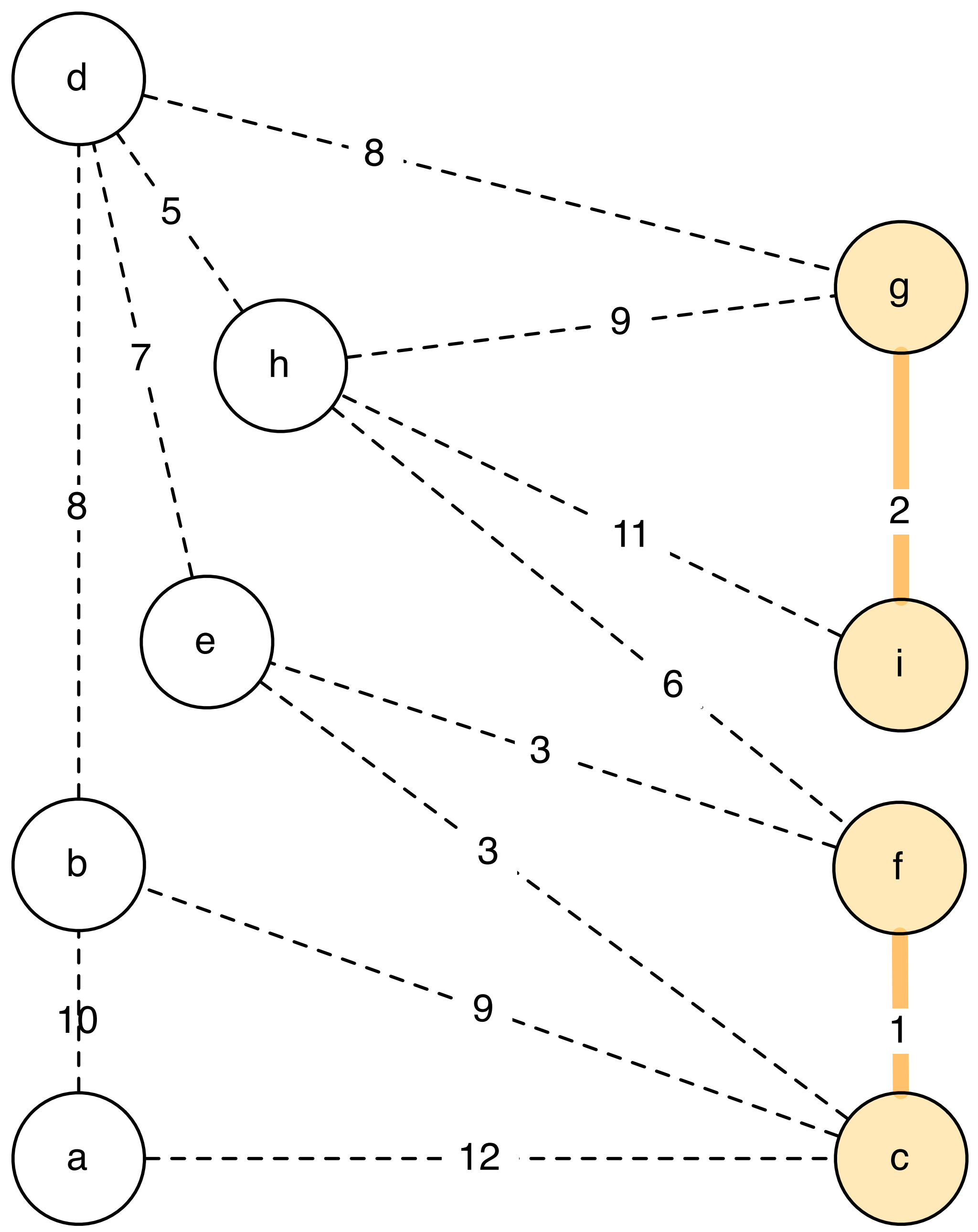
Let $(S, V - S)$ be any cut that respects A.

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

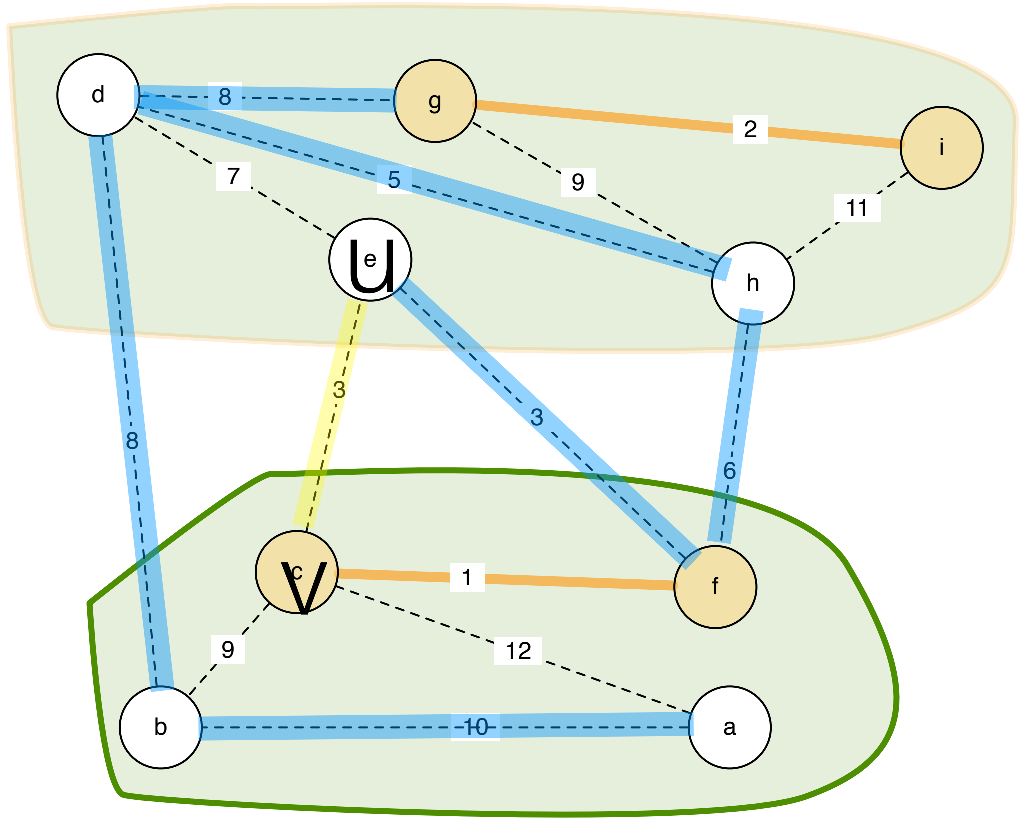



d
8
5
g
9
h
7
8
2
11
e
i
6
3
b
3
f
9
10
1
a
12
c

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 of cut thm



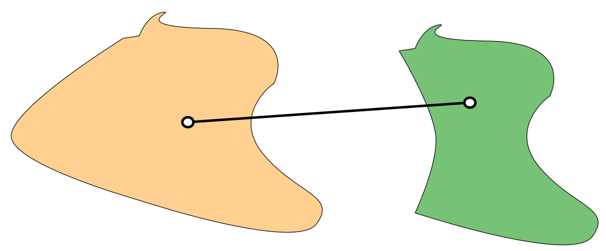
correctness

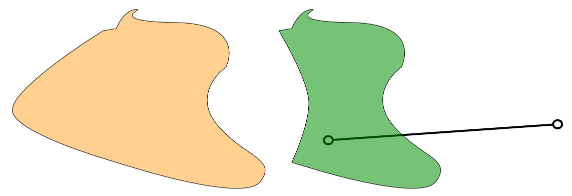
```
KRUSKAL-PSEUDOCODE(G)
A ← ∅
repeat  V − 1 times:
        add to A the lightest edge e ∈ E that does not create a cycle
```

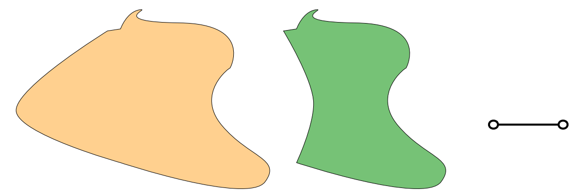
correctness

```
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).

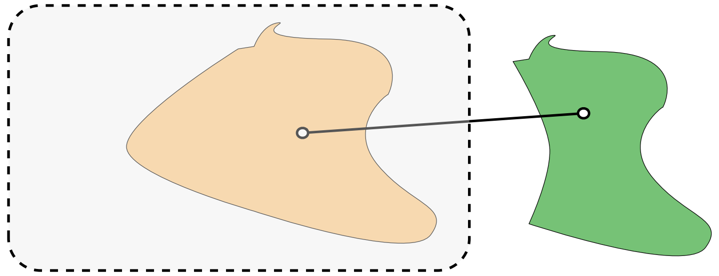






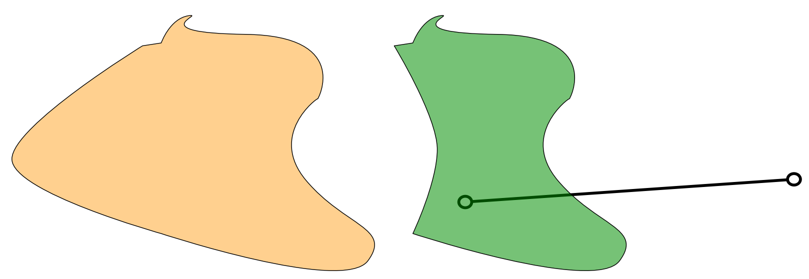
3 cases for edge e.

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



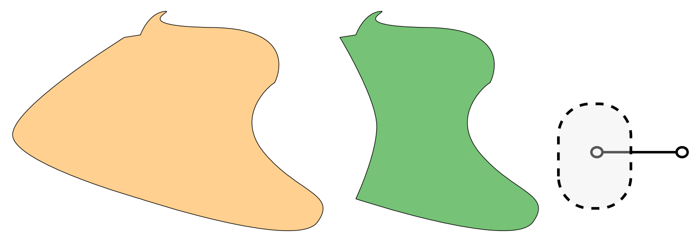
3 cases for edge e.

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



3 cases for edge e.

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



analysis?

KRUSKAL-PSEUDOCODE(G)
1 $A \leftarrow \emptyset$
2 **repeat** $V - 1$ times:
3 add to A the lightest edge $e \in E$ that does not create a cycle

we can interpret this step as a way of picking a cut $(S, V-S)$ such that A respects $(S, V-S)$ and e is the lightest edge that crosses $(S, V-S)$

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

$A \leftarrow \emptyset$

repeat $V - 1$ times:

  Pick a cut $\underbrace{(S, V - S)}$ that $\underbrace{\text{respects } A}$

  Let e be min-weight edge over cut $(S, V - S)$

  $A \leftarrow A \cup \{e\}$

CUT thm

A be a tree. and $S = A$.

Prim's algorithm

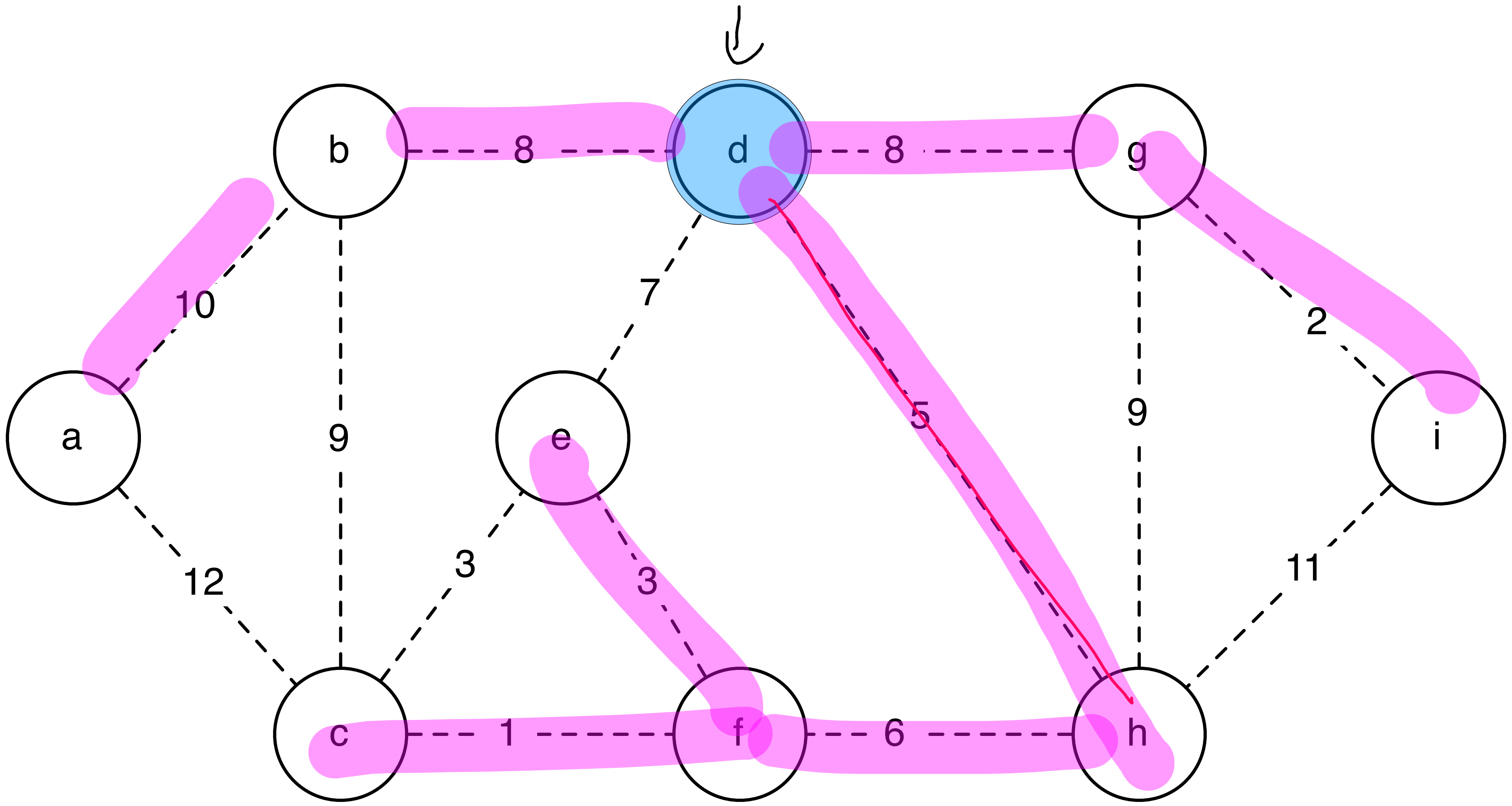
GENERAL-MST-STRATEGY$(G = (V, E))$

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

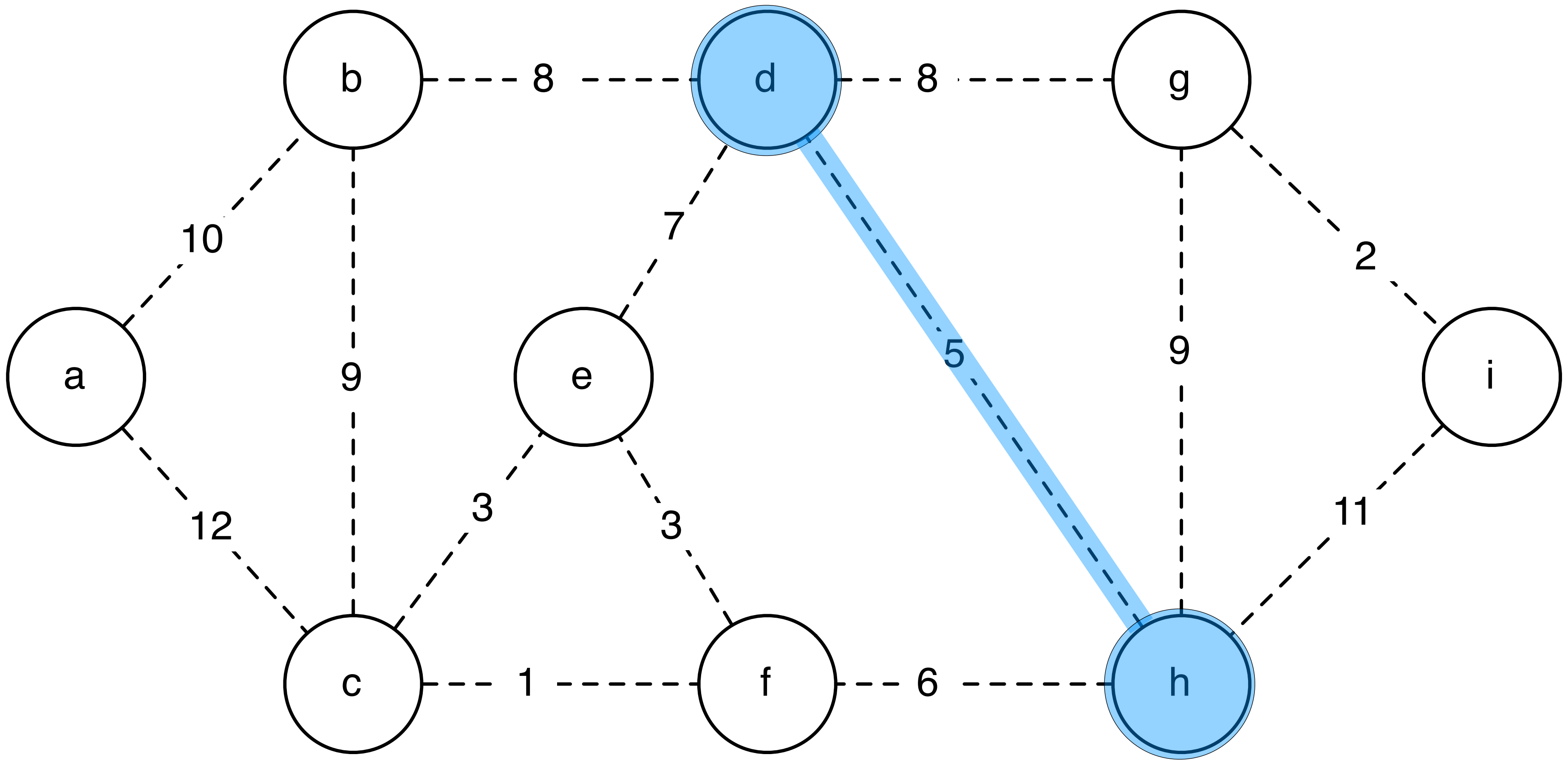
A is a subtree

edge e is lightest edge that grows the subtree

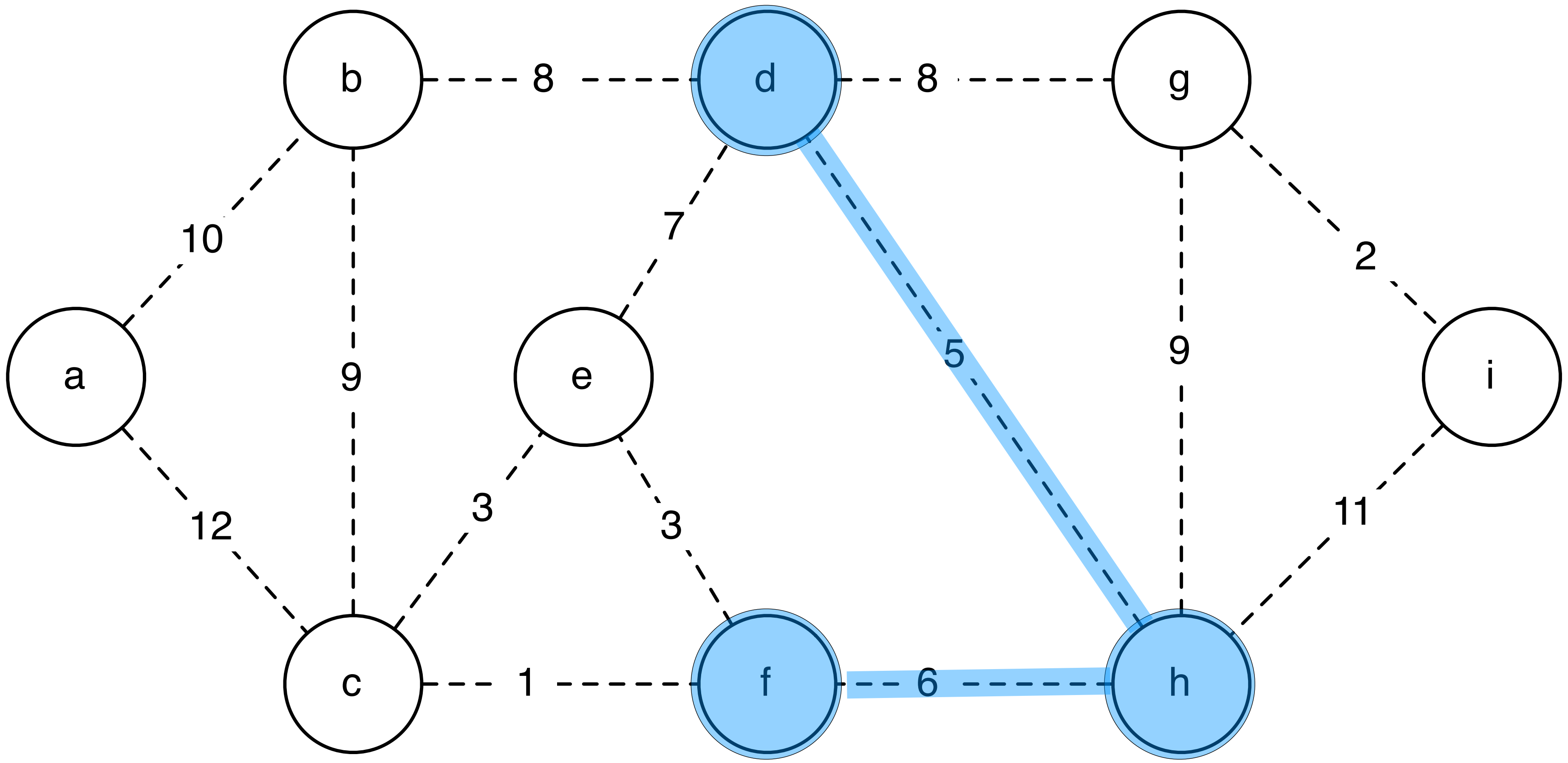
prim



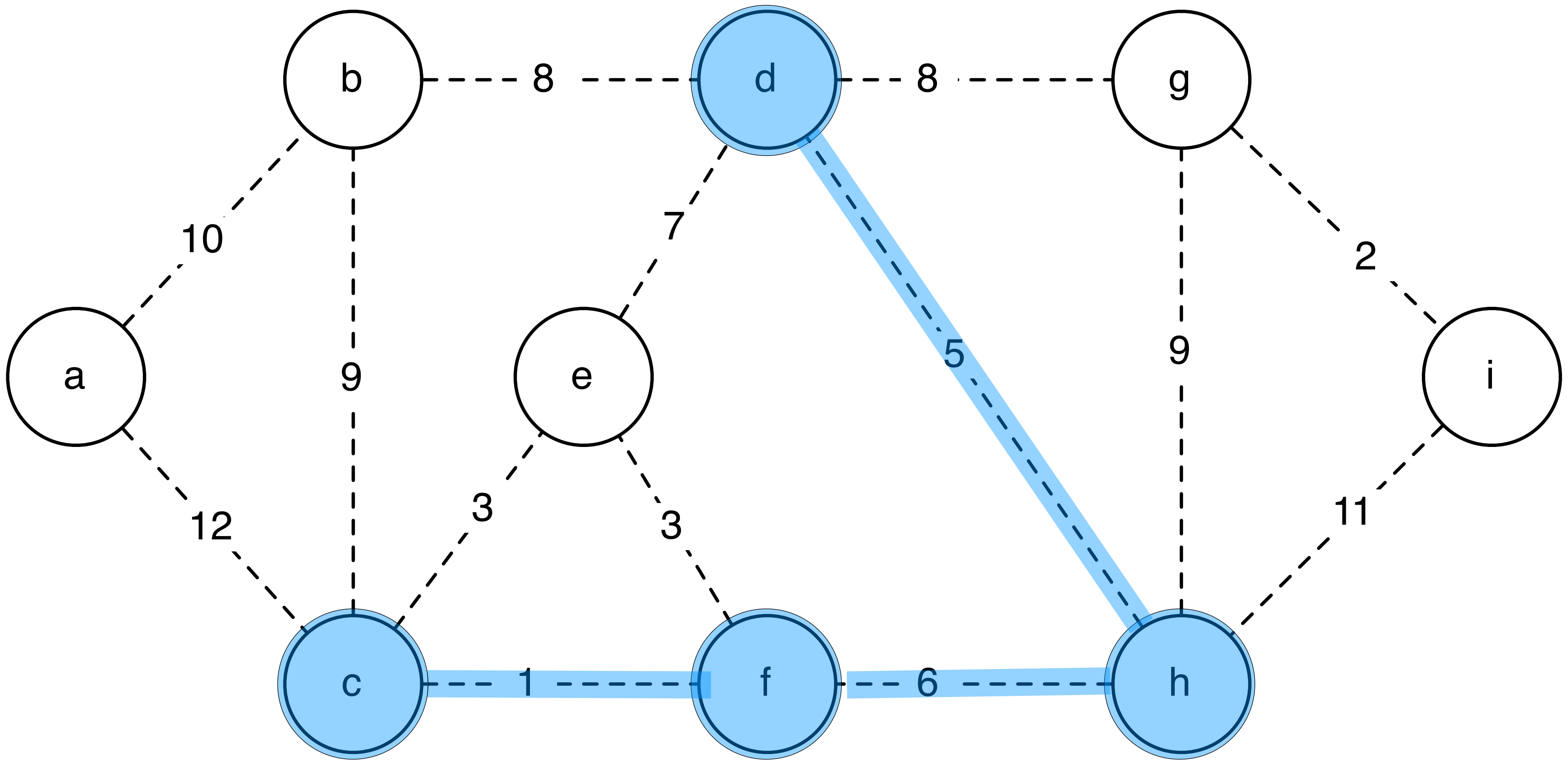
prim



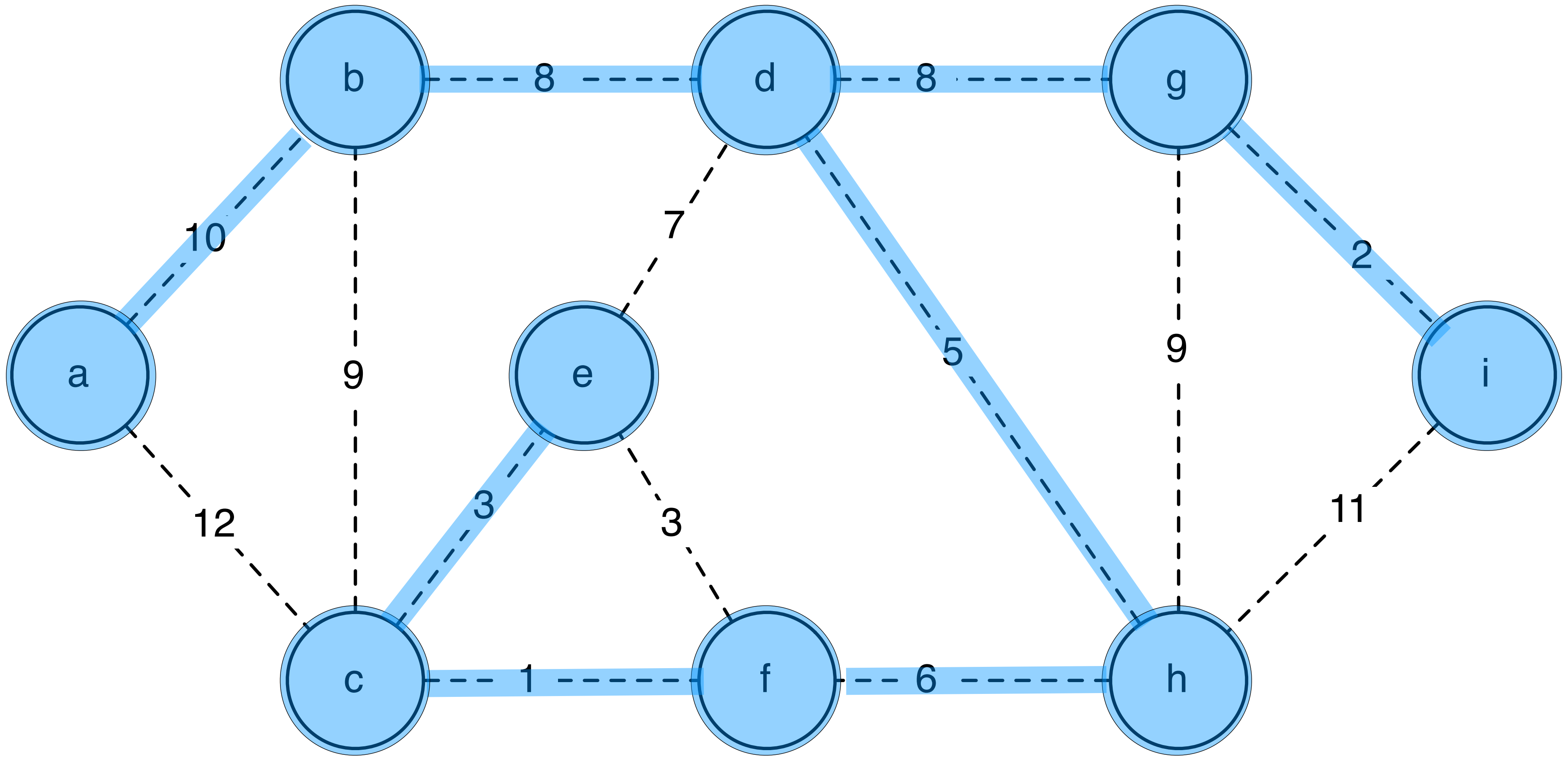
prim



prim



prim



implementation

idea: keep a data structure which identifies the "lightest edge that crosses $(A=S, V-S)$"

— priority queue.

implementation

new data structure $(\text{node}, \text{key})$ integer?

- <u>makequeue</u>: insert a list of $(n_1, k_1)\ (n_2, k_2) \dots (n_\ell, k_\ell)$ into Q
  <u>insert</u>

- <u>extractmin operation</u>: removes the node (n_i, k_i) where k_i is the smallest key in Q.

- <u>decrease key operation</u>: given a key $(n_i, \underline{k_i})$, decrease the value of k_i to k_i^*.

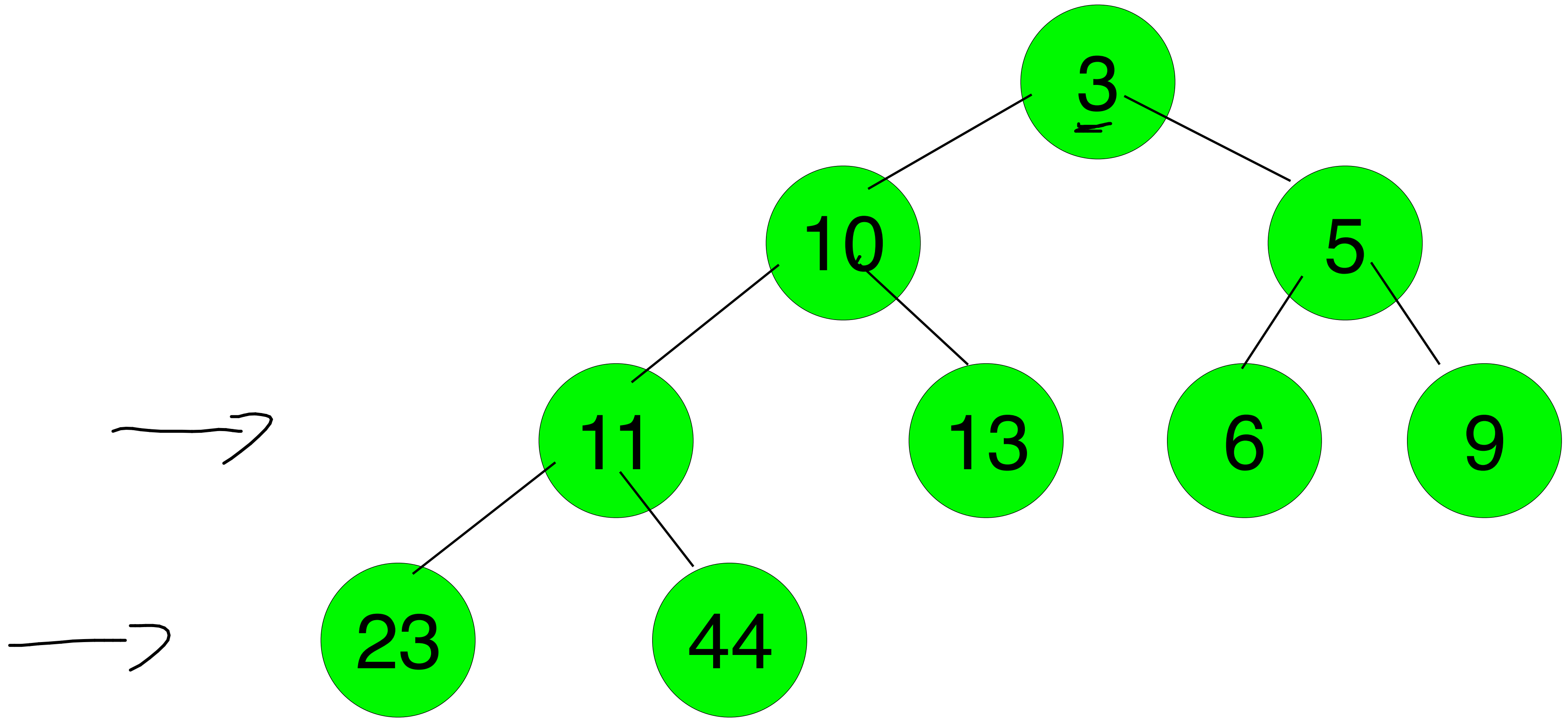
binary heap

1 full tree, key value <= to key of children

2

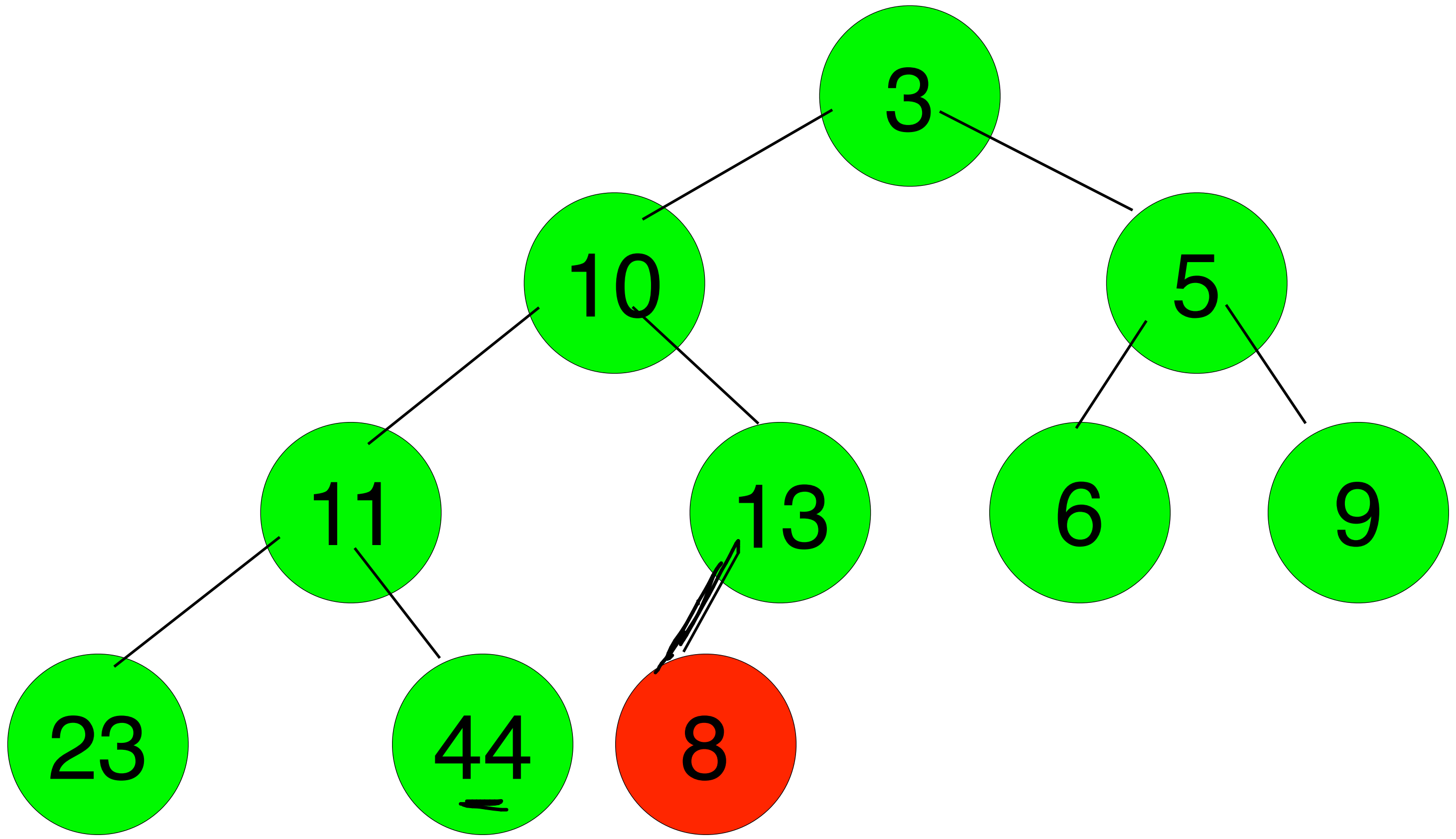
binary heap

full tree, key value <= to key of children



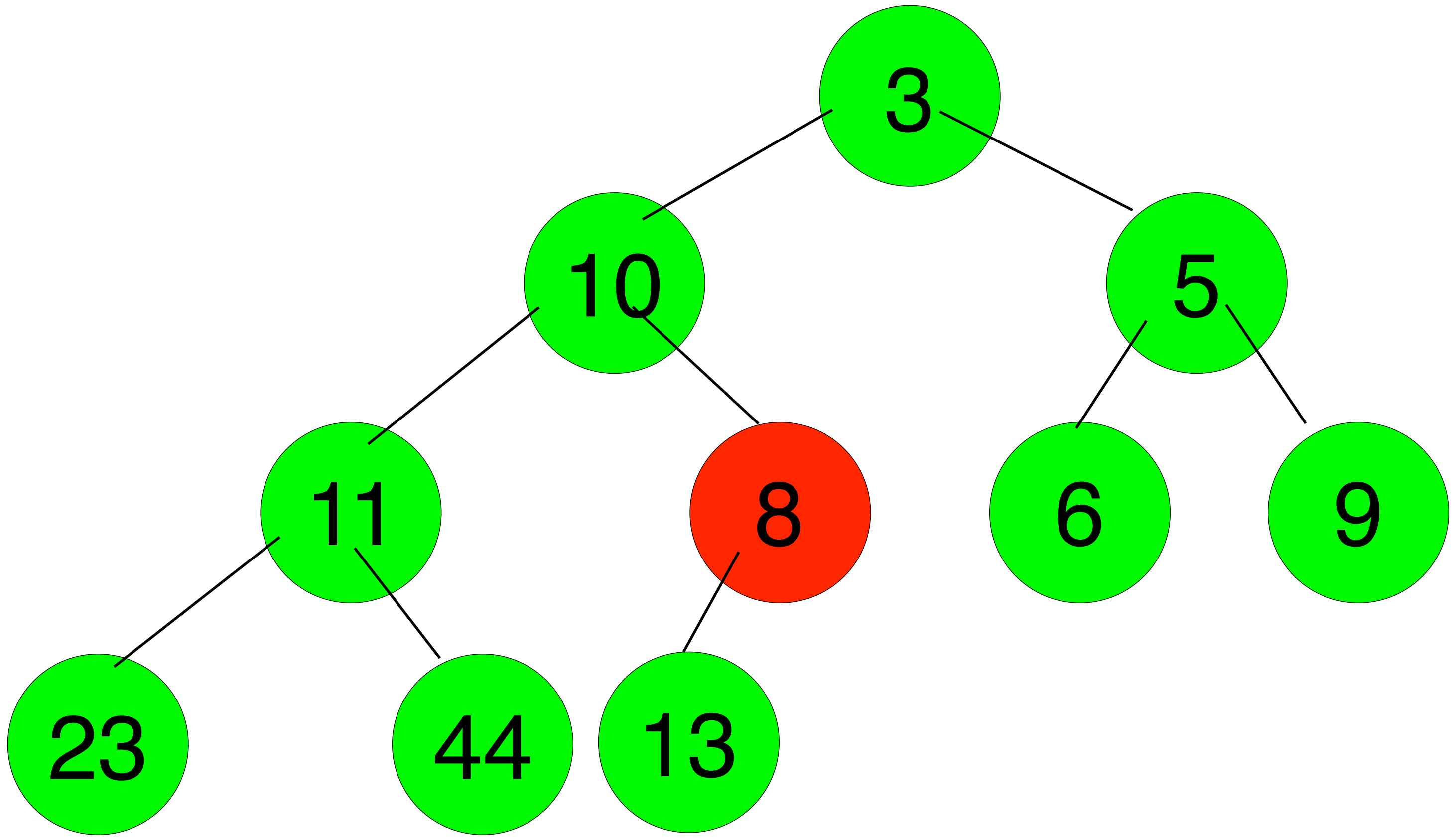
binary heap

full tree, key value <= to key of children



binary heap

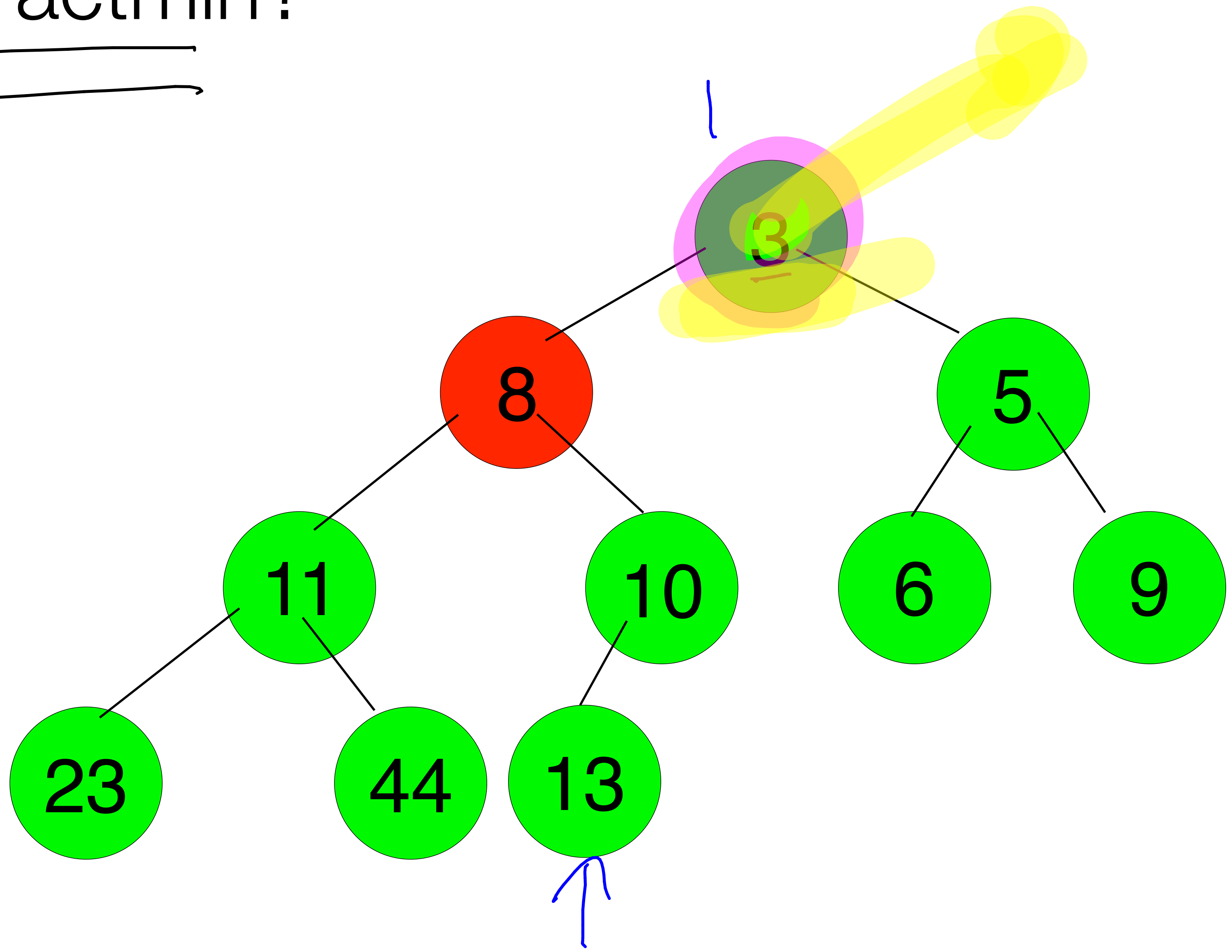
full tree, key value <= to key of children



binary heap

full tree, key value <= to key of children

how to extractmin?



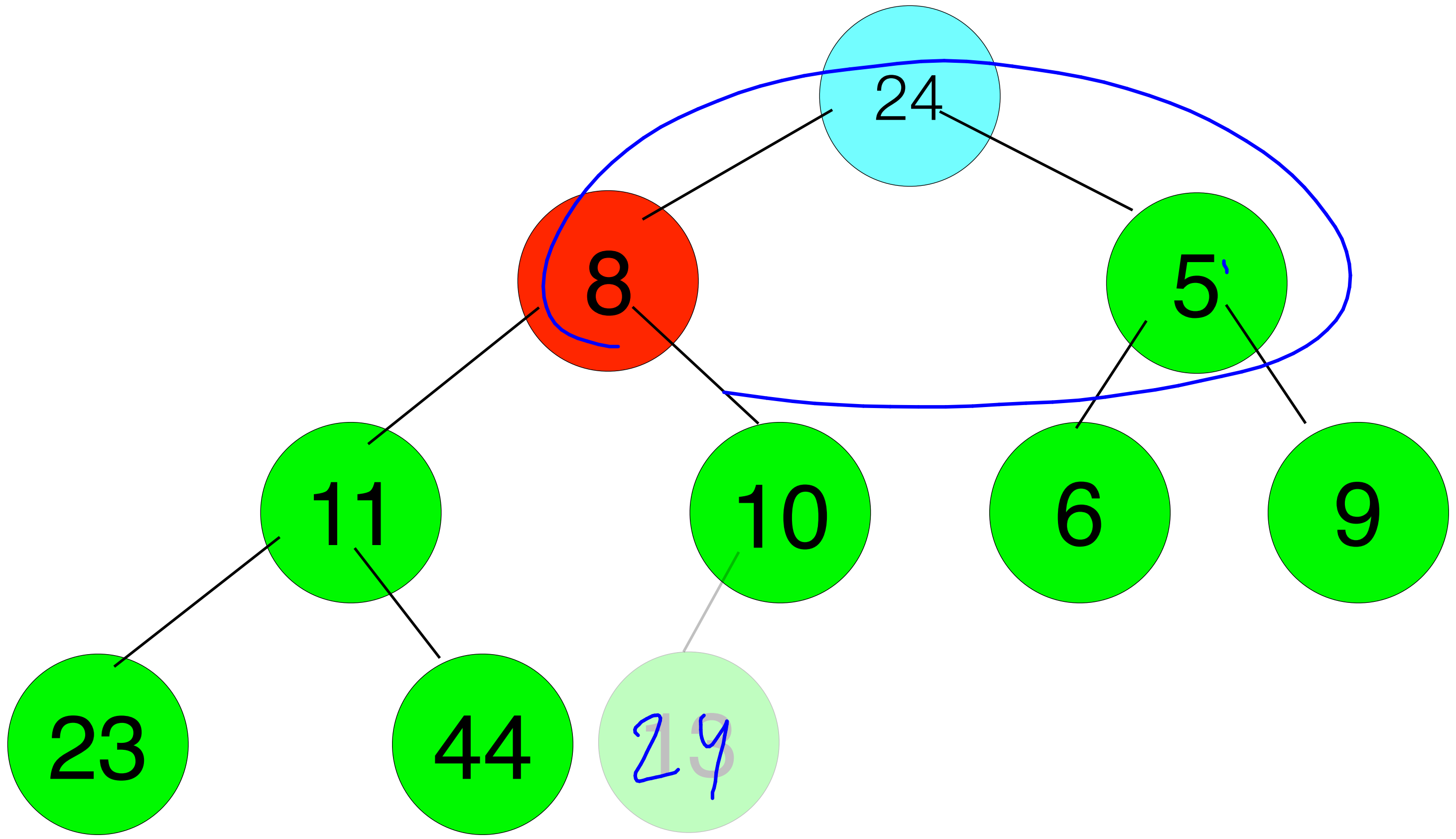
insert: time ?

$\Theta(\log n)$ where n is the size of the heap.

binary heap

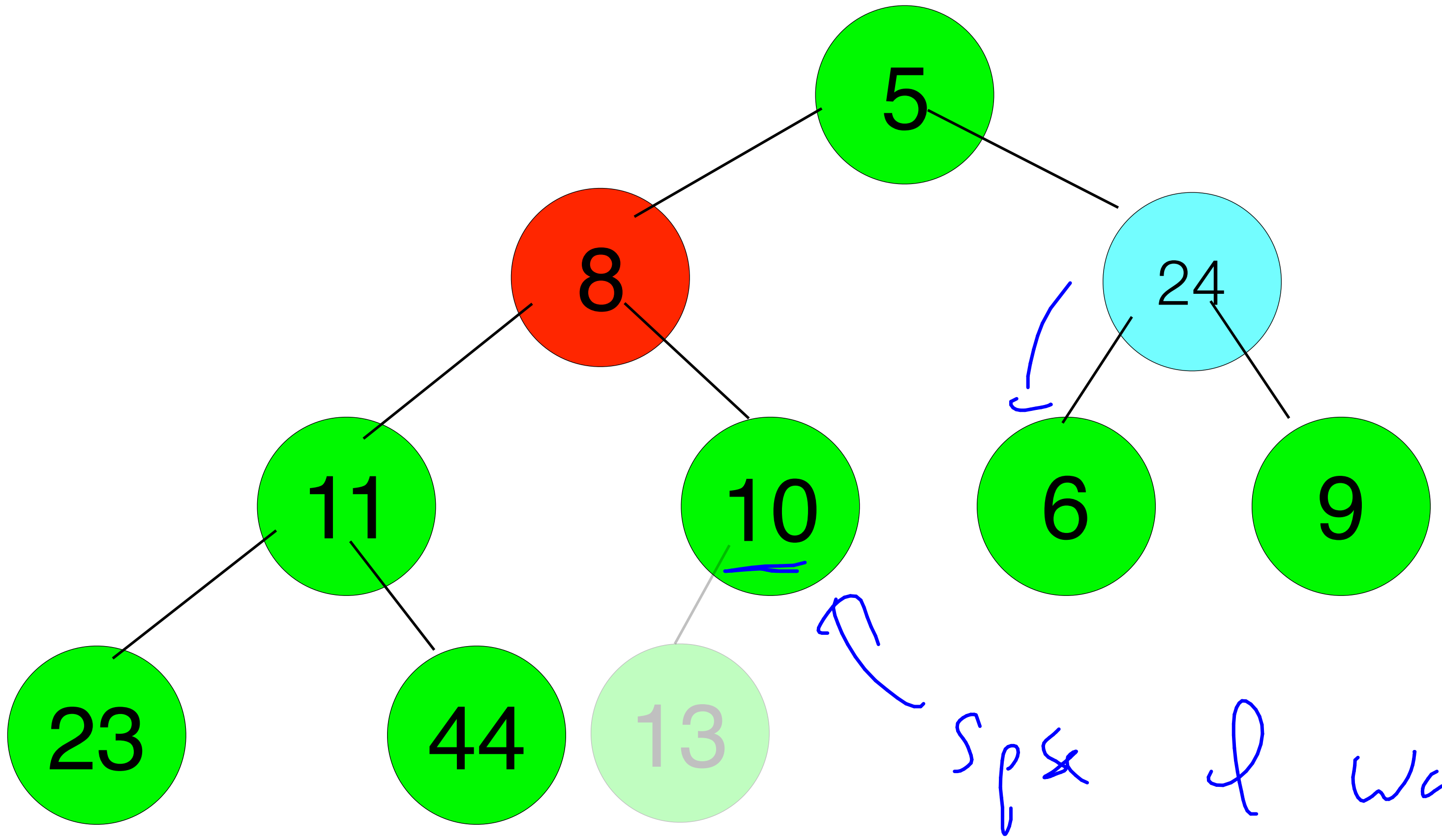
full tree, key value <= to key of children

how to extractmin?



(fix me)

binary heap



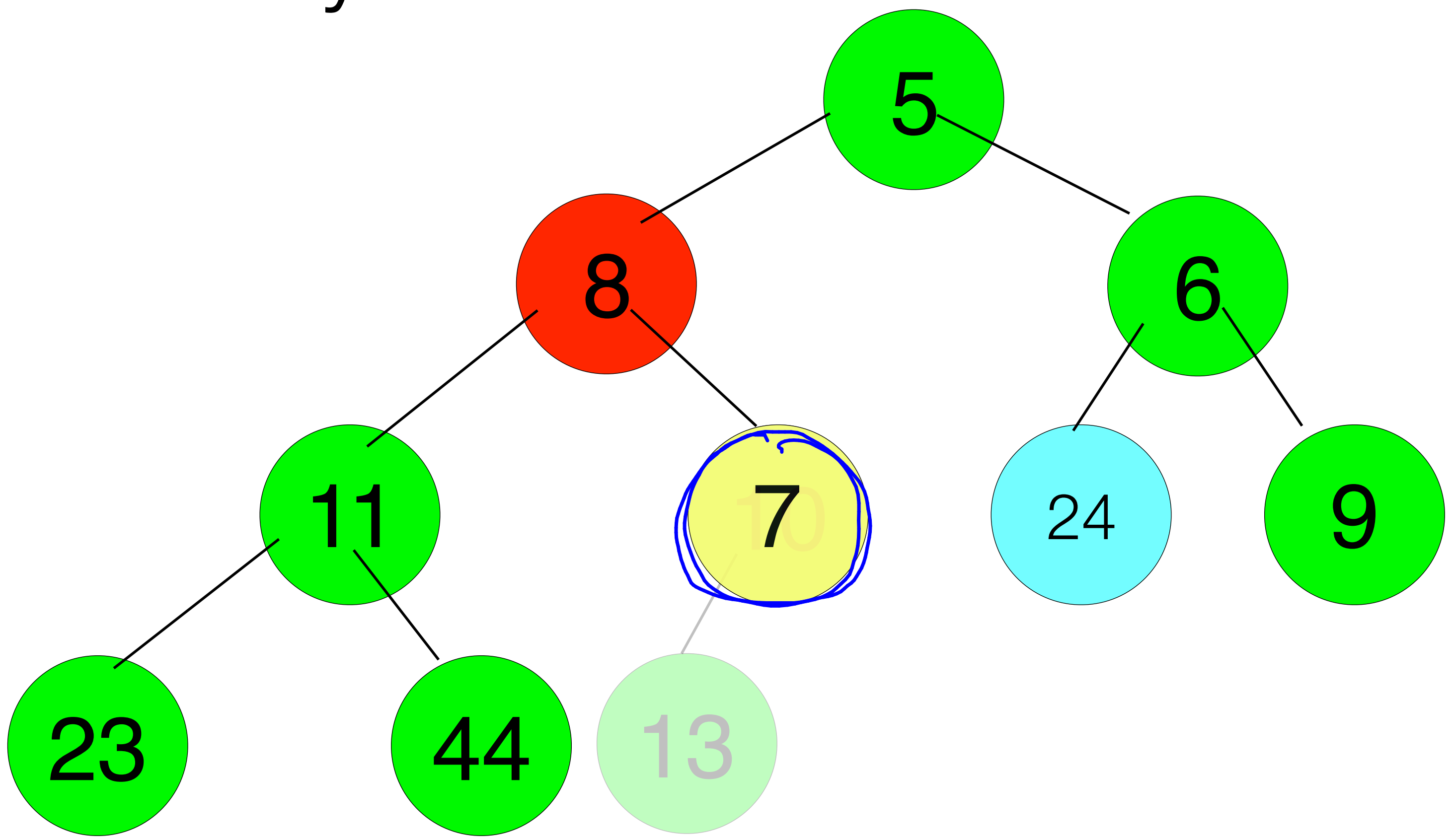
spse I want to decrease key 10 to 7!!

binary heap

full tree, key value <= to key of children

how to extractmin? → $\Theta(\log n)$ time

how to decreasekey?

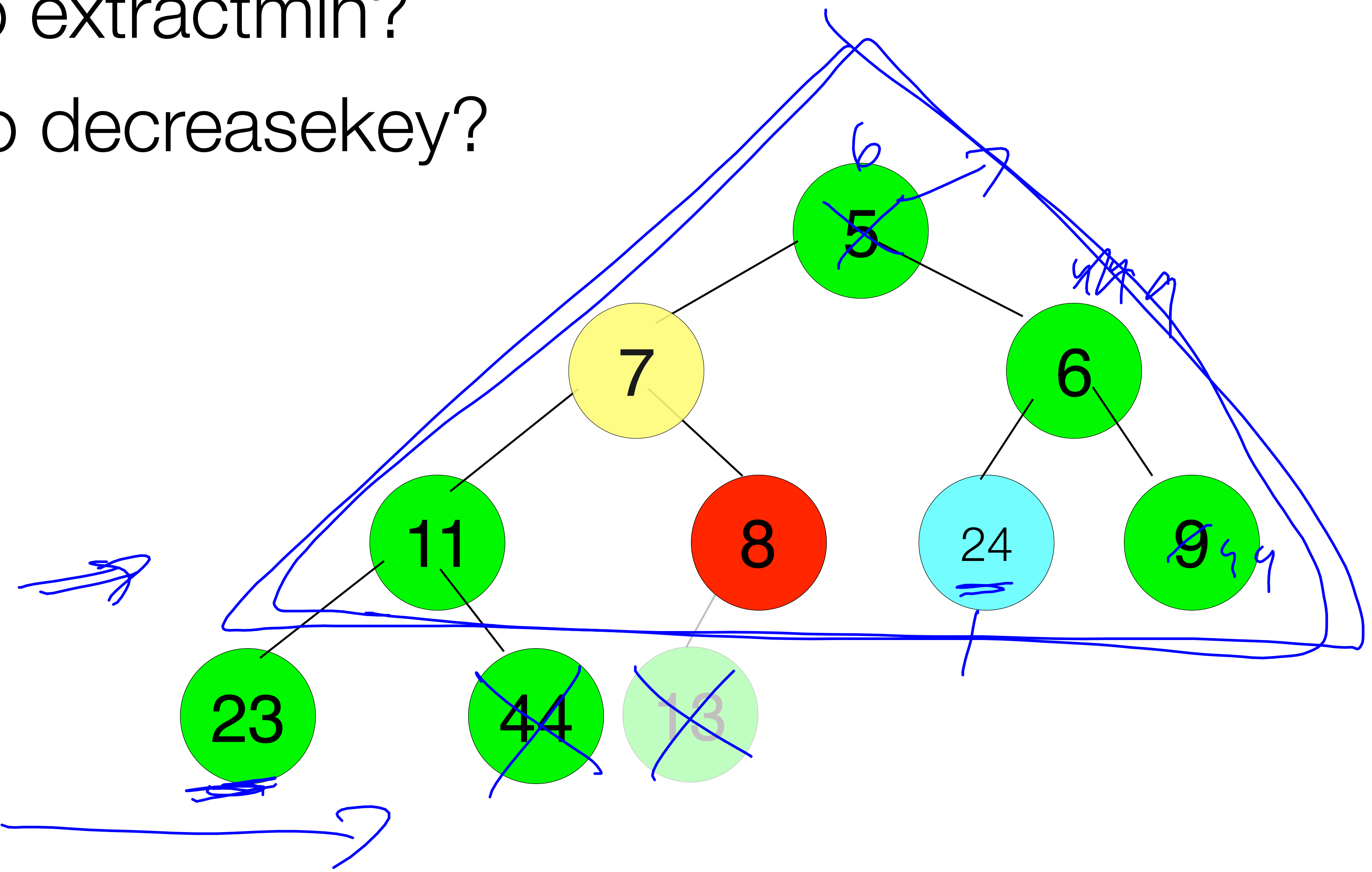


binary heap

full tree, key value <= to key of children

how to extractmin?

how to decreasekey?



O(log n) time

```
node {
    key int
    parent *node
    left,right *node
}
```

implementation

use a priority queue to keep track of light edges

insert:

makequeue: $\rightarrow$ $O(n)$

extractmin: $O(\log n)$

decreasekey: $O(\log n)$

Prim's algorithm

Initialize each node w/ $k_v \leftarrow \infty$

Pick some node r, and set $k_r \leftarrow 0$

$Q \leftarrow$ MakeQueue { all vertices }

while Q is not empty

$v \leftarrow$ extractmin (Q)

for all neighbors u of v,

if $u \in Q$ and $k_u > w(u,v)$

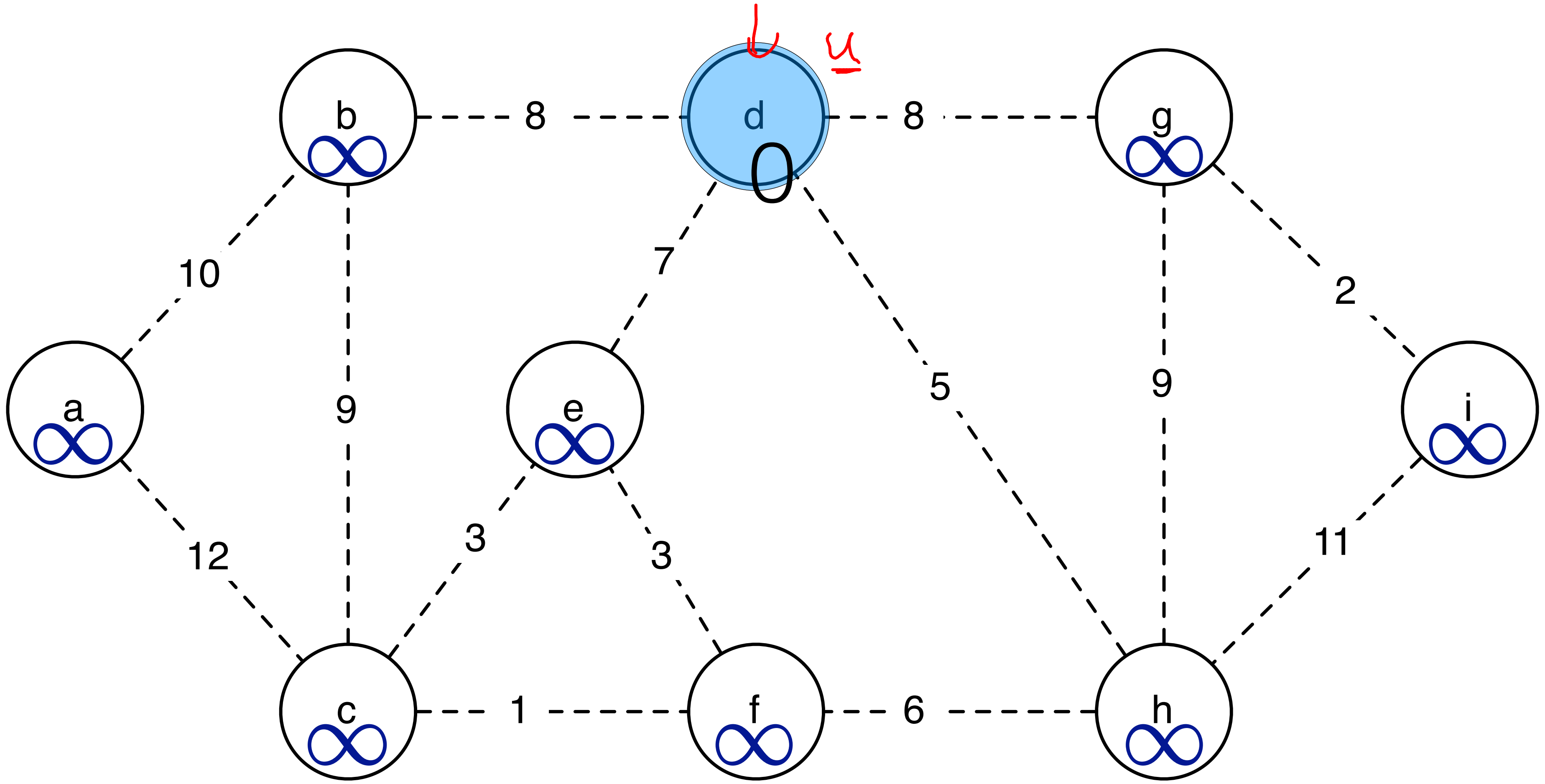
DecreaseKey$(Q, u, w(u,v))$

implementation

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

$Q \leftarrow \emptyset$ $\quad \triangleright$ 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

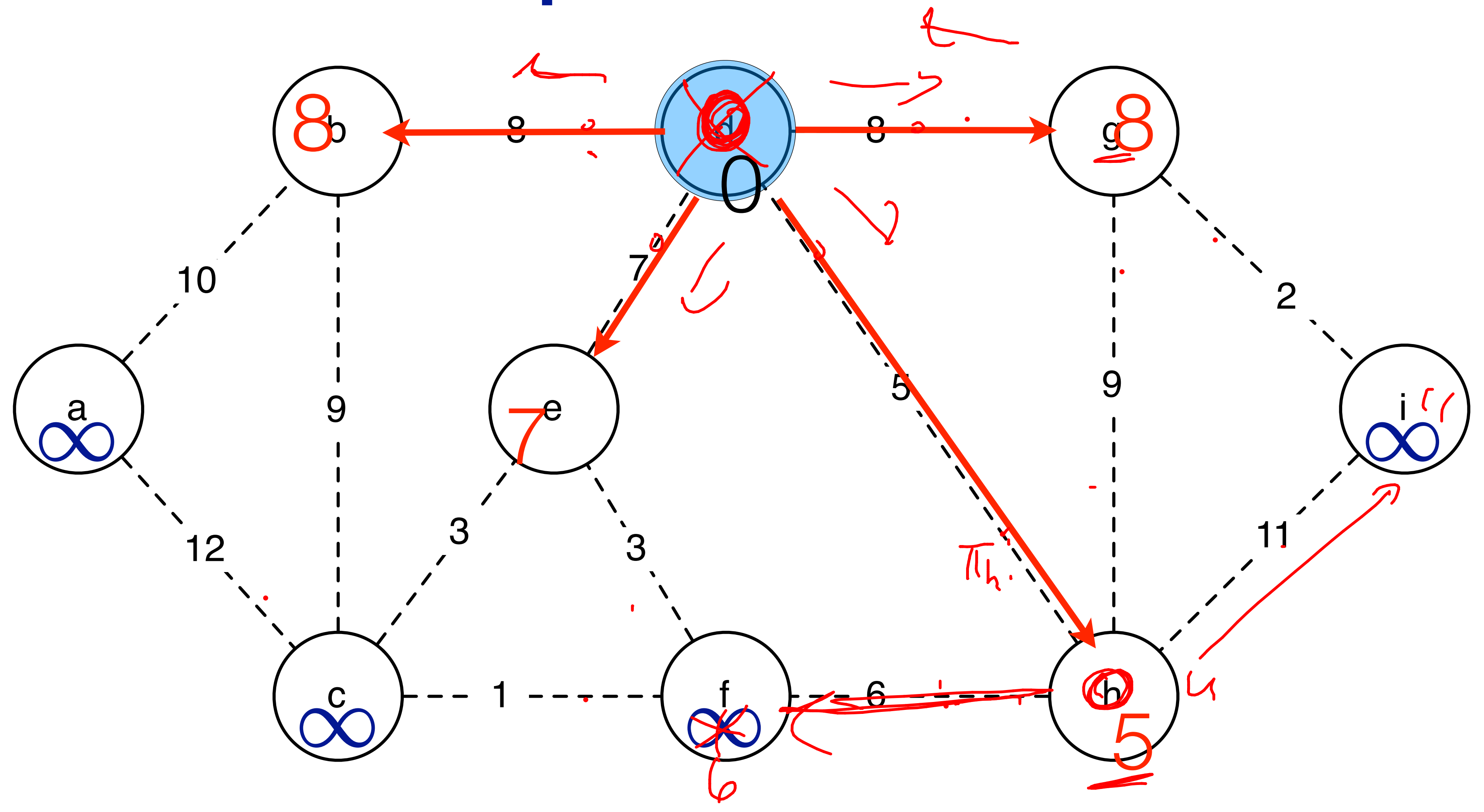


PRIM$(G = (V, E))$

$Q \leftarrow \emptyset$ $\quad \triangleright$ 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

4 4 3



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)

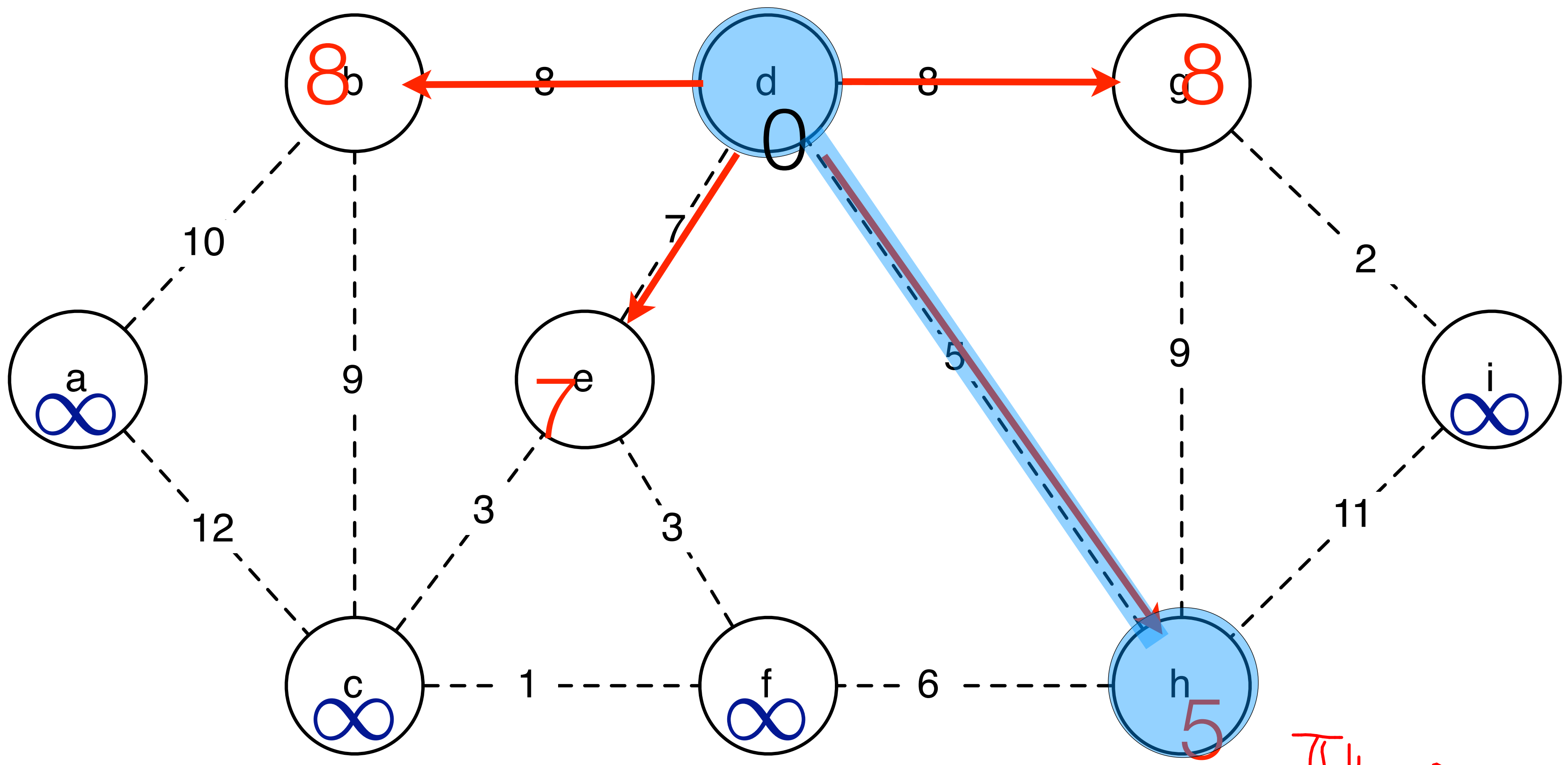
    for each $v \in Adj(u)$

      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

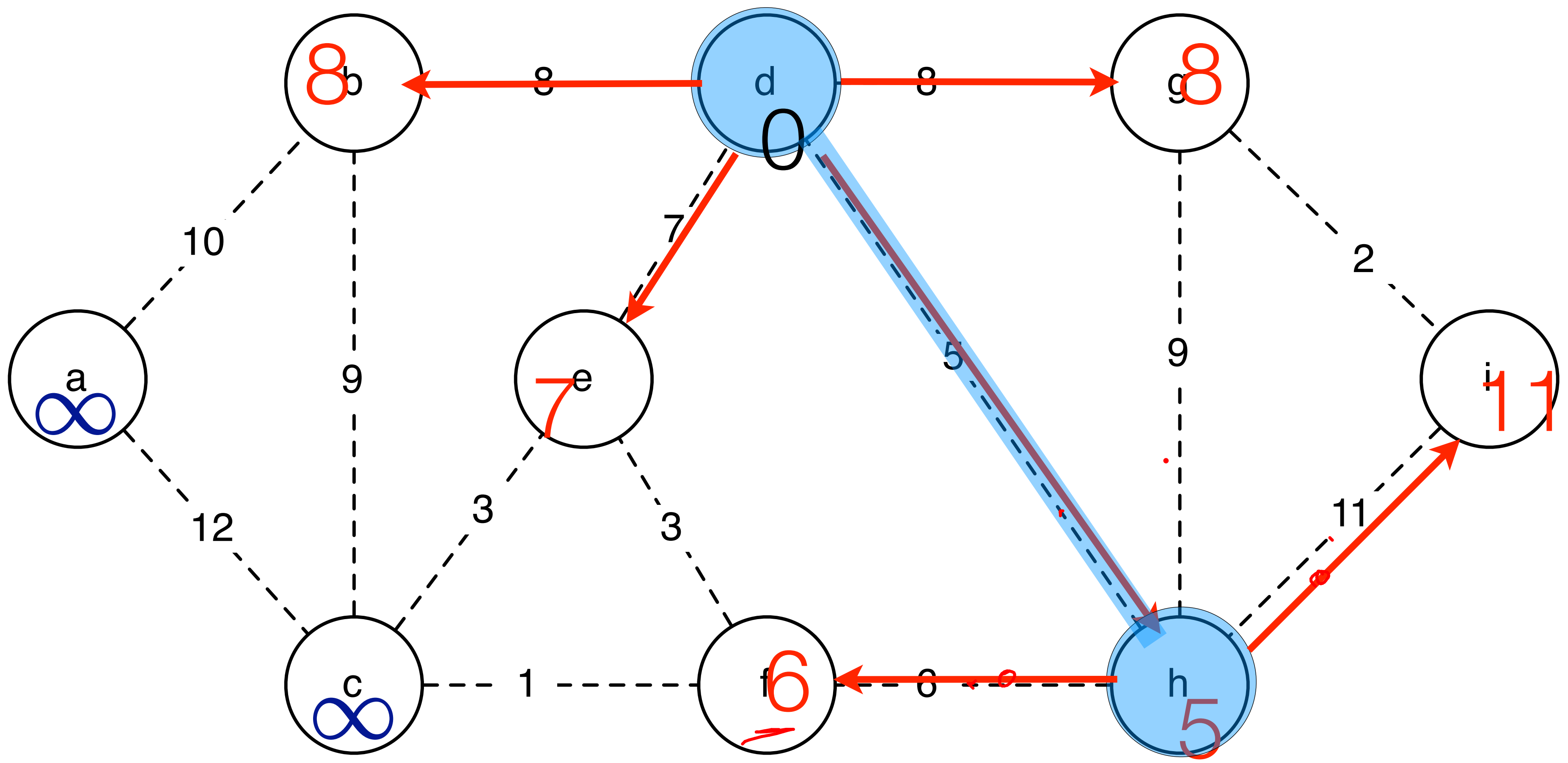


This is part of my MST

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 $k_v \leftarrow w(u, v)$

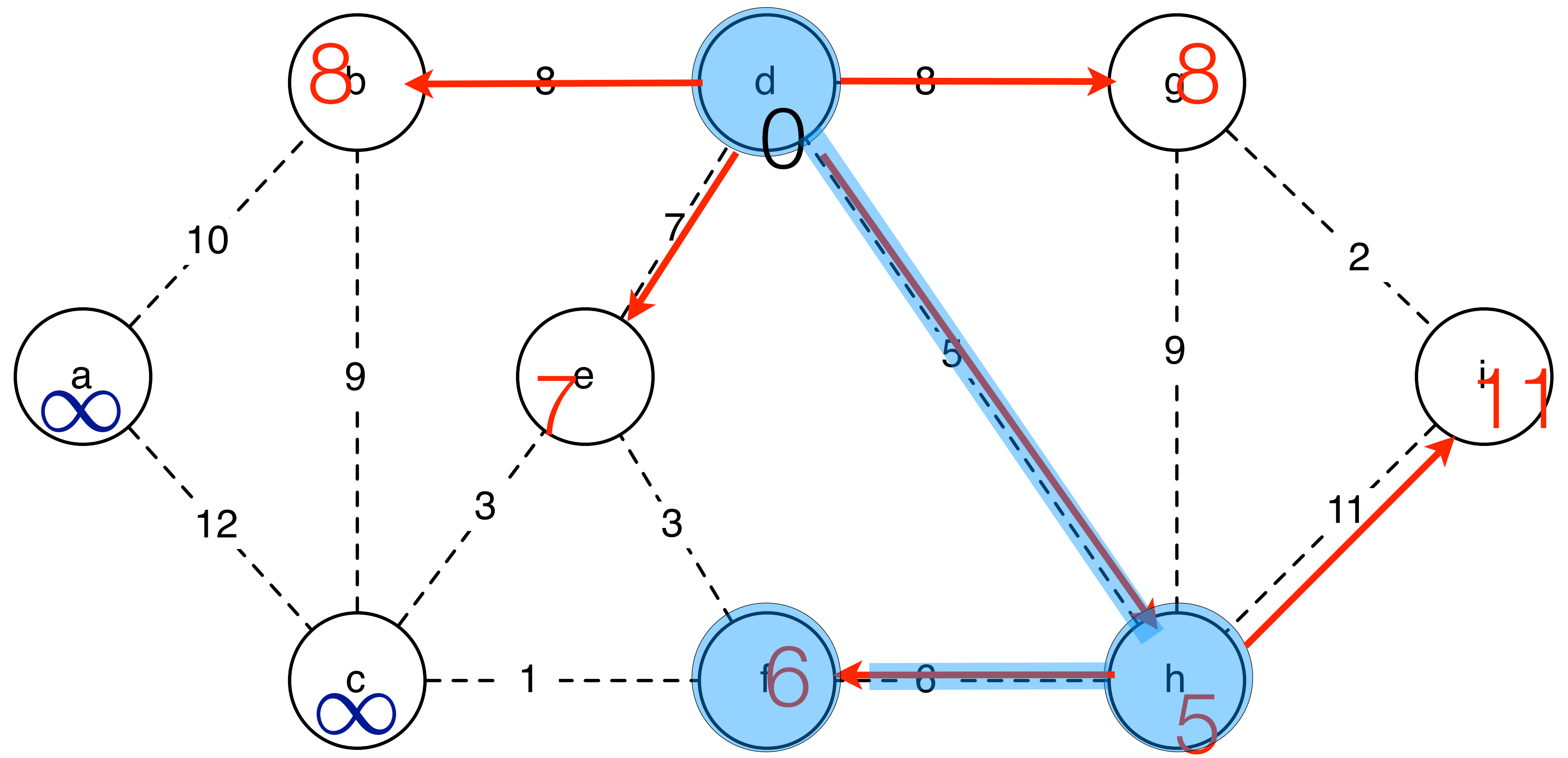
prim



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

$Q \leftarrow \emptyset$ $\quad \triangleright$ 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



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)

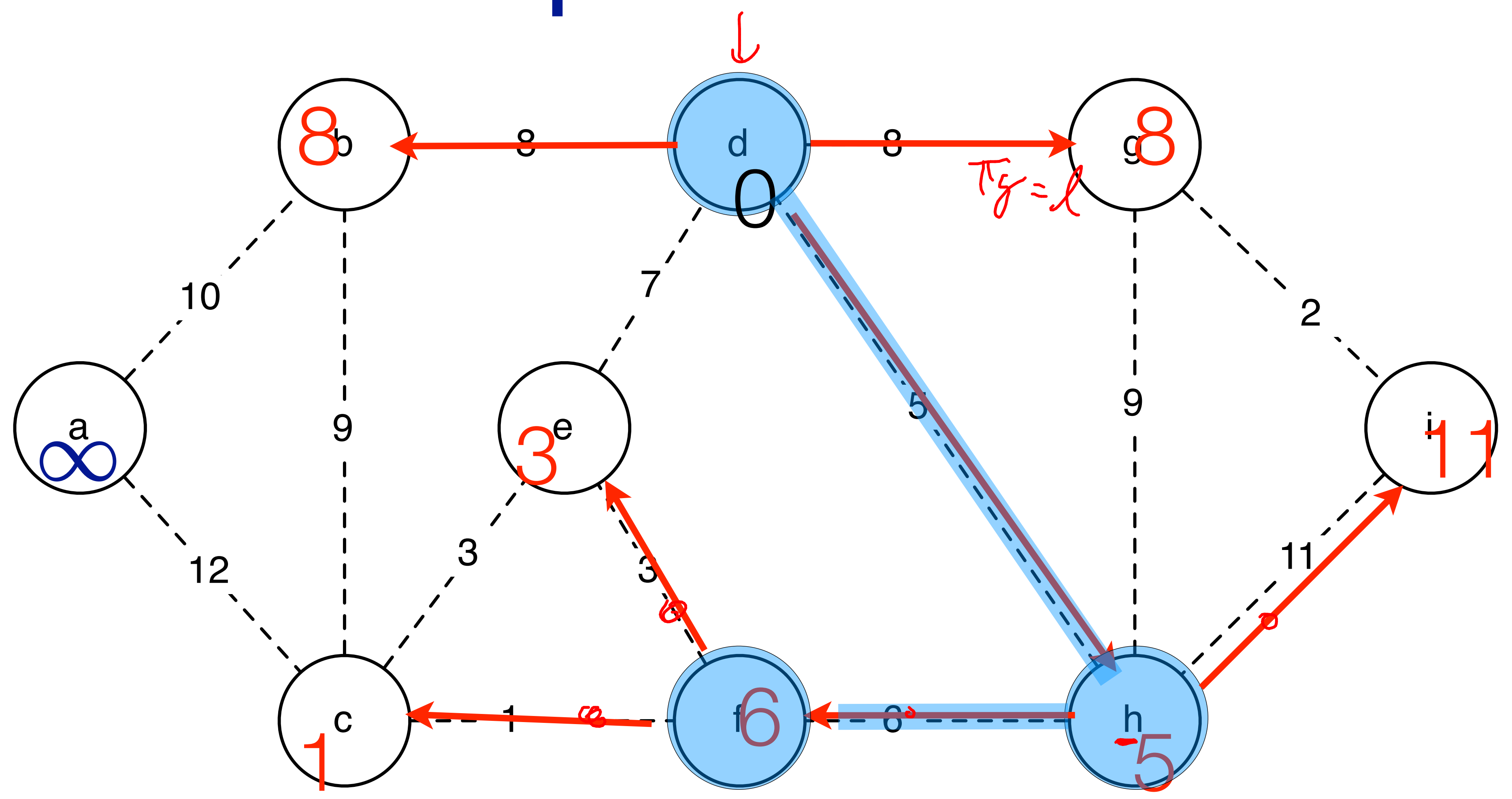
   for each $v \in Adj(u)$

    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



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

$Q \leftarrow \emptyset$ ▷ 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$(Q, v, w(u, v))$ ▷ Sets $k_v \leftarrow w(u, v)$

# of calls to decrease key is < E.

running time

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 $k_v \leftarrow w(u, v)$

$\rightarrow \Theta(V)$

$\Theta(V \lg V)$

$\rightarrow$ called V times $\rightarrow \Theta(V \lg V)$ time

$\leftarrow \Theta(E)$

$\rightarrow \Theta(E)$

$\rightarrow \Theta(E \operatorname{Log}(V))$

this work.

called at most E times over entire execution

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

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)
```

$O(\log V)$

$$O(V \log V + E \log V) = \underline{O(E \log V)}$$

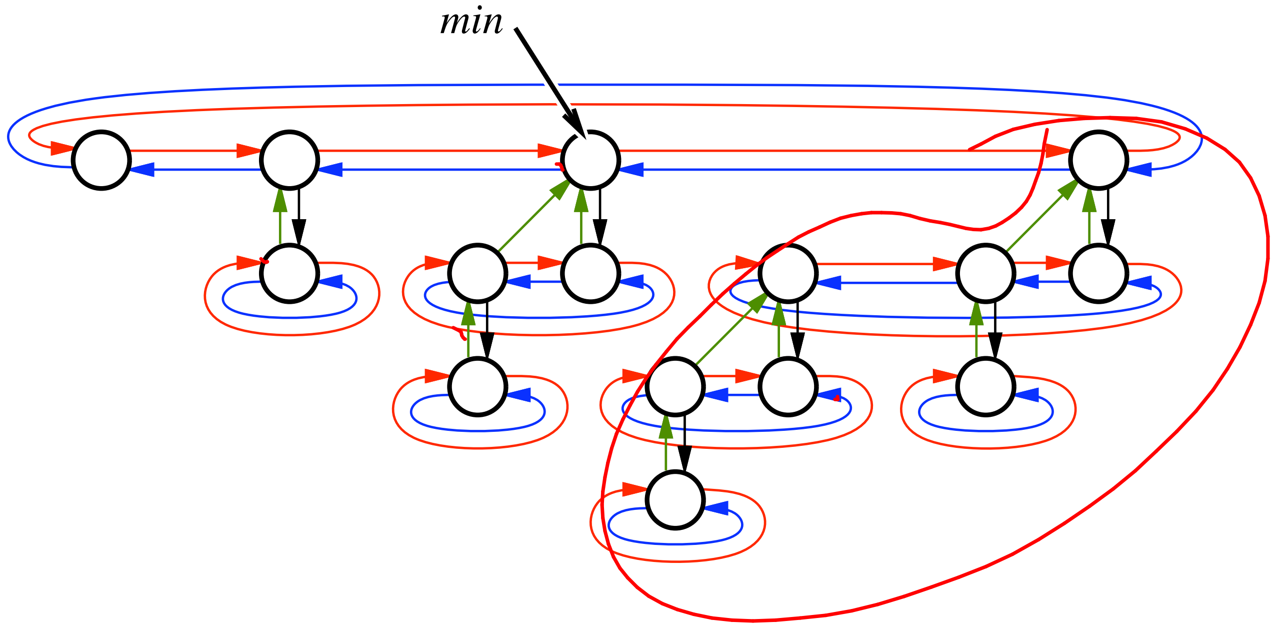
$E \geq V$

implementation

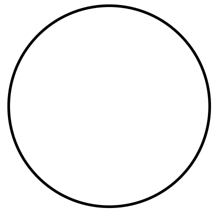
use a priority queue to keep track of light edges

	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



fibonacci sequence



each node has 4 pointers

2 fields:

degree

marked

$D(n)$

faster implementation

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

$Q \leftarrow \emptyset$ ▷ 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) → $V \lg V$
    for each $v \in Adj(u)$
      do if $v \in Q$ and $w(u, v) < k_v$
        then $\pi_v \leftarrow u$
          DECREASE-KEY$(Q, v, w(u, v))$ ▷ Sets $k_v \leftarrow w(u, v)$

$\Theta(1)$

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

Research in mst

$\alpha(2^{300}) = 4$

FREDMAN-TARJAN 84: $E + V \log V$

GABOW-GALIL-SPENCER-TARJAN 86: $E \log(\log^* V)$

CHAZELLE 97 $E\alpha(V) \log \cancel{\alpha}(V)$

CHAZELLE 00 $E\alpha(V)$

PETTIE-RAMACHANDRAN 02: Θ (optimal)

deterministic

KARGER-KLEIN-TARJAN 95: E

(randomized)

Euclidean mst: $V \log V$

Ackerman function

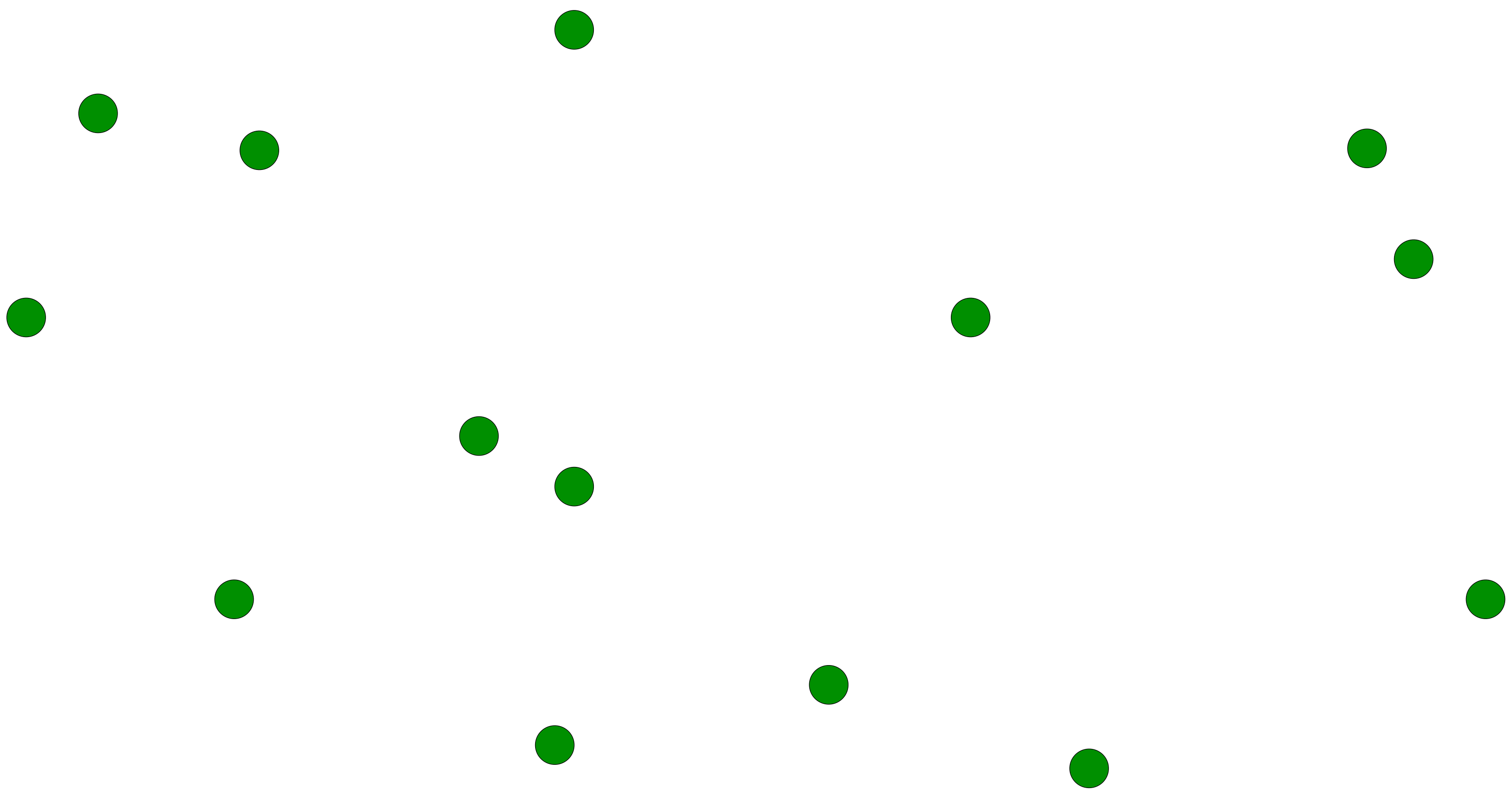
$$A(m,n) = \begin{cases} n+1 & m = 0 \\ A(m-1,1) & m > 0,\ n = 0 \\ A(m-1, A(m,n-1)) & m,n > 0 \end{cases}$$

$$A(4,2) =$$

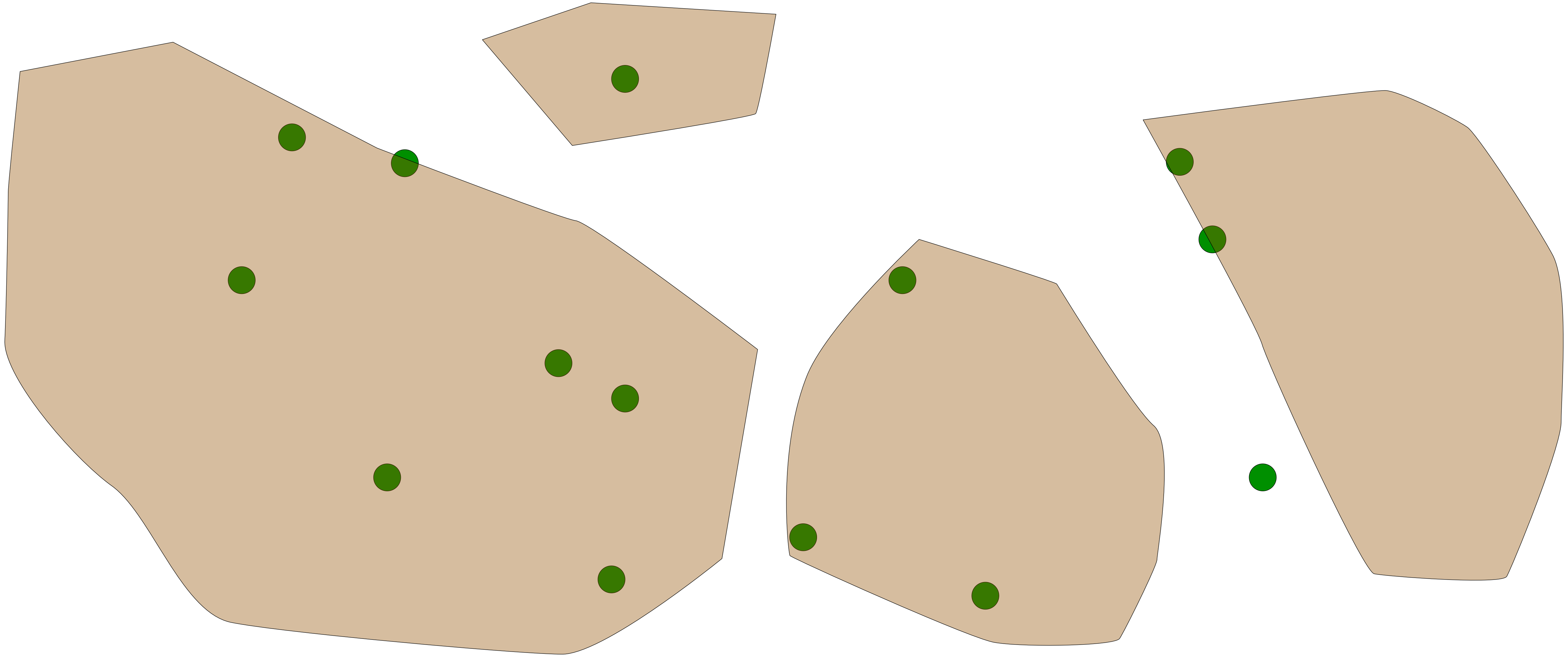
inverse ackerman

$$\alpha(n) =$$

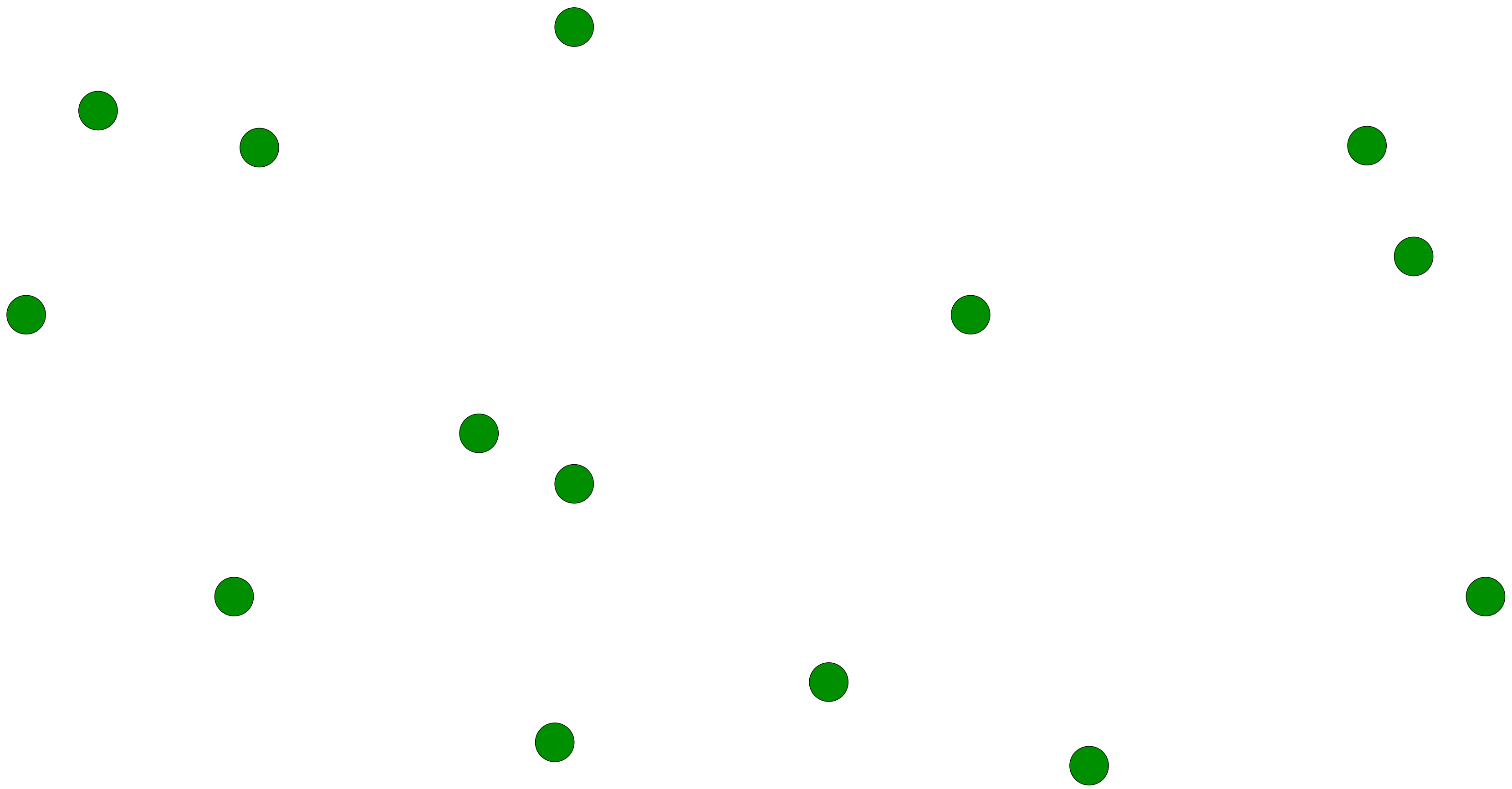
application of mst



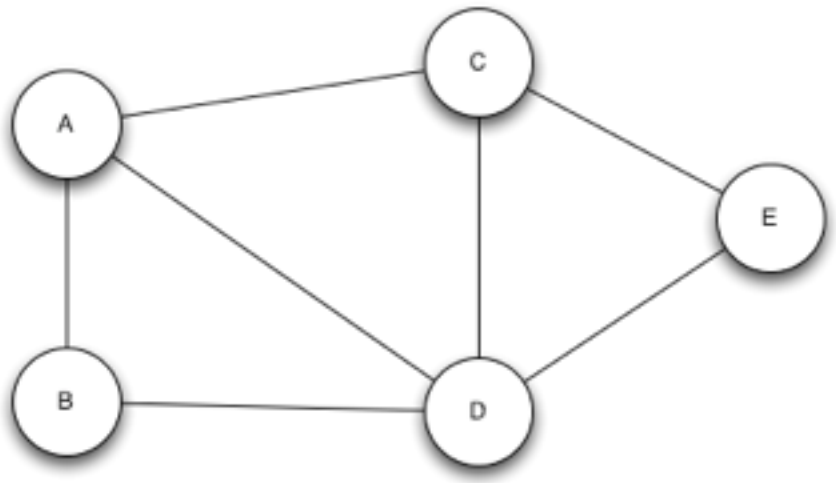
application of mst



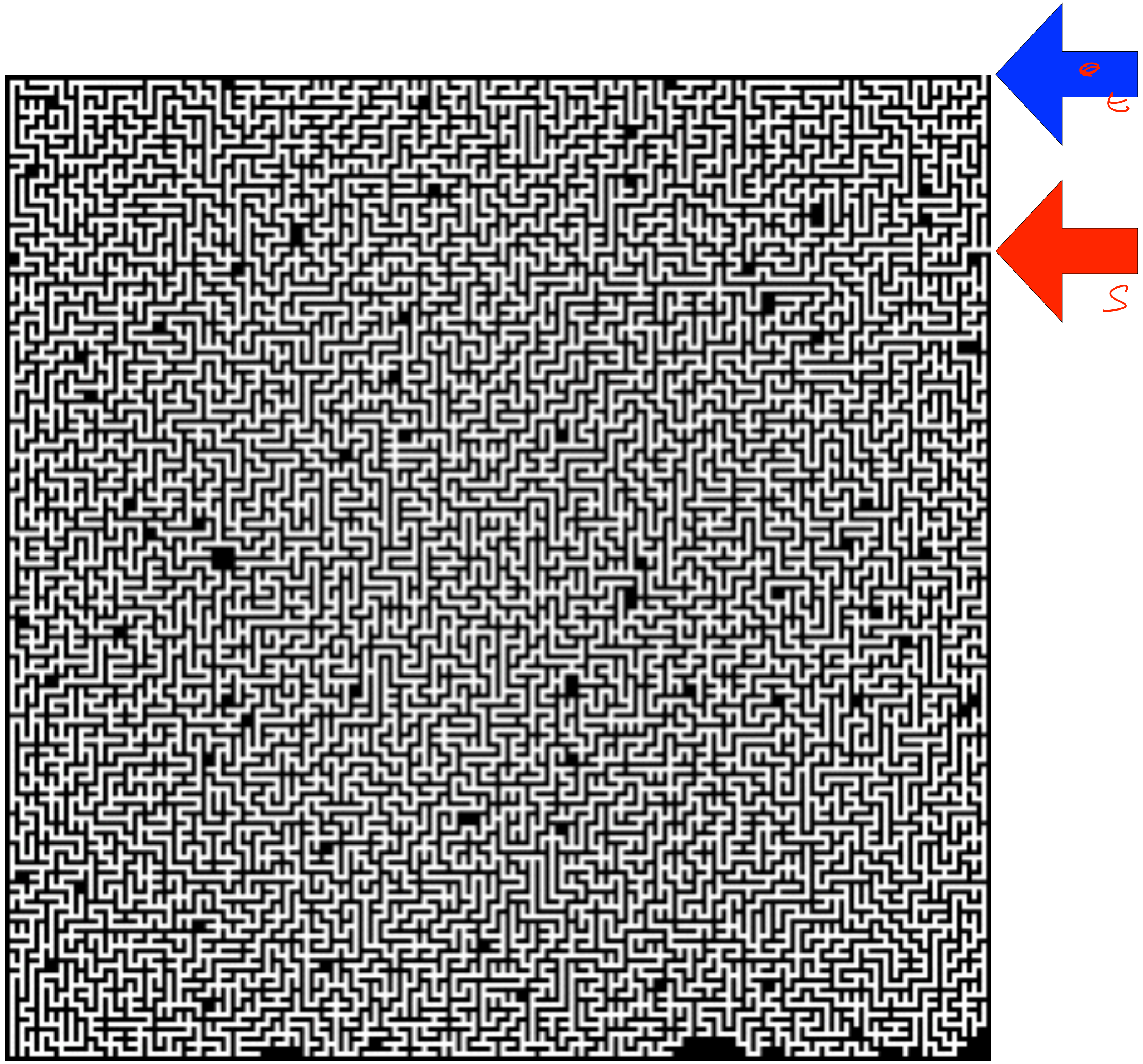
application of mst



simple graph questions



what is the length of the path from a to e?

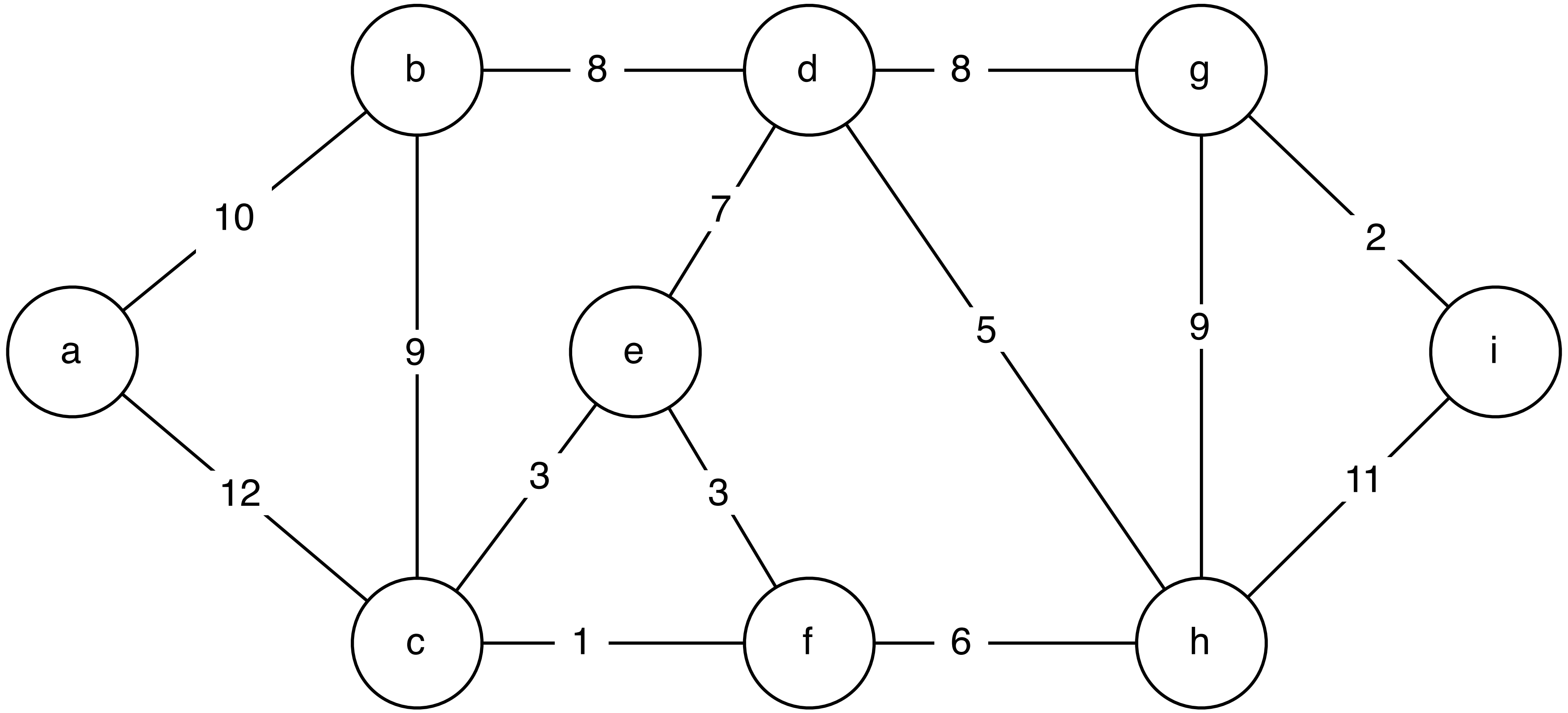

e
S

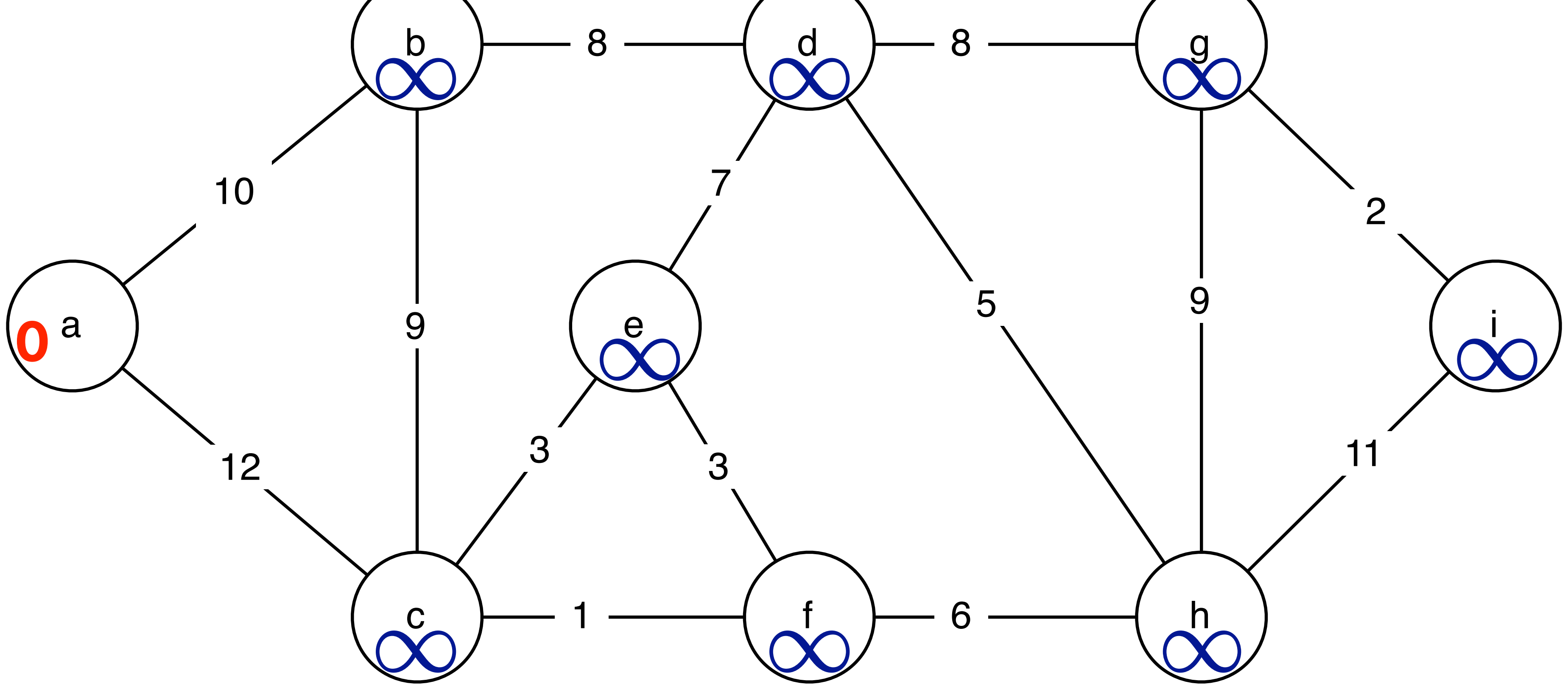
shortest path property

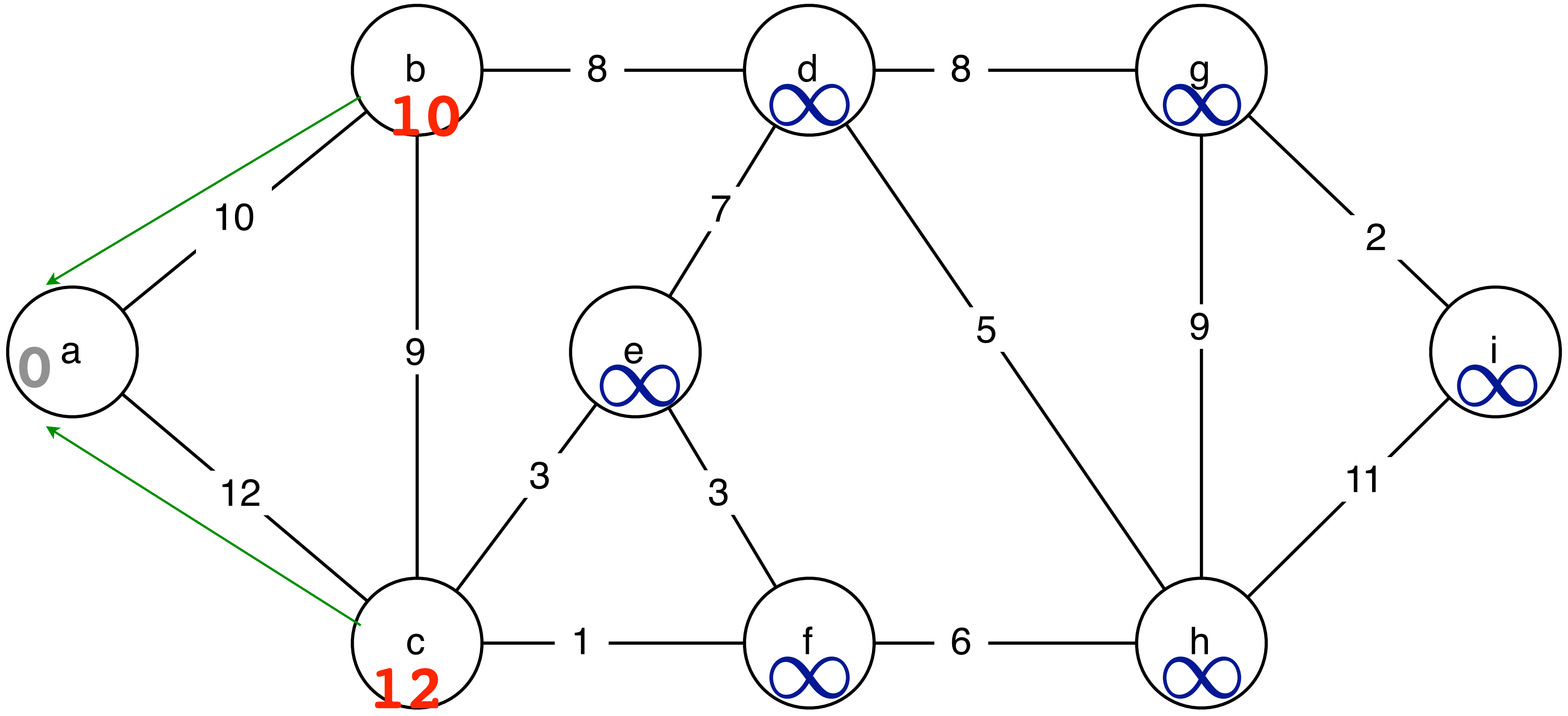
definition:

$$\delta(s, v)$$

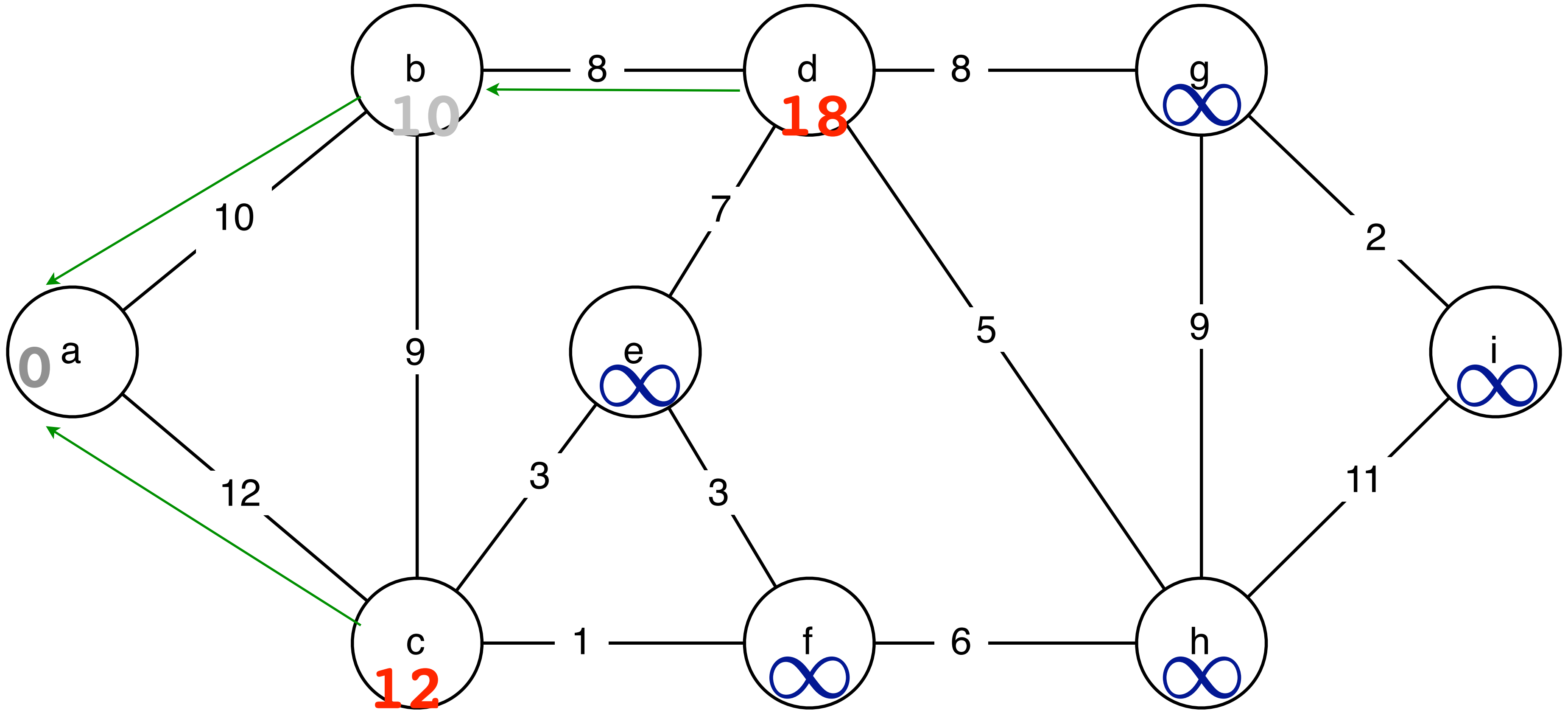
shortest paths

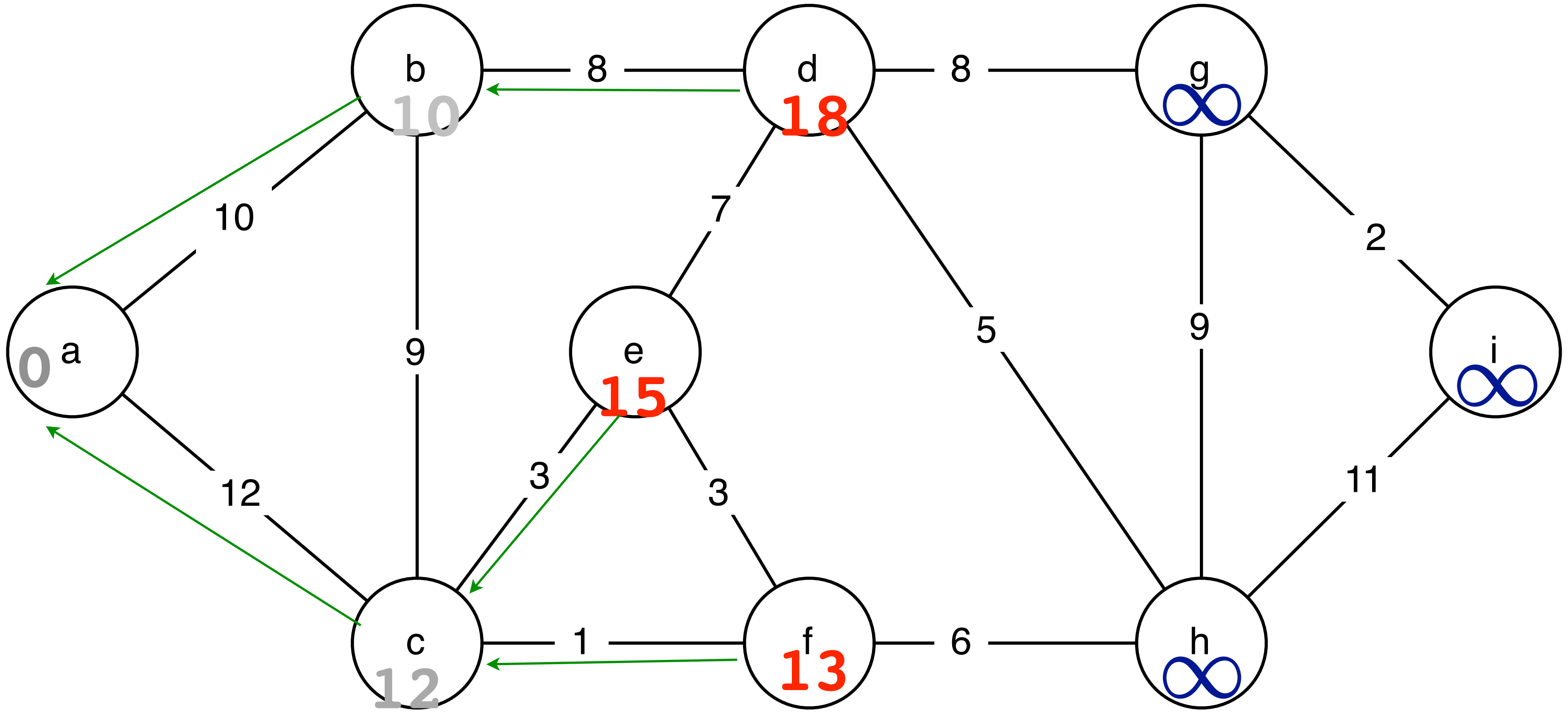


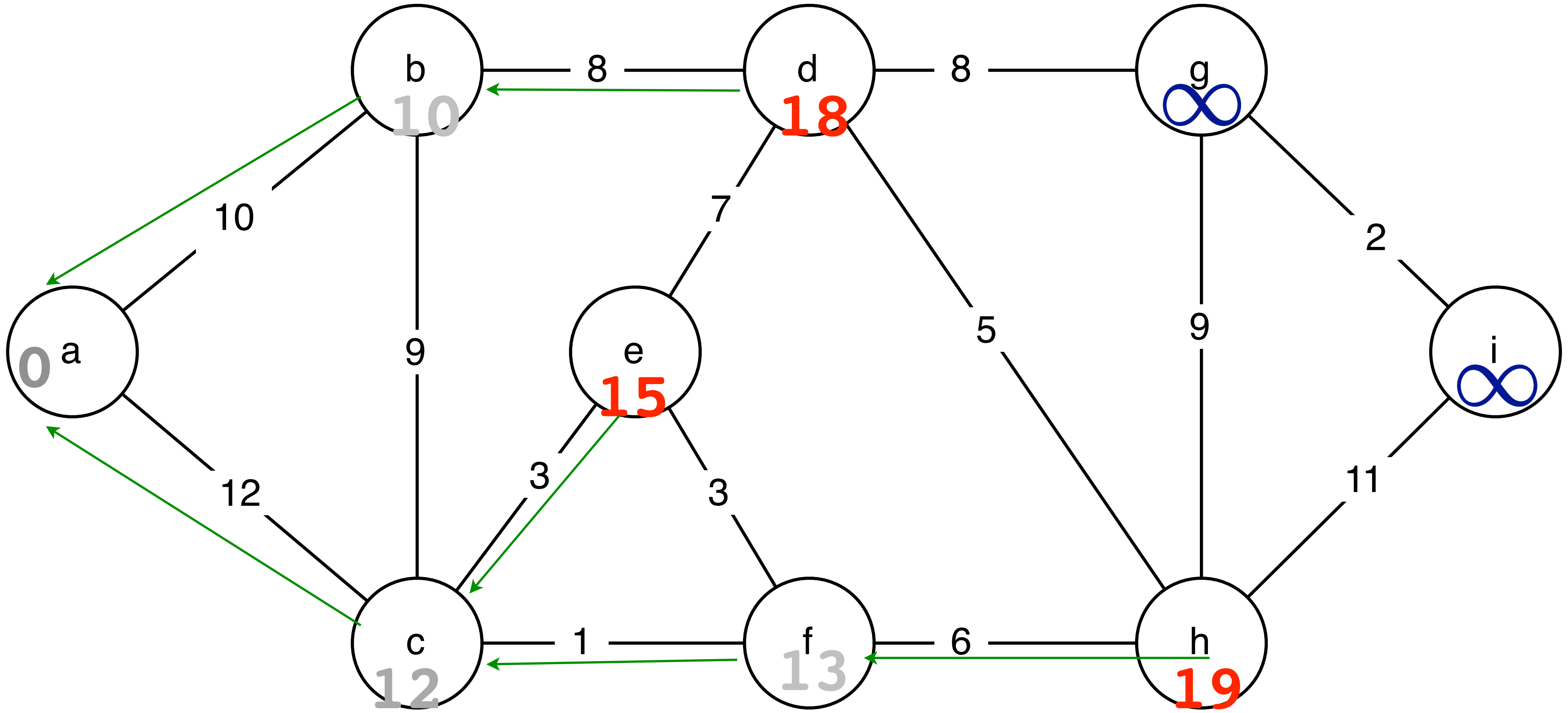

0
a
b
c
d
e
f
g
h
i
10
12
8
9
7
3
3
1
8
5
6
9
2
11
∞



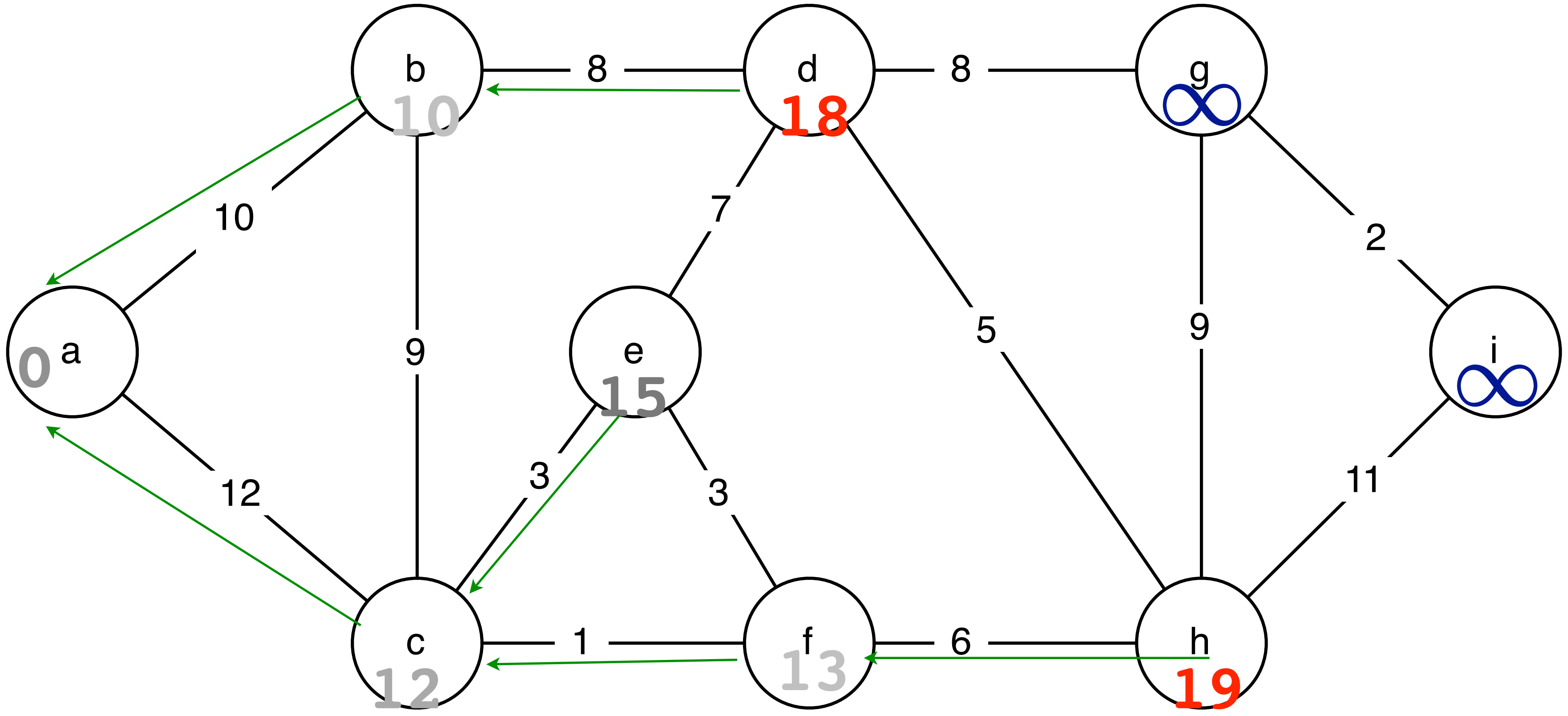
b
10
d
∞
g
∞
8
8
10
7
2
a
0
9
e
∞
5
9
i
∞
12
3
3
11
c
12
1
f
∞
6
h
∞


a
0
b
10
c
12
d
18
e
∞
f
∞
g
∞
h
∞
i
∞
8
8
10
9
7
5
9
2
12
3
3
1
6
11

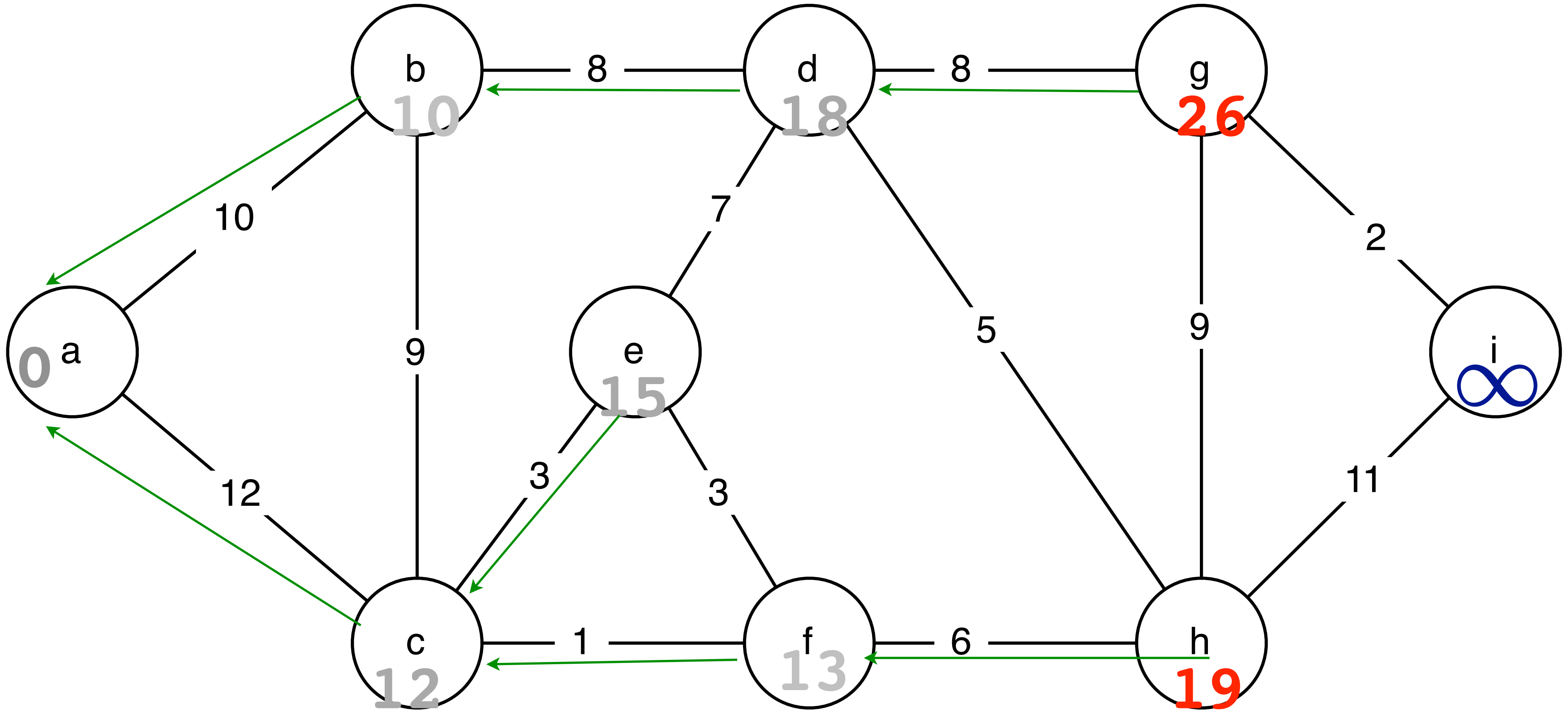

a
0
b
10
c
12
d
18
e
15
f
13
g
∞
h
∞
i
∞
10
12
9
8
7
3
3
1
8
5
9
6
2
11



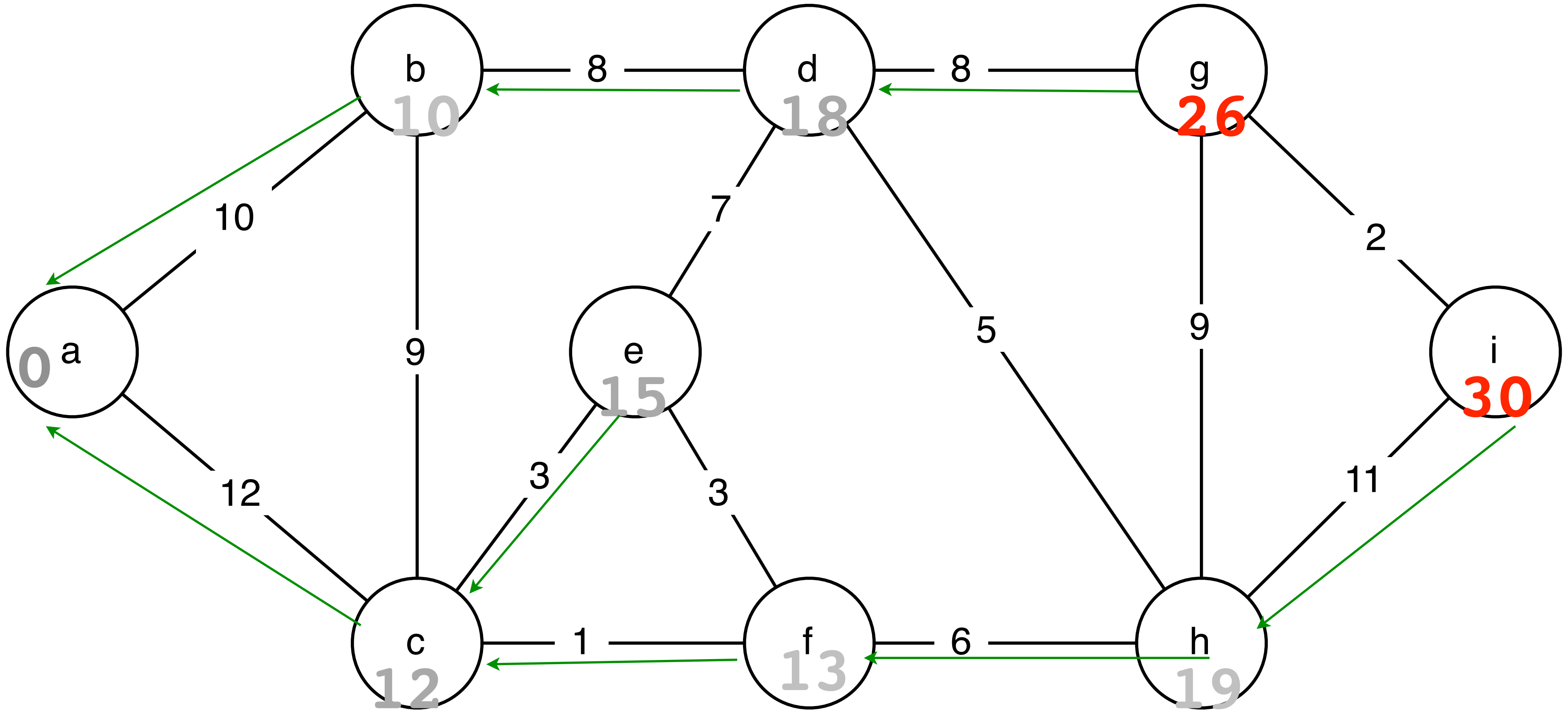
b
10
8
d
18
8
g
∞
10
7
5
9
2
a
0
9
e
15
i
∞
12
3
3
11
c
12
1
f
13
6
h
19



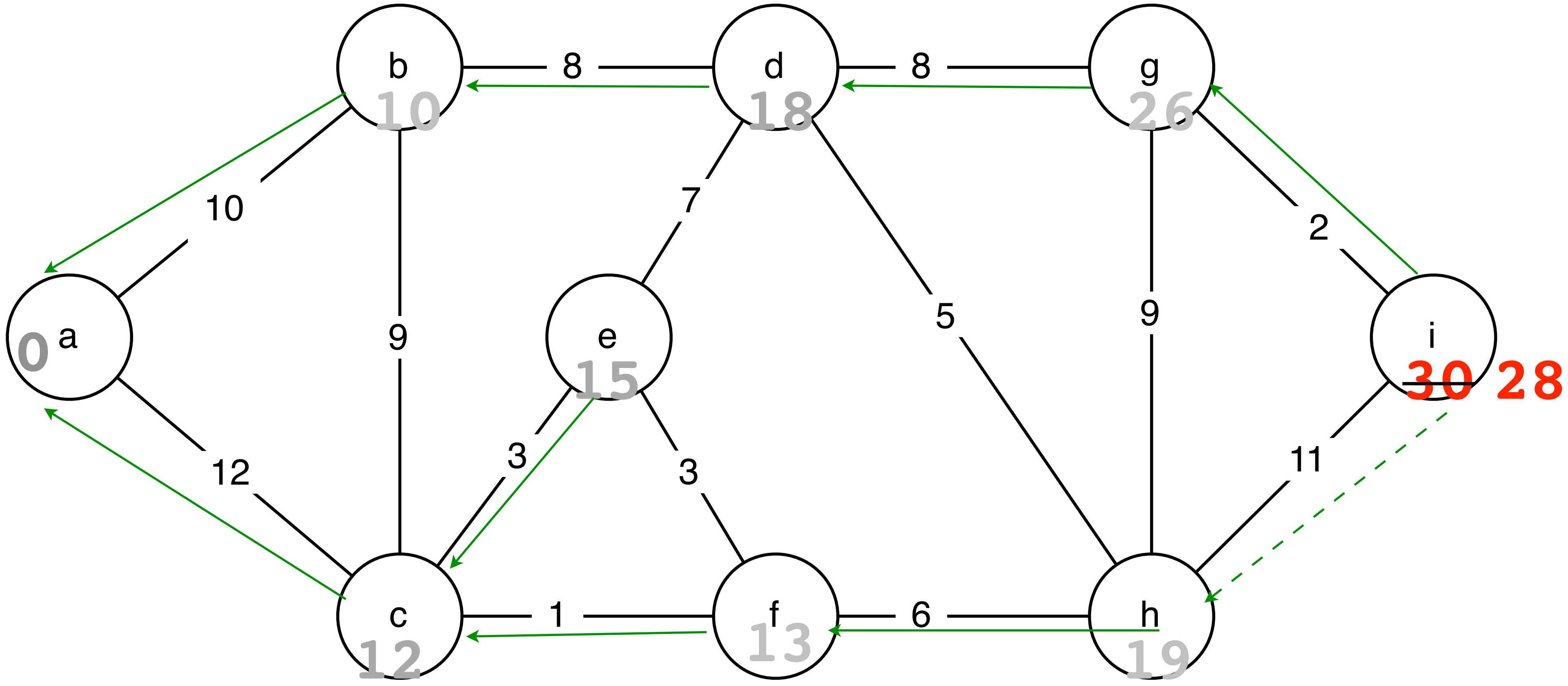
a
0
b
10
c
12
d
18
e
15
f
13
g
∞
h
19
i
∞
8
8
10
9
7
5
9
2
12
3
3
1
6
11



a
0
b
10
c
12
d
18
e
15
f
13
g
26
h
19
i
∞
8
8
10
9
7
5
9
2
12
3
3
11
1
6



a
0
b
10
c
12
d
18
e
15
f
13
g
26
h
19
i
30
8
8
10
9
7
5
9
2
12
3
3
1
6
11



a
0
b
10
c
12
d
18
e
15
f
13
g
26
h
19
i
~~30~~ 28
8
8
10
9
7
5
9
2
12
3
3
1
6
11

algorithm

DIJKSTRA$(G = (V, E), s)$

for all $v \in V$
  do $d_u \leftarrow \infty$
    $\pi_u \leftarrow$ NIL
$d_s \leftarrow 0$
$Q \leftarrow$ MAKEQUEUE(V) $\triangleright$ use d_u as key
while $Q \neq \emptyset$
  do $u \leftarrow$ EXTRACTMIN(Q)
    for each $v \in Adj(u)$
      do if $d_v > d_u + w(u, v)$
        then $d_v \leftarrow d_u + w(u, v)$
          $\pi_v \leftarrow u$
          DECREASEKEY(Q, v)

DIJKSTRA$(G = (V, E), s)$

for all $v \in V$
  do $d_u \leftarrow \infty$
    $\pi_u \leftarrow$ NIL
$d_s \leftarrow 0$
$Q \leftarrow$ MAKEQUEUE(V) ▷ use d_u as key
while $Q \neq \emptyset$
  do $u \leftarrow$ EXTRACTMIN(Q)
    for each $v \in Adj(u)$
      do if $d_v > d_u + w(u, v)$
        then $d_v \leftarrow d_u + w(u, v)$
          $\pi_v \leftarrow u$
          DECREASEKEY(Q, v)

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

$Q \leftarrow \emptyset$ ▷ 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$(Q, v, w(u, v))$ ▷ Sets $k_v \leftarrow w($

running time

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

for all $v \in V$
  do $d_u \leftarrow \infty$
    $\pi_u \leftarrow \textsc{nil}$
$d_s \leftarrow 0$
$Q \leftarrow \textsc{makequeue}(V)$ $\triangleright$ use d_u as key
while $Q \neq \emptyset$
  do $u \leftarrow \textsc{extractmin}(Q)$
    for each $v \in Adj(u)$
      do if $d_v > d_u + w(u, v)$
        then $d_v \leftarrow d_u + w(u, v)$
          $\pi_v \leftarrow u$
          $\textsc{decreasekey}(Q, v)$

why does dijkstra work?

triangle inequality:

$$\forall (u,v) \in E,\ \delta(s,v) \leq \delta(s,u) + w(u,v)$$

upper bound:

$$d_v \geq \delta(s,v)$$

breadth first search

input: $G = (V, E), s$

output:

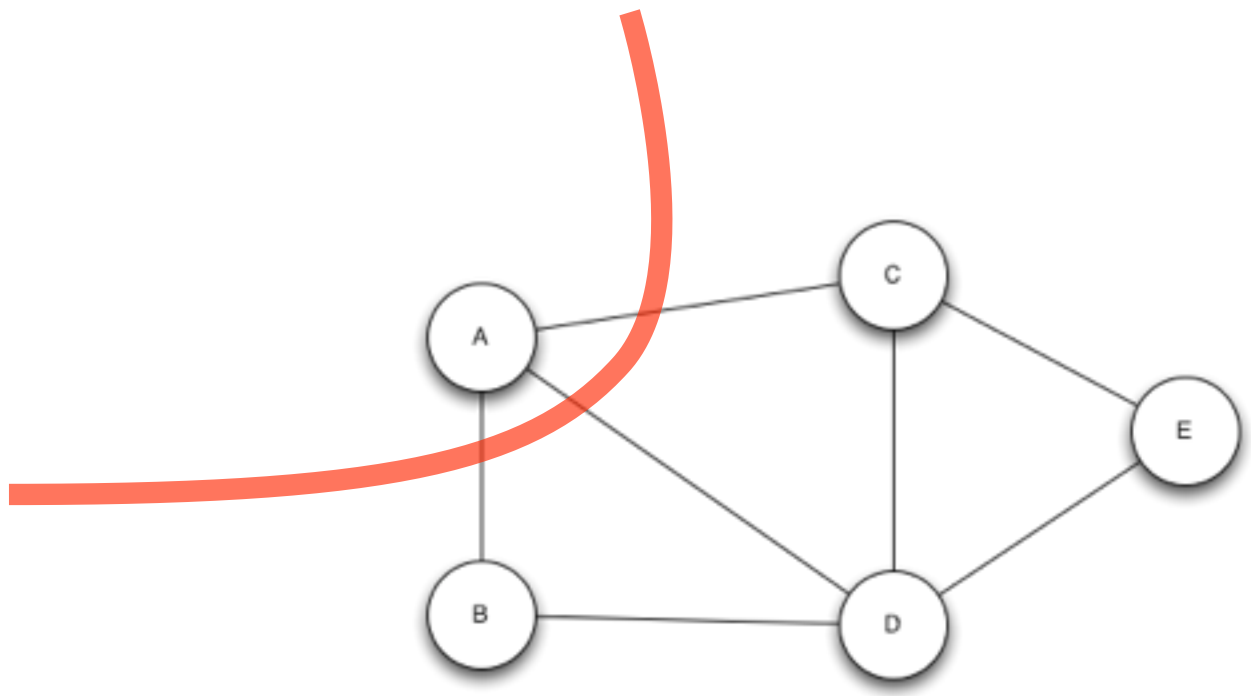
breadth first search

input: $G = (V, E), s$

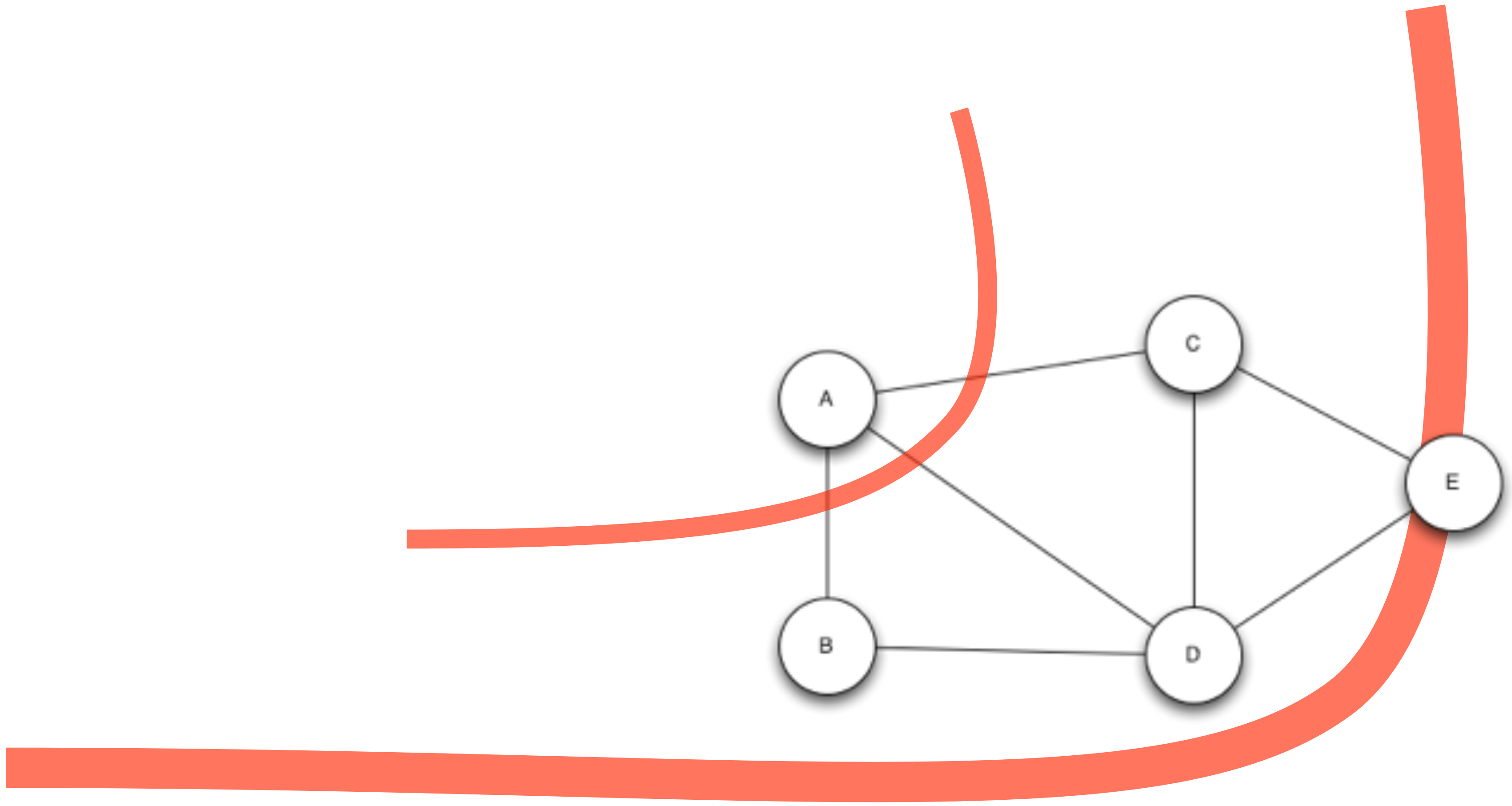
output: $\forall v \in V \quad d_v = \delta(s, v)$

smallest # of edges from s to v

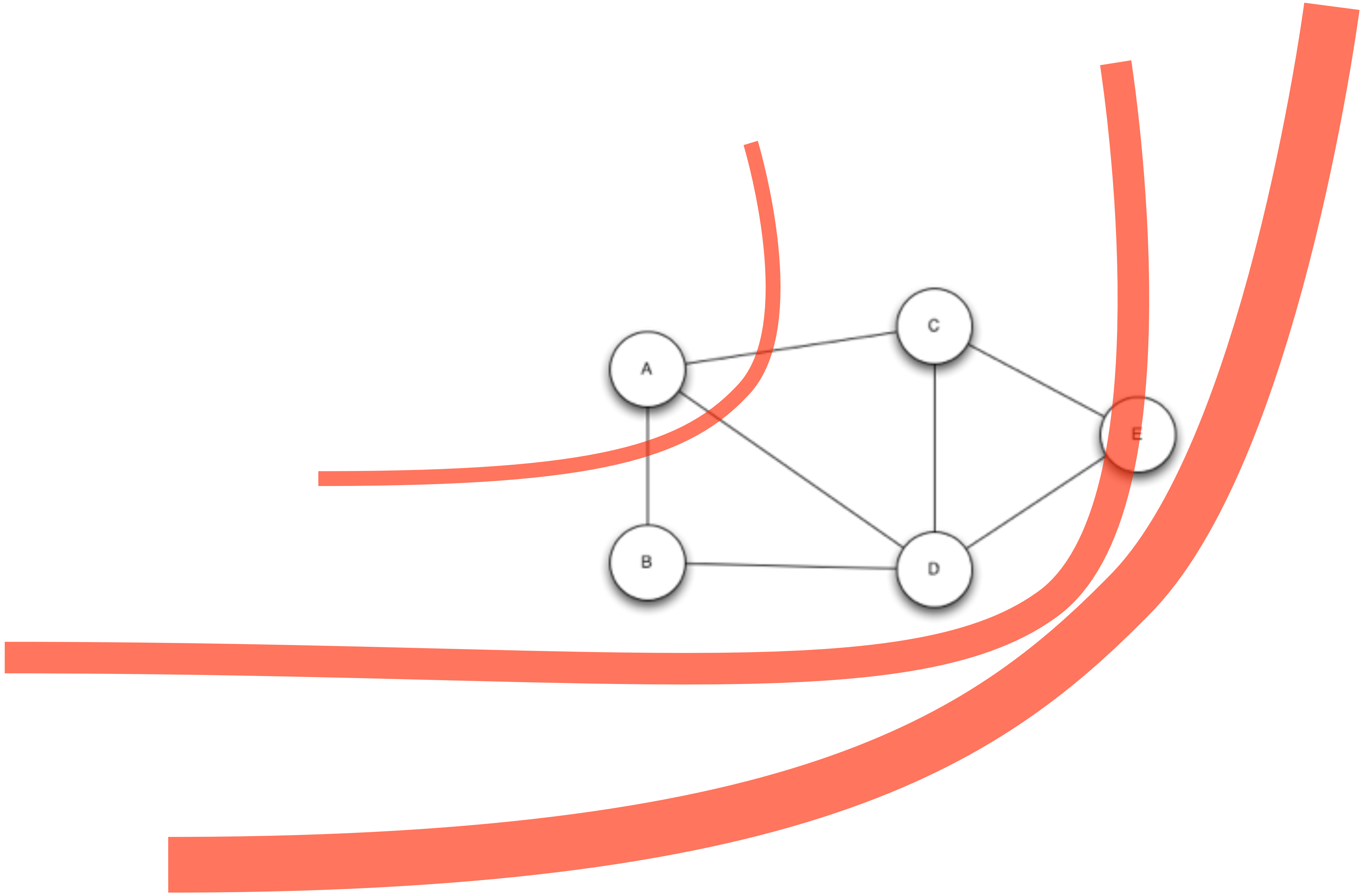
breadth-first search



breadth-first search



breadth-first search

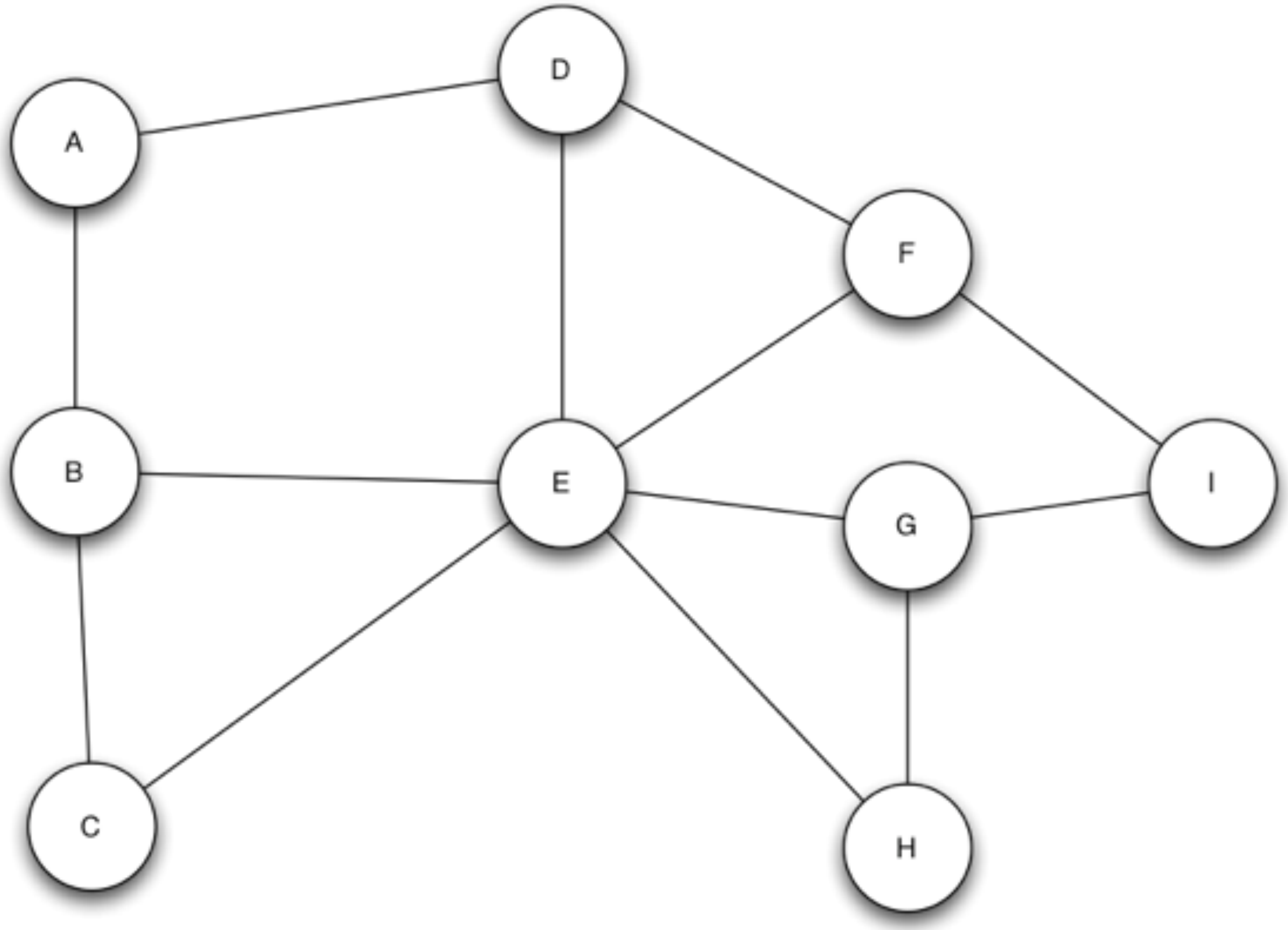


breadth first search

input: $G = (V, E), s$

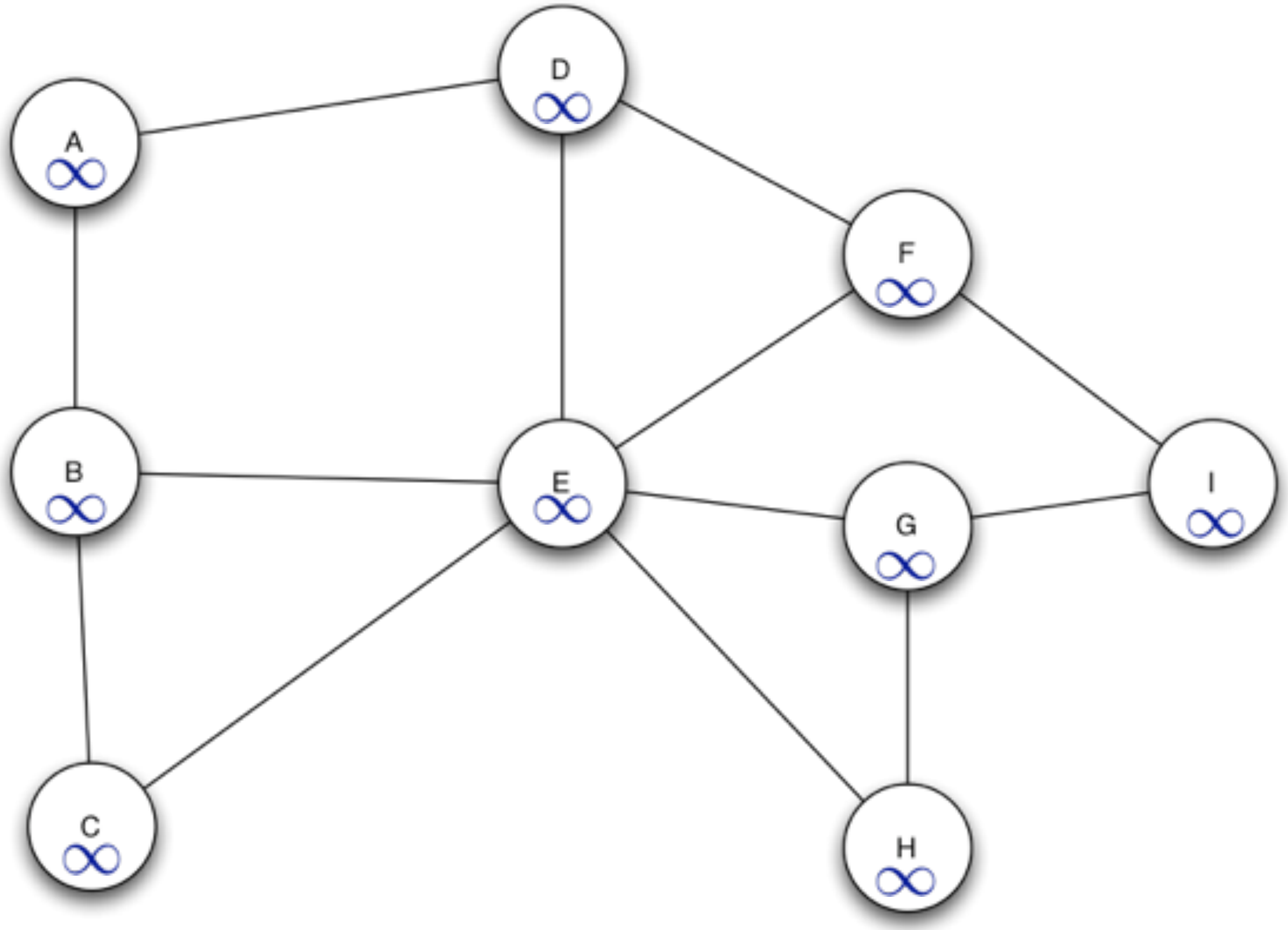
output: smallest # of edges from s to v $d_v \quad \forall v \in V$

bfs(G, a)



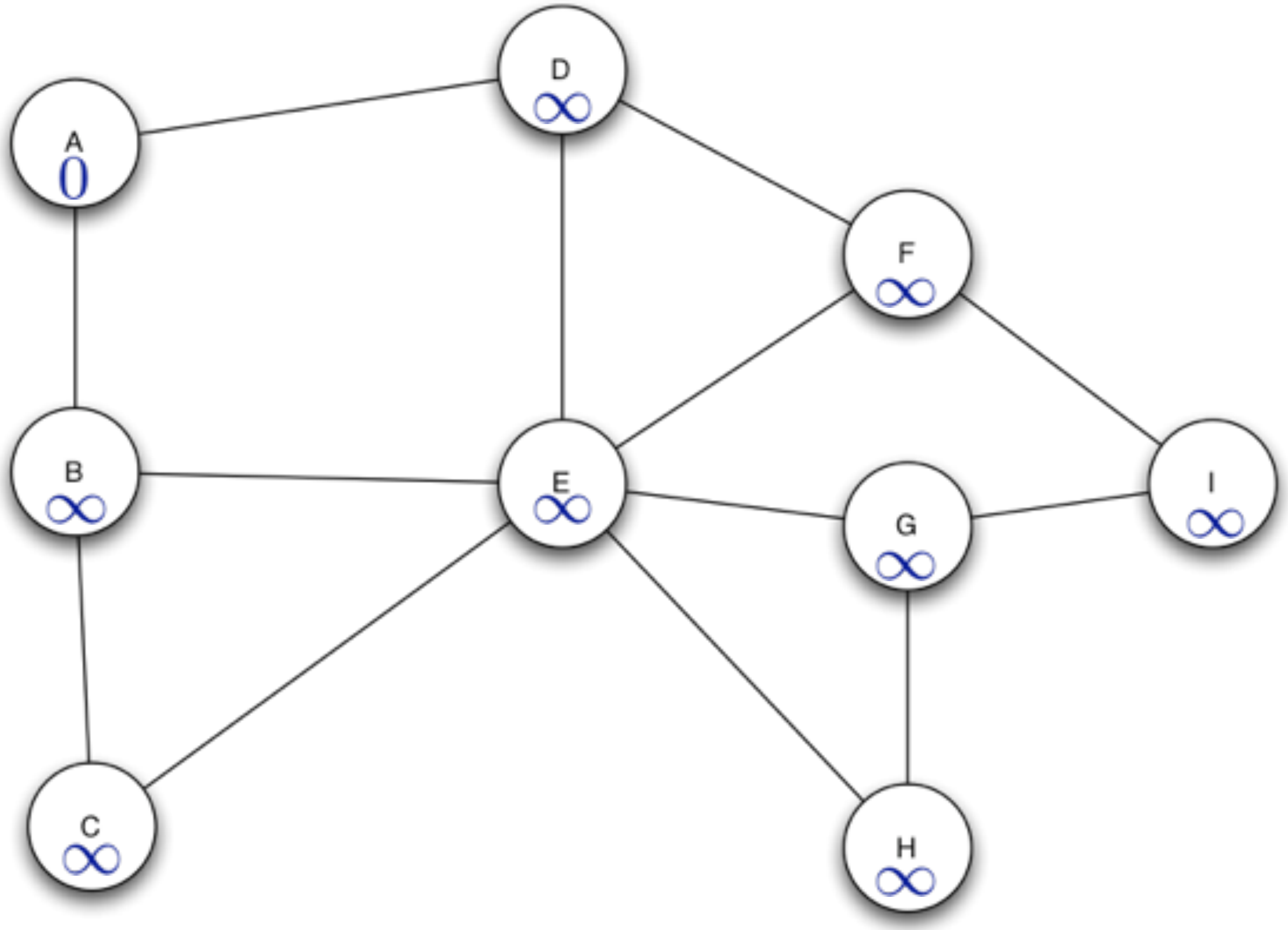
Q

bfs(G, a)



Q

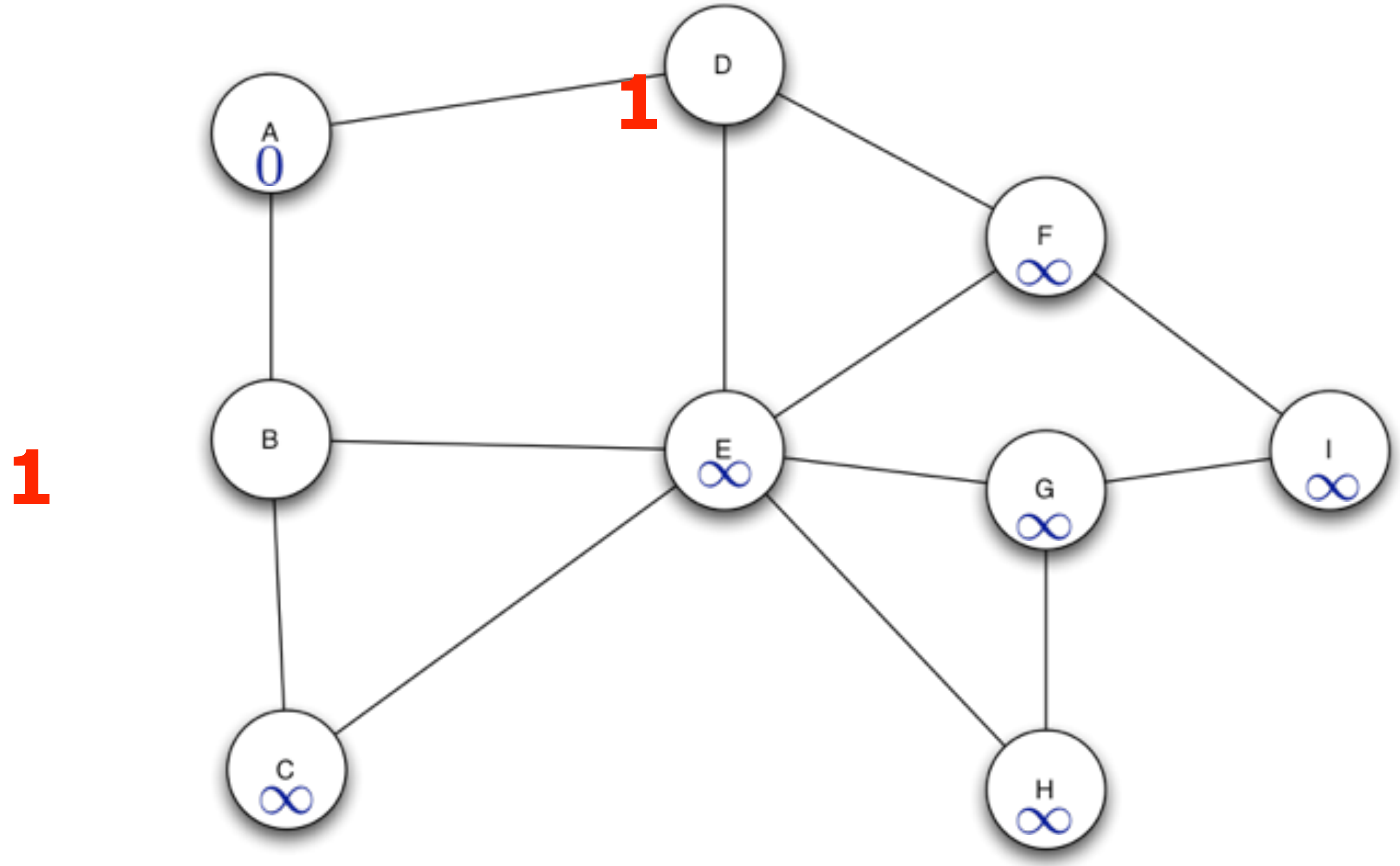
bfs(G, a)



Q

a

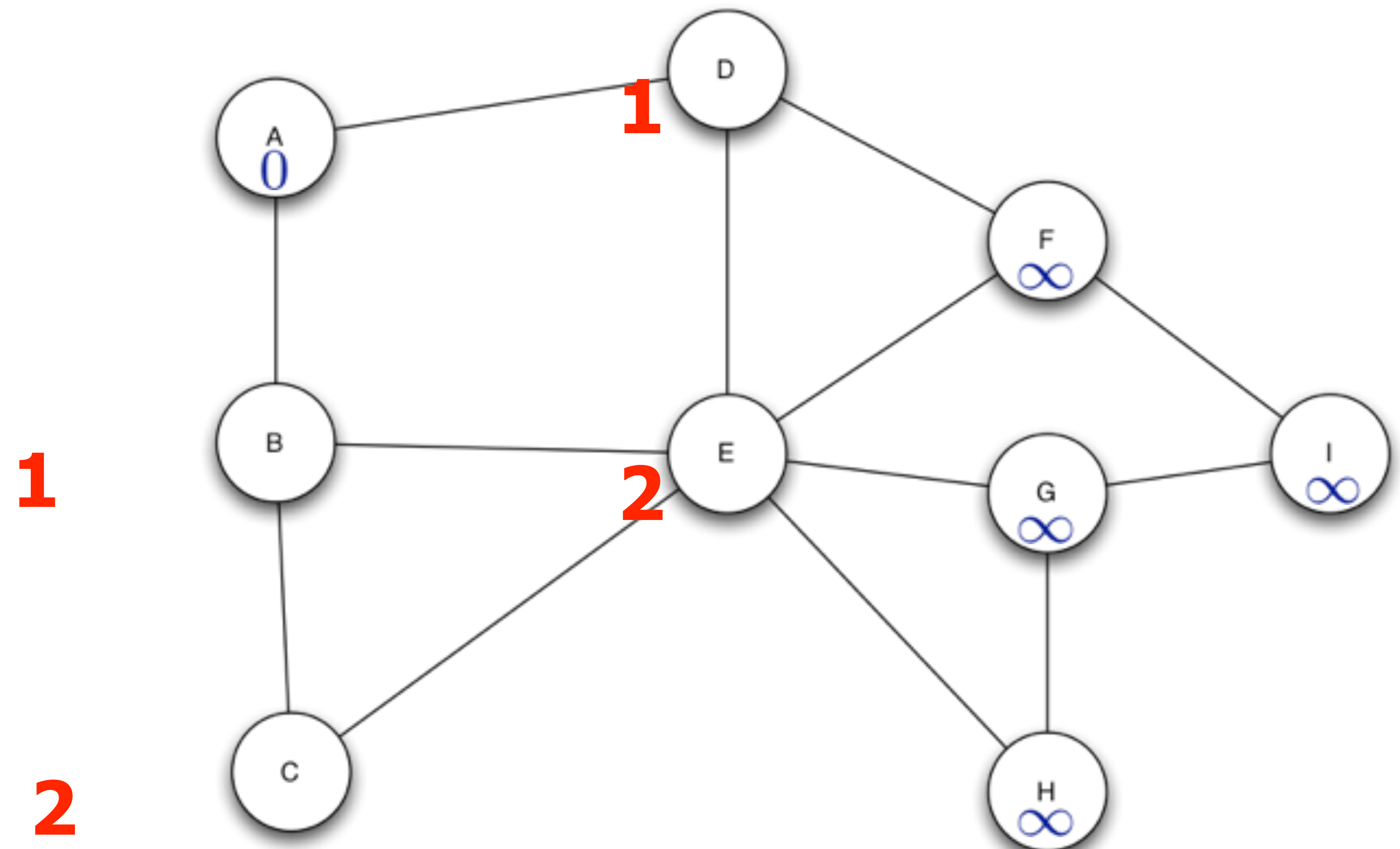
bfs(G, a)



Q

a
b
d

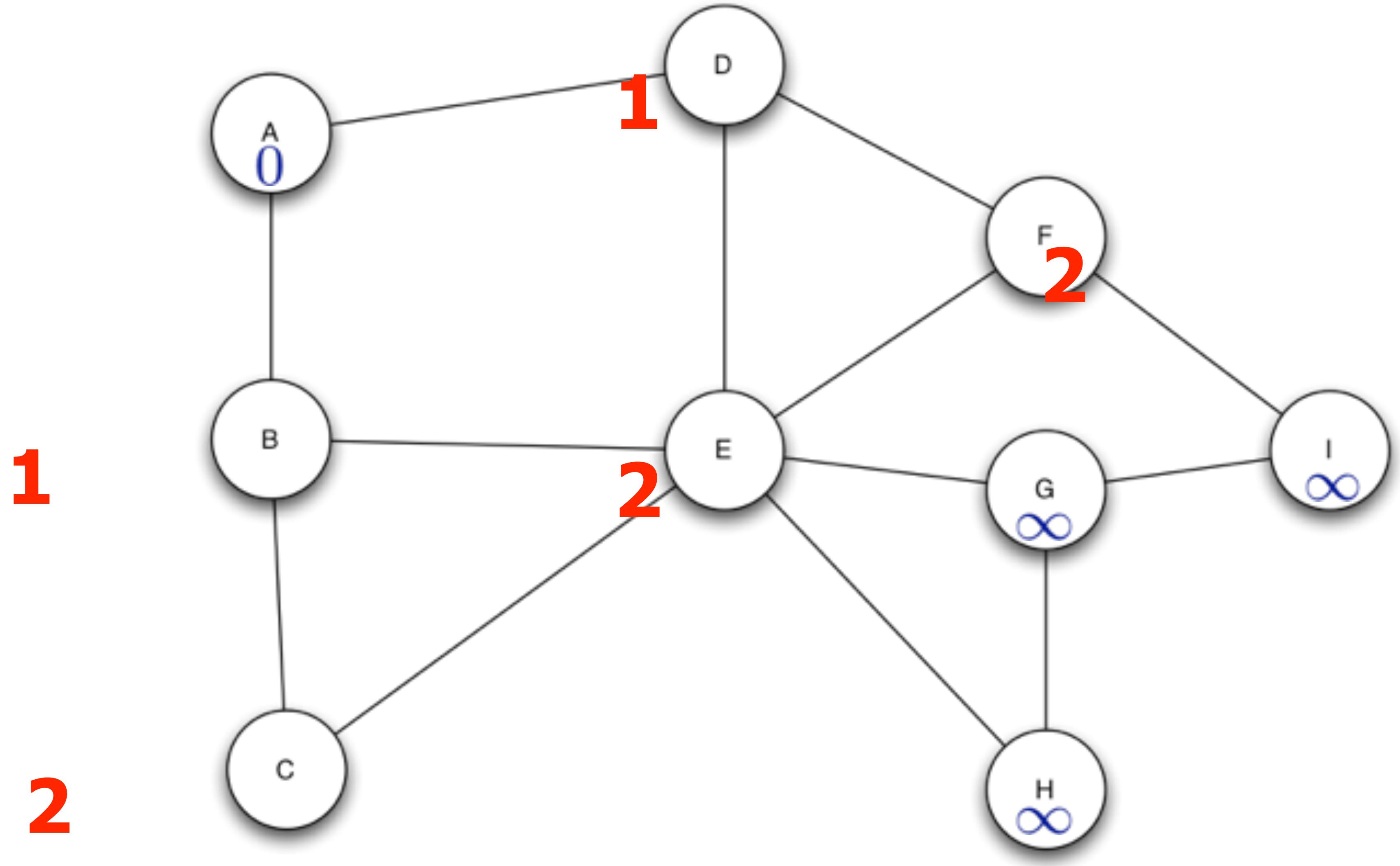
bfs(G, a)



$\mathcal{Q}$

a
b
d
c
e

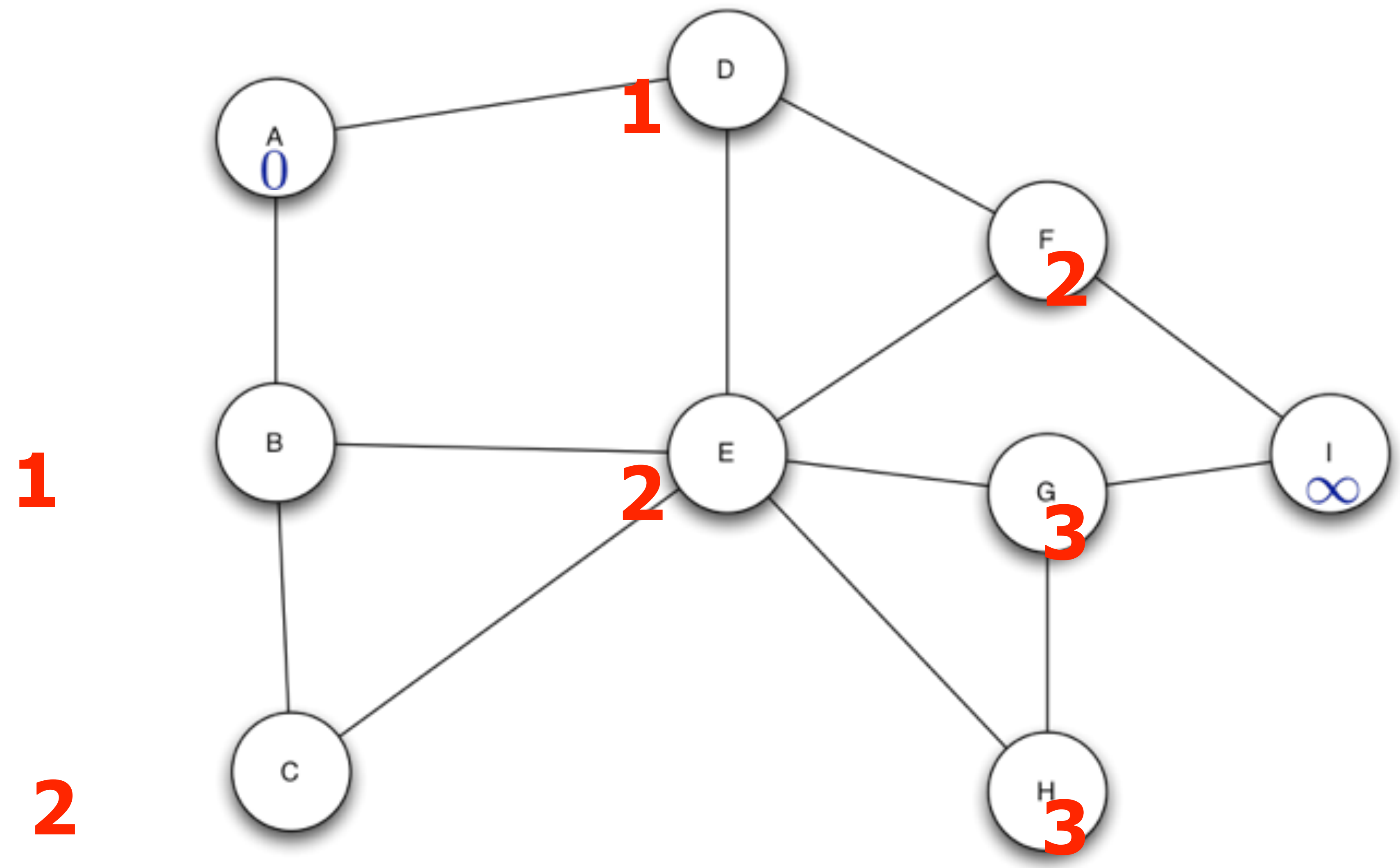
bfs(G, a)



Q

a
b
d
c
e
f

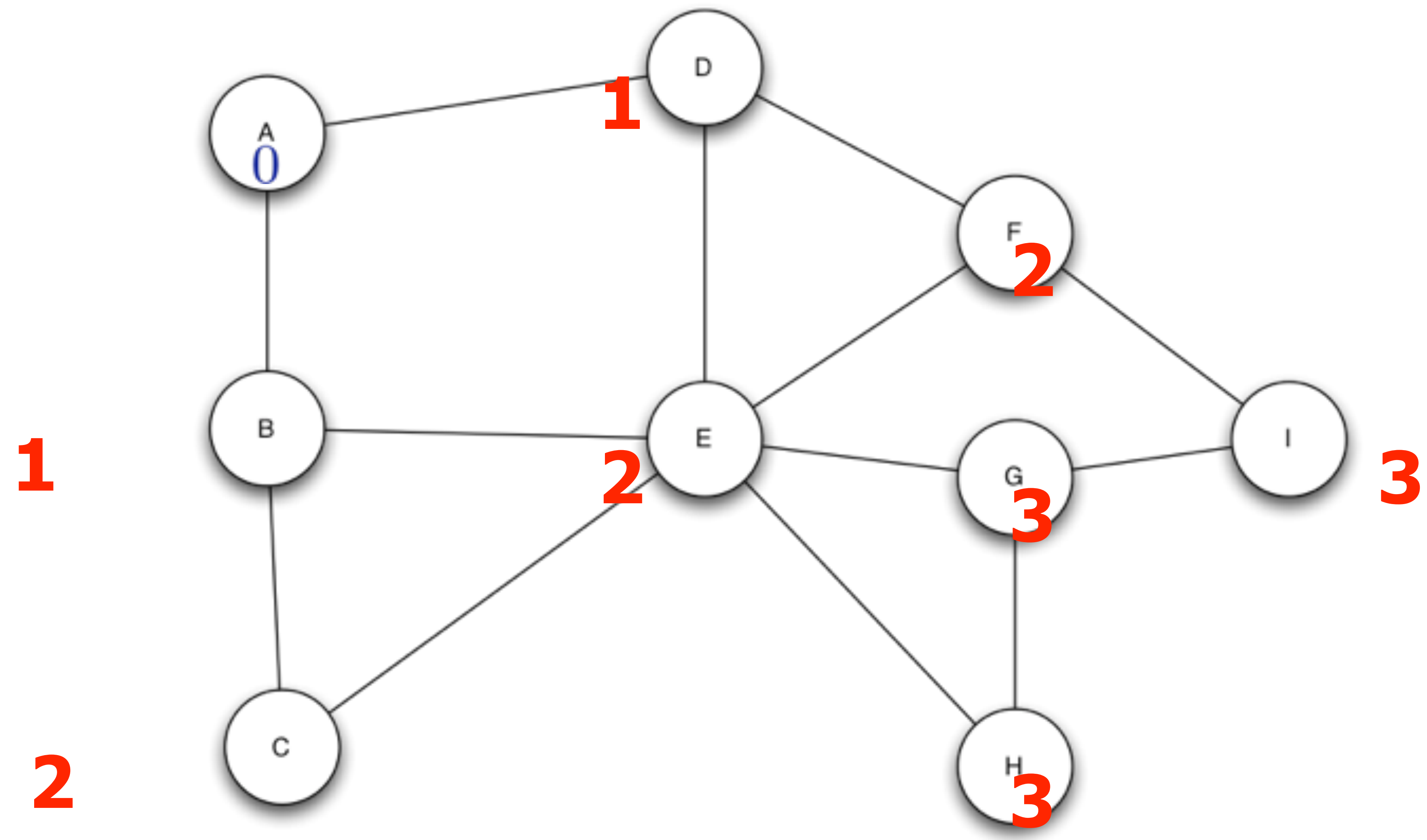
bfs(G, a)



$$Q$$

a
b
d
c
e
f
g
h

bfs(G, a)



Q

a
b
d
c
e
f
g
h

bfs(G, a)

breadth first search

BFS(V, E, s)
for each $u \in V - \{s\}$
  do $d[u] \leftarrow \infty$
$d[s] \leftarrow 0$
$Q \leftarrow \emptyset$
ENQUEUE(Q, s)
while $Q \neq \emptyset$
  do $u \leftarrow$ DEQUEUE(Q)
    for each $v \in Adj[u]$
      do if $d[v] = \infty$
        then $d[v] \leftarrow d[u] + 1$
          ENQUEUE(Q, v)

bfs theorem