L15

4102

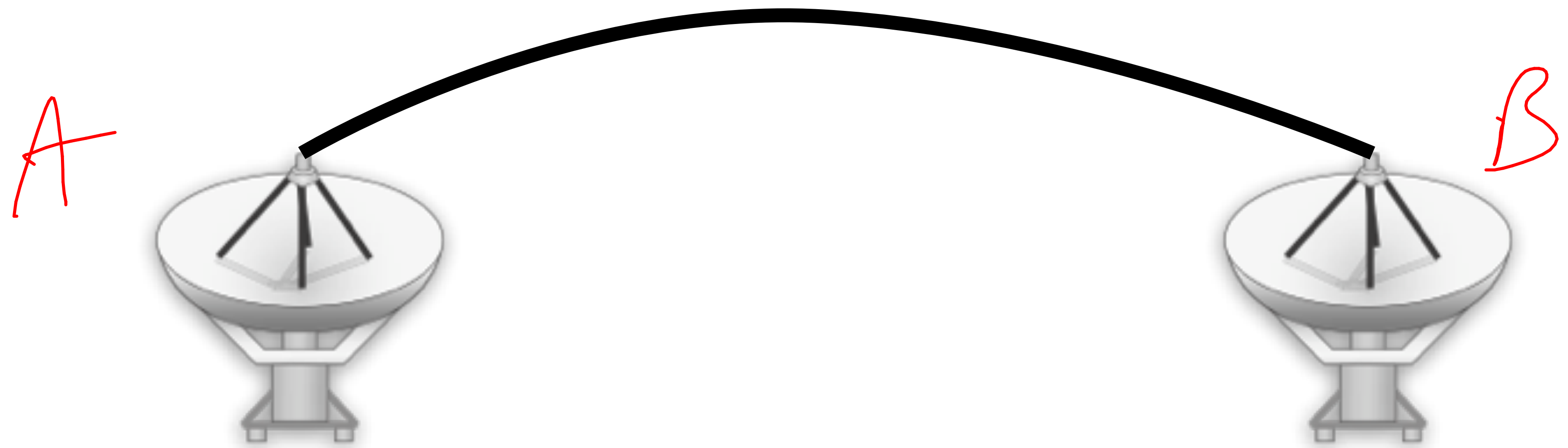
10.17.2013

abhi shelat

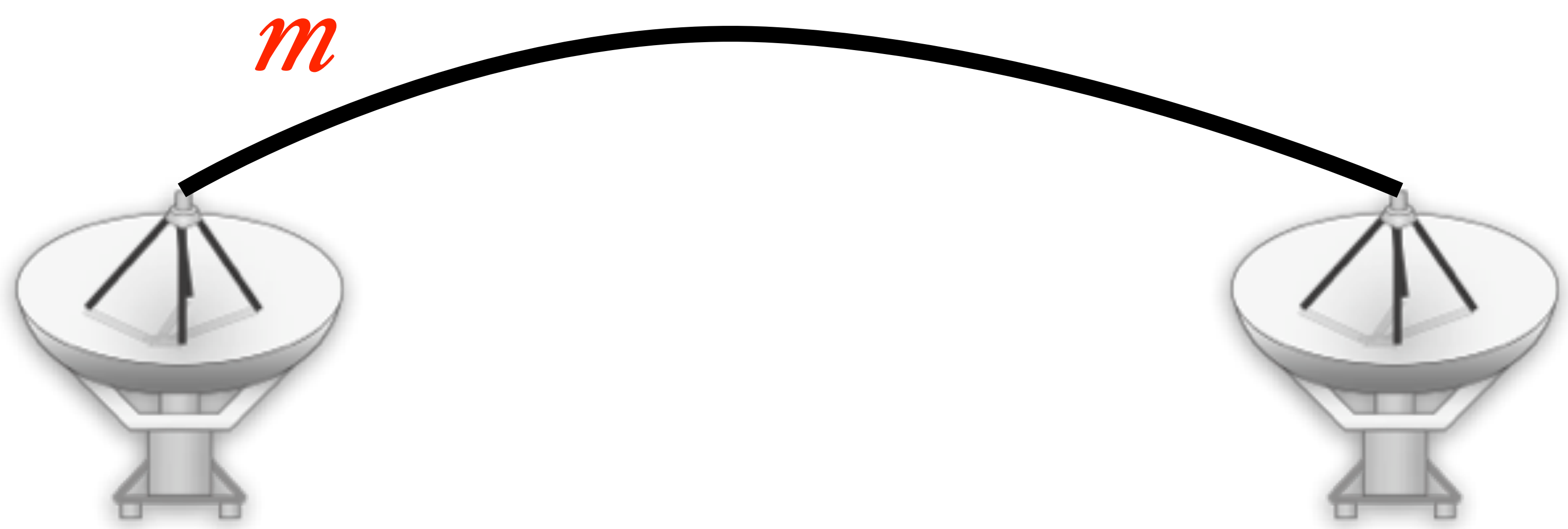
Greedy Alg:
Huffman

Huffman Coding


image: wikimedia



In testimony before the committee, Mr. Lew stressed that the Treasury Department would run out of “extraordinary measures” to free up cash in a matter of days. At that point, the country’s bills might overwhelm its cash on hand plus any receipts from taxes or other sources, leading to an unprecedented default.Mr. Lew said that Treasury had no workarounds to avoid breaching the debt ceiling. “There is no plan other than raising the debt limit,” he said. “The legal issues, even regarding interest and principal on the debt, are complicated.”



In testimony before the committee, Mr. Lew stressed that the Treasury Department would run out of “extraordinary measures” to free up cash in a matter of days. At that point, the country’s bills might overwhelm its cash on hand plus any receipts from taxes or other sources, leading to an unprecedented default.Mr. Lew said that Treasury had no workarounds to avoid breaching the debt ceiling. “There is no plan other than raising the debt limit,” he said. “The legal issues, even regarding interest and principal on the debt, are complicated.”

def: cost of an encoding

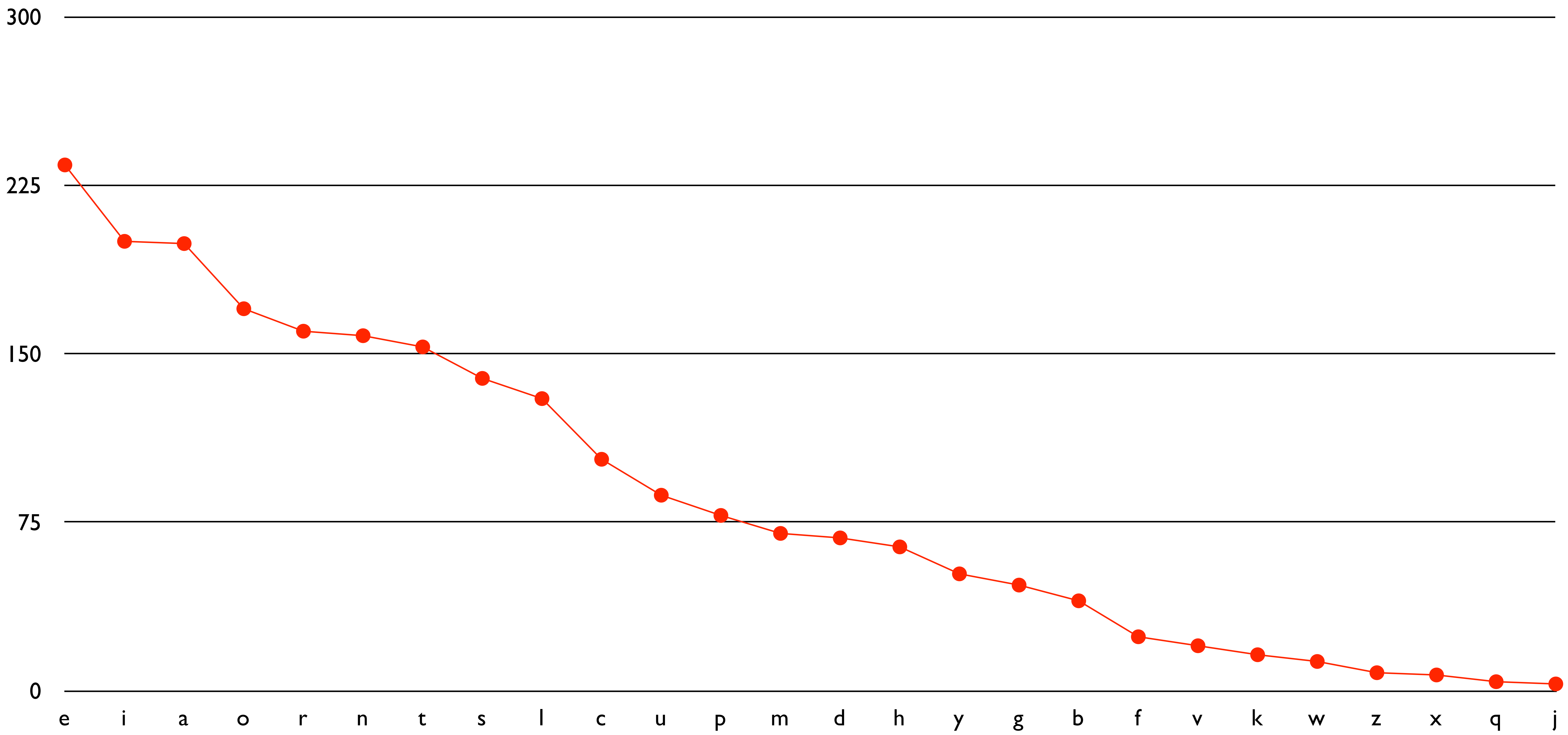
cost encoding for frequencies

$$B(T, \{f_c\}) = \sum_{c \in C} f_c \cdot \ell_c$$

$c \in C$	f_c	T	ℓ_c
e:	235	000	3
i:	200	001	3
o:	170	010	3
u:	87	011	3
p:	78	100	3
g:	47	101	3
b:	40	110	3
f:	24	111	3
	881		

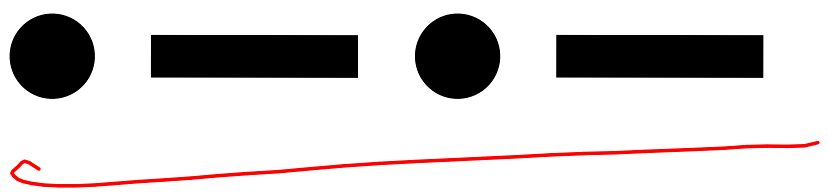
character frequency

e: 234803
i: 200613
a: 198938
o: 170392
r: 160491
n: 158281
t: 152570
s: 139238
l: 130172
c: 103307
u: 87211
p: 78077
m: 70504
d: 68007
h: 64165
y: 51527
g: 47011
b: 40351
f: 24110
v: 20103
k: 16012
w: 13825
z: 8439
x: 6926
q: 3729
j: 3075



morse code


International Morse Code
- 1 dash = 3 dots.
- The space between parts of the same letter = 1 dot.
- The space between letters = 3 dots.
- The space between words = 7 dots.
A B C D E F G H I J K L M N O P Q R S T U
V W X Y Z . , ? / @ 1 2 3 4 5 6 7 8 9 0



def: prefix-free code

$$\forall x, y \in C, x \neq y \implies \textsc{code}(x) \text{ not a prefix of } \textsc{code}(y)$$

e:	235	0
i:	200	10
o:	170	110
u:	87	1110
p:	78	11110
g:	47	111110
b:	40	1111110
f:	24	11111110

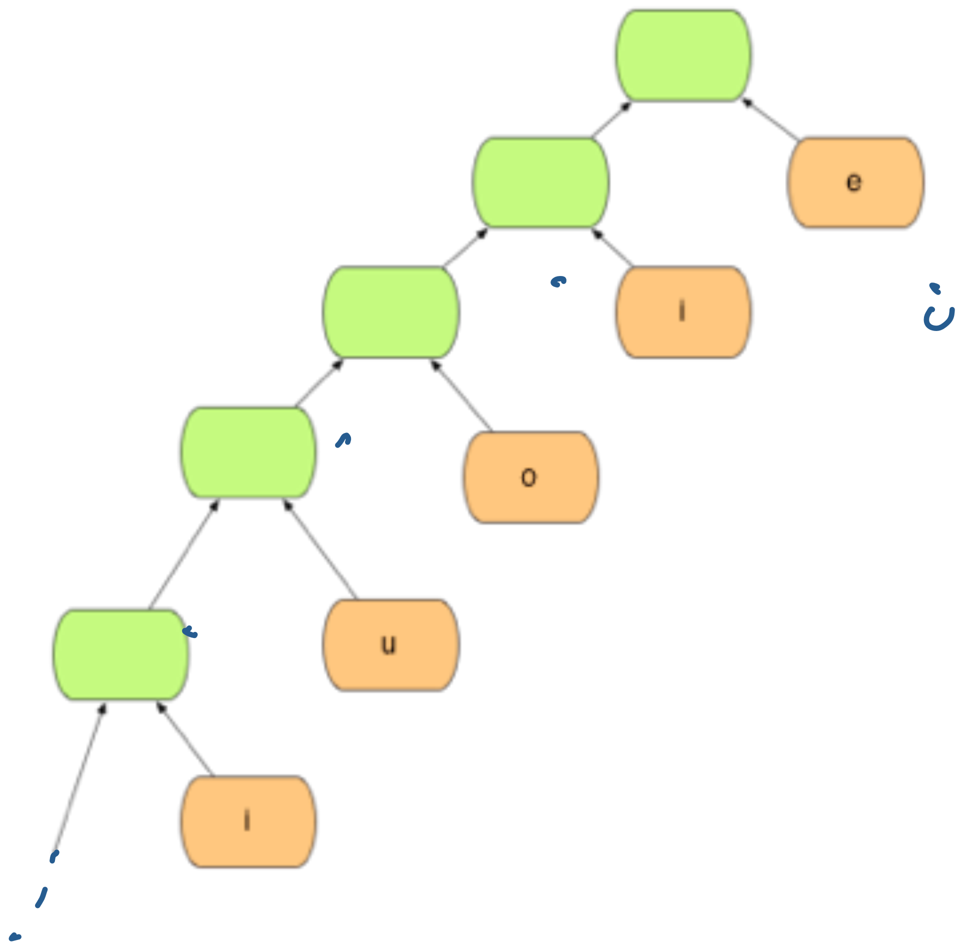
prefix free code

prefix code

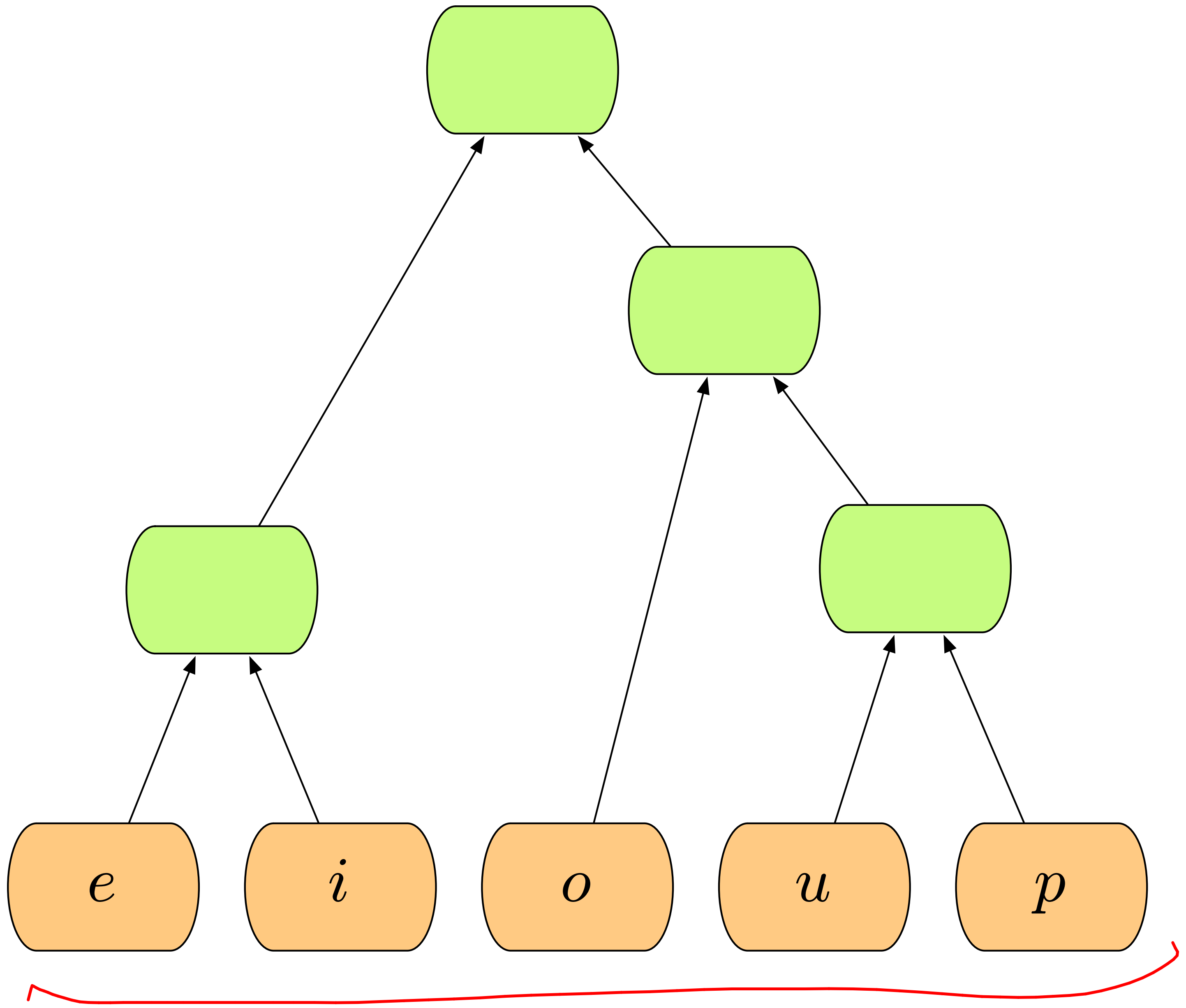
binary tree

code to binary tree

e:	235	0
i:	200	10
o:	170	110
u:	87	1110
p:	78	11110
g:	47	111110
b:	40	1111110
f:	24	11111110



1111110101111110



$c \in C$	f_c	T	ℓ_c
e:	235	00	2
i:	200	01	2
o:	170	10	2
u:	87	110	3
p:	78	111	3

use tree to encode messages

goal

INPUT

given the character frequencies

$$\{f_c\}_{c \in C}$$

Alphabet

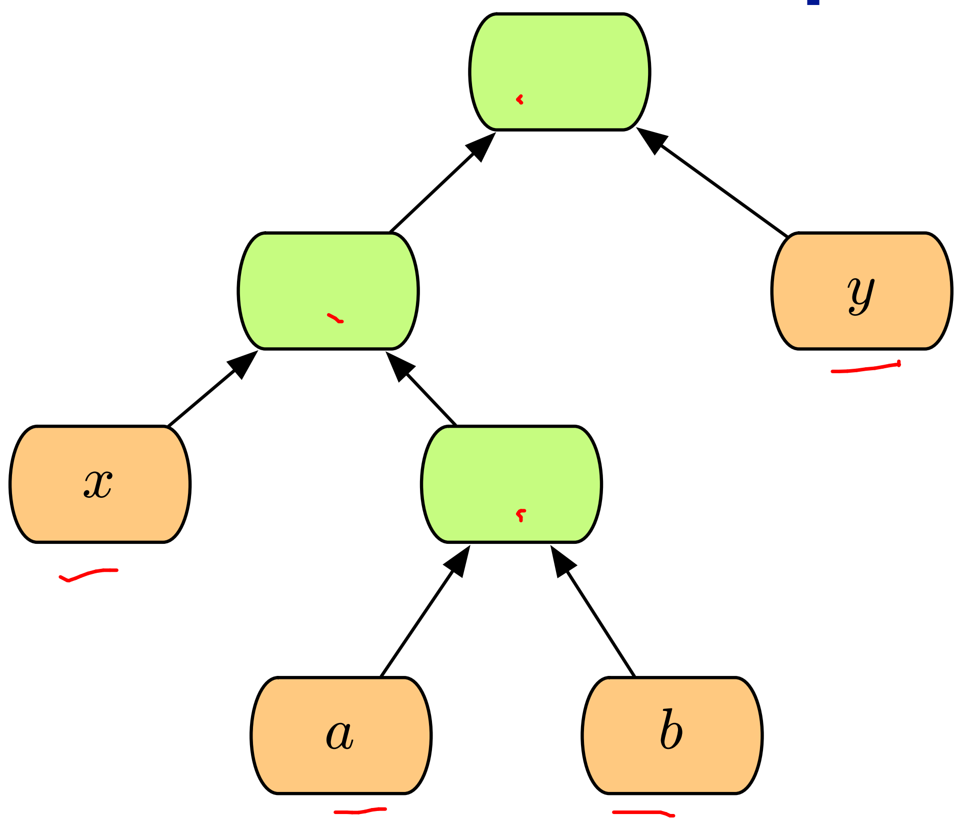
produce a prefix code T with smallest cost

free

$$\min_T B(T, \{f_c\})$$

smallest cost for the input frequencies

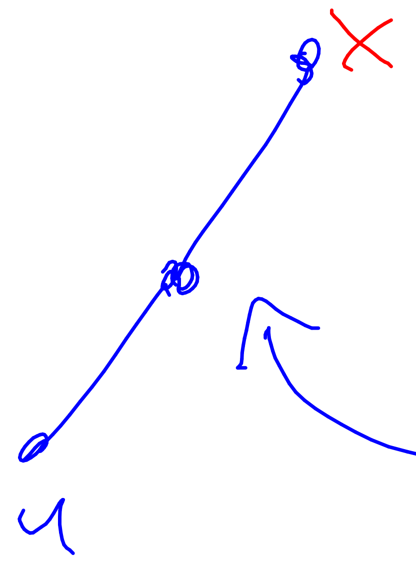
property



Lemma: optimal tree must be full.

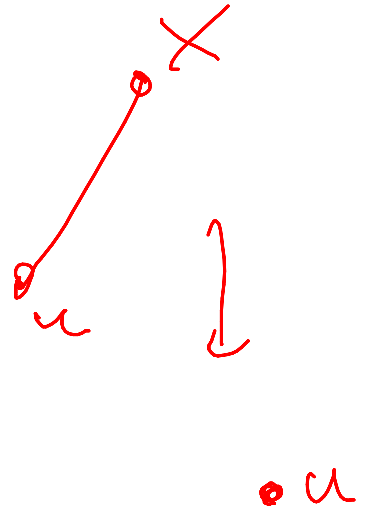
→ each node in the tree has either 0 or 2 children

Why??



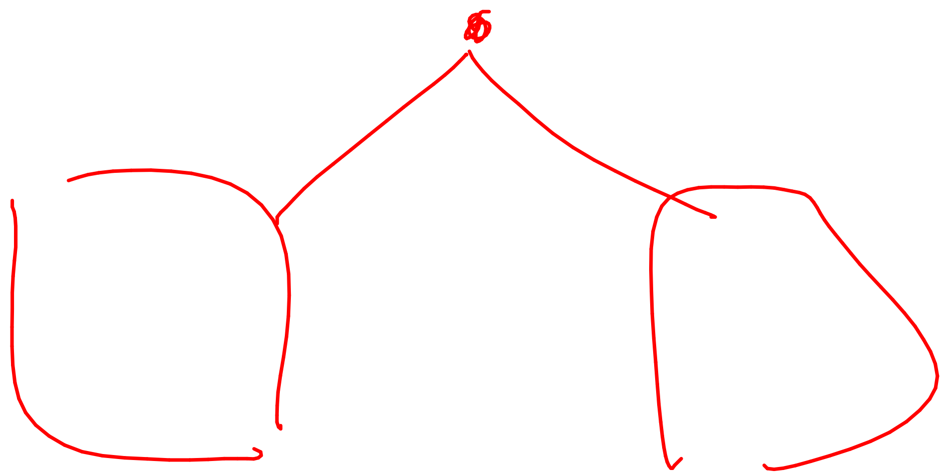
if some node has only 1 child, then it

→ can be removed from the tree to produce a code tree that is shorter for all the children of that node.



divide & conquer?

– partition the frequencies into 2 roughly equal halves & solve recursively.

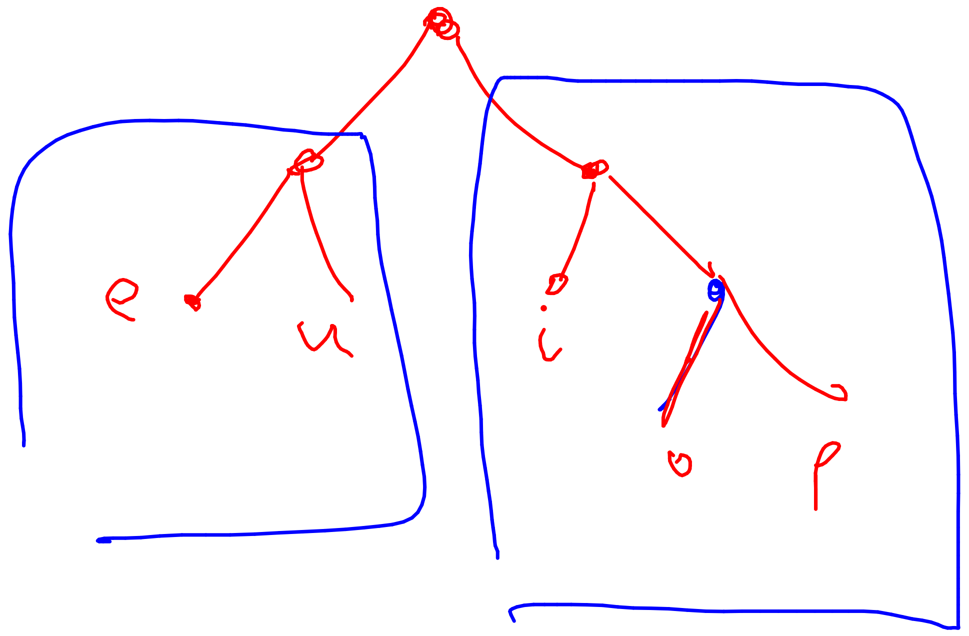


counter-example

50

e: 32
i: 25
o: 20
u: 18
p: 5

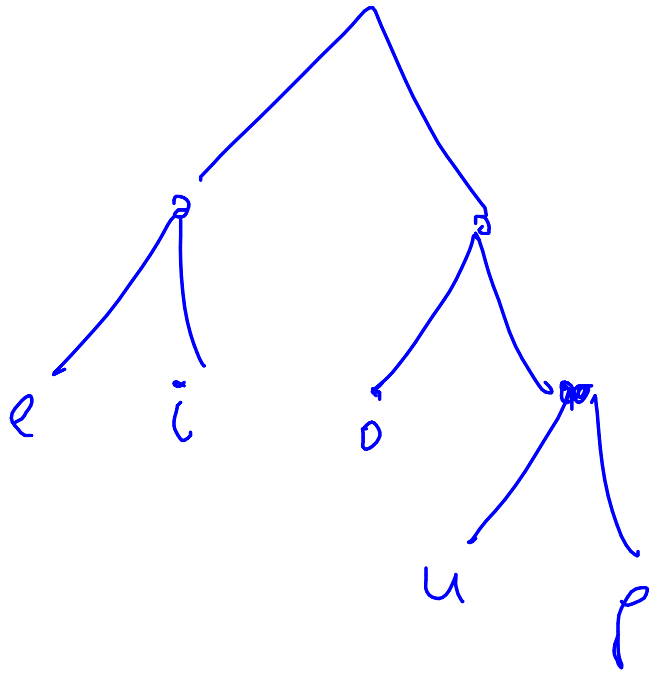
50



not optimal

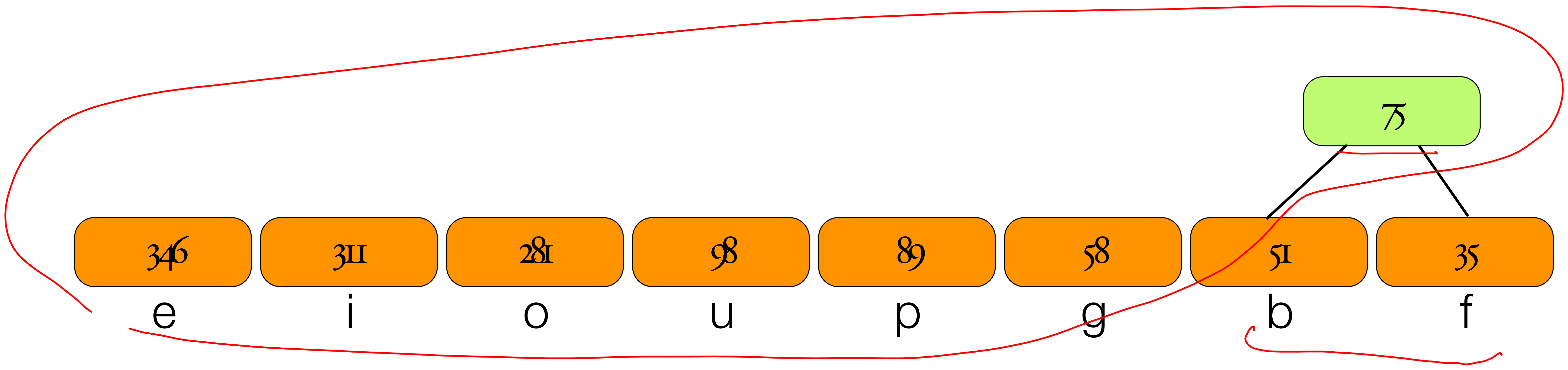
$l_o = 3$

$l_u = 2$

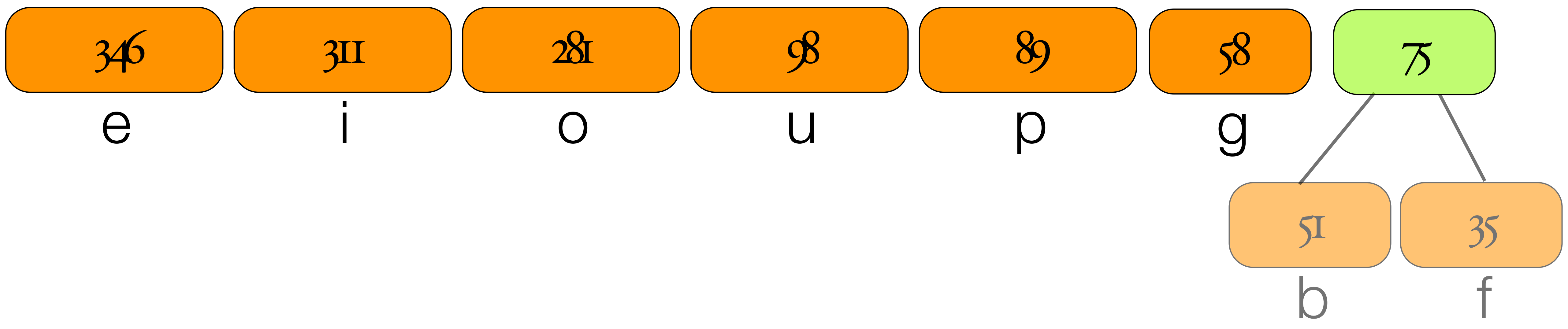


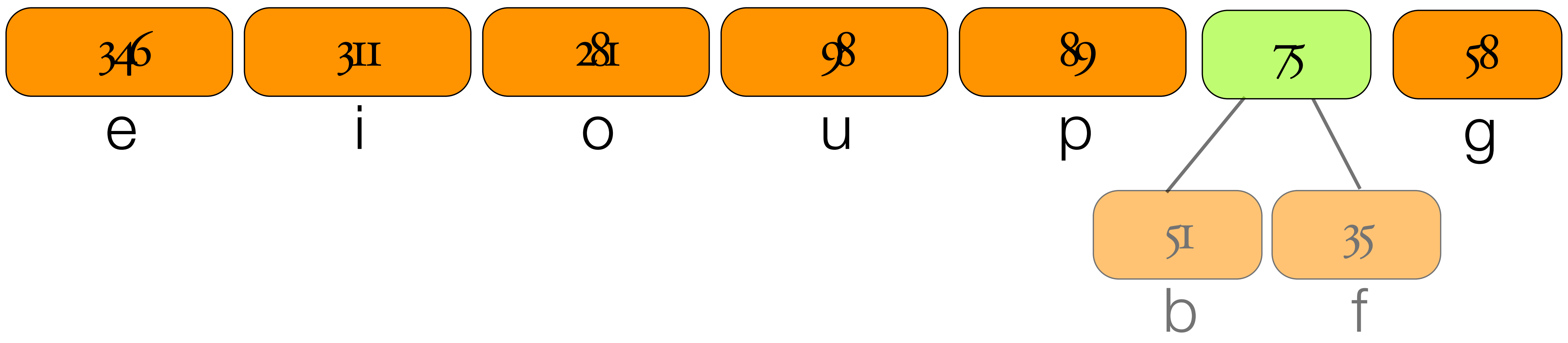
$l_u < l_o$ but
$f_u < f_o$

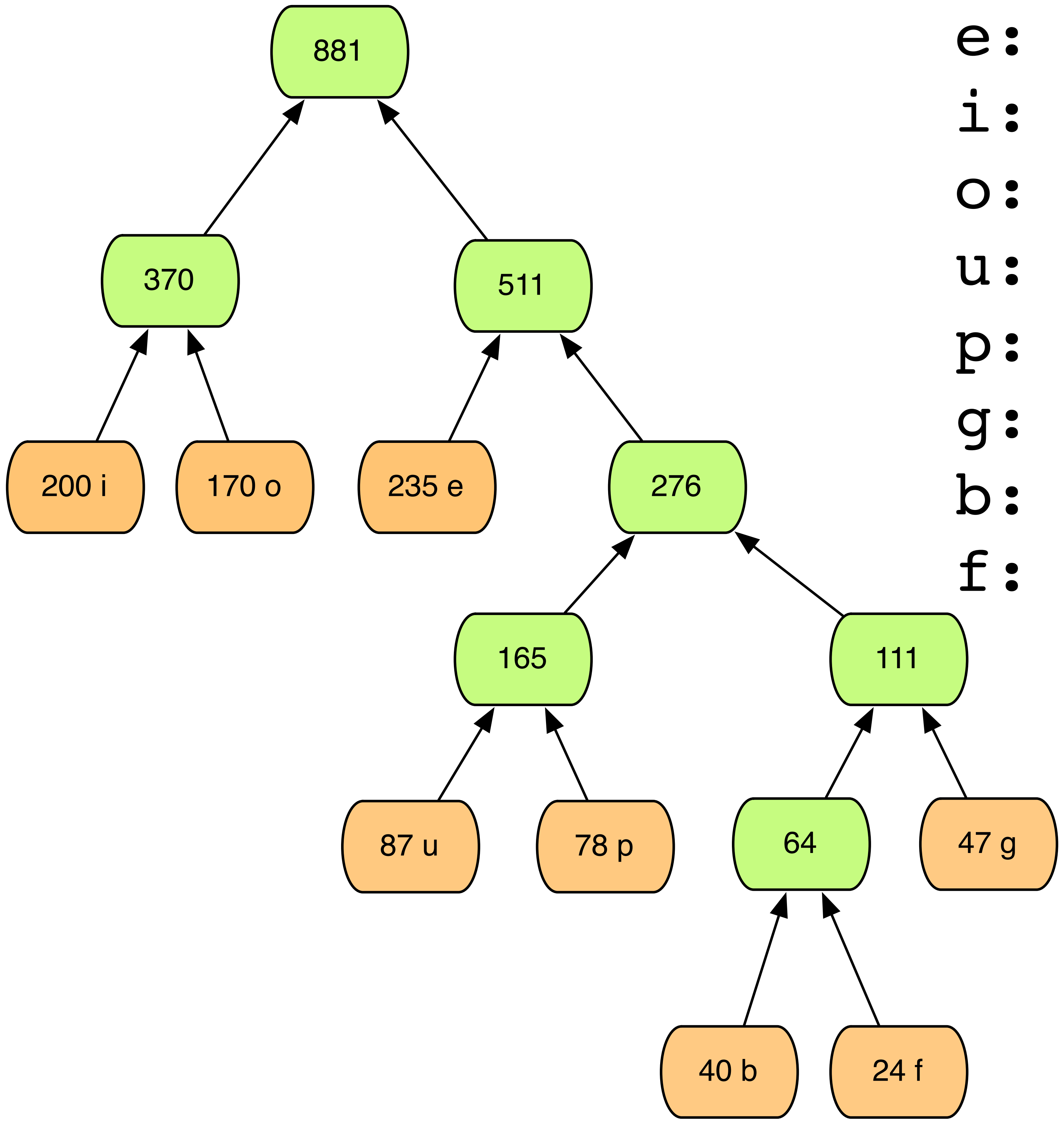
not good



75
346
311
281
98
89
58
51
35
e
i
o
u
p
g
b
f


346
e
311
i
281
o
98
u
89
p
58
g
75
51
b
35
f


346
e
311
i
281
o
98
u
89
p
75
58
g
51
b
35
f



e: 235 01 470
i: 200 11 400
o: 170 10 340
u: 87 0011 348
p: 78 0010 312
g: 47 0000 188
b: 40 00011 200
f: 24 00010 120

2378

objective

Show that the Huffman construction is optimal prefix-free encoding.

$\Rightarrow$ 2 lemmas that we prove.

exchange argument

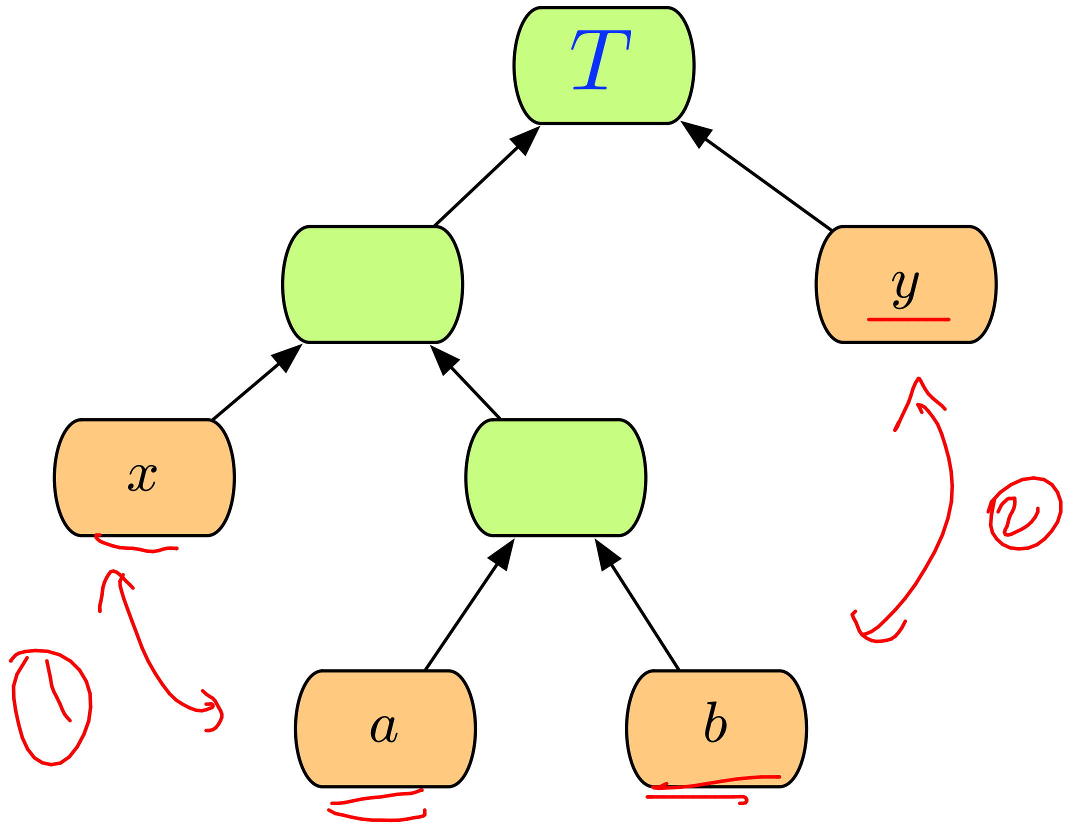
lemma: If x, y are the 2 least frequent characters in $\{f_c\}$, then there exists an optimal Tree in which x, y are siblings.

Proof: Consider an optimal solution T in which x, y are not siblings.

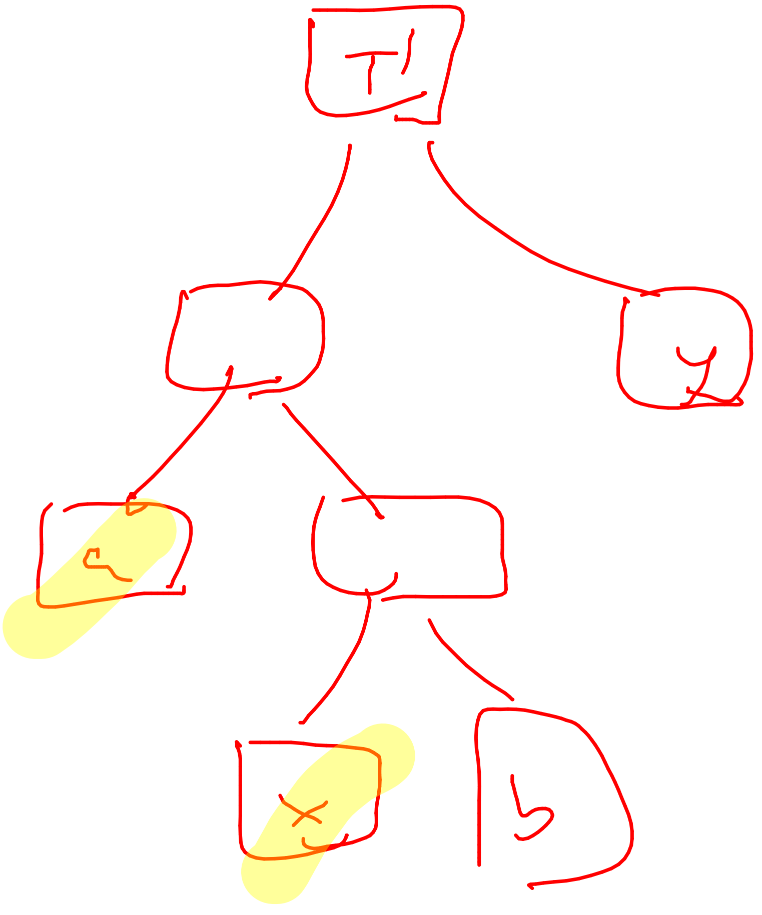
① Since T is optimal, T is full tree. Let a, b be the leaves with the greatest depth. We know such a pair exists b/c T is full.

exchange argument

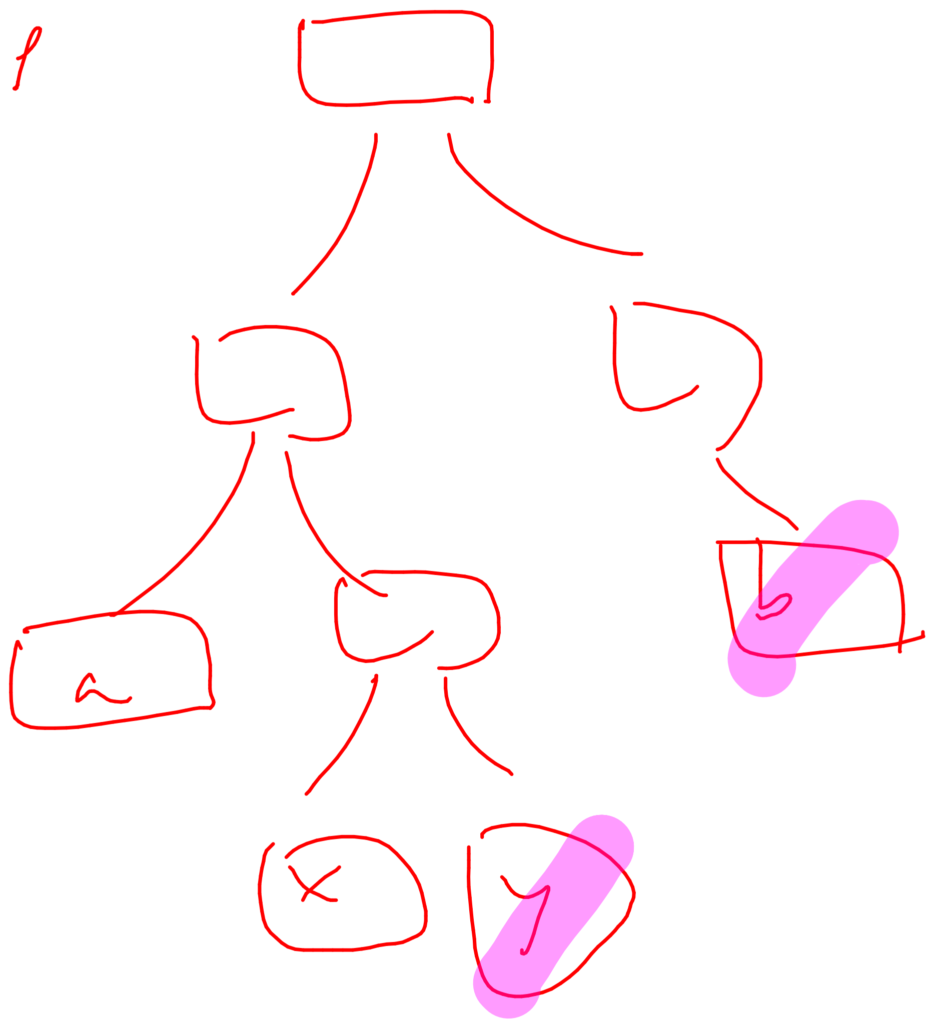
lemma: Let $x, y \in C$ be characters with smallest frequencies f_x, f_y. There exists an optimal prefix code T'' for C in which x, y are siblings. That is, the codes for x, y have the same length and only differ in the last bit.



first step



second step



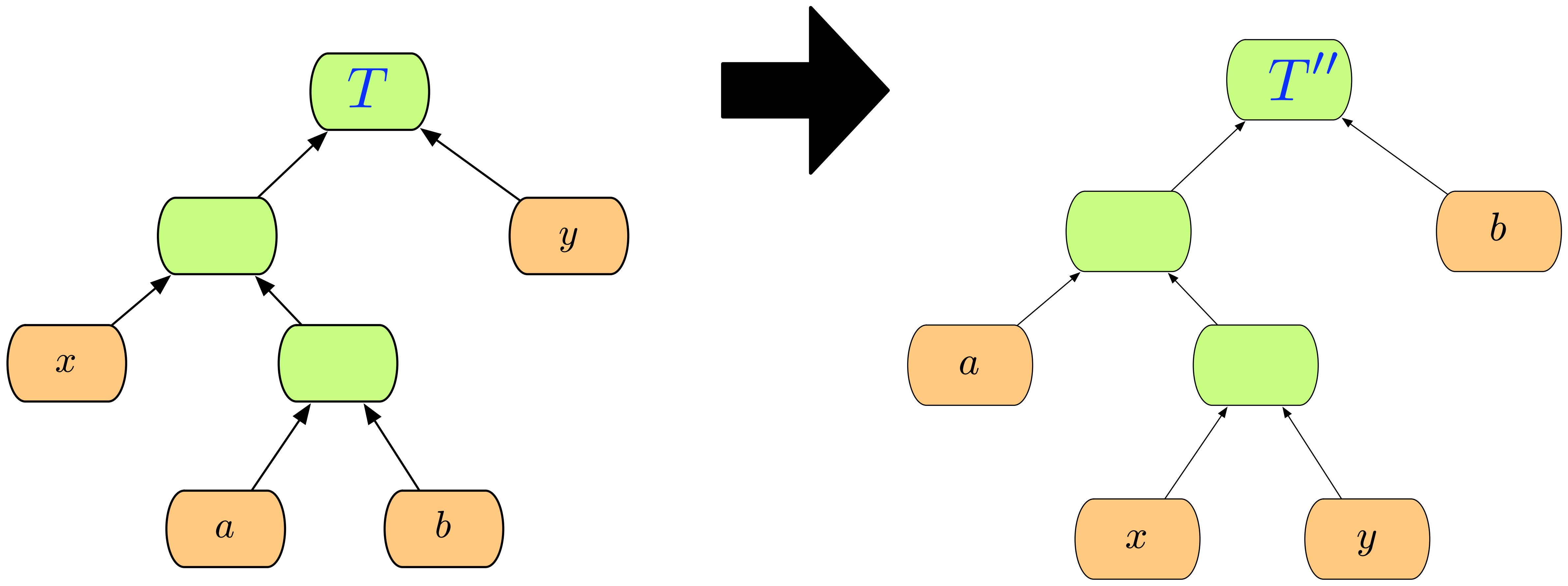
Wlog, lets say that

$f_x \leq f_a$

$f_y \leq f_b$

exchange argument

lemma: Let $x, y \in C$ be characters with smallest frequencies f_x, f_y. There exists an optimal prefix code T'' for C in which x, y are siblings. That is, the codes for x, y have the same length and only differ in the last bit.



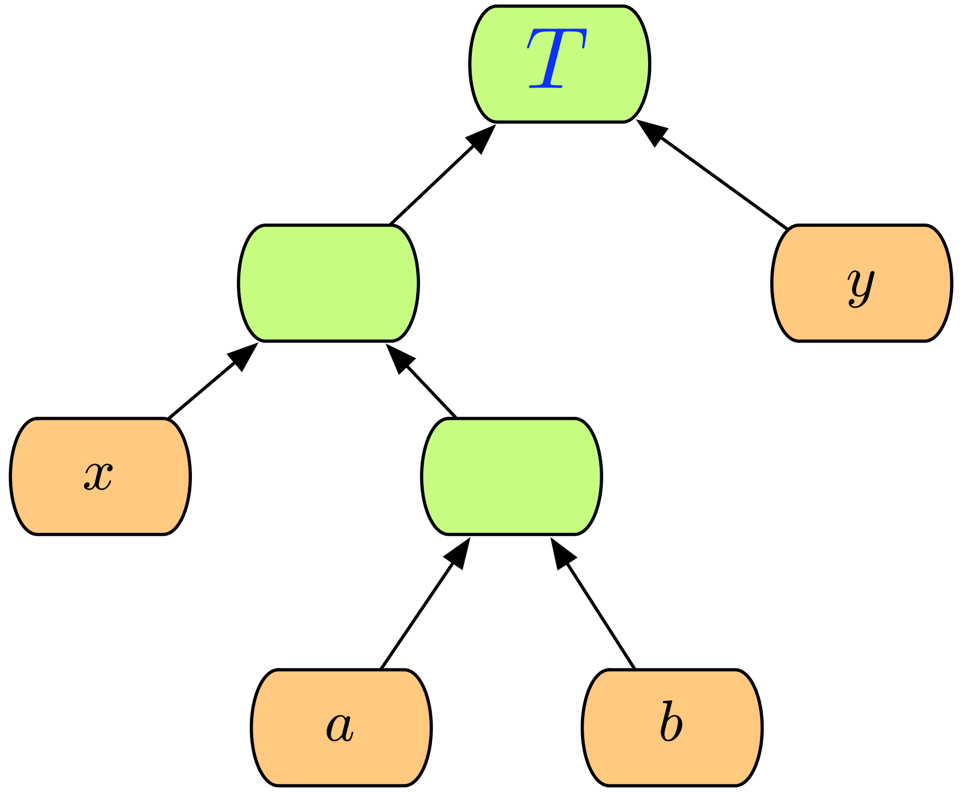
exchange argument

lemma: Let $x, y \in C$ be characters with smallest frequencies f_x, f_y. There exists an optimal prefix code T'' for C in which x, y are siblings. That is, the codes for x, y have the same length and only differ in the last bit.

proof:

exchange argument

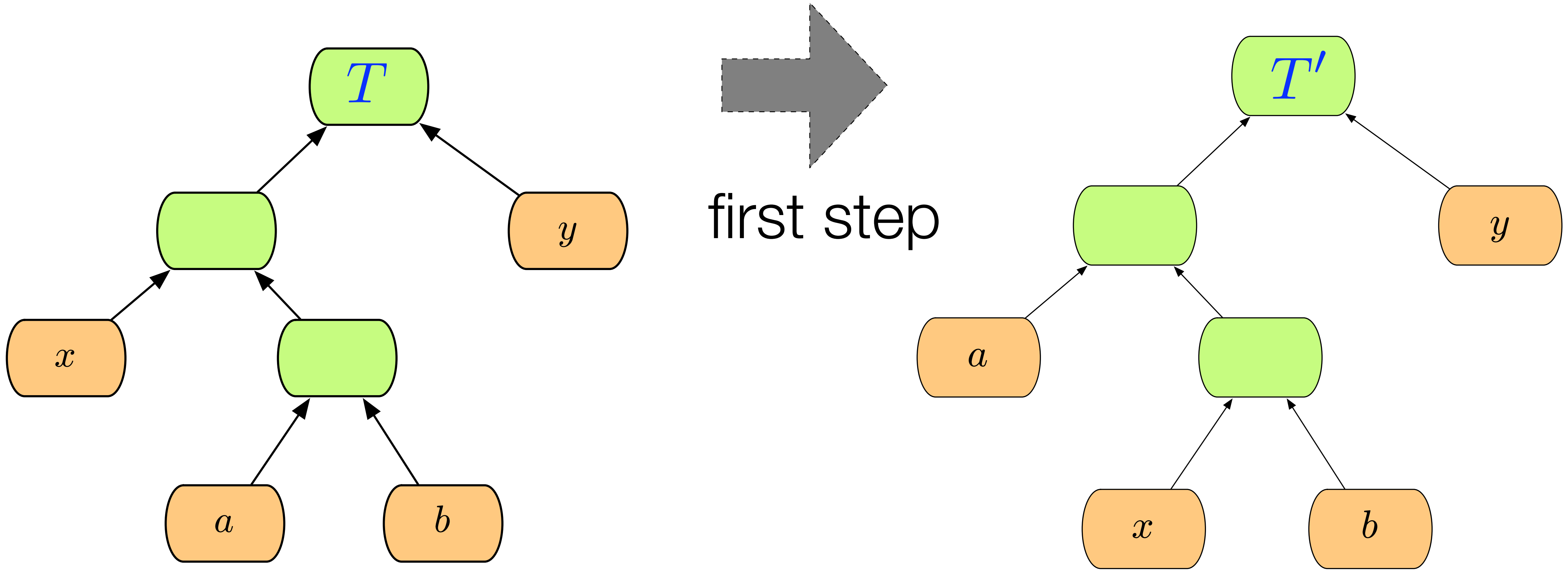
lemma: Let $x, y \in C$ be characters with smallest frequencies f_x, f_y. There exists an optimal prefix code T'' for C in which x, y are siblings. That is, the codes for x, y have the same length and only differ in the last bit.



first step

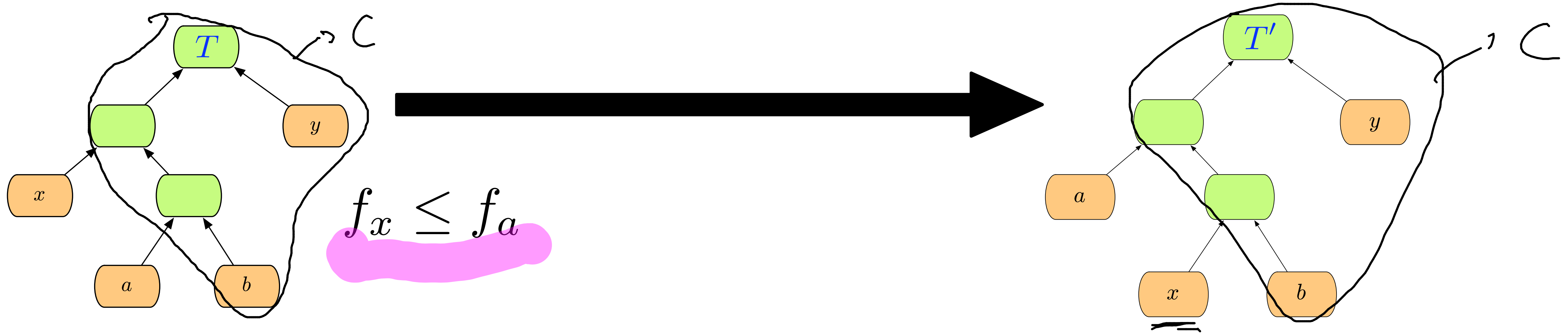
exchange argument

lemma: Let $x, y \in C$ be characters with smallest frequencies f_x, f_y. There exists an optimal prefix code T'' for C in which x, y are siblings. That is, the codes for x, y have the same length and only differ in the last bit.



$$f_a \leq f_b \qquad f_x \leq f_a$$

$$f_x \leq f_y \qquad f_y \leq f_b$$



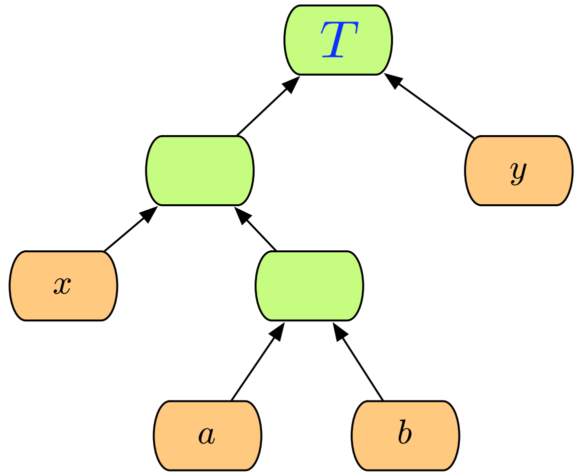
$$B(T) = C + f_x \cdot l_x + f_a \cdot l_a$$

$$B(T') = C + f_x \cdot l_a + f_a \cdot l_x$$

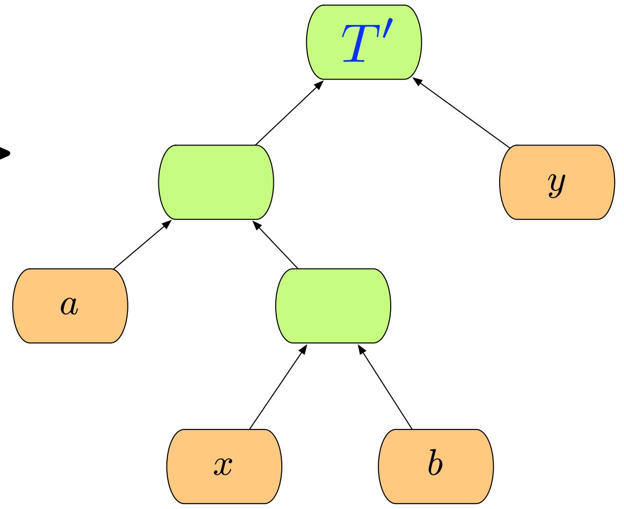
$$B(T) - B(T') = f_x \cdot l_x + f_a \cdot l_a - f_x \cdot l_a - f_a \cdot l_x$$

$$= \overset{-l_x}{l_x}\left(\overset{-f_x + f_a}{f_x - f_a}\right) + l_a\left(f_a - f_x\right)$$

$$= \underbrace{(l_a - l_x)}_{\geq 0}\underbrace{(f_a - f_x)}_{\geq 0} \geq 0$$



$$f_x \leq f_a$$



$$B(T) = \sum_c f_c \ell_c + f_x \ell_x + f_a \ell_a \qquad B(T') = \sum_c f_c \ell'_c + f_x \ell'_x + f_a \ell'_a$$

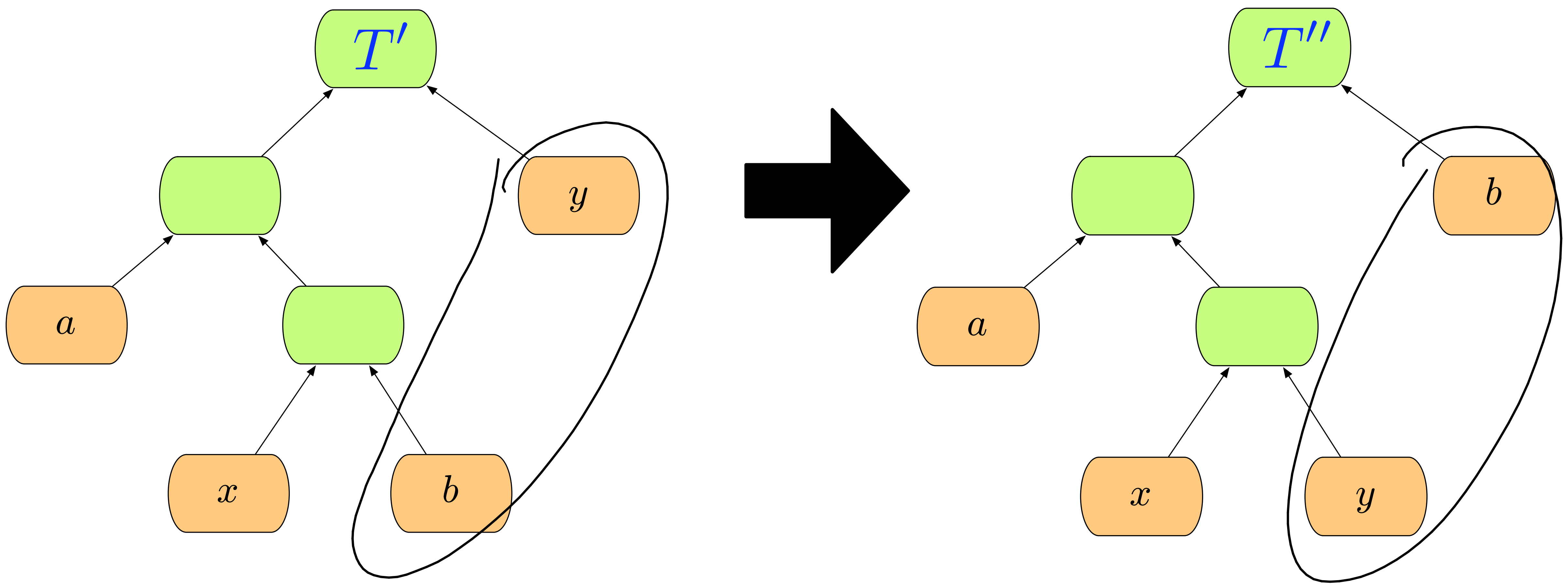
$$B(T) - B(T') \geq 0$$

optimal

our tree

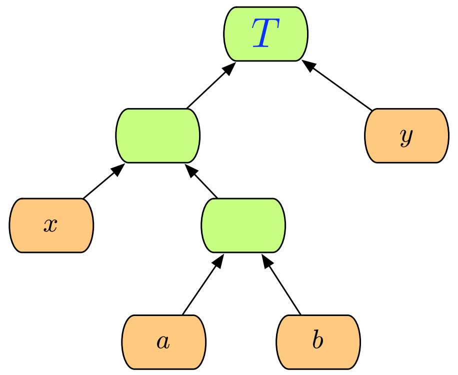
0.

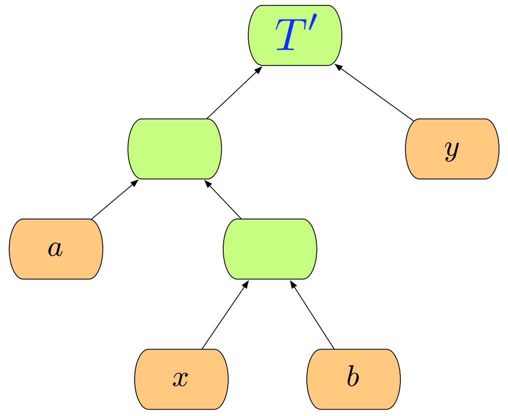
B(T') is also optimal!!

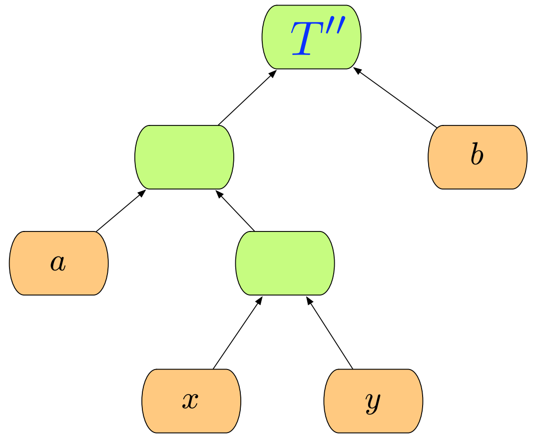


$$B(T') - B(T'') \geq 0$$

Same argument

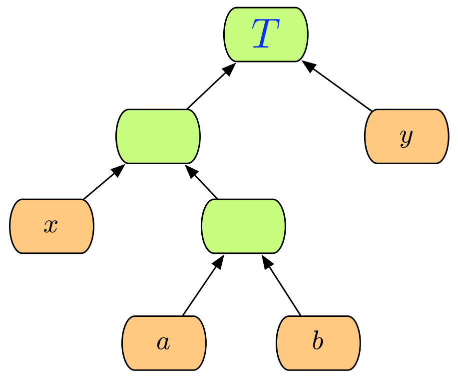


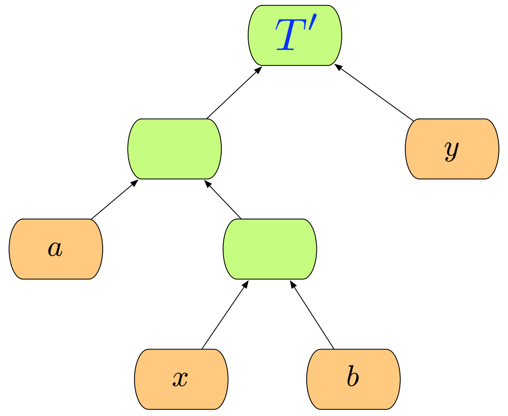


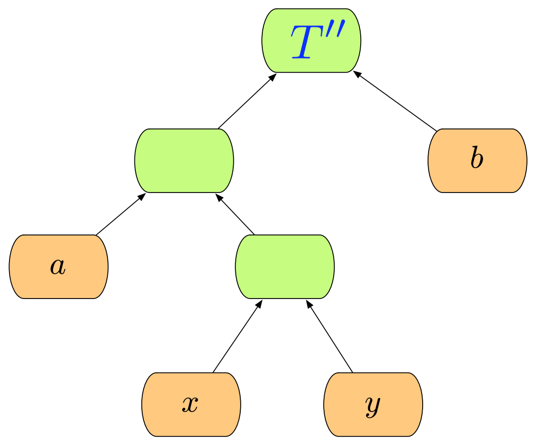


$B(T) \geq B(T') \geq B(T'') \Rightarrow T''$ is also optimal

and x, y are siblings in T''!





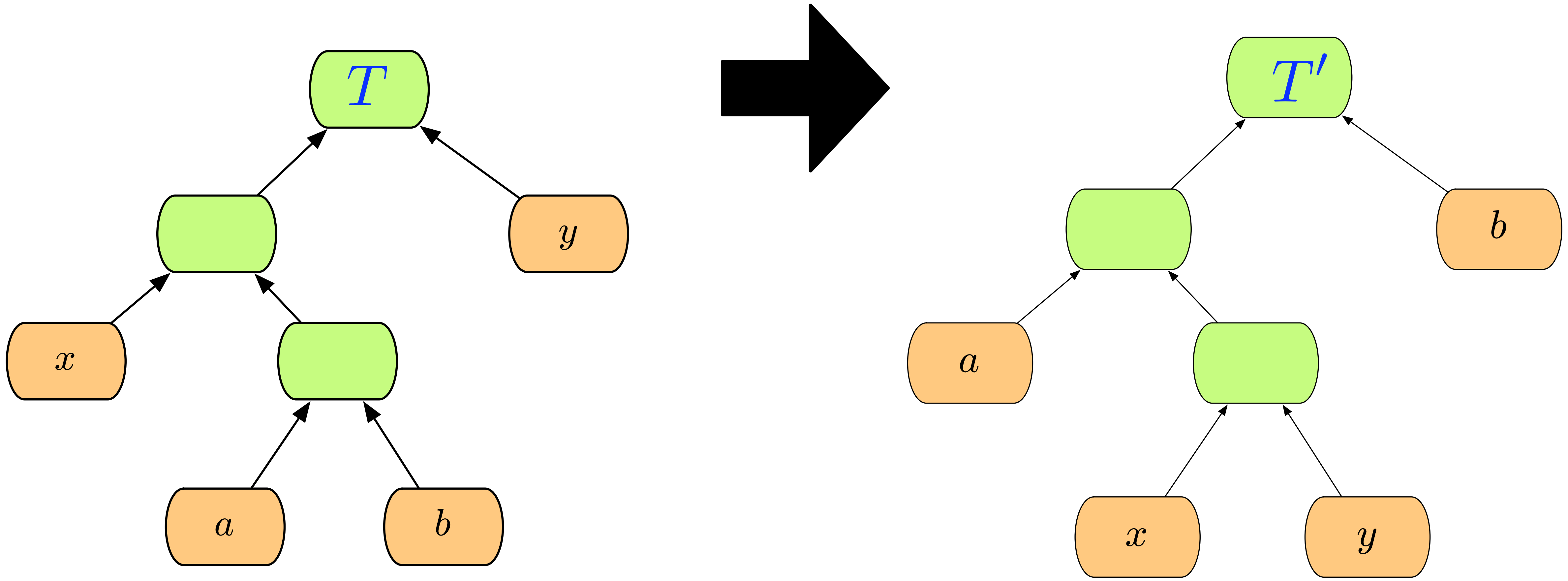


$$B(T) - B(T') \geq 0 \qquad B(T') - B(T'') \geq 0$$

T'' is also optimal

exchange argument

lemma: Let $x, y \in C$ be characters with smallest frequencies f_x, f_y. There exists an optimal prefix code T'' for C in which x, y are siblings. That is, the codes for x, y have the same length and only differ in the last bit.

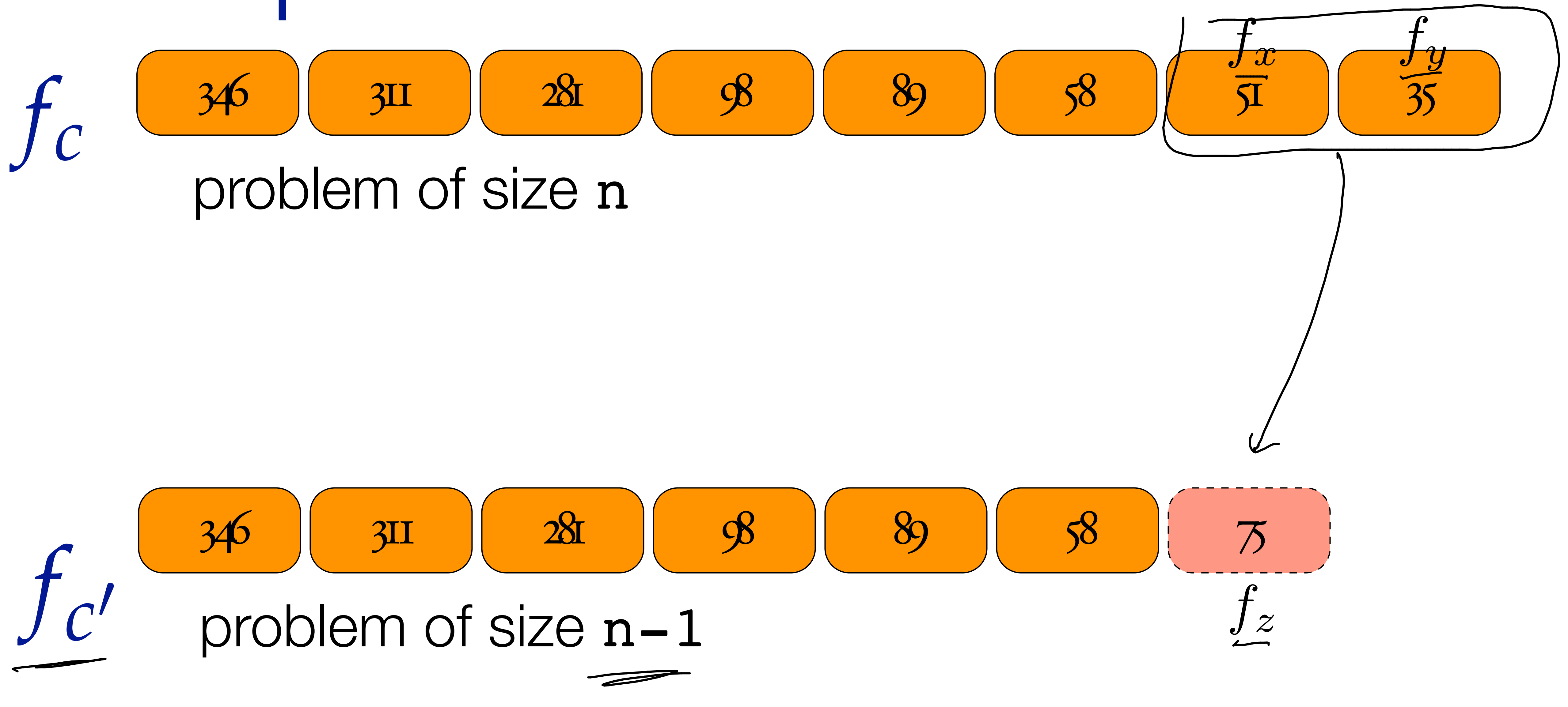


optimal sub-structure

f_c

						f_x	f_y
346	311	281	98	89	58	51	35

optimal sub-structure



why does this work ???

optimal sub-structure

f_c

						f_x	f_y
346	311	281	98	89	58	51	35

problem of size **n**

$f_{c'}$

346	311	281	98	89	58	75
						f_z

problem of size **n-1**

Lemma: The optimal solution for $\{f_c\}$ is to produce the optimal solution for $\{f_{c'}\}$ and replace node z with

$f_x \quad f_y$.

optimal sub-structure

f_c

						f_x	f_y
346	311	281	98	89	58	51	35

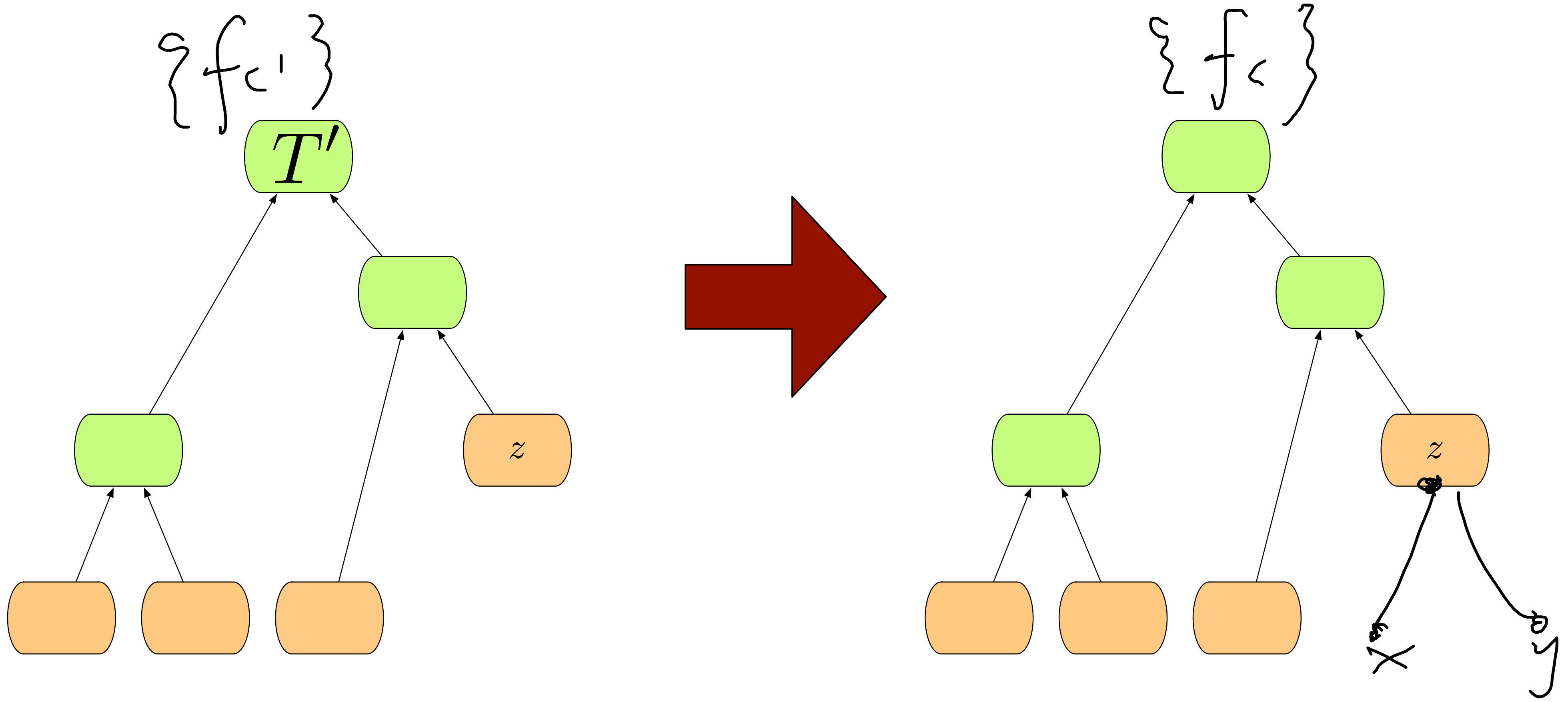
problem of size **n**

$f_{c'}$

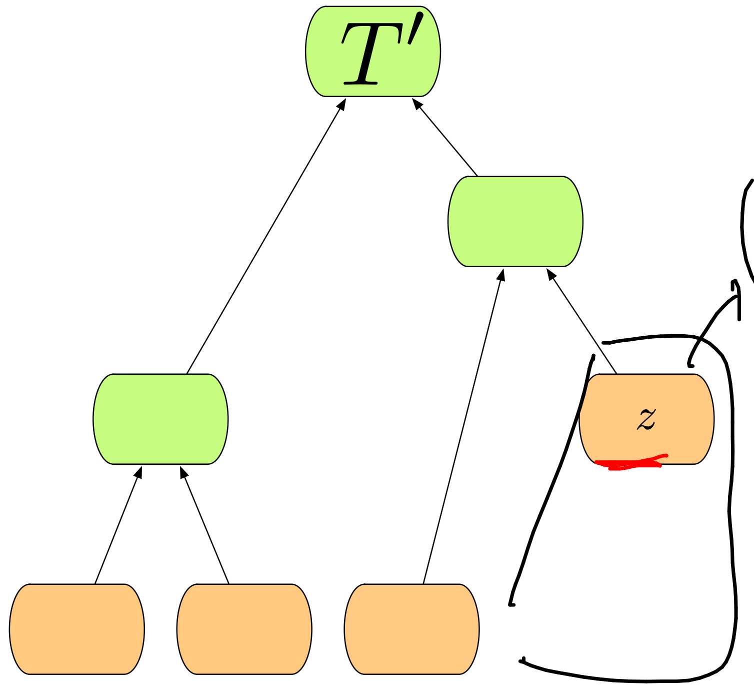
346	311	281	98	89	58	75
						f_z

problem of size **n-1**

Lemma: The optimal solution for T consists of computing an optimal solution for T' and replacing the left z with a node having children x, y.

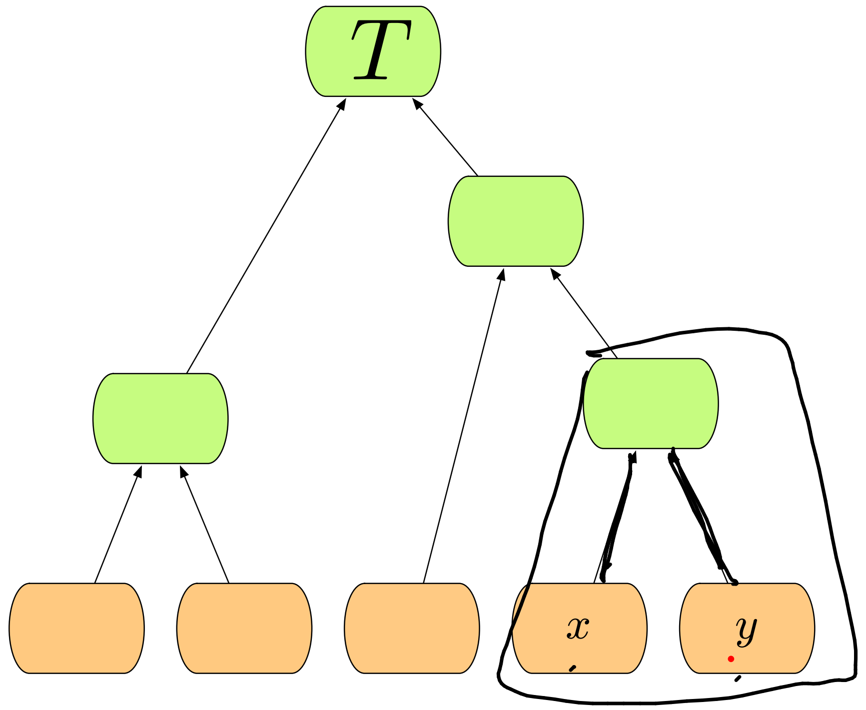


$\{f_{c'}\}$
T'
z
$\{f_c\}$
z
x
y



$(f_x+f_y)\,l_z$

$B(T')$

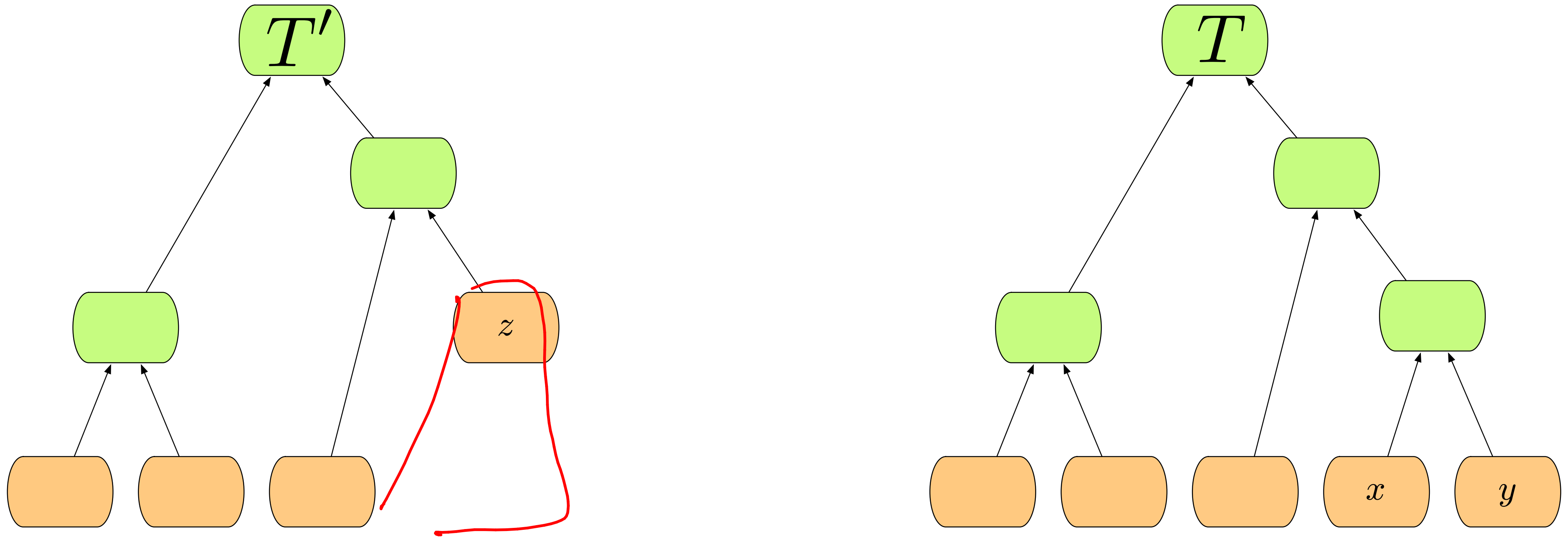


$(f_x+f_y)(l_z+1)$

$B(T)$

Let T' be an optimal solution for $\{f_c\}$.

$$B(T') = B(T) - f_x - f_y$$



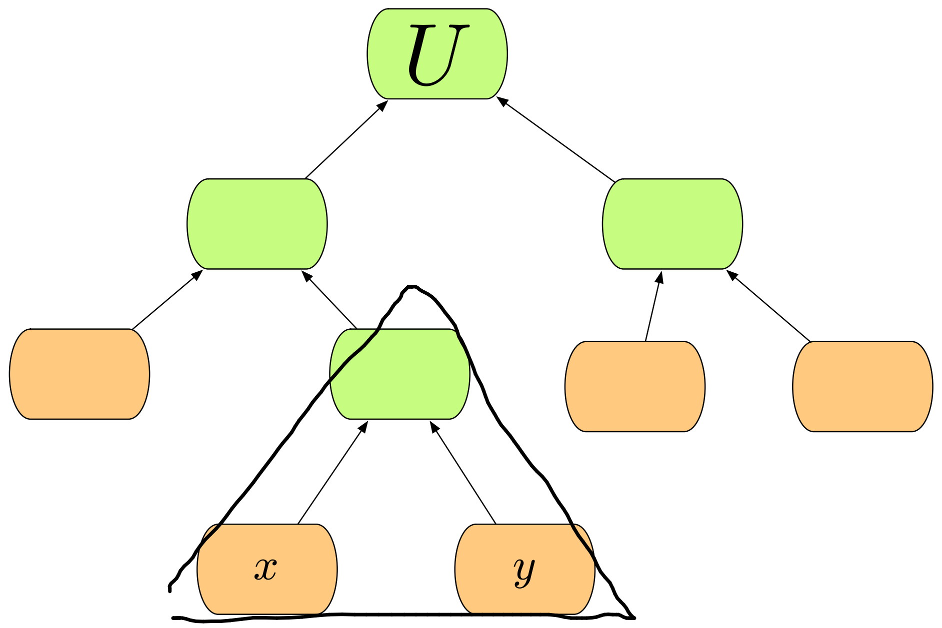
$B(T')$ $B(T)$

$$B(T') = B(T) - f_x - f_y$$

Suppose T is not optimal

① i.e. there is some other tree U s.t. $B(U) < B(T)$

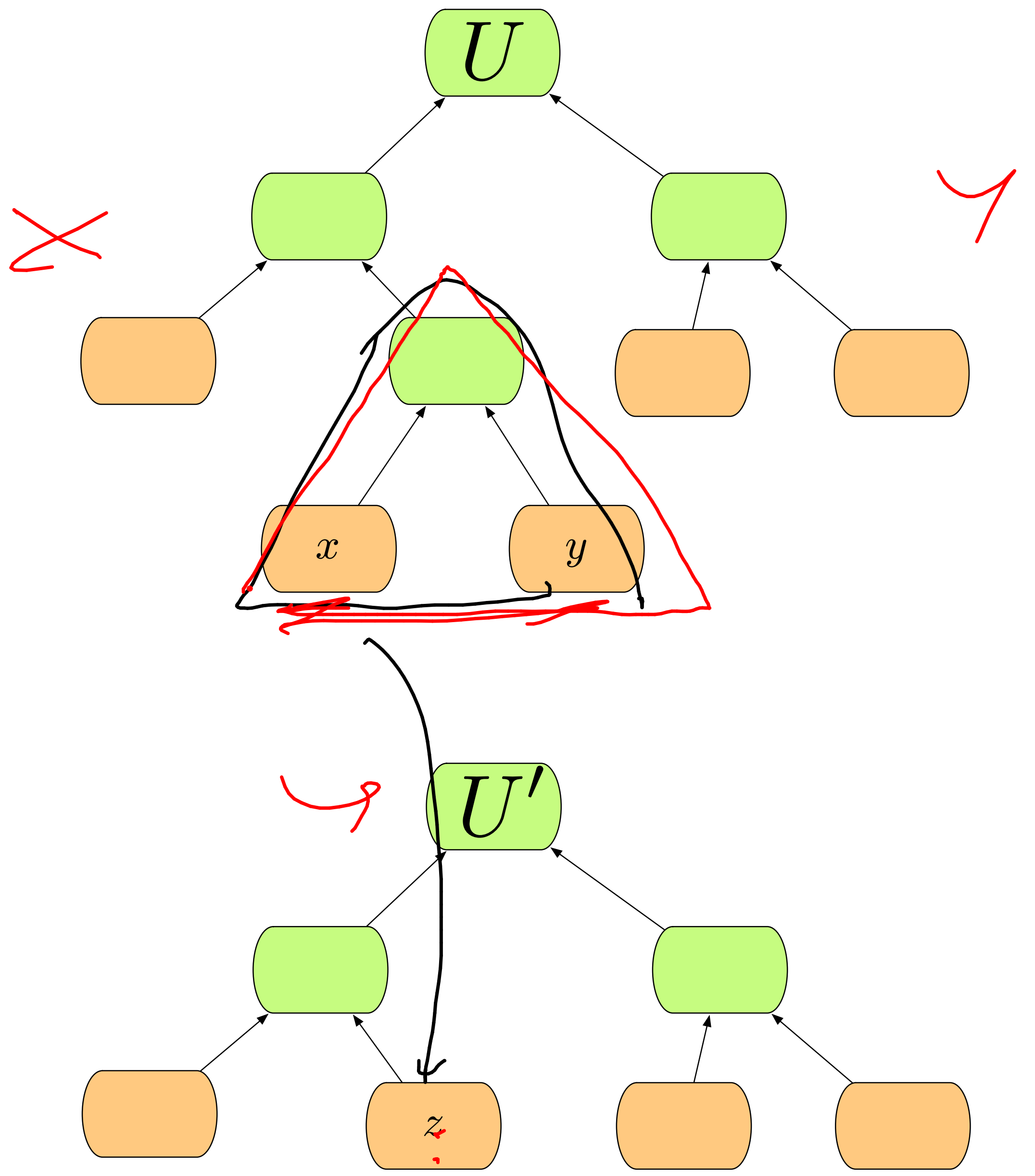
Suppose T is not optimal



$$B(U) < B(T)$$

→ by Lemma 1, we know that x, y must be siblings @ the lowest depth in U.

Suppose T is not optimal



$$B(U) < B(T)$$

$$B(U') = B(U) - f_x - f_y$$

$$< B(T) - f_x - f_y$$

$$= B(T')$$

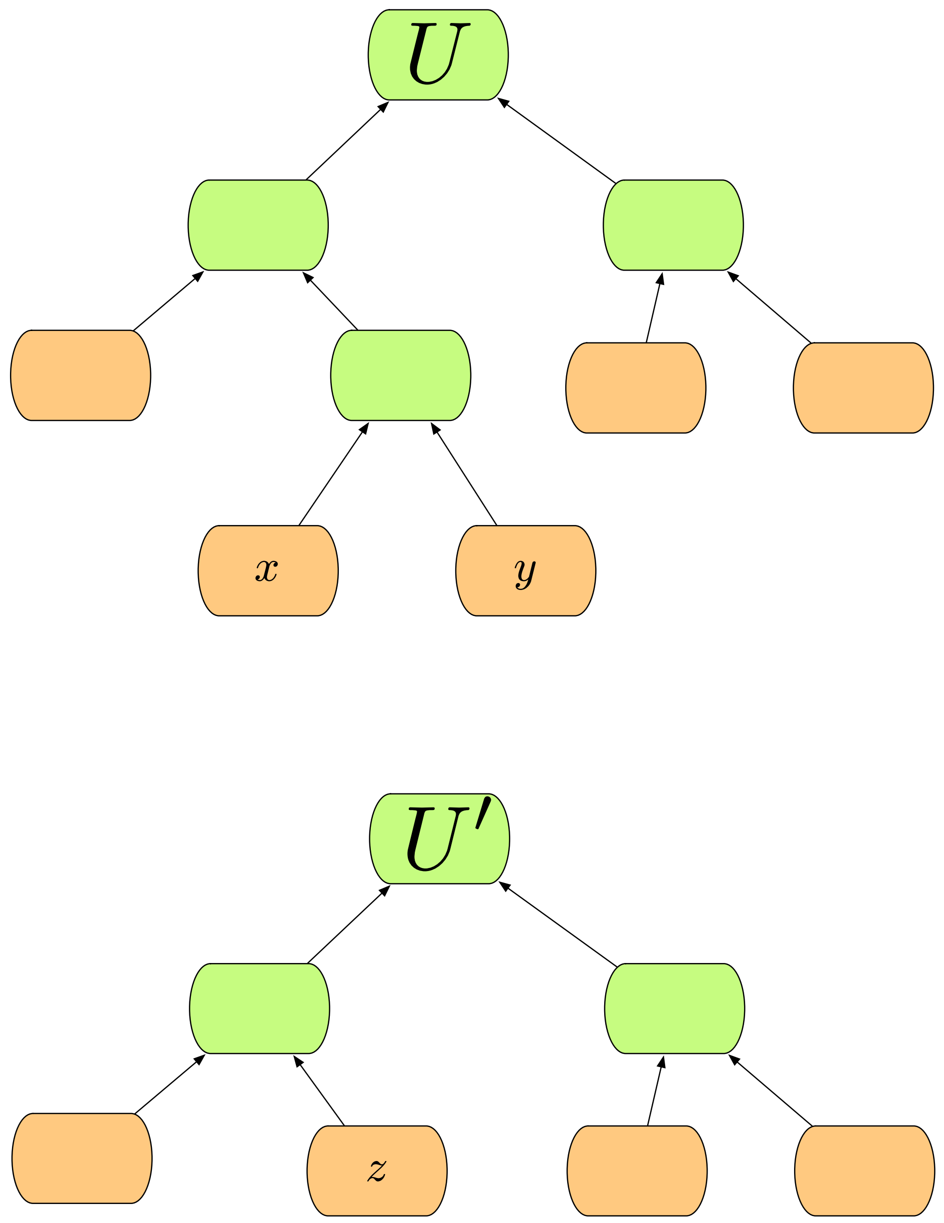
b/c x, y are siblings, that can be combined into a node z

$\Rightarrow B(U') < B(T')$

$\Rightarrow$ this would mean that T' is not optimal for $\{f_c\}$

which is a contradiction !!

Suppose T is not optimal



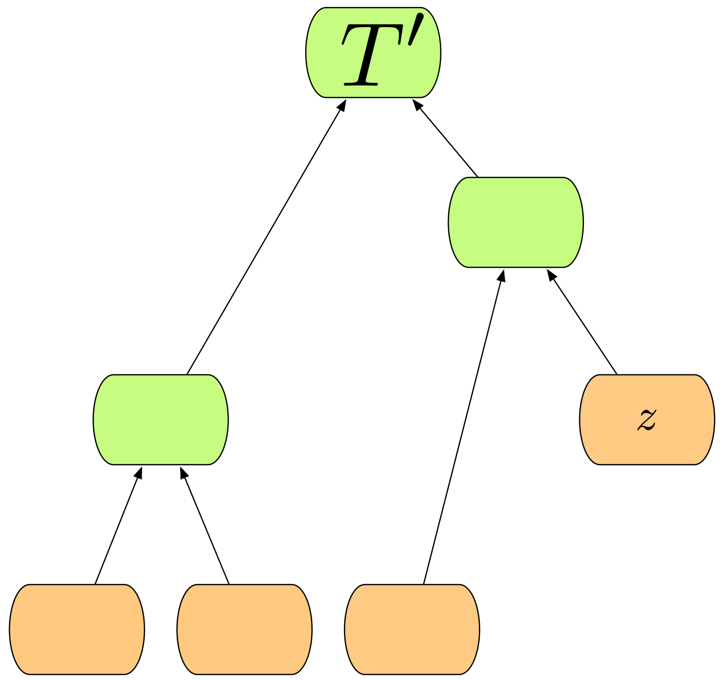
$$B(U) < B(T)$$

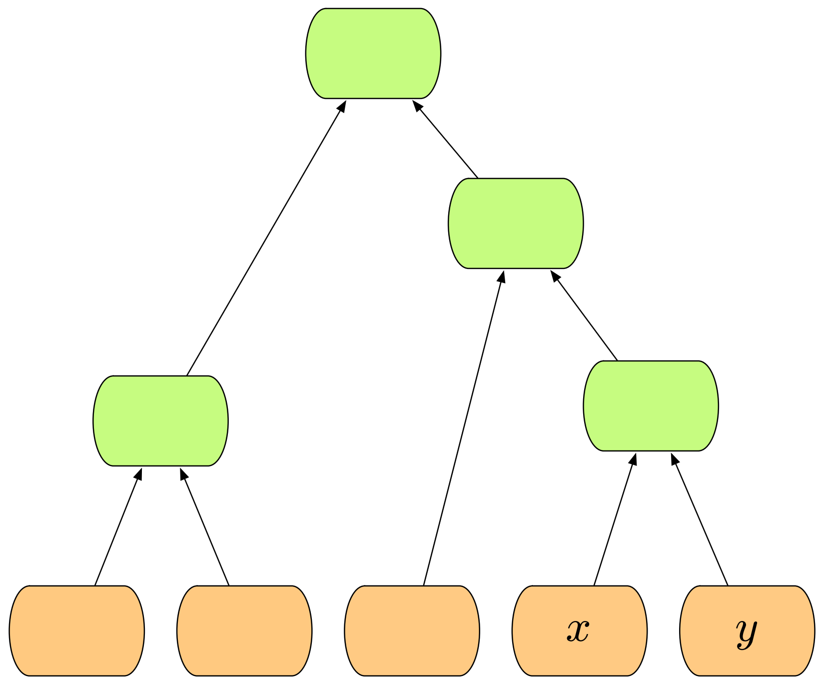
$$B(U') = B(U) - f_x - f_y$$

< B(t) - fx - fy

But this implies that B(T') was not optimal.

therefore





summary of argument

MST

connecting houses

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

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

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

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

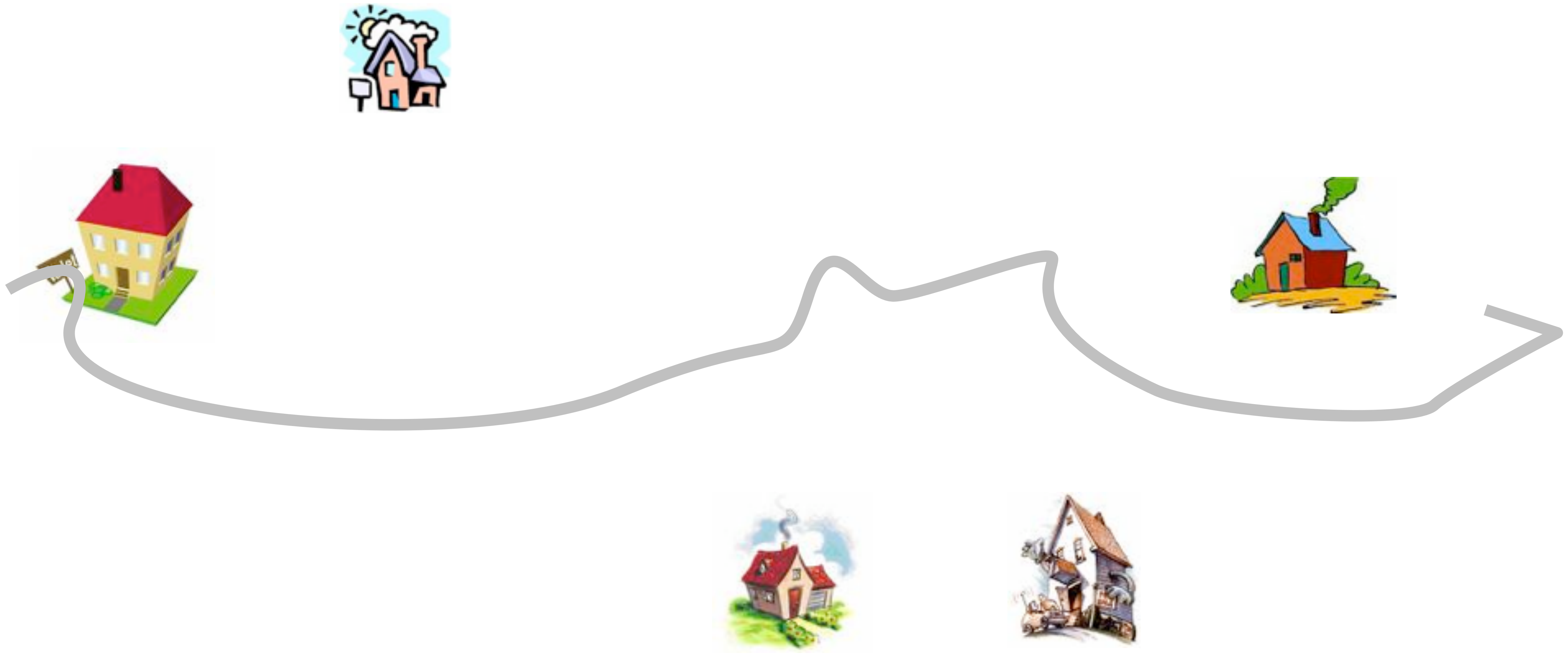


image:www.princegeorgeva.org, thefranciscofamily.org, www.rightdriveacademy.co.uk, www.ccscambridge.org, www.drawingcoach.com, www.pastoral.org.uk, www.daasgallery.com

connecting houses

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

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

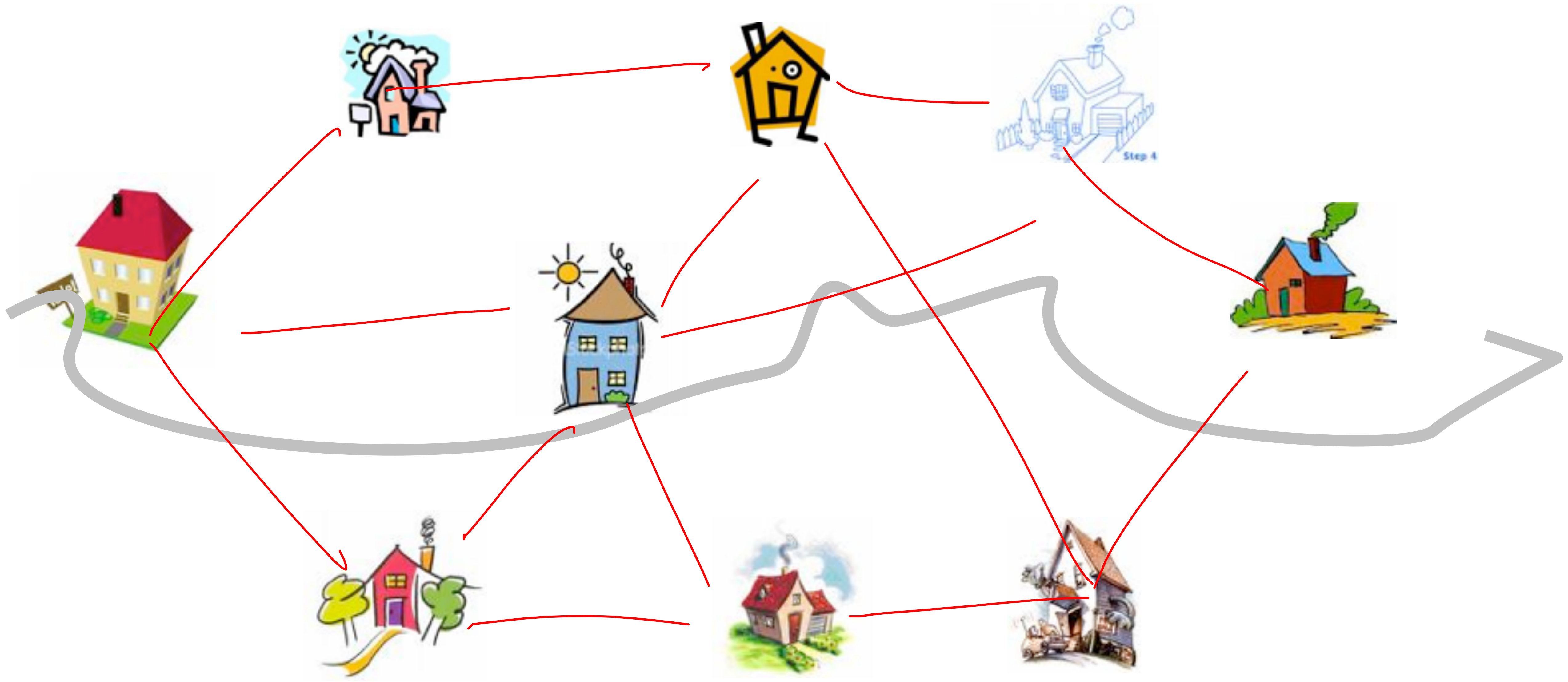
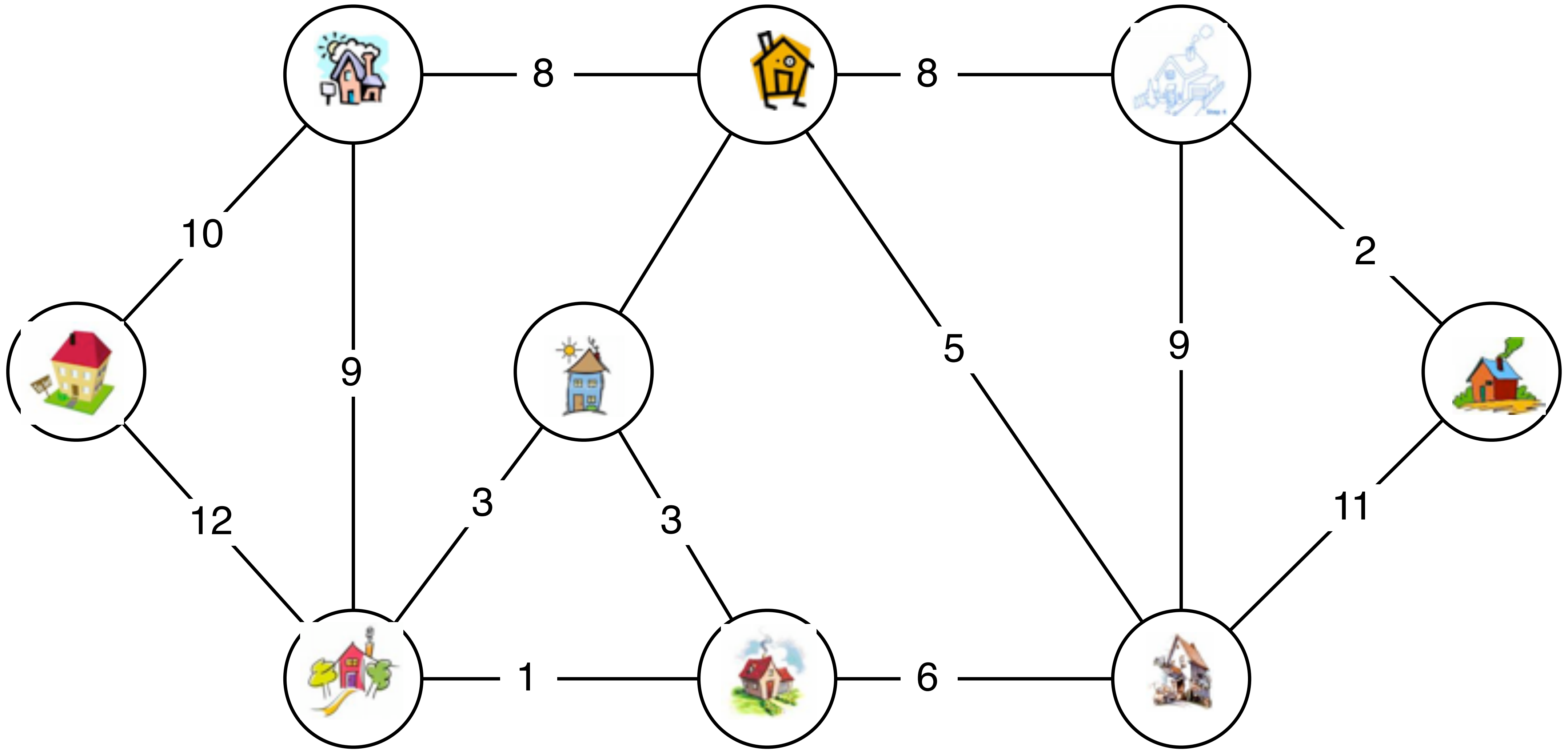


image:www.princegeorgeva.org, thefranciscofamily.org, www.rightdriveacademy.co.uk, www.ccscambridge.org, www.drawingcoach.com, www.pastoral.org.uk, www.daasgallery.com


8
8
10
9
5
9
2
3
3
12
11
1
6

graphs

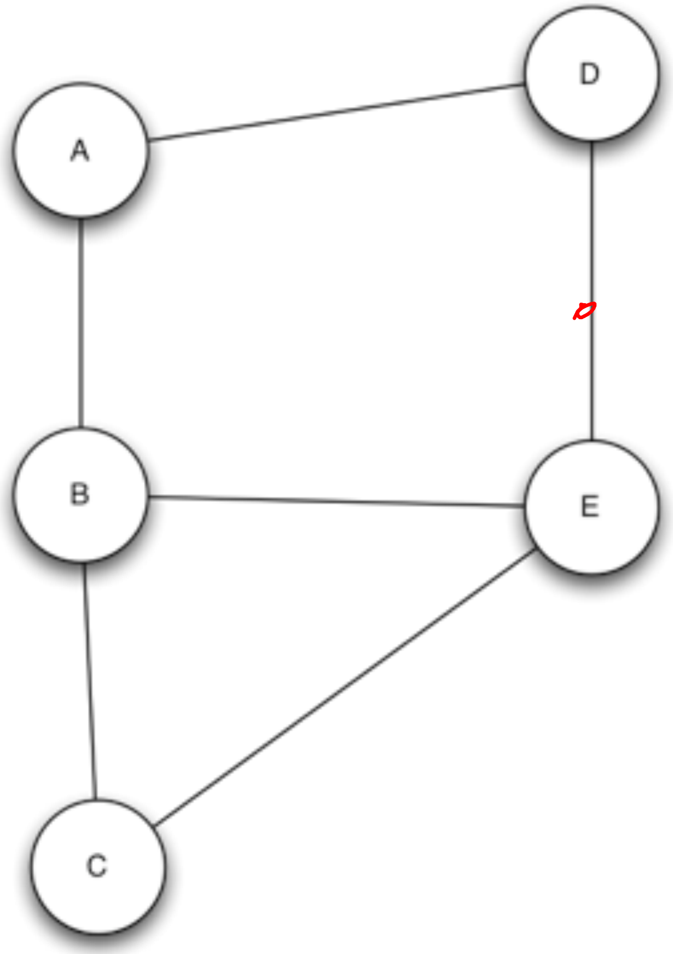
clrs [ch 22]

$$G = (V, E), \quad w: E \to \mathbb{R}^{+}$$

vertices

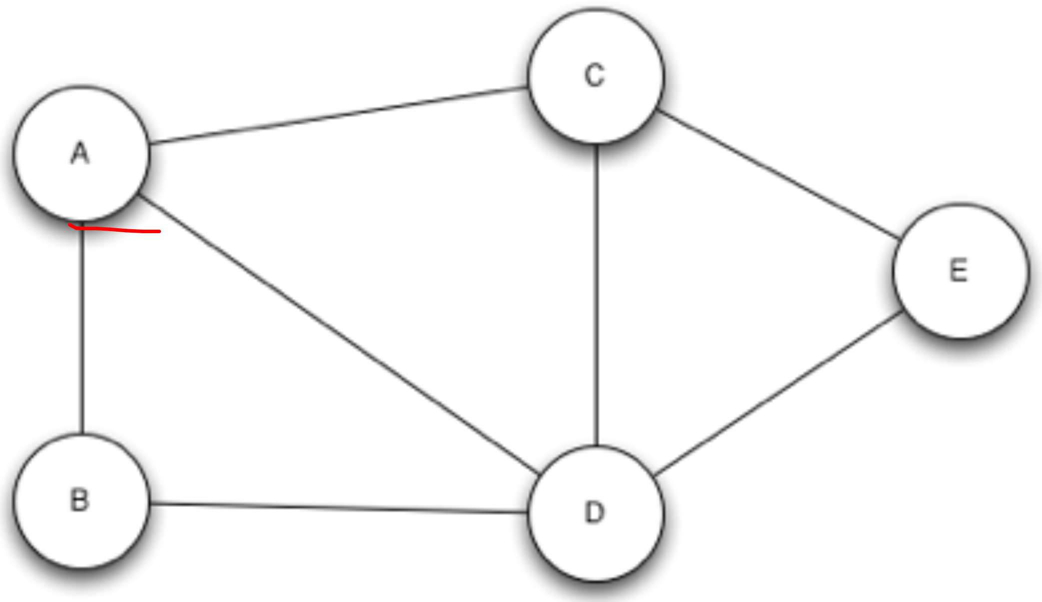
edges

weight for an edge $e \in E$

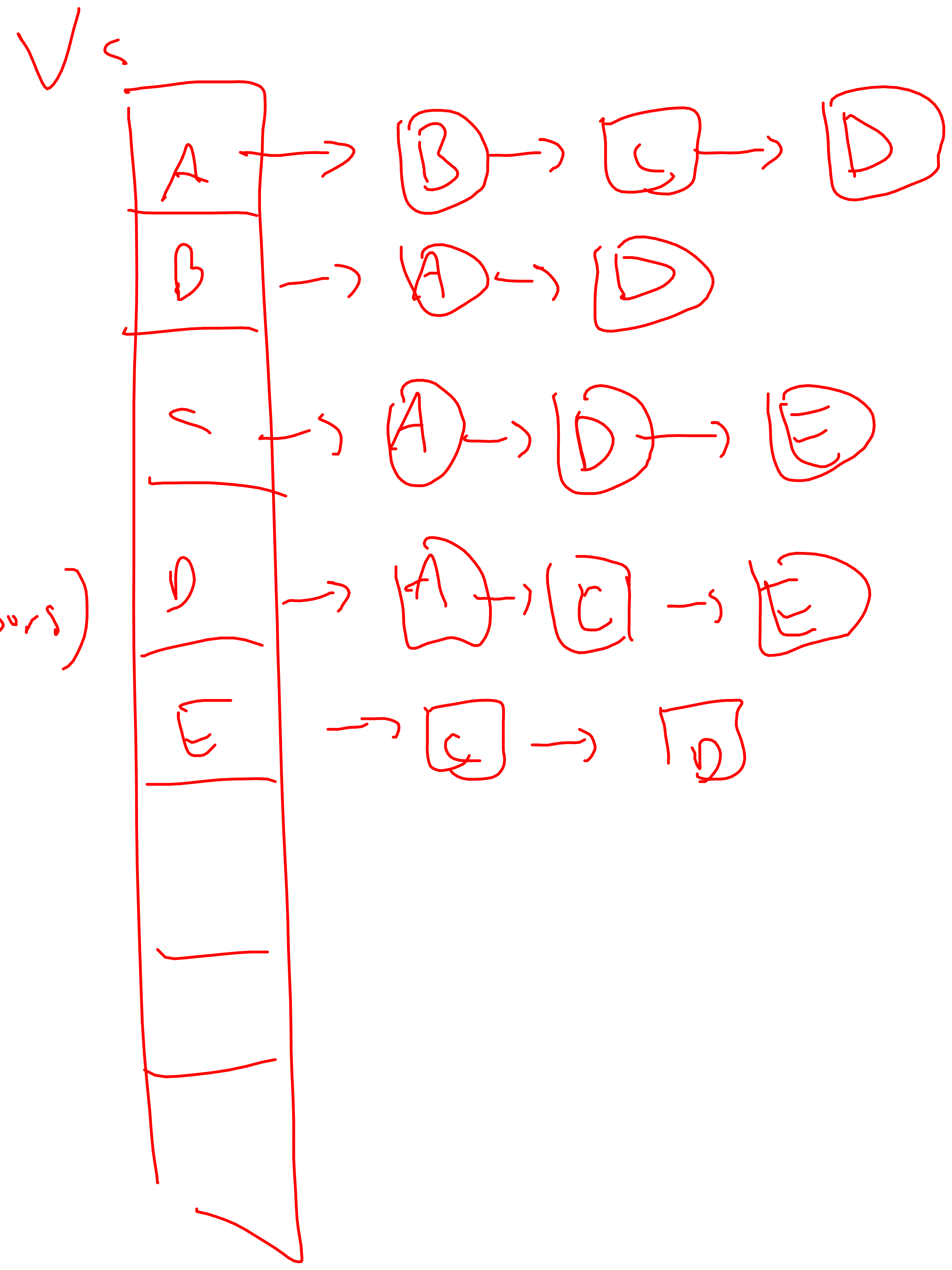


representation

$$G = (V, E)$$



adjacency list



space: $\Theta(|V| + |E|)$

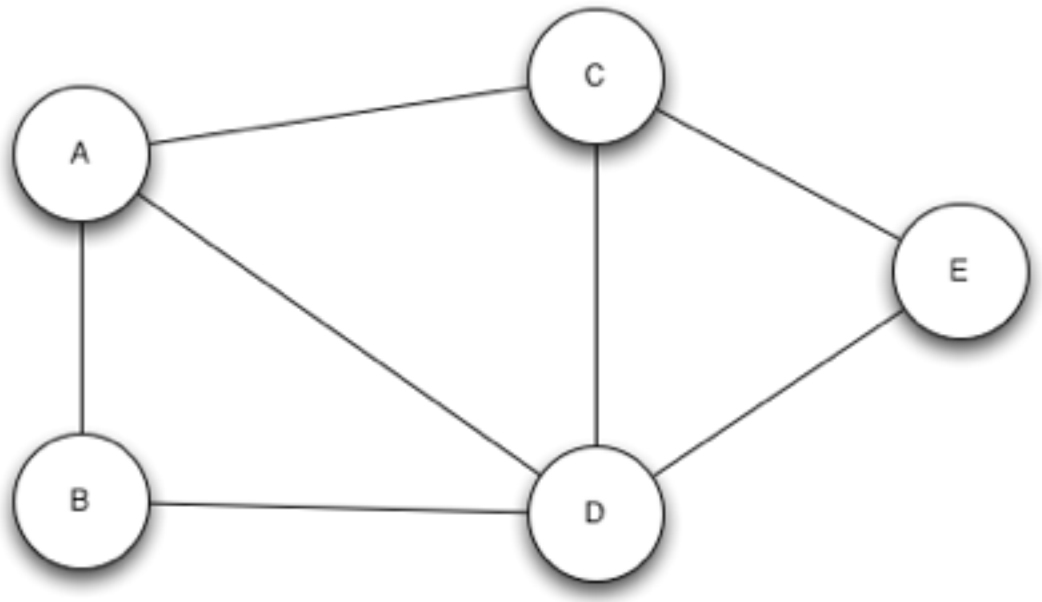
time list neighbors: $\Theta(\#\text{ of neighbors})$

time check an edge:

$\Theta(\#\text{ of neighbors}) = \Theta(V)$

representation

$$G = (V, E)$$



adjacency matrix

$$\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & & 1 & 1 & 1 & \\ B & 1 & & & 1 & \\ C & 1 & & & 1 & 1 \\ D & 1 & & 1 & & 1 \\ E & & & 1 & 1 & \end{array}$$

space: $\Theta(V^2)$

time list neighbors: $\Theta(V)$

time check an edge: $\Theta(1)$

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:

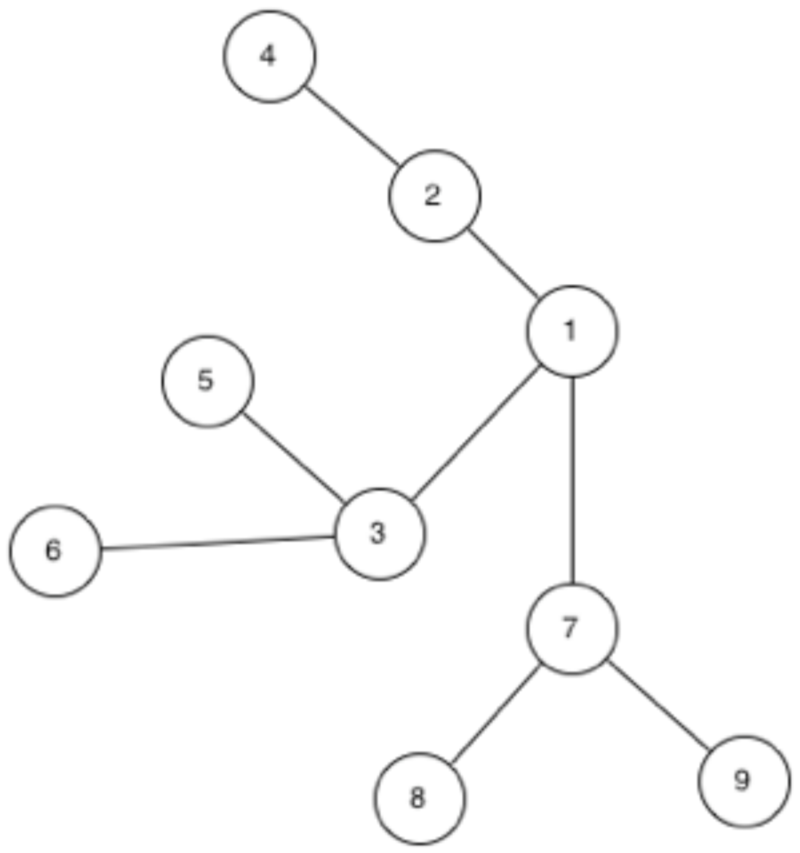
cycle:

definition:tree

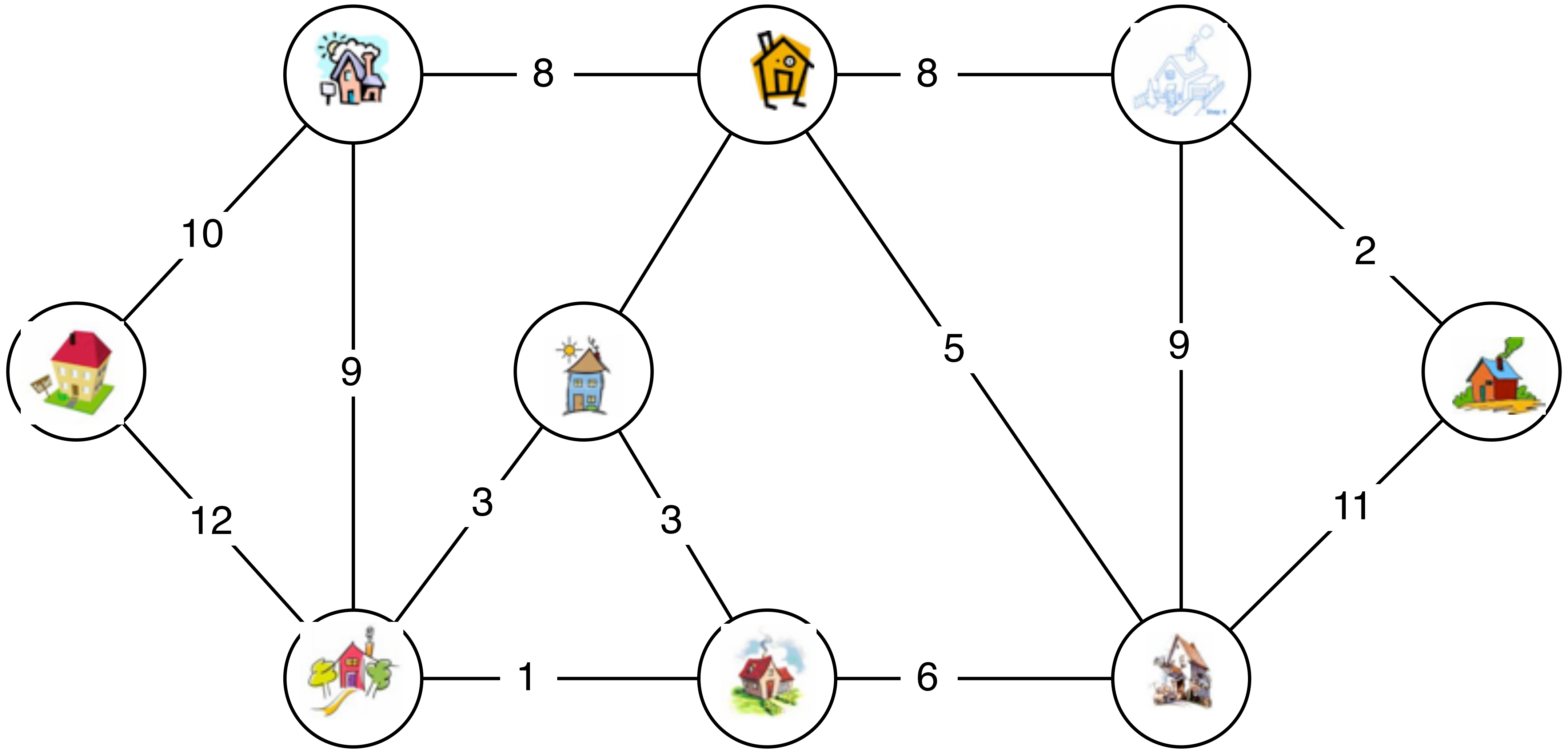
connected graph:

a tree is

connected graph w/No cycles!



what we want:



① way to connect each pair of nodes that has minimum cost ⇒ MST.

minimum spanning tree

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

(a) connects all vertices

(b) has the least cost

$$\min \sum_{(u,v)\in T} w(u,v)$$

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

(a) connects all vertices

(b) has the least cost

$$\min \sum_{(u,v) \in T} w(u,v)$$

facts

how many edges does solution have ?

does solution have a cycle?

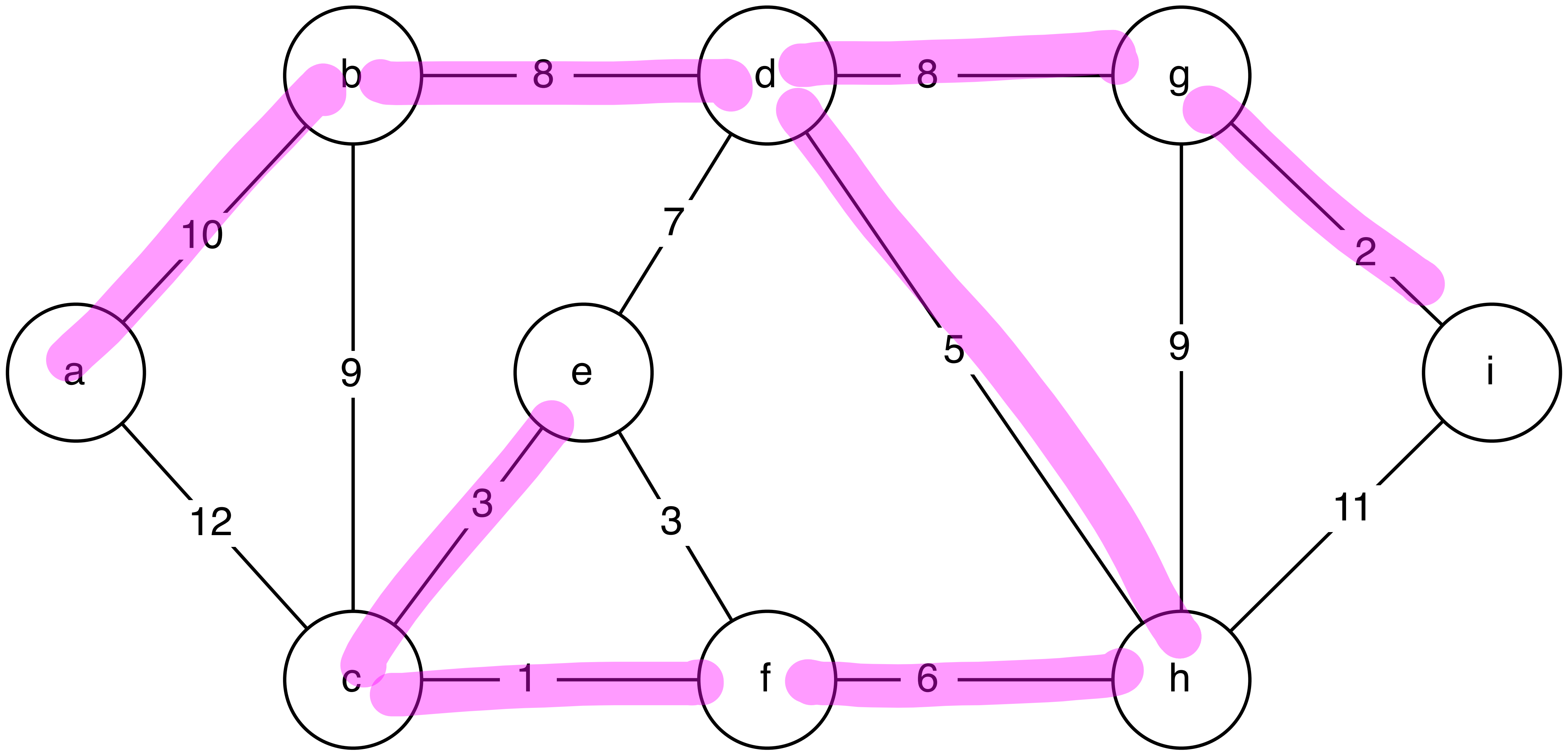
strategy

start with an empty set of edges A

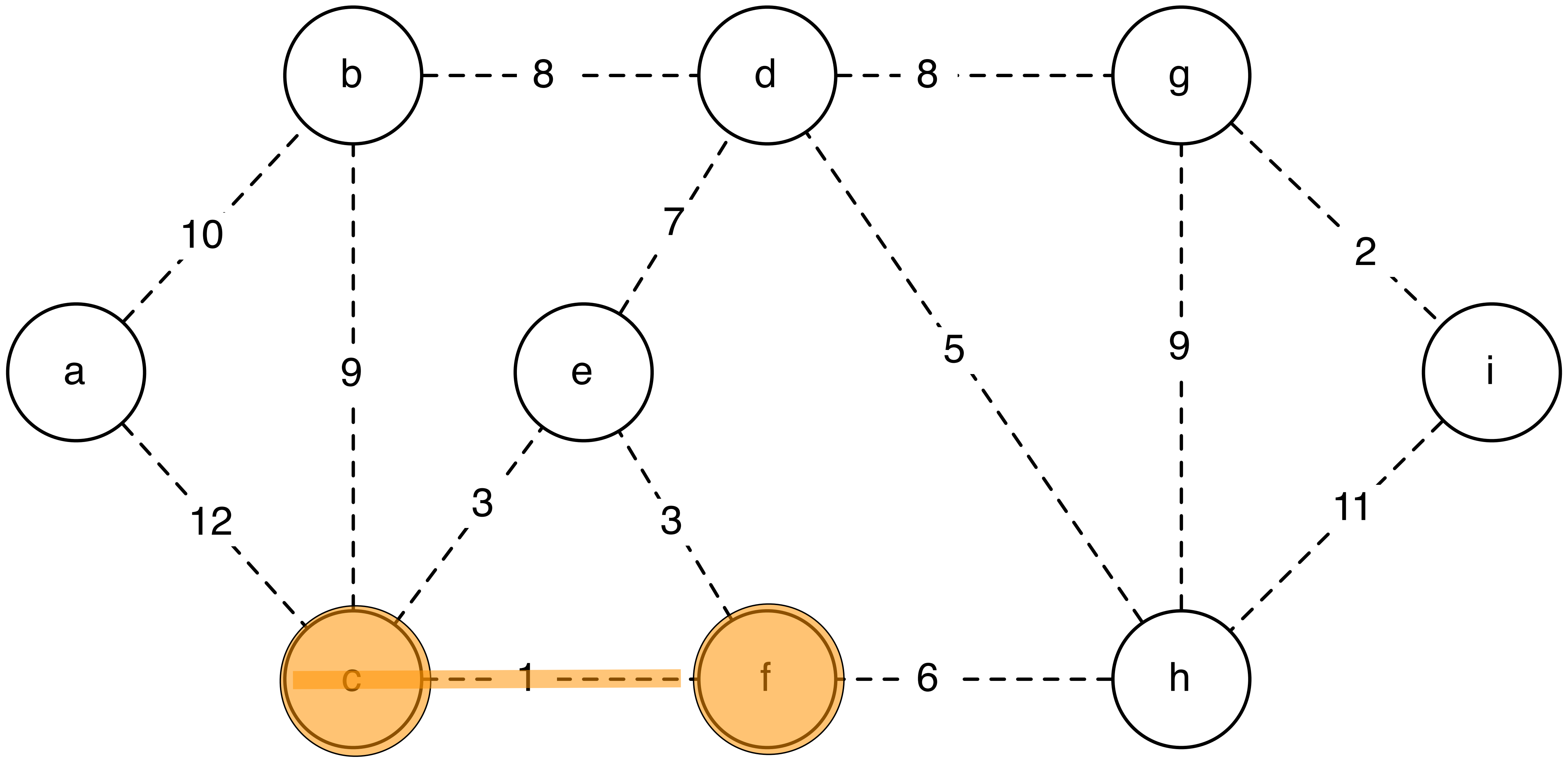
repeat ~~for v-1 times:~~

add lightest edge that does not create a cycle

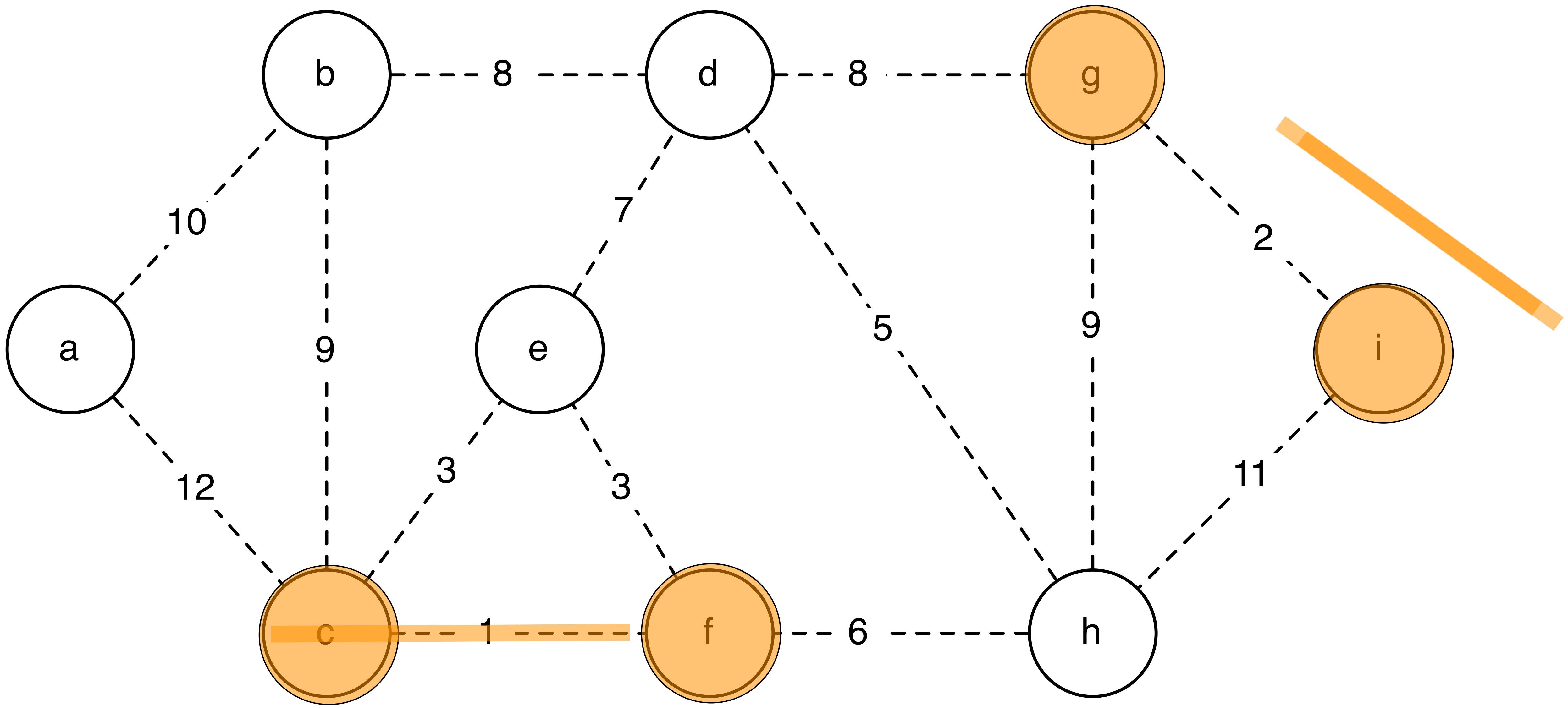
example



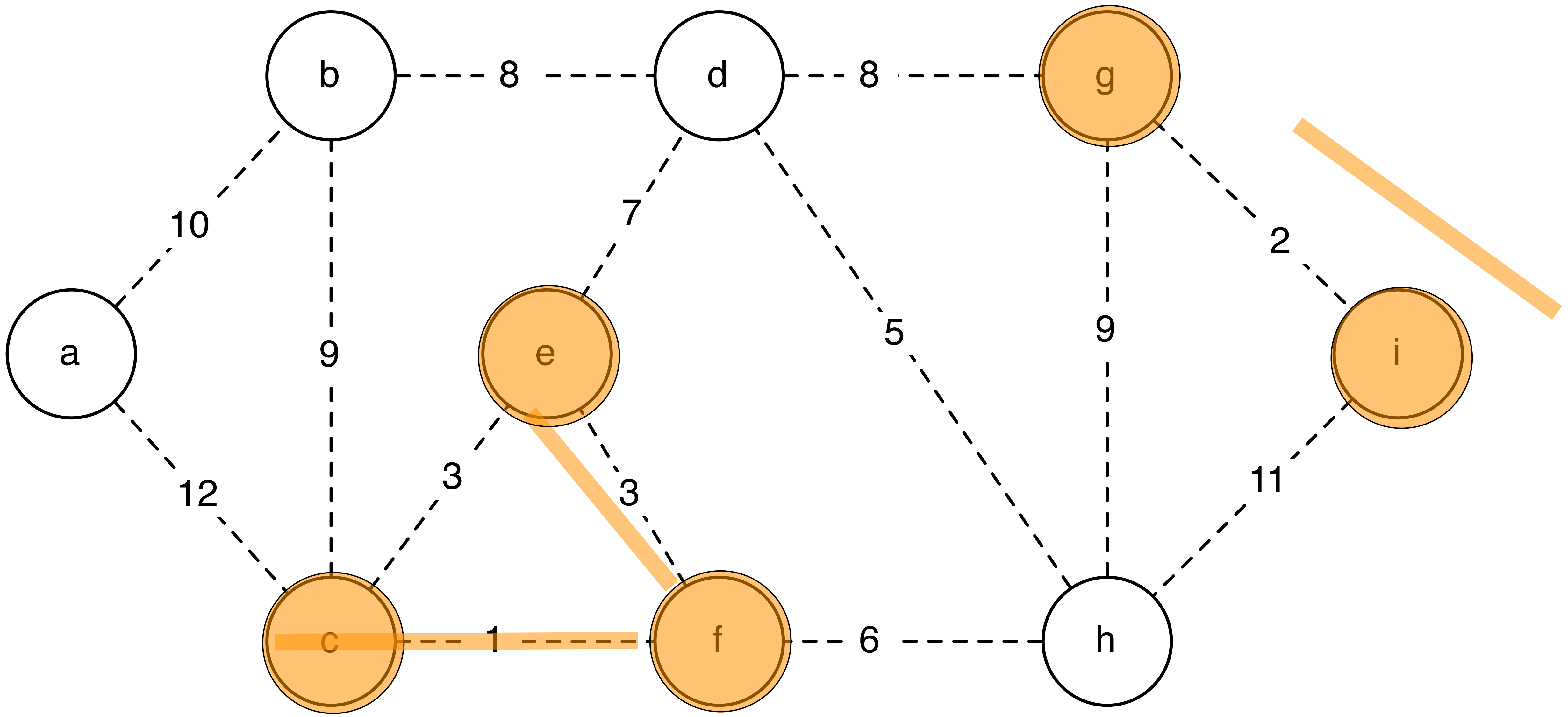
kruskal



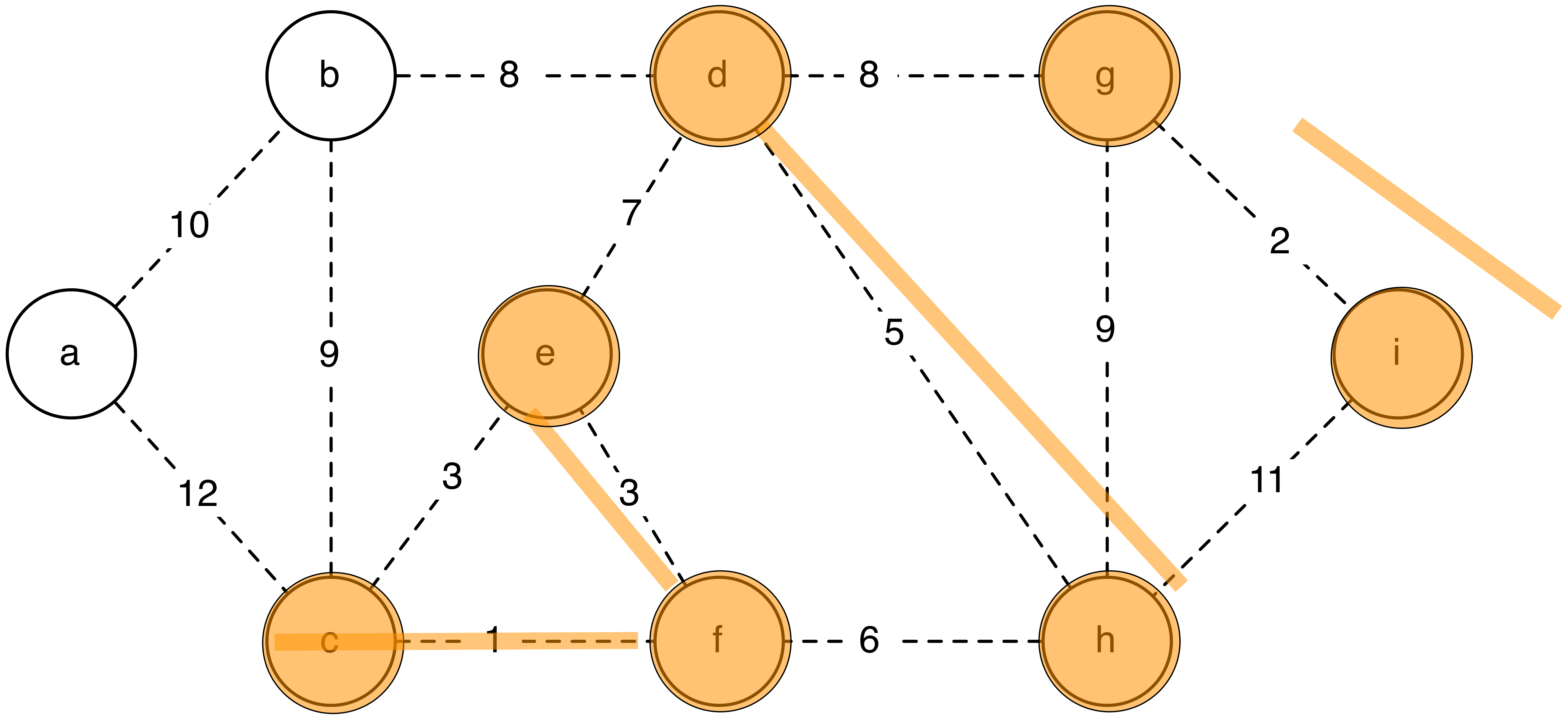
kruskal



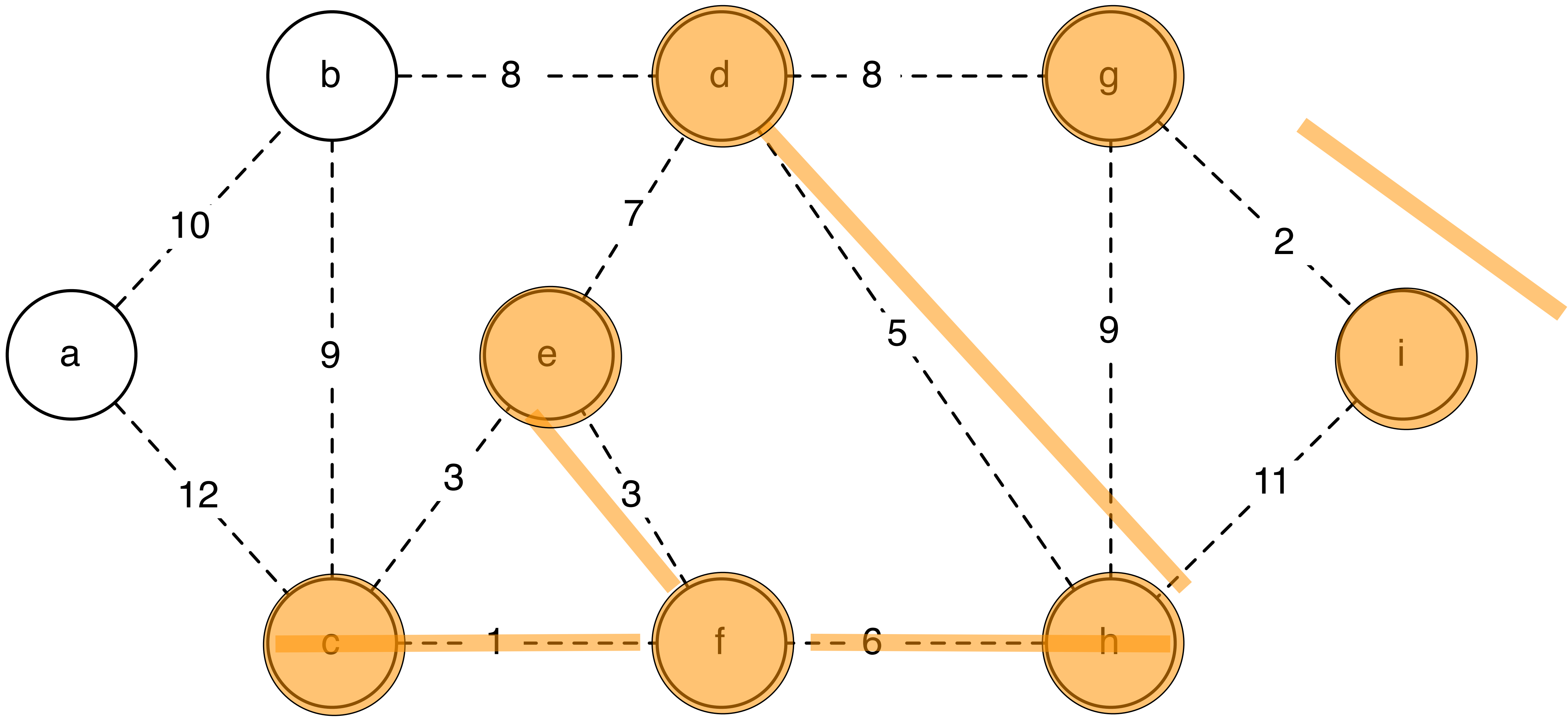
kruskal



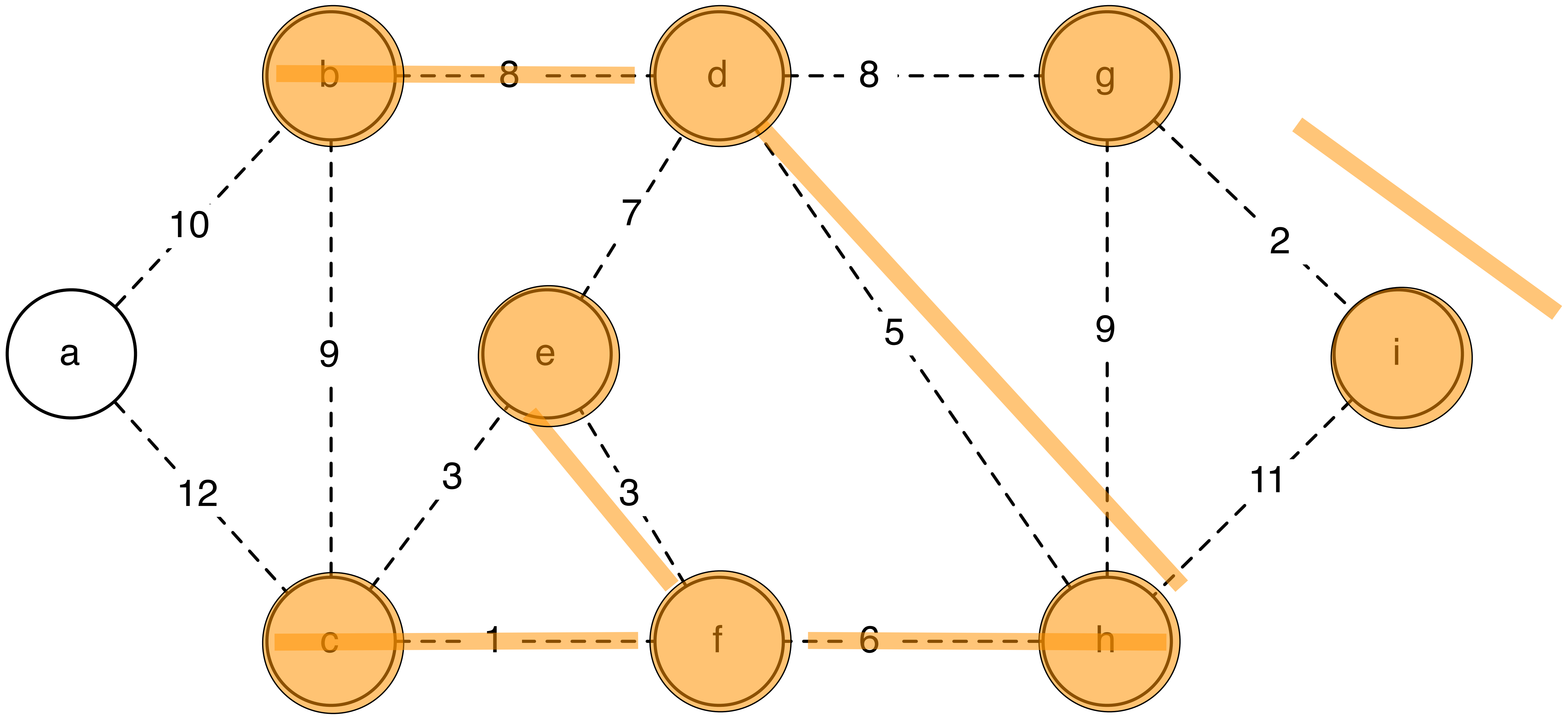
kruskal



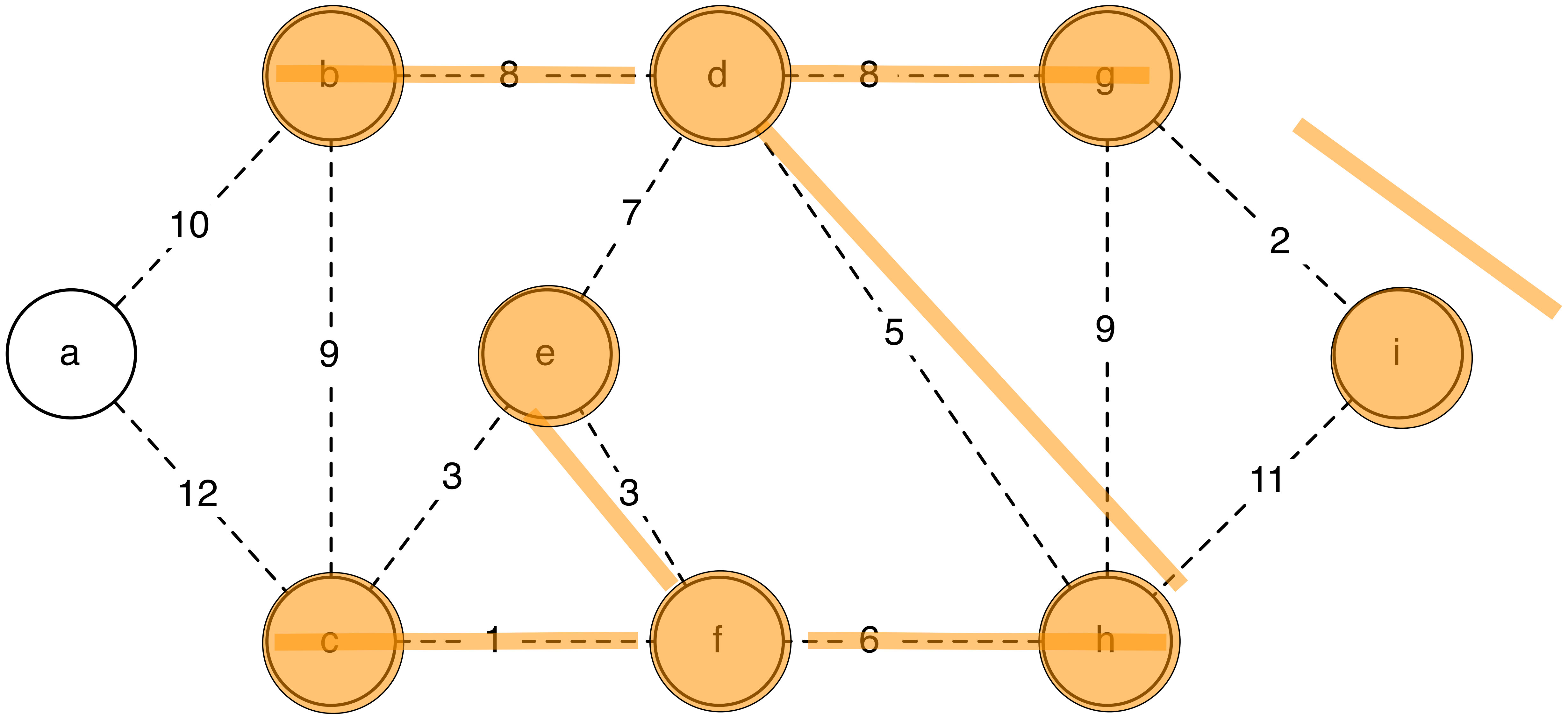
kruskal



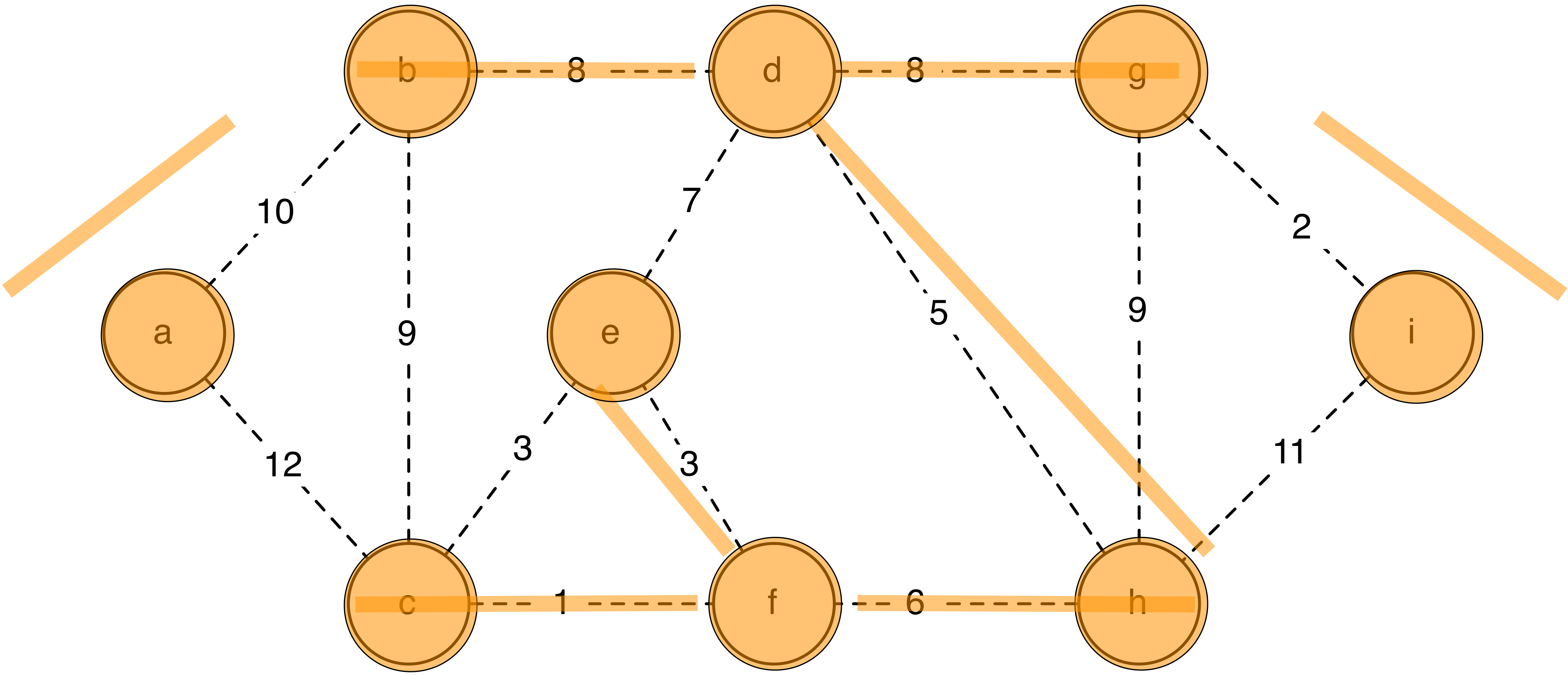
kruskal



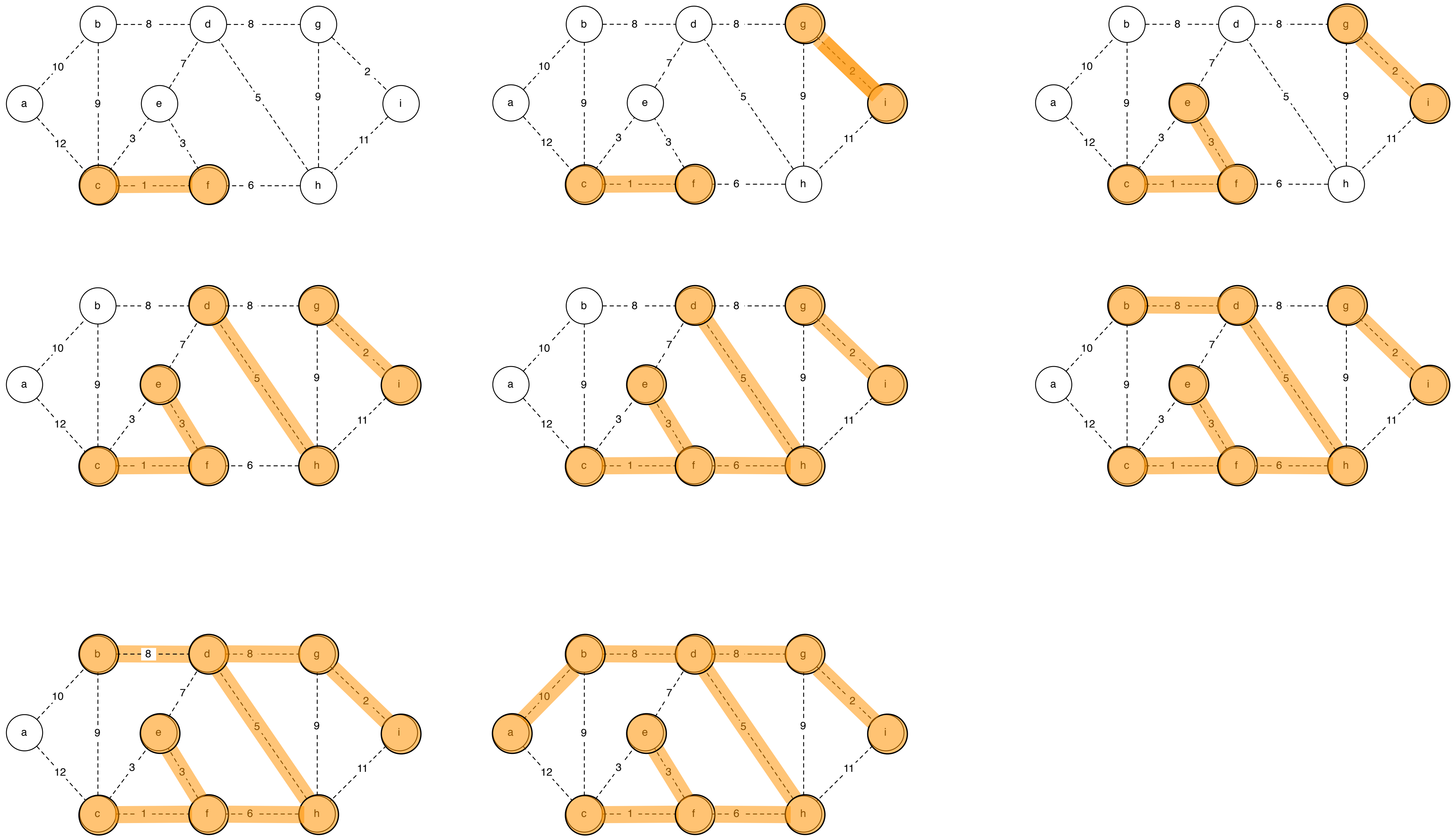
kruskal



kruskal



kruskal

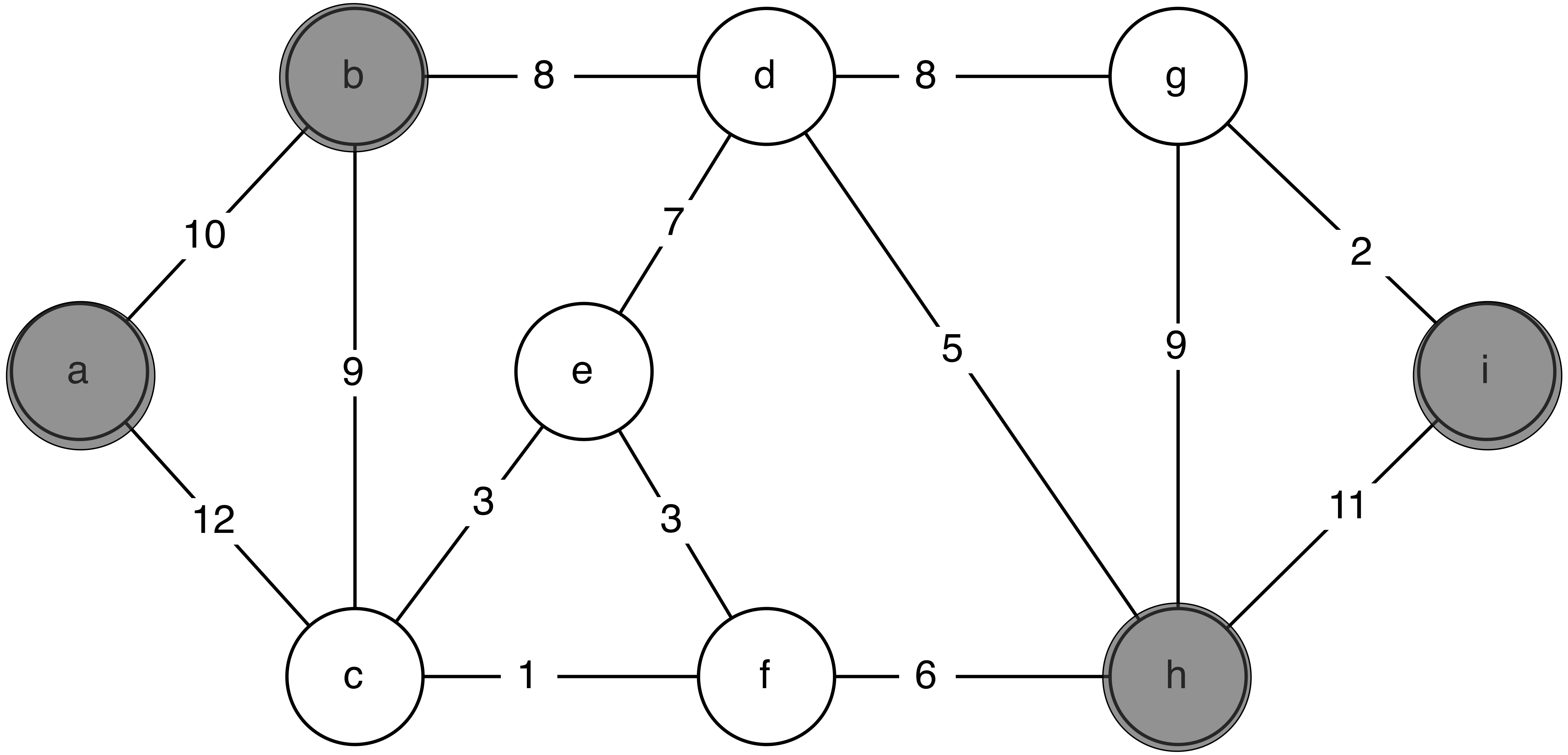


why does this work?

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

definition: cut

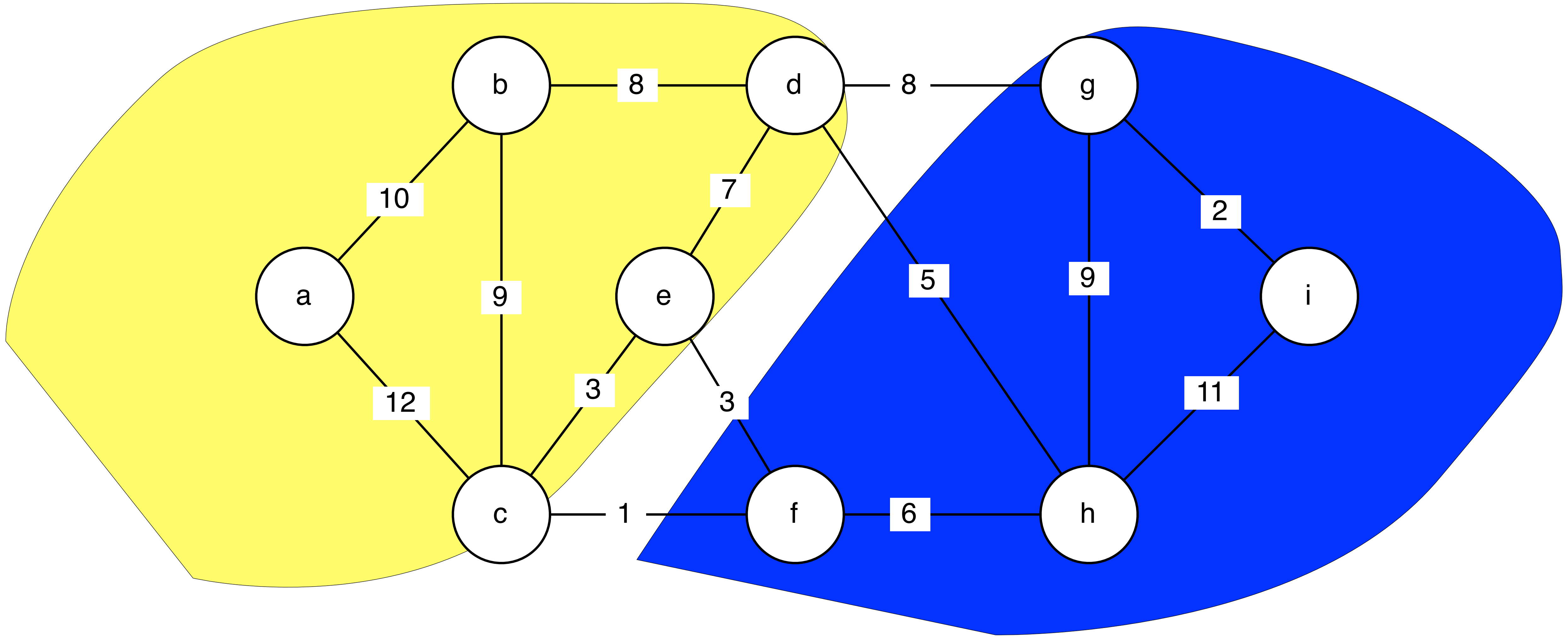
example of a cut



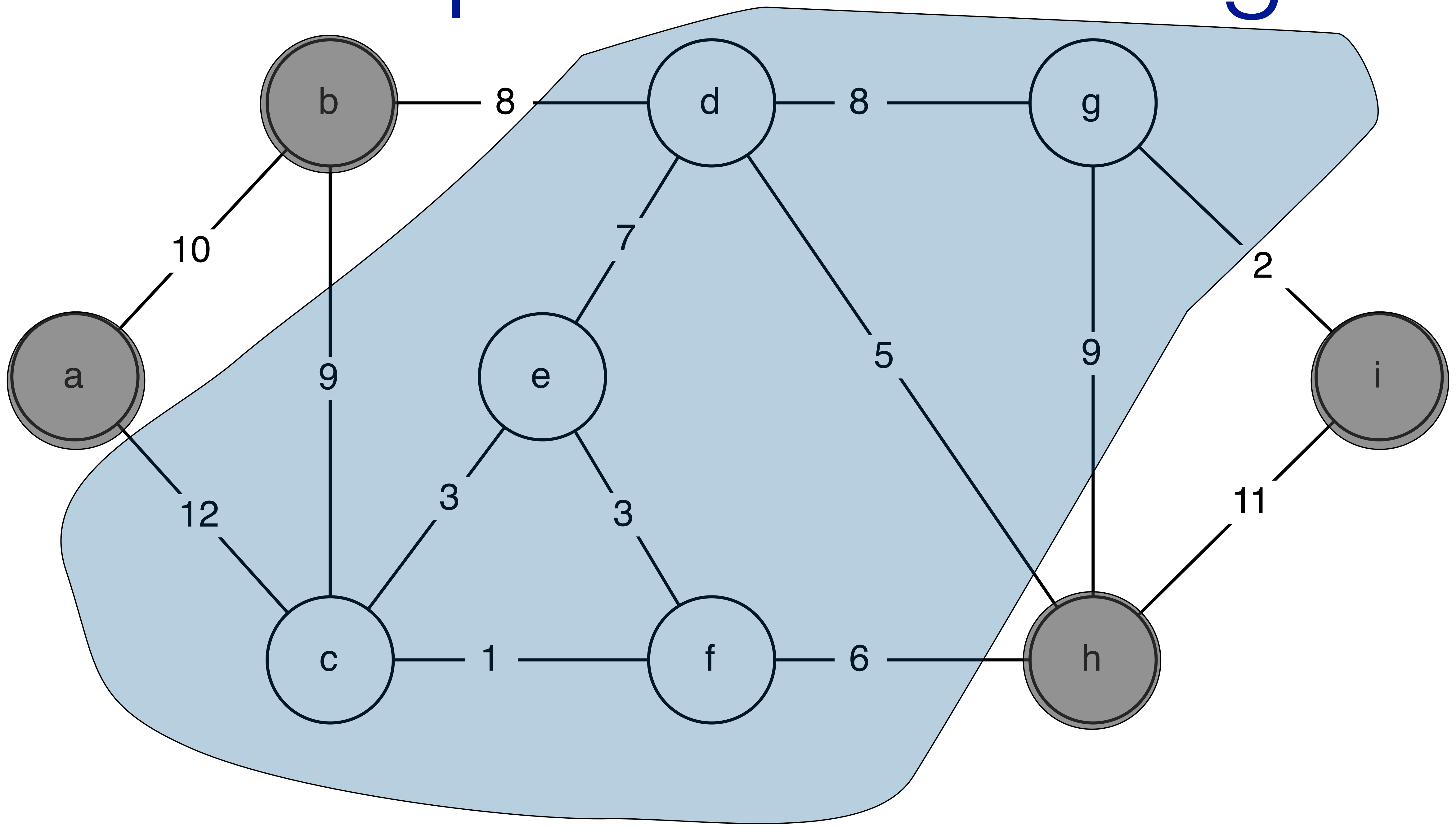
definition: crossing a cut

definition: crossing a cut

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



example of a crossing



definition: respect

theorem: cut property

thm: cut property

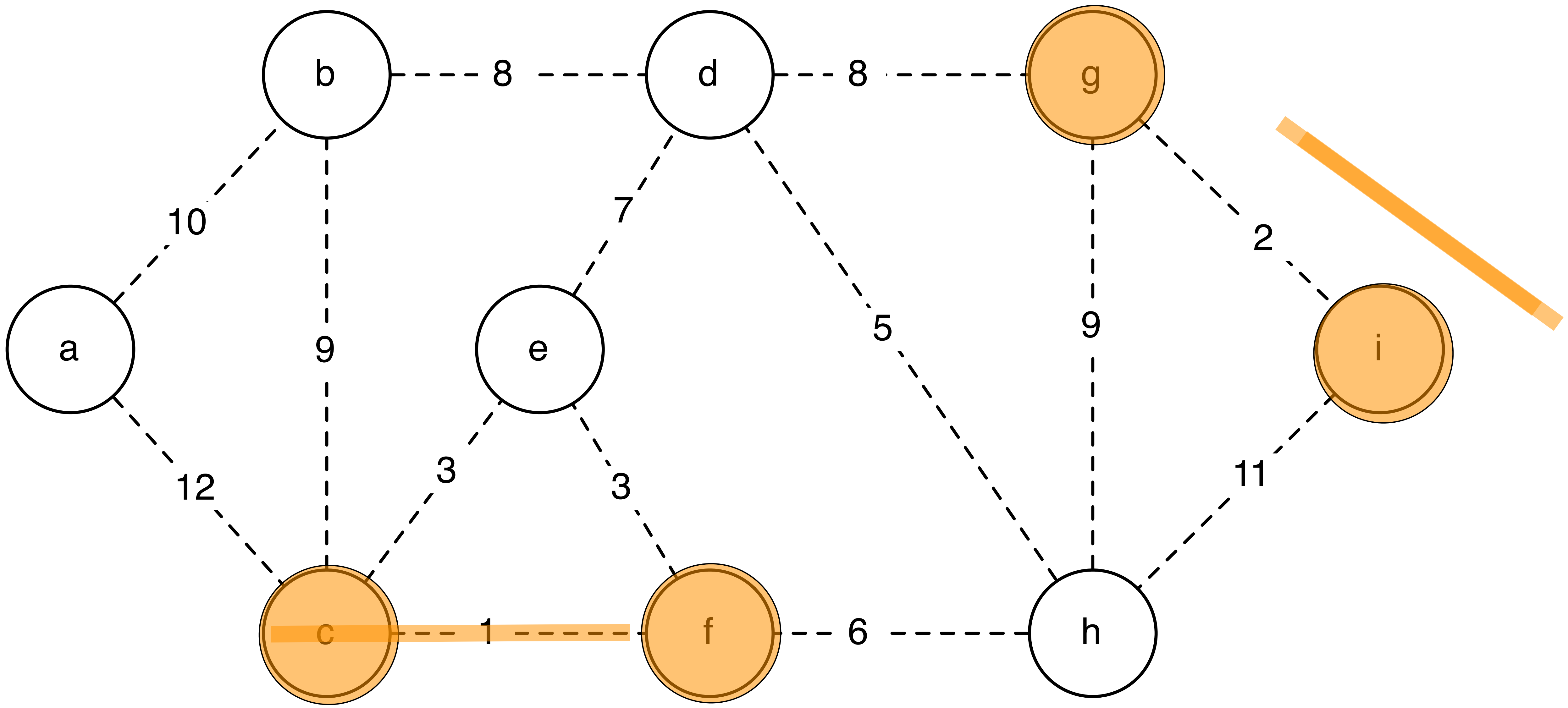
suppose the set of edges A is part of an m.s.t.

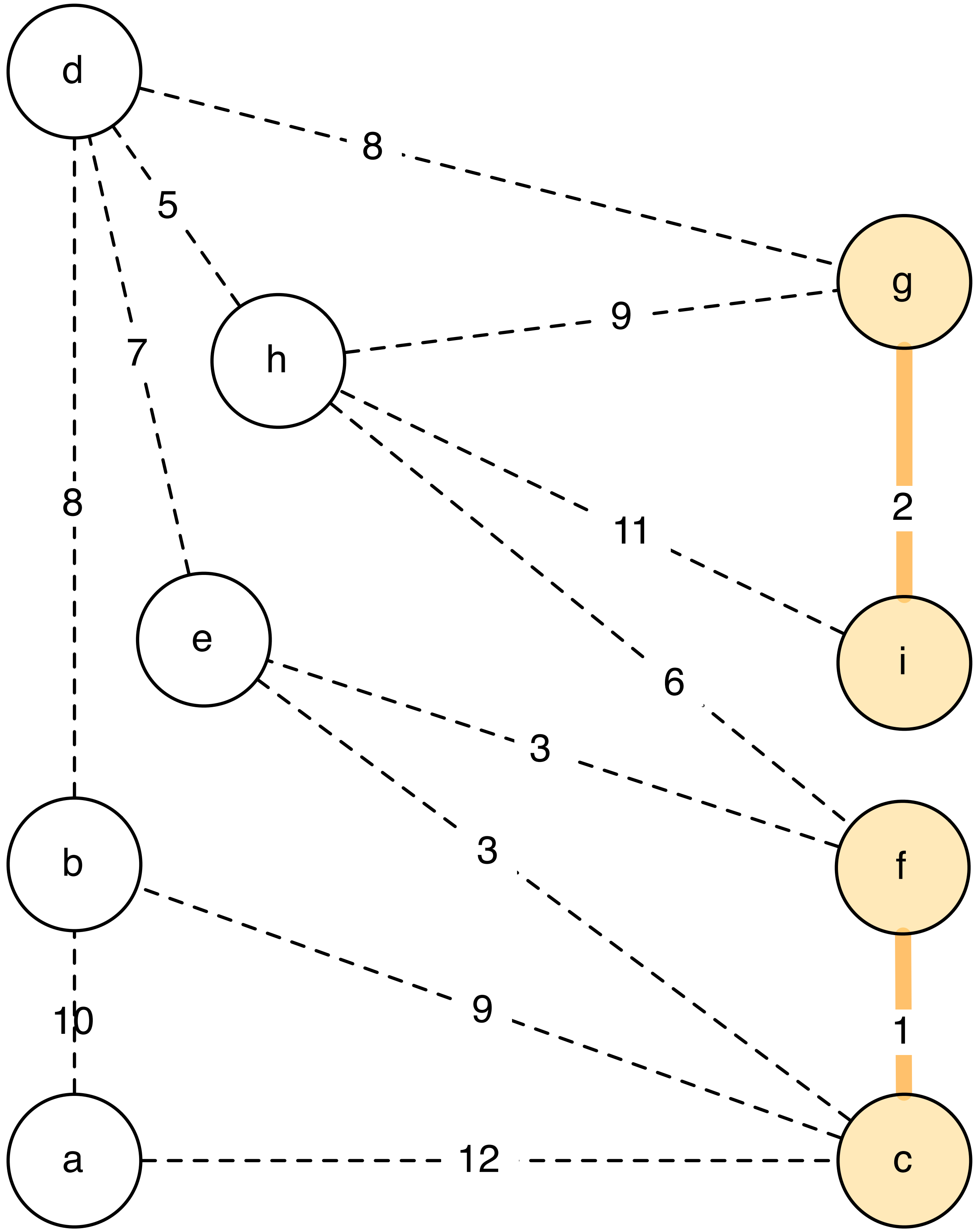
let $(S, V-S)$ be any cut that respects A.

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

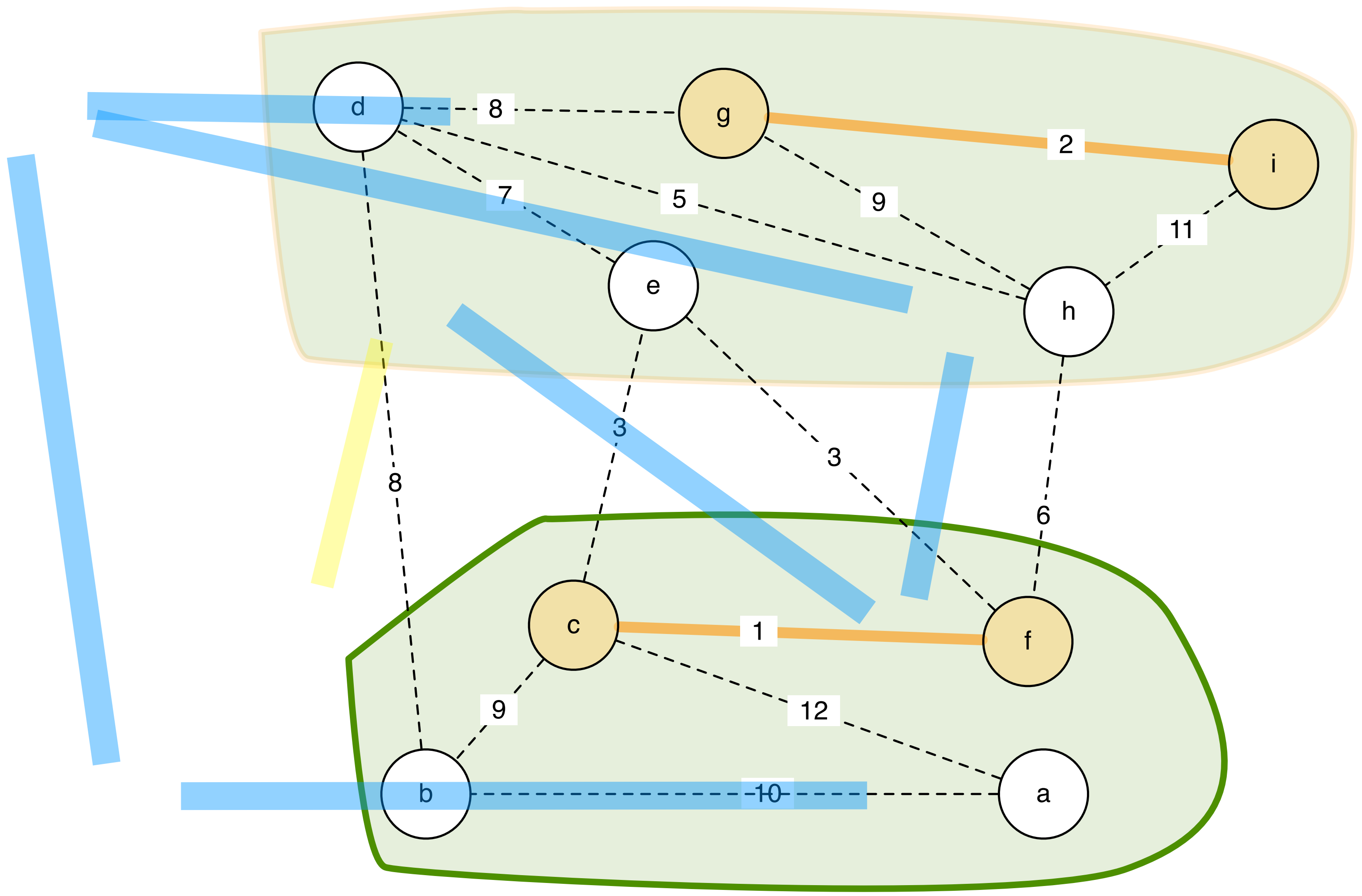




d
8
5
g
9
h
7
8
11
2
e
6
i
3
f
3
b
9
1
10
a
12
c

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



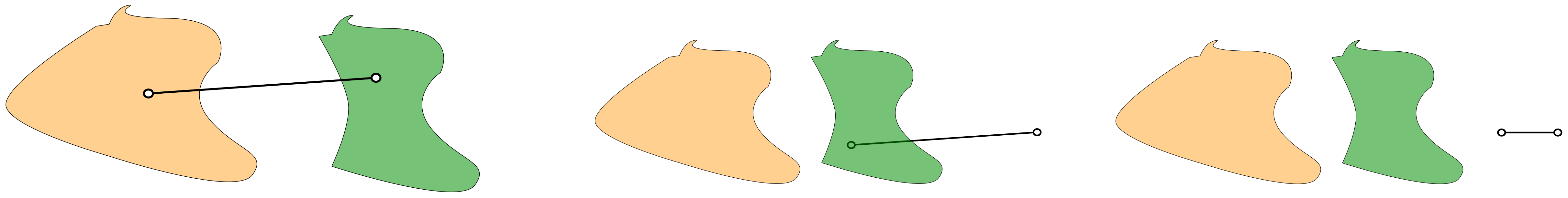
correctness

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

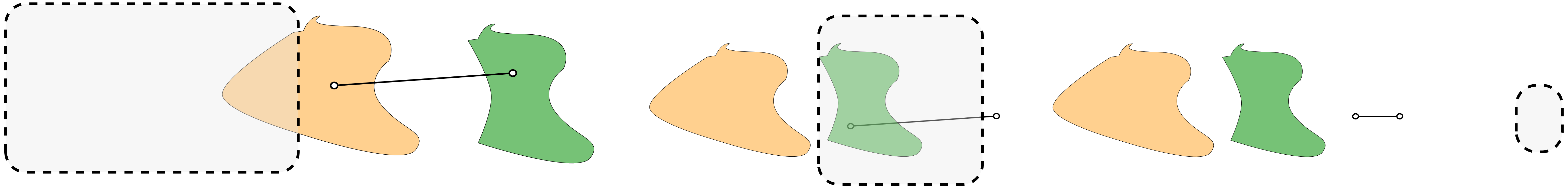
```
KRUSKAL-PSEUDOCODE(G)
1  A ← ∅
2  repeat  V − 1 times:
3              add to A the lightest edge e ∈ E that does not create a cycle
```

correctness

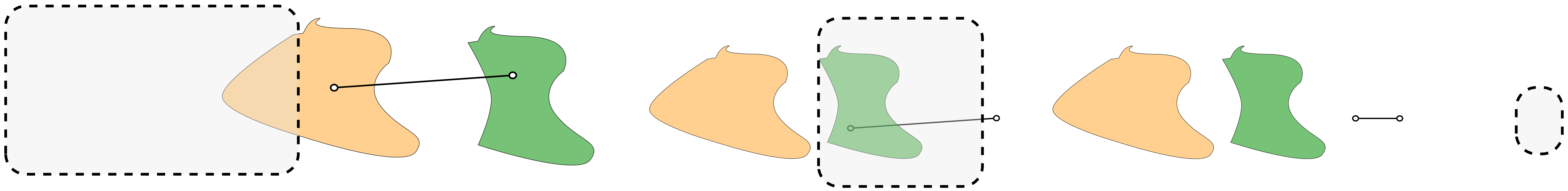
proof: by induction. in step 1, A is part of some MST.
suppose that after k steps, A is part of some MST (line 2).
in line 3, we add an edge e to A.



3 cases for edge e



3 cases for edge e



e must be lightest edge crossing

analysis?

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

```
General-MST-Strategy(G = (V, E))
A ← ∅
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 ← A ∪ {e}
```

prim

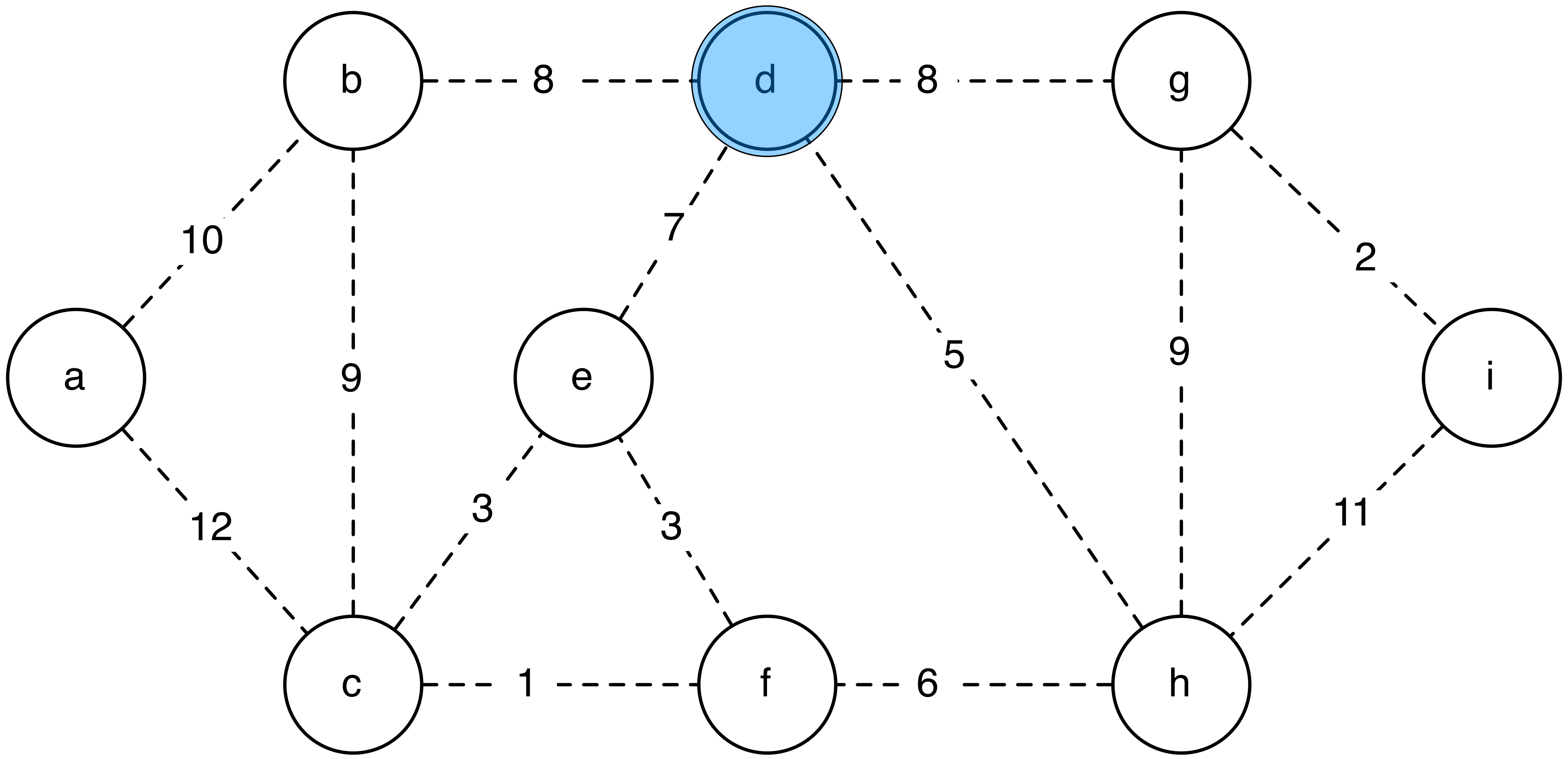
GENERAL-MST-STRATEGY$(G = (V, E))$

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

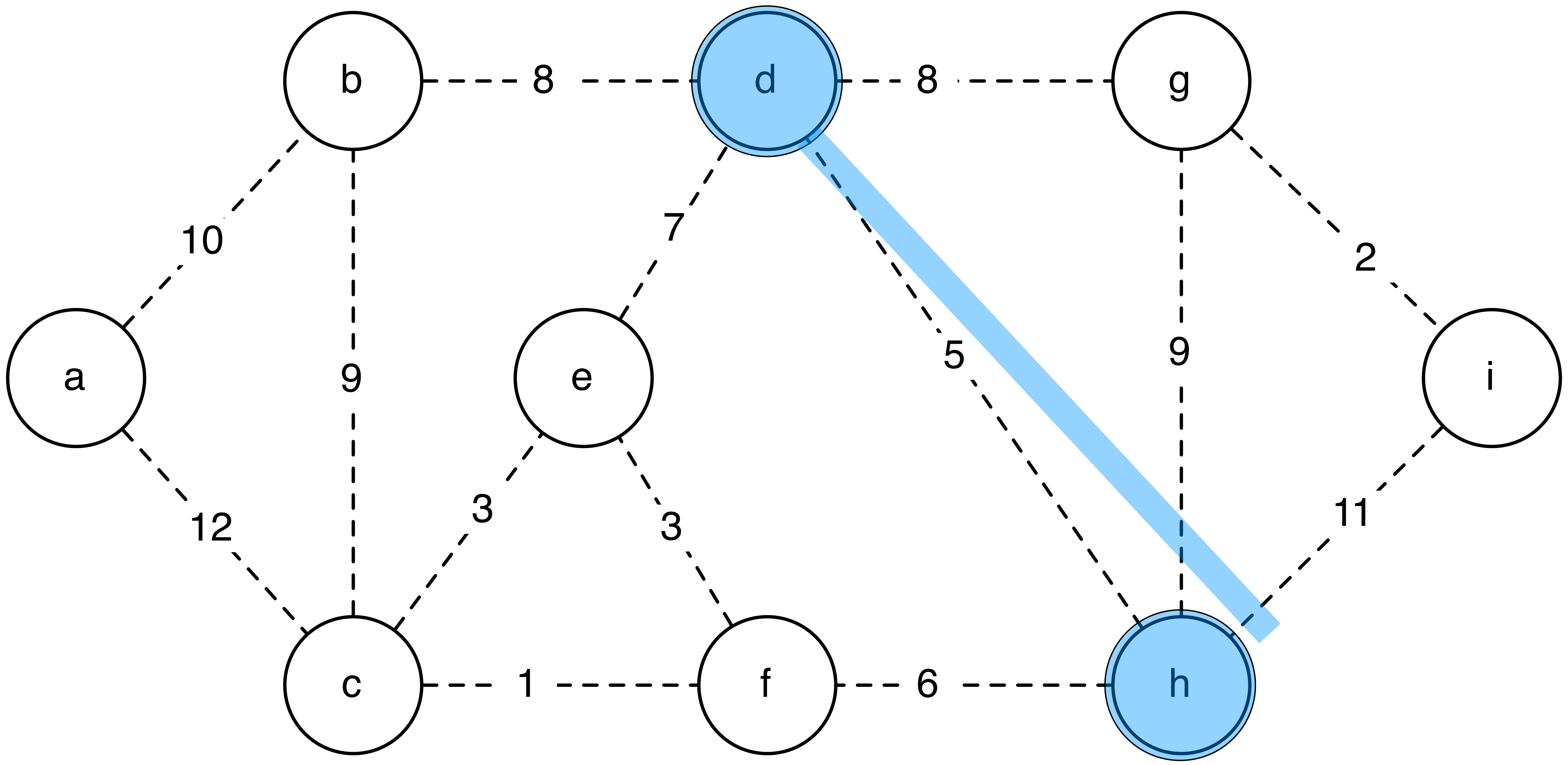
A is a subtree

edge e is lightest edge that grows the subtree

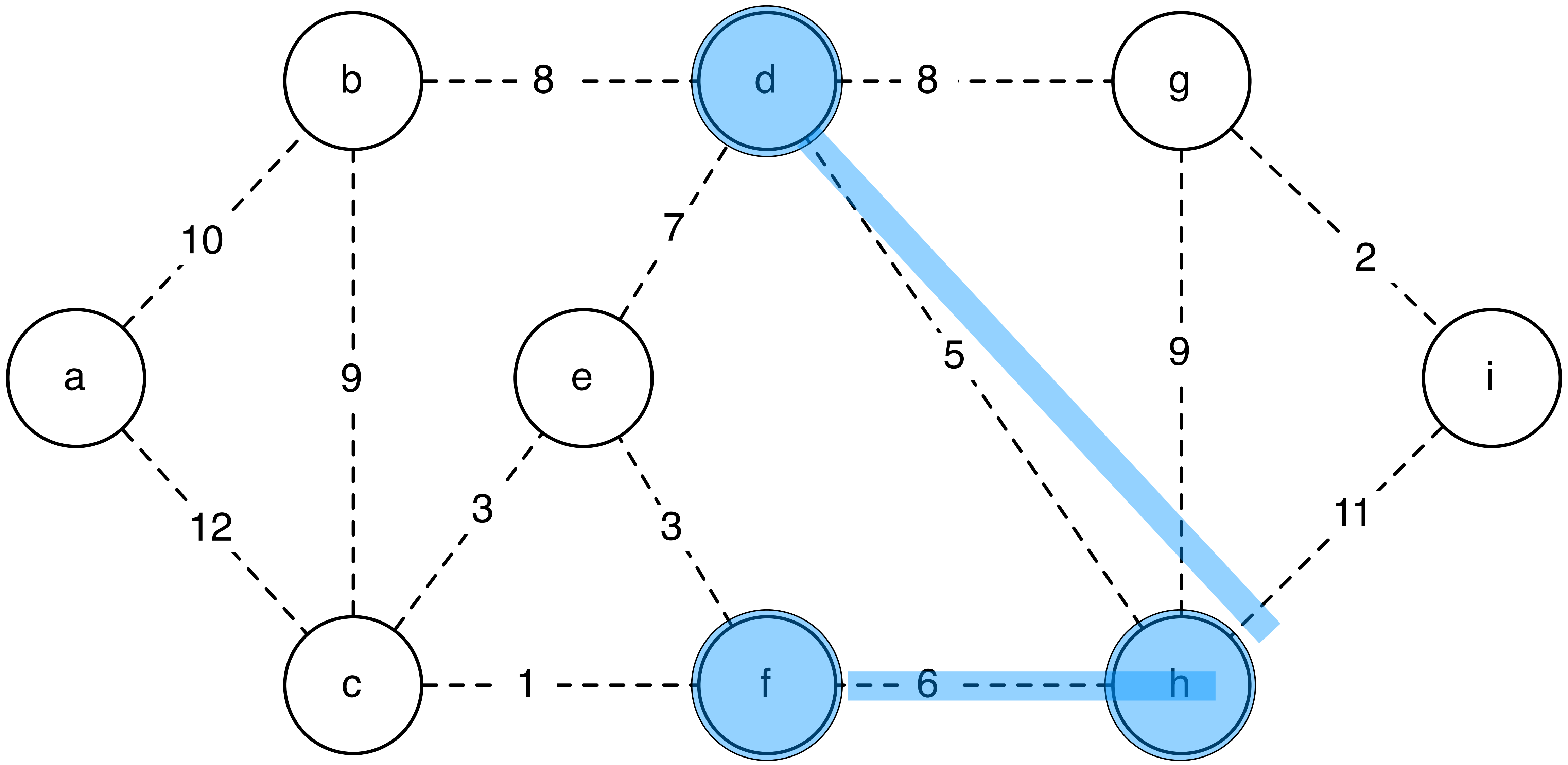
prim



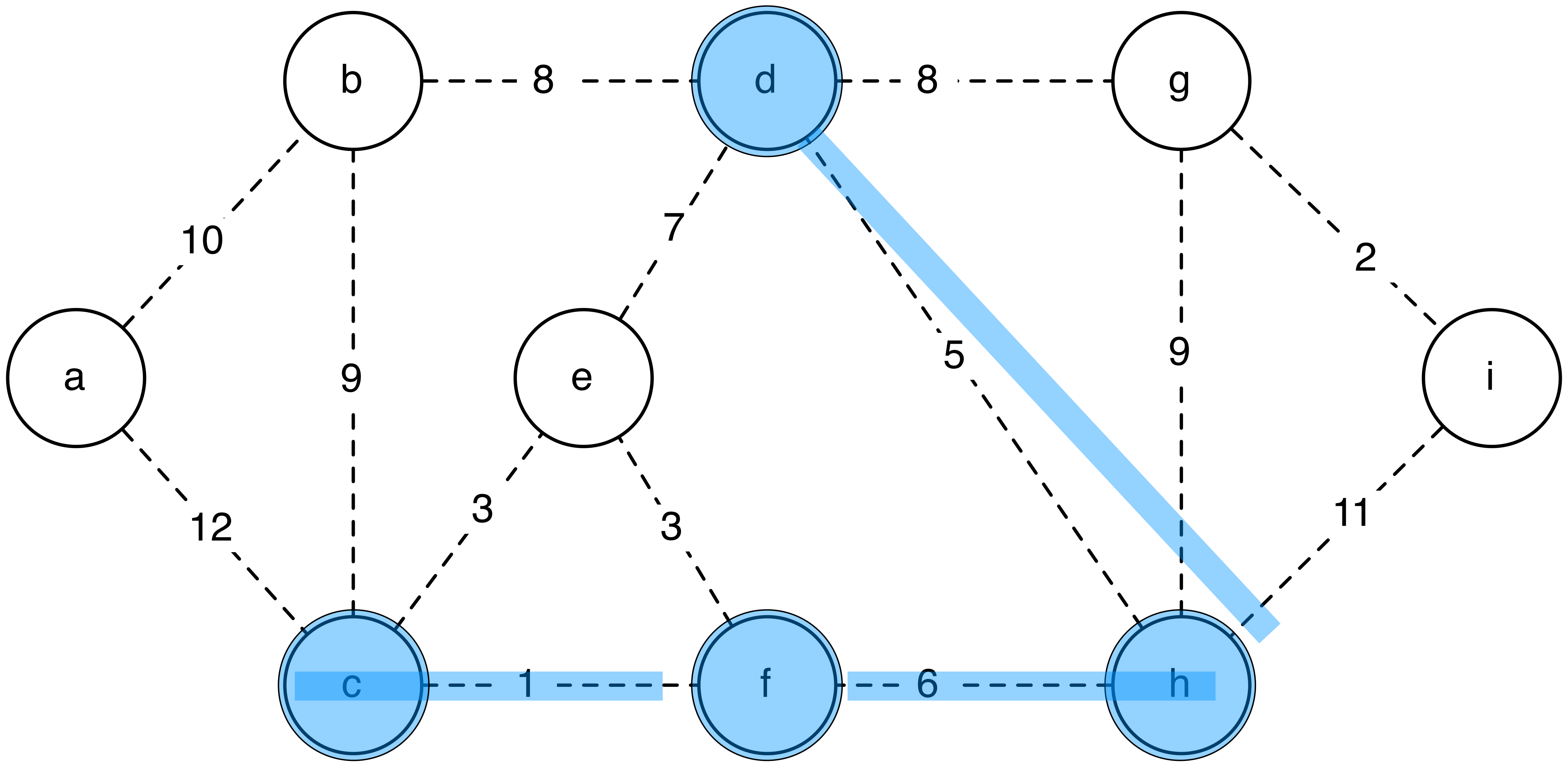
prim



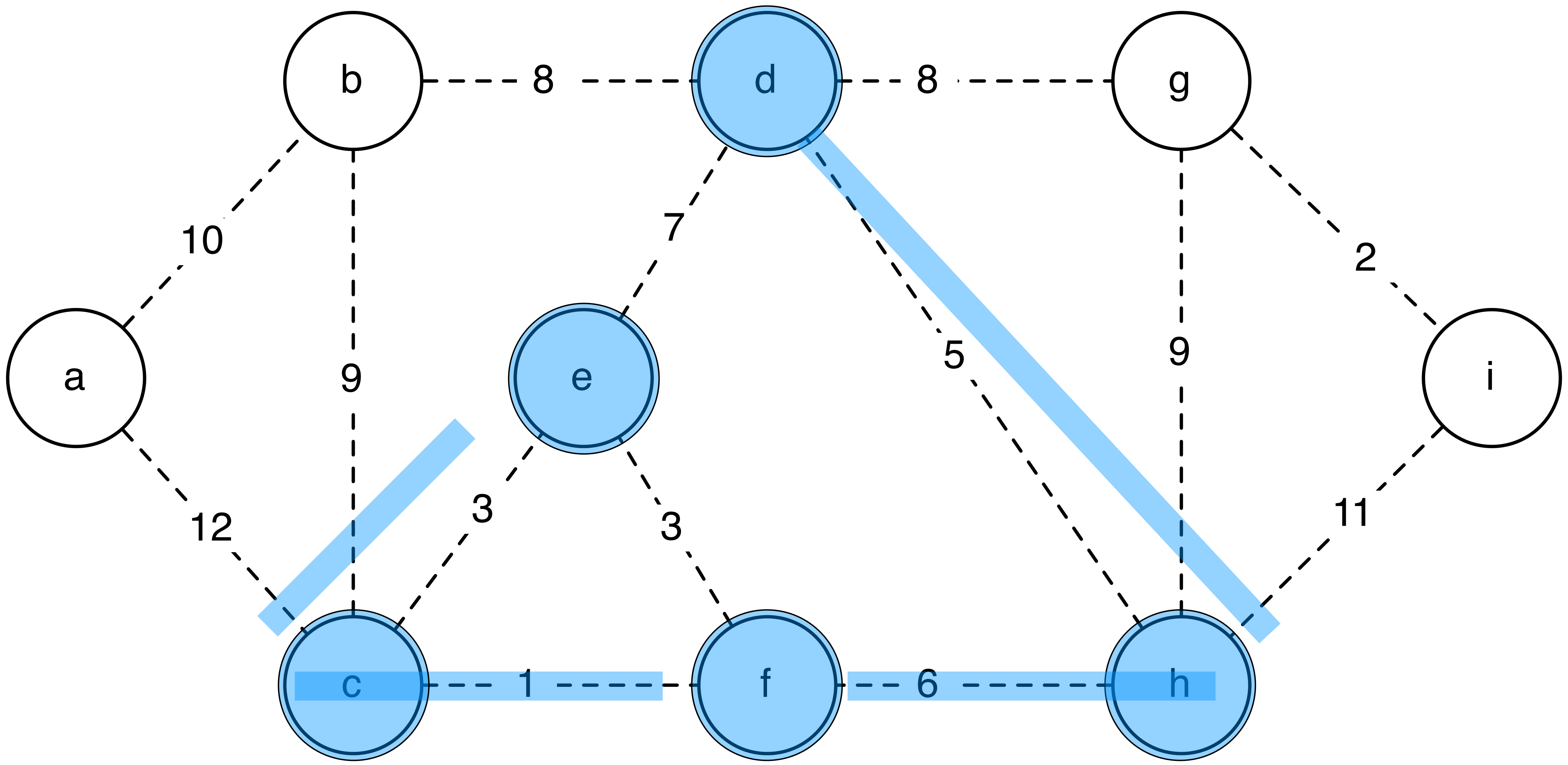
prim



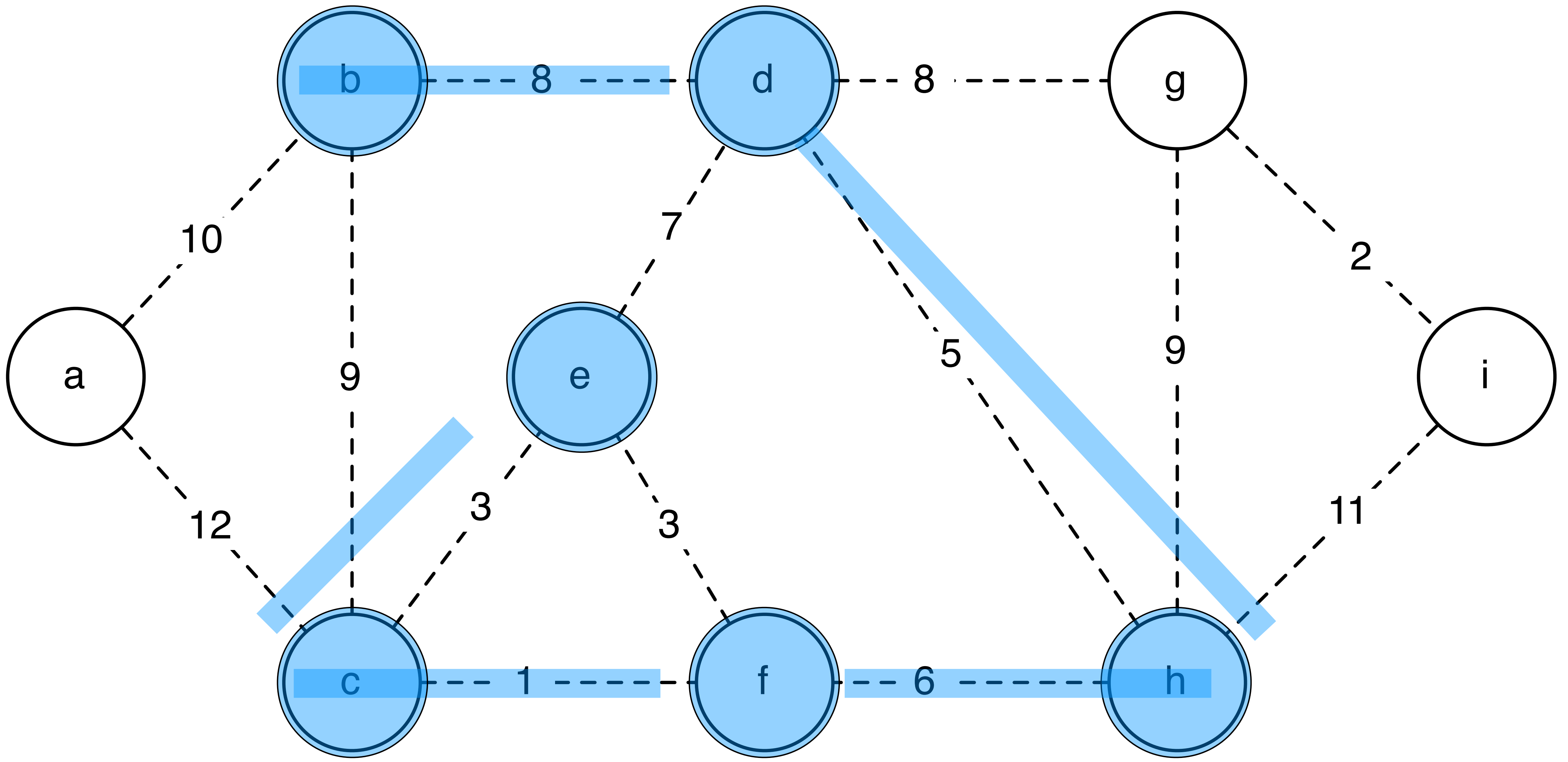
prim



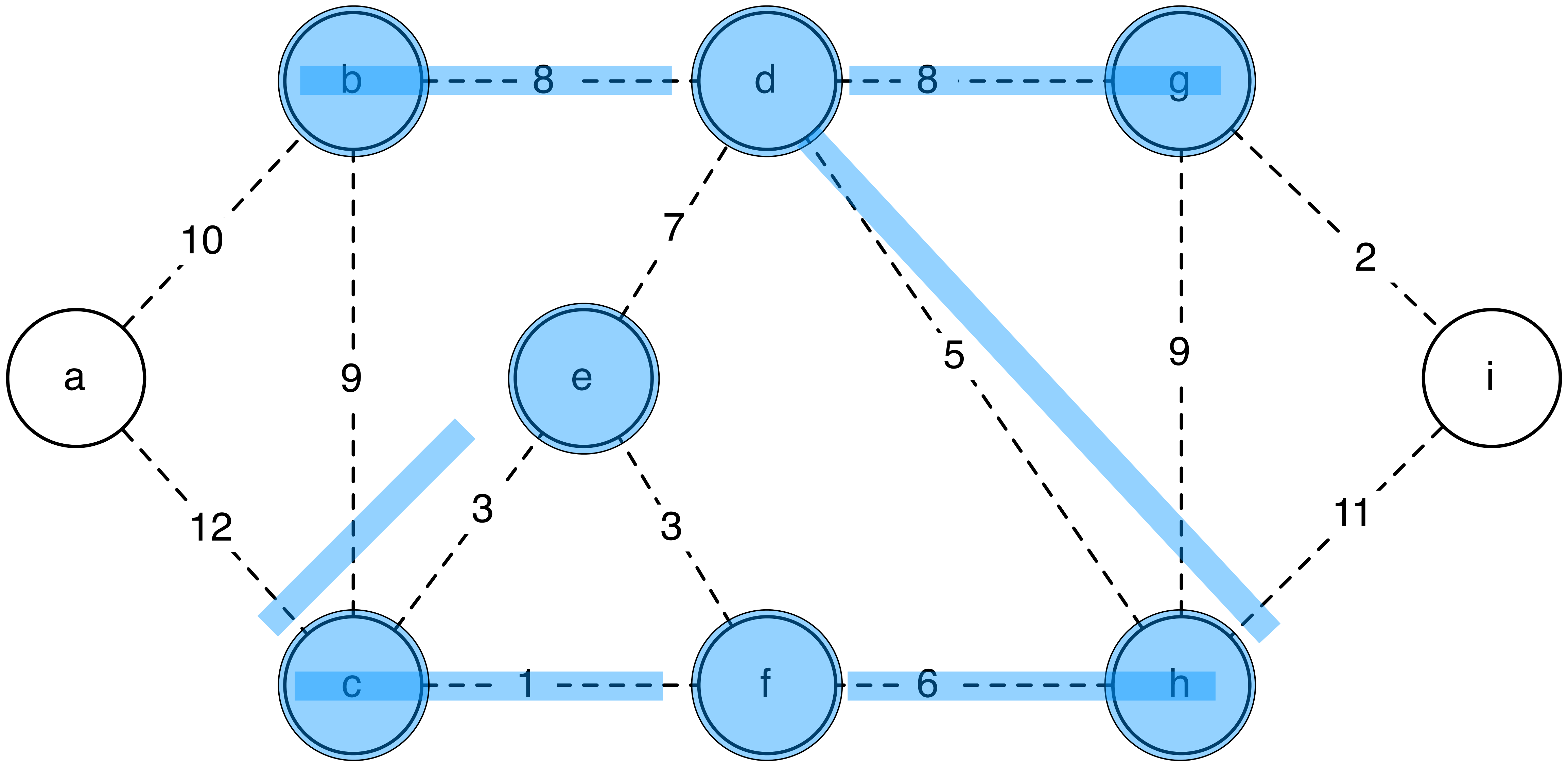
prim



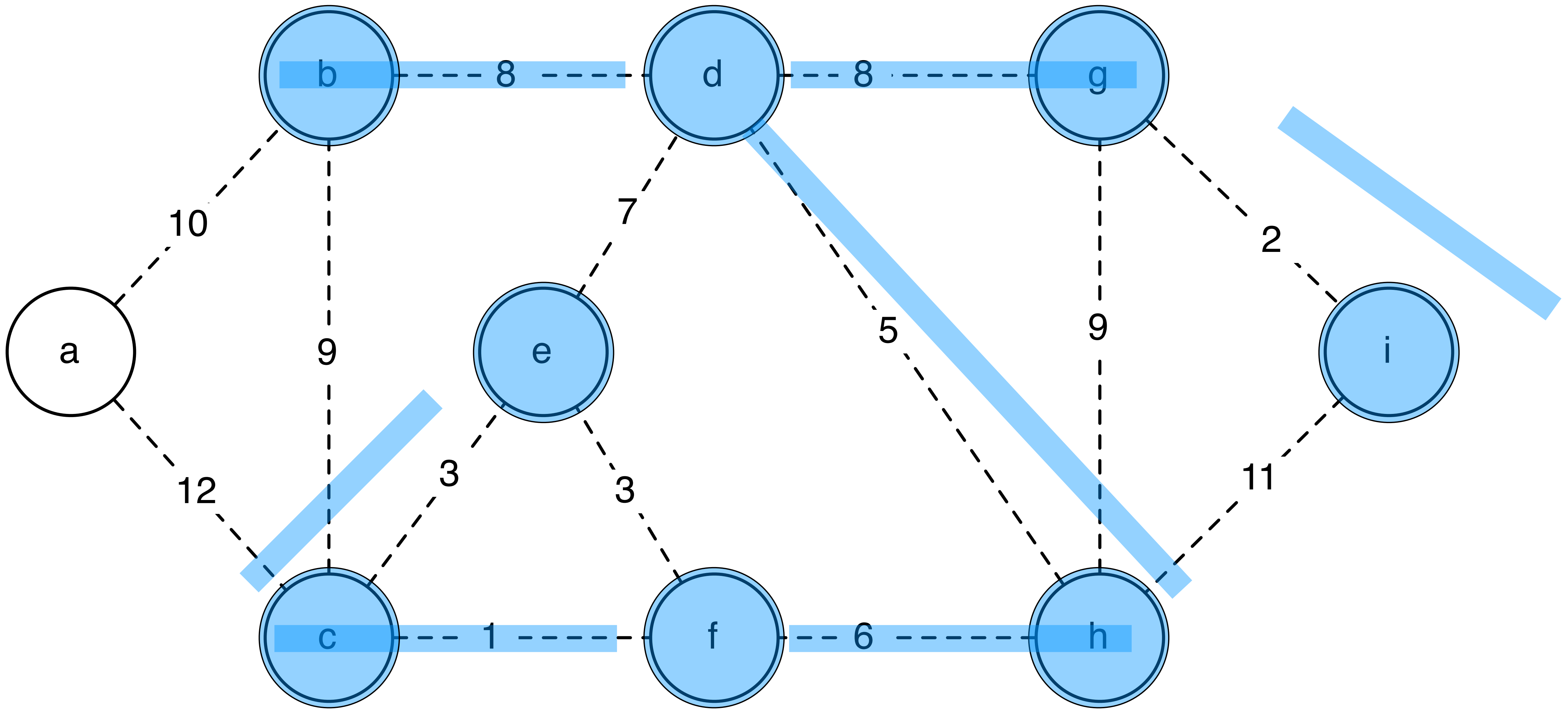
prim



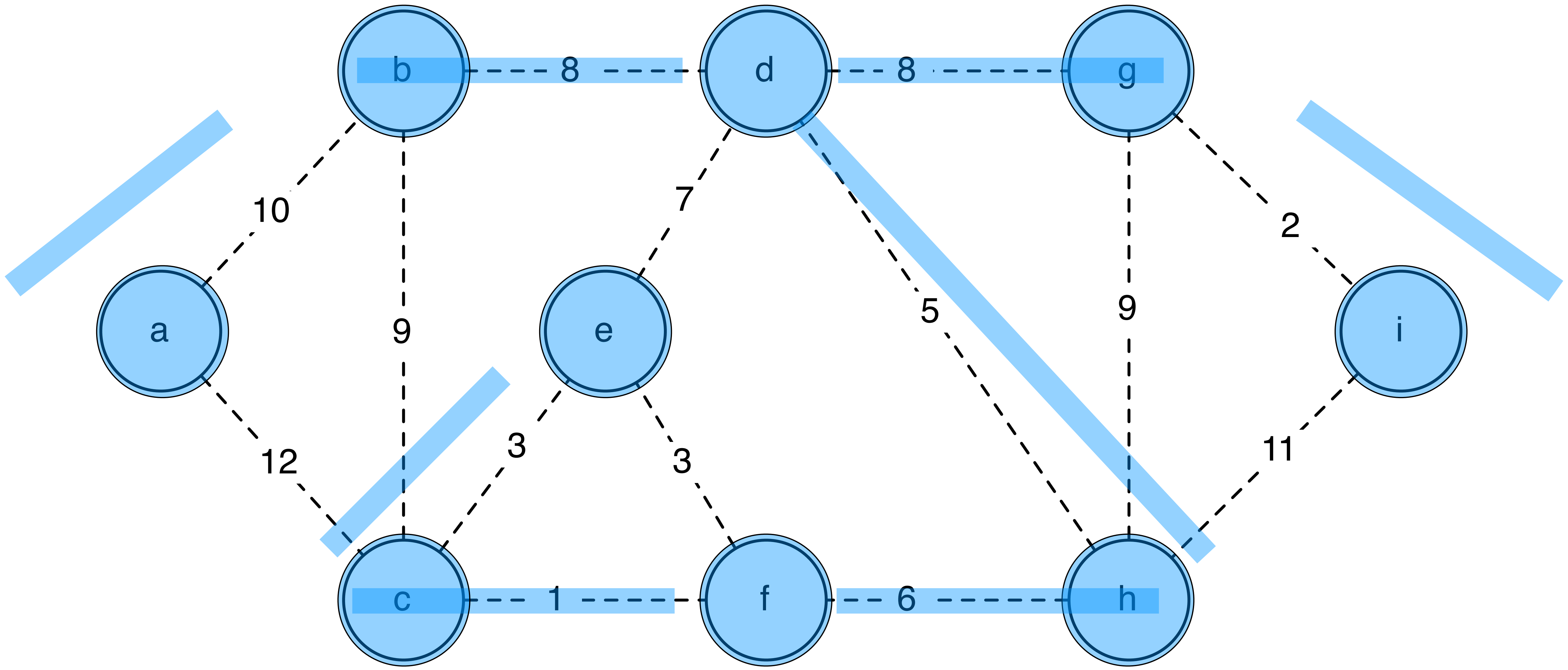
prim



prim



prim



implementation

idea:

implementation

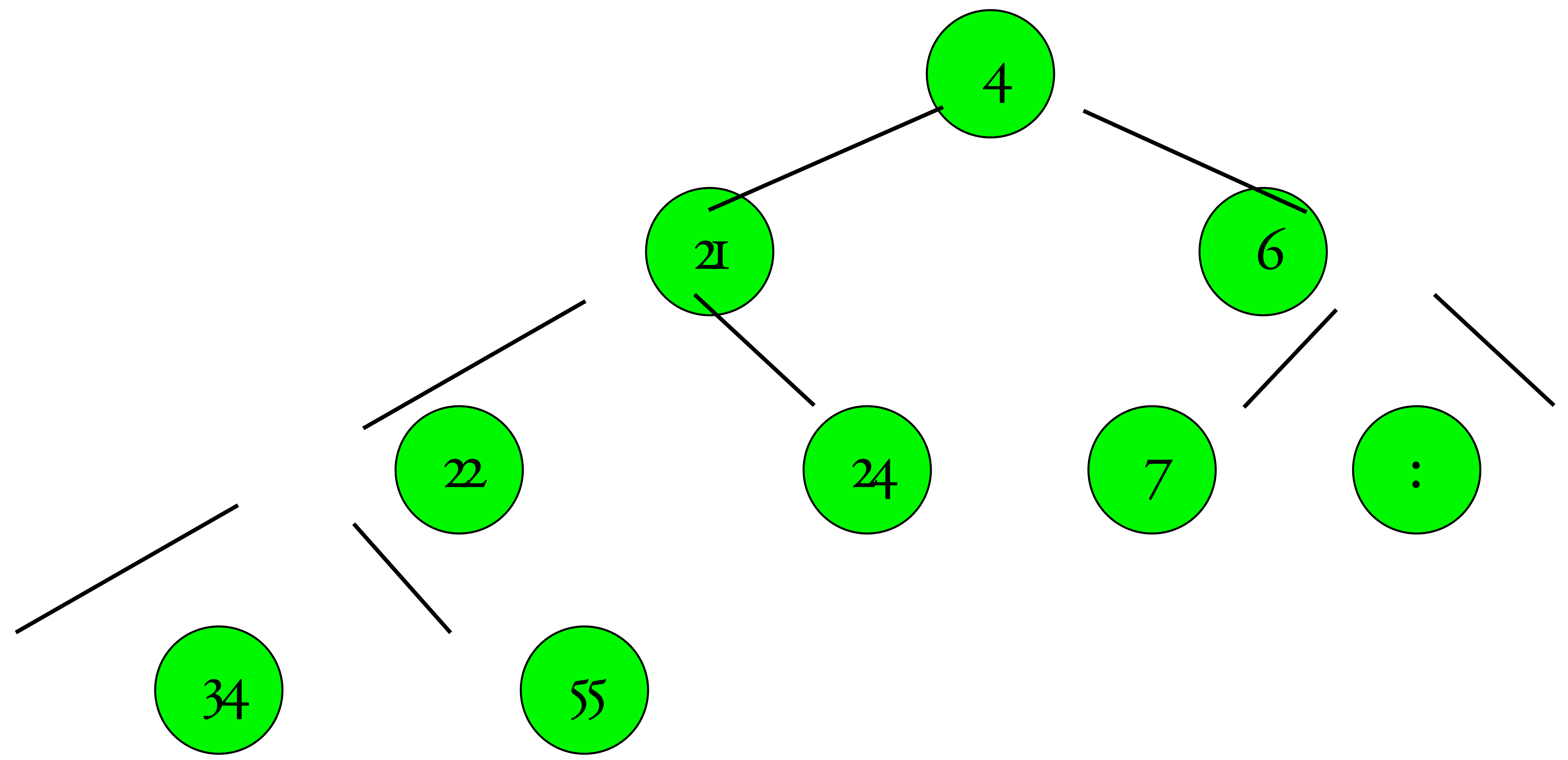
new data structure

binary heap

full tree, key value <= to key of children

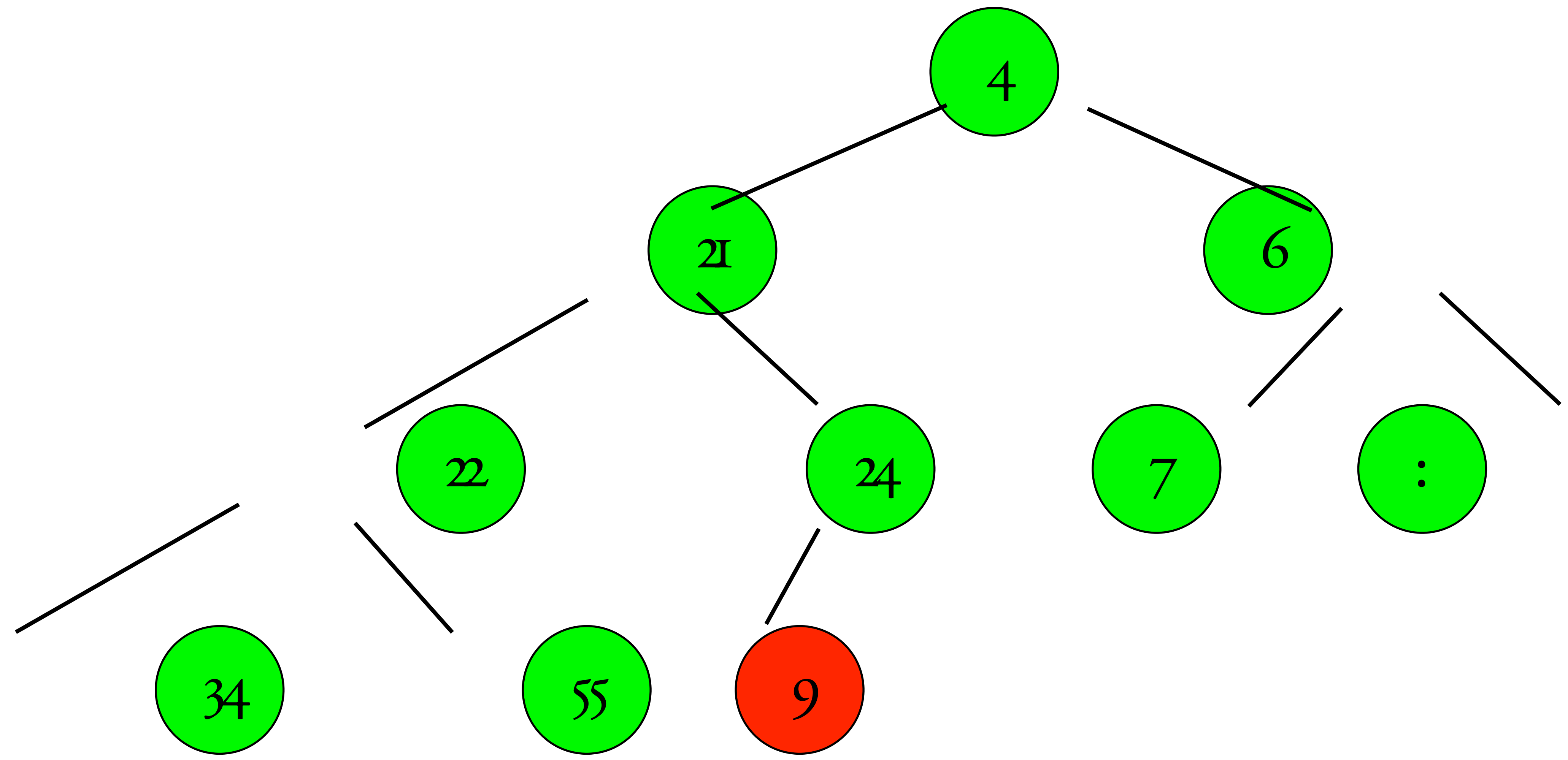
binary heap

full tree, key value <= to key of children



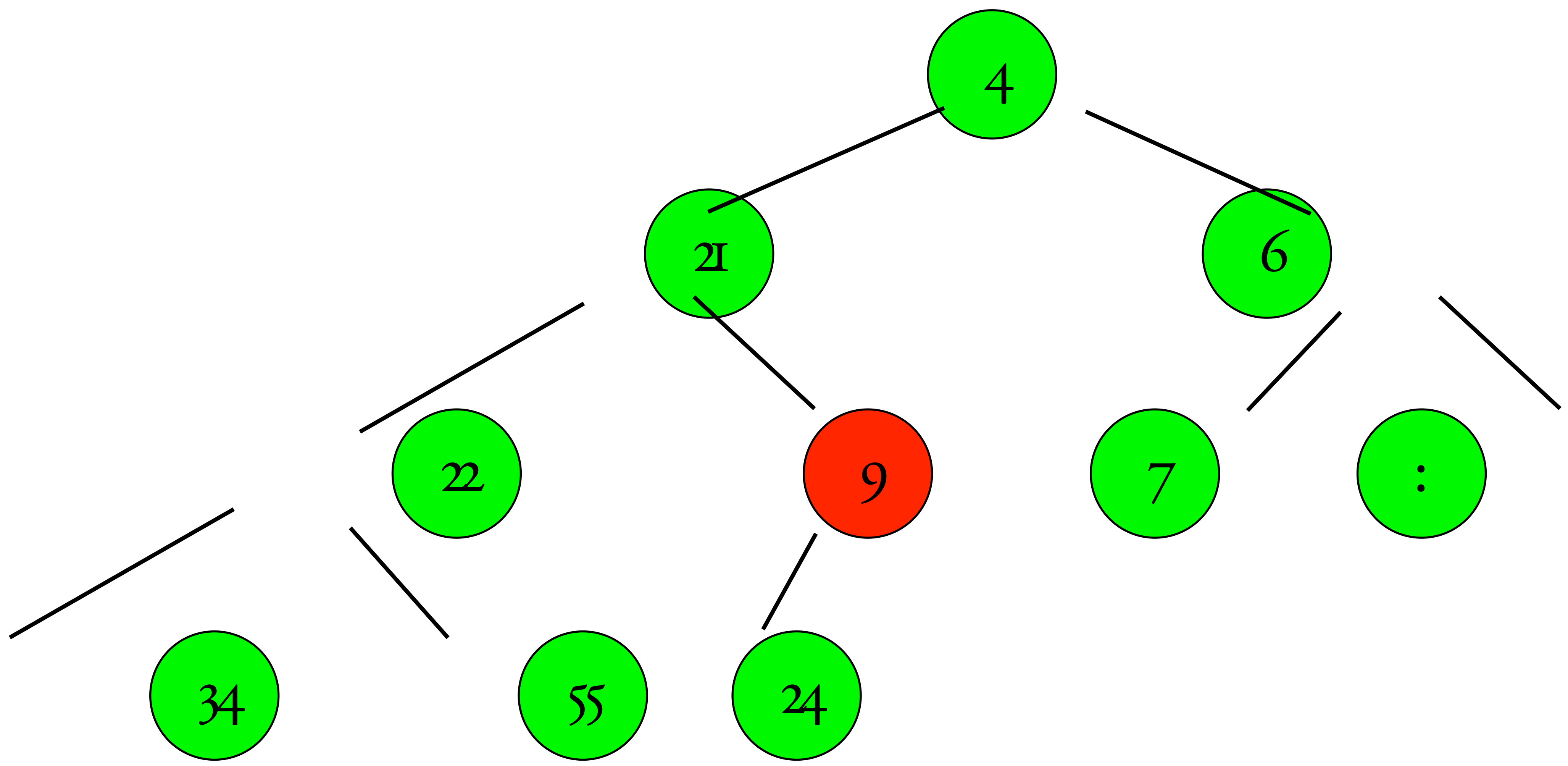
binary heap

full tree, key value <= to key of children



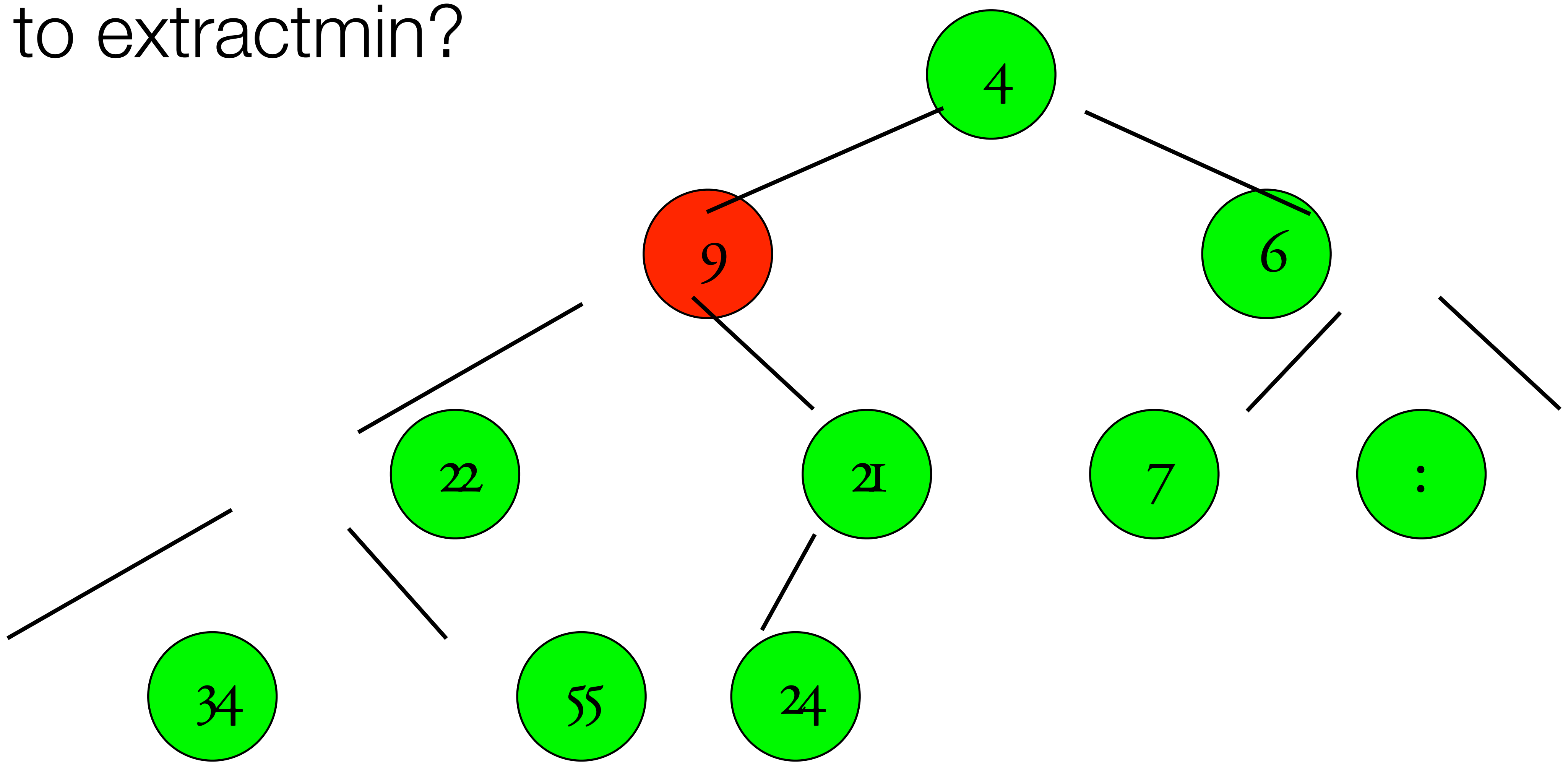
binary heap

full tree, key value <= to key of children



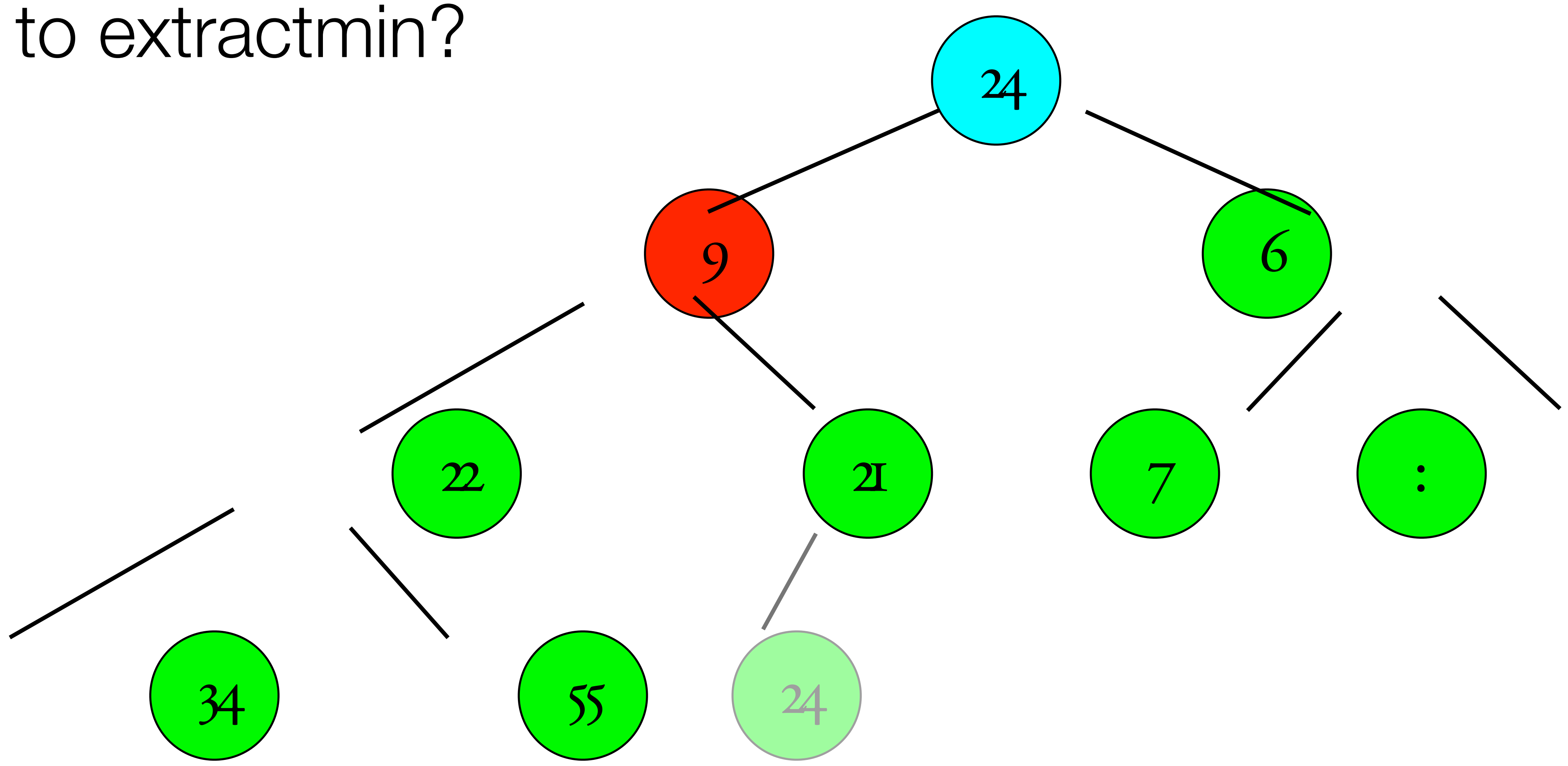
binary heap

full tree, key value <= to key of children
how to extractmin?



binary heap

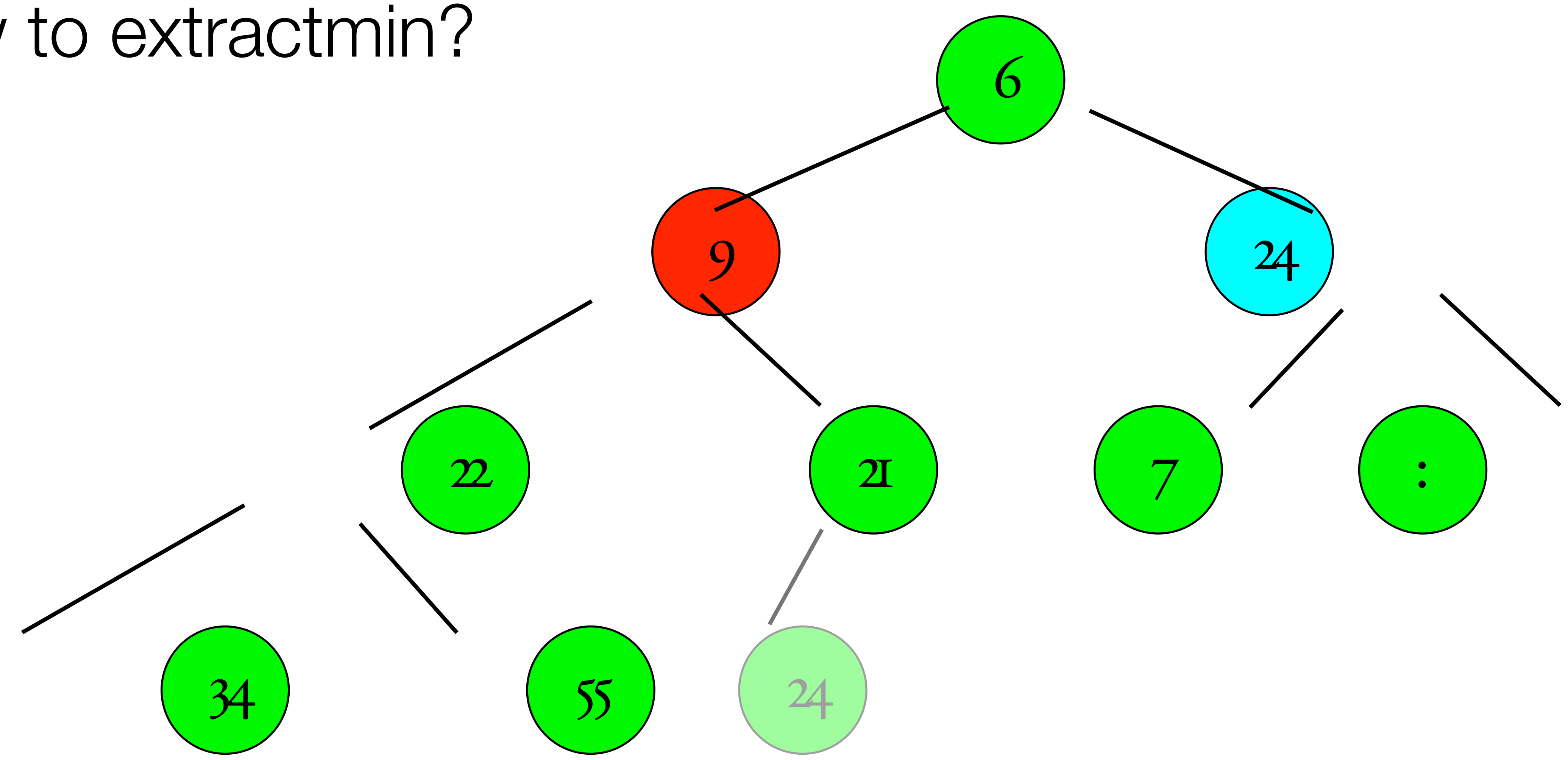
full tree, key value <= to key of children
how to extractmin?



binary heap

full tree, key value <= to key of children

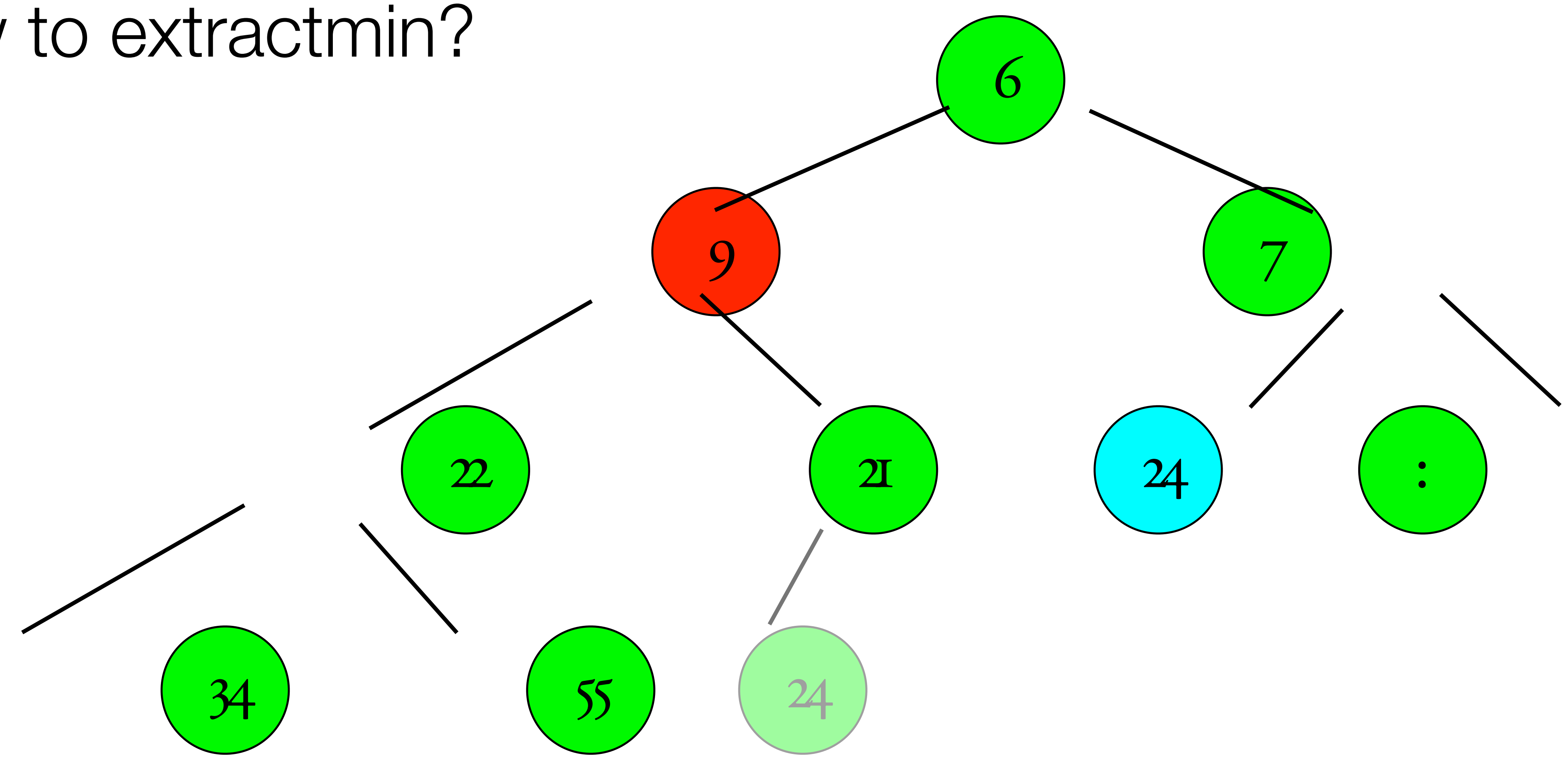
how to extractmin?



binary heap

full tree, key value <= to key of children

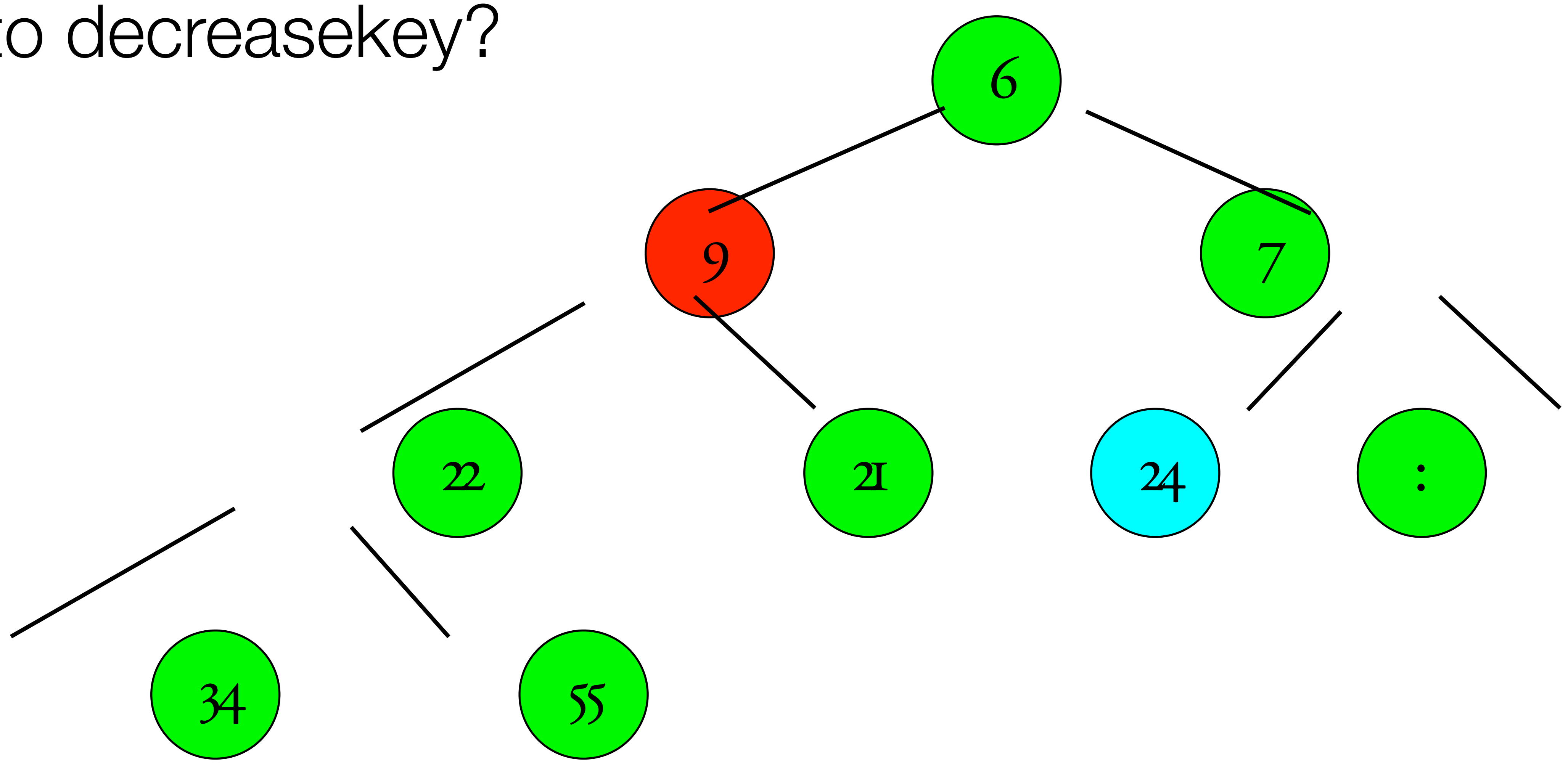
how to extractmin?



binary heap

full tree, key value <= to key of children

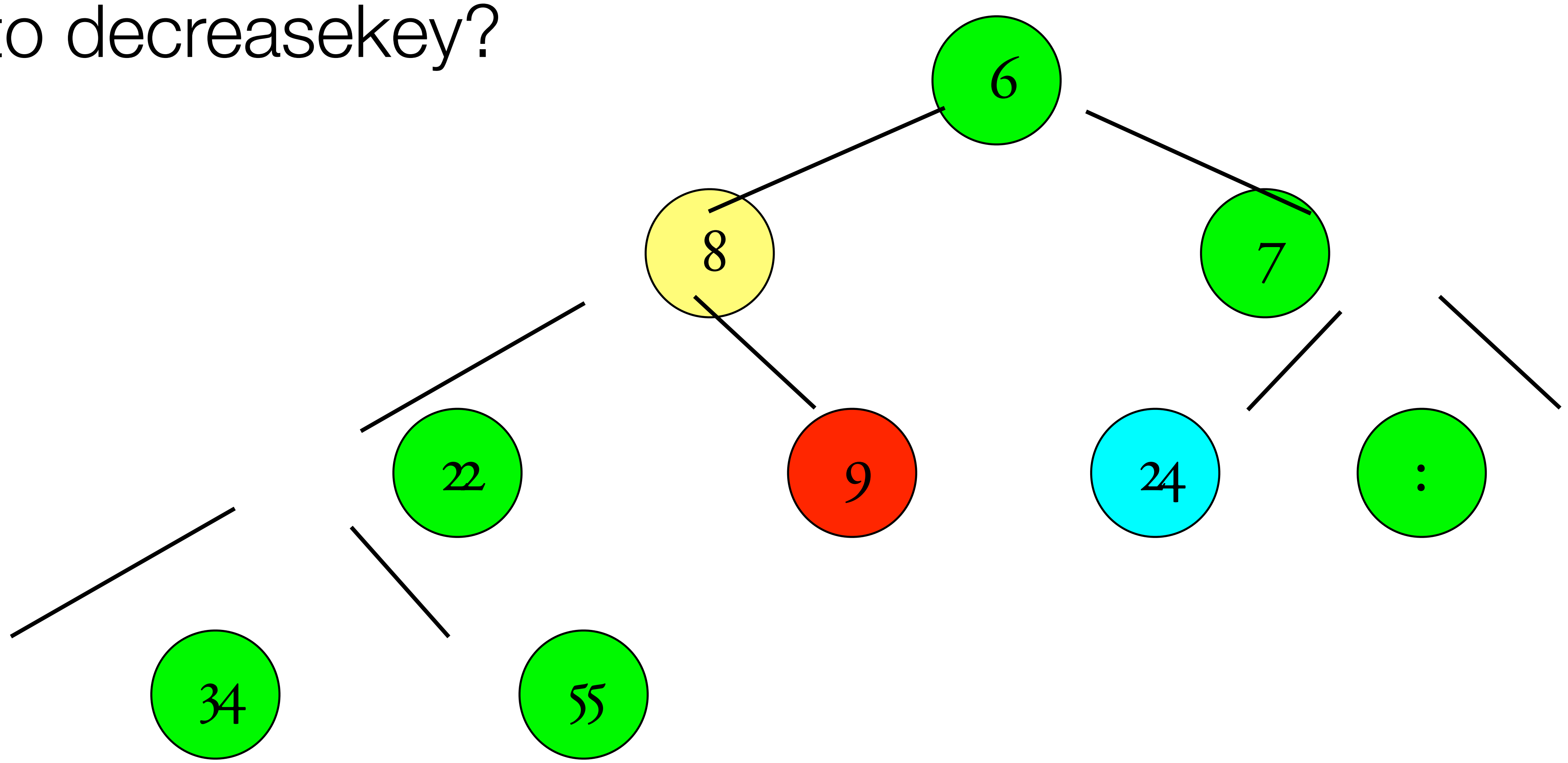
how to decreasekey?



binary heap

full tree, key value <= to key of children

how to decreasekey?



implementation

use a priority queue to keep track of light edges

insert:

makequeue:

extractmin:

decreasekey:

algorithm

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)
```