Outline

1. Minimum Spanning Tree
   - Kruskal's Algorithm
   - Reverse-Kruskal's Algorithm
   - Prim's Algorithm
2. Single Source Shortest Paths
   - Dijkstra's Algorithm
3. Shortest Paths in Graphs with Negative Weights
4. All-Pair Shortest Paths and Floyd-Warshall

Two Methods to Build a MST

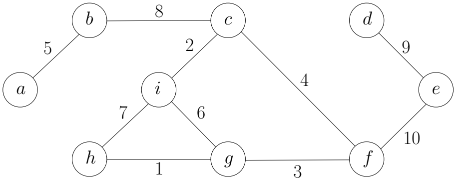
1. Start from $F \leftarrow \emptyset$, and add edges to F one by one until we obtain a spanning tree

Two Methods to Build a MST

1. Start from $F \leftarrow \emptyset$, and add edges to F one by one until we obtain a spanning tree
2. Start from $F \leftarrow E$, and remove edges from F one by one until we obtain a spanning tree

Two Methods to Build a MST

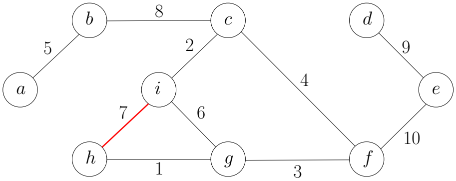
1. Start from $F \leftarrow \emptyset$, and add edges to F one by one until we obtain a spanning tree
2. Start from $F \leftarrow E$, and remove edges from F one by one until we obtain a spanning tree



Q: Which edge can be safely excluded from the MST?

Two Methods to Build a MST

1. Start from $F \leftarrow \emptyset$, and add edges to F one by one until we obtain a spanning tree
2. Start from $F \leftarrow E$, and remove edges from F one by one until we obtain a spanning tree

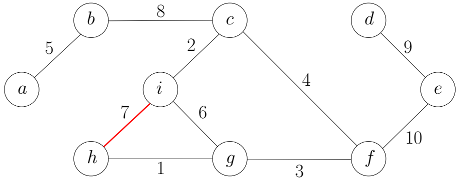


Q: Which edge can be safely excluded from the MST?

A: The heaviest non-bridge edge.

Two Methods to Build a MST

1. Start from $F \leftarrow \emptyset$, and add edges to F one by one until we obtain a spanning tree
2. Start from $F \leftarrow E$, and remove edges from F one by one until we obtain a spanning tree



Q: Which edge can be safely excluded from the MST?

A: The heaviest non-bridge edge.

Def. A bridge is an edge whose removal disconnects the graph.

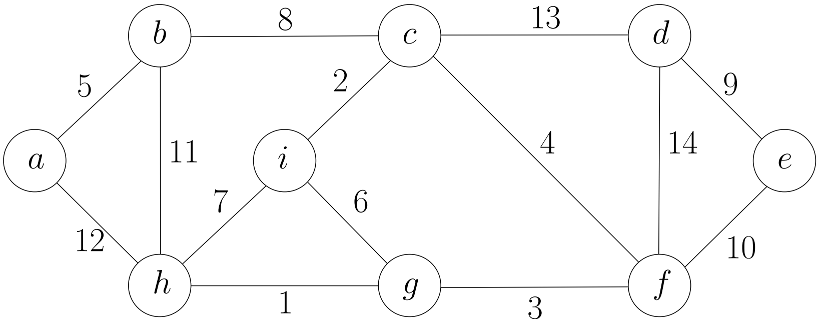
Lemma It is safe to exclude the heaviest non-bridge edge: there is a MST that does not contain the heaviest non-bridge edge.

Reverse Kruskal's Algorithm

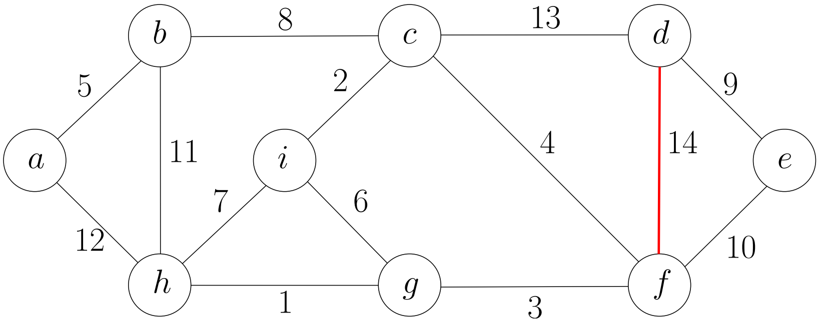
MST-Greedy(G, w)

```
F ← E
sort E in non-increasing order of weights
for every e in this order do
    if (V, F \ {e}) is connected then
        F ← F \ {e}
return (V, F)
```

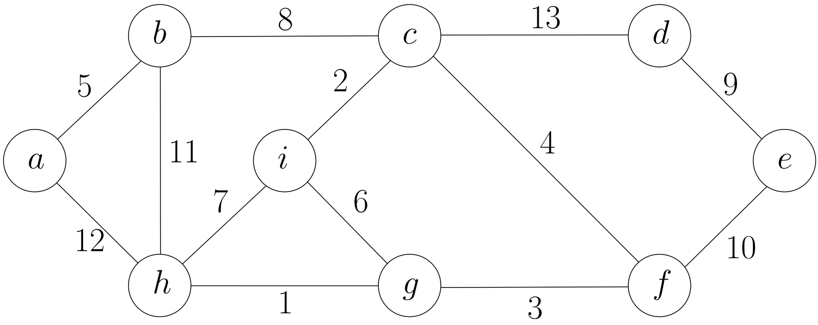
Reverse Kruskal's Algorithm: Example



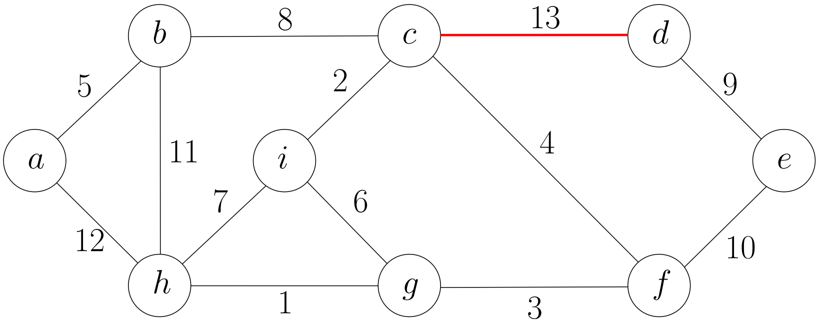
Reverse Kruskal's Algorithm: Example



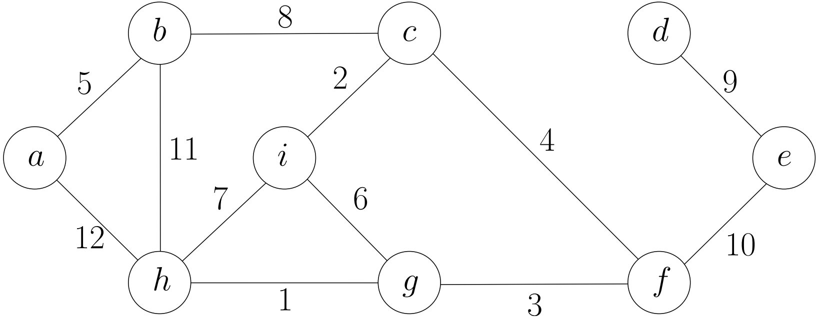
Reverse Kruskal's Algorithm: Example



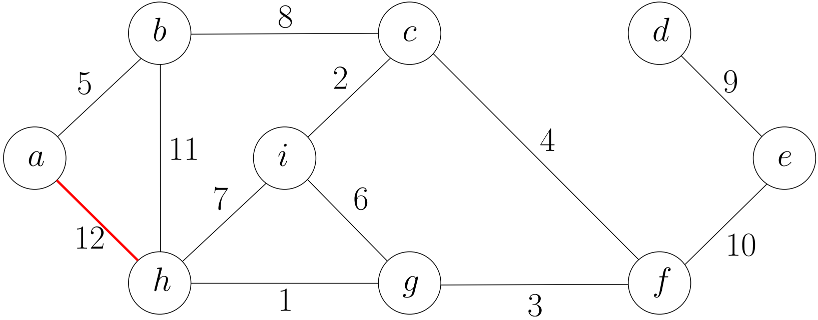
Reverse Kruskal's Algorithm: Example



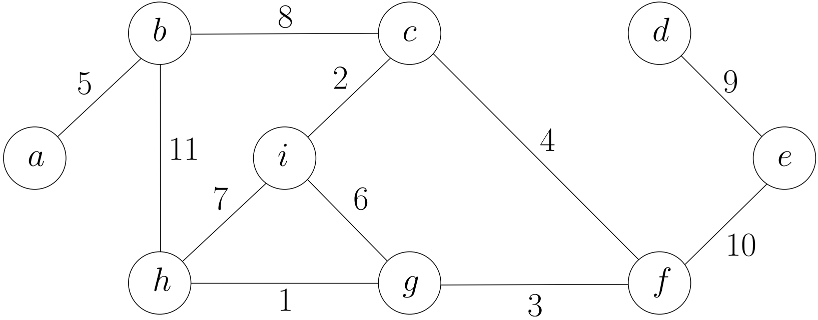
Reverse Kruskal's Algorithm: Example



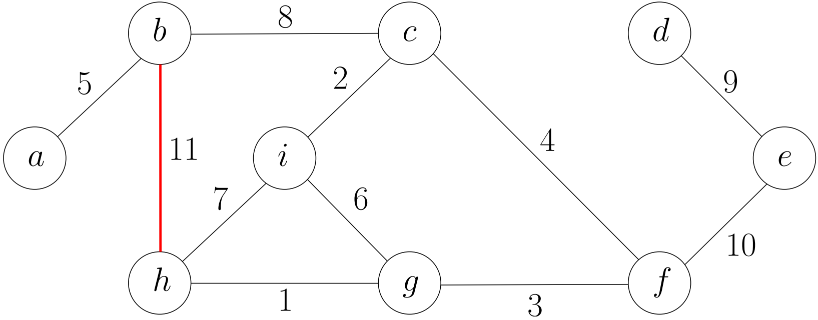
Reverse Kruskal's Algorithm: Example



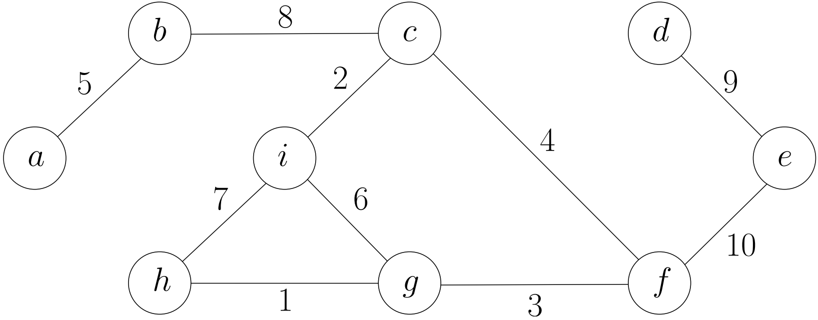
Reverse Kruskal's Algorithm: Example



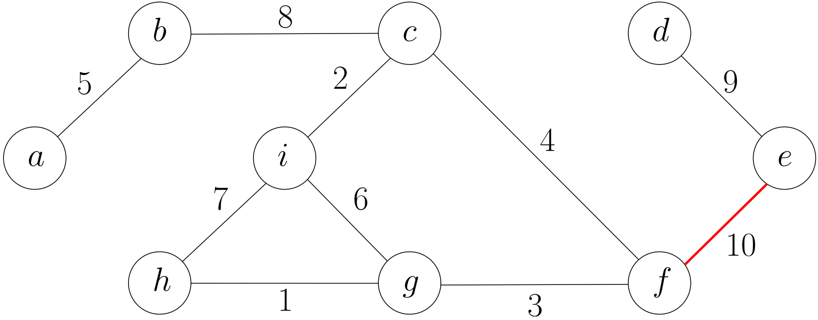
Reverse Kruskal's Algorithm: Example



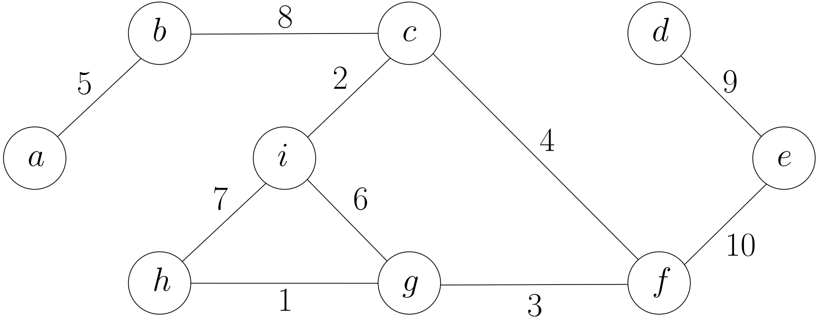
Reverse Kruskal's Algorithm: Example



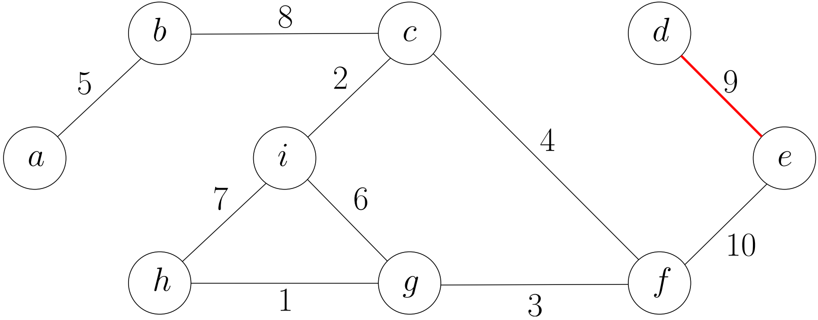
Reverse Kruskal's Algorithm: Example



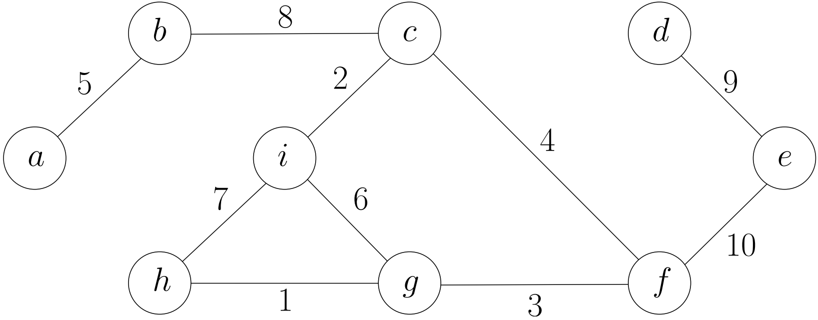
Reverse Kruskal's Algorithm: Example



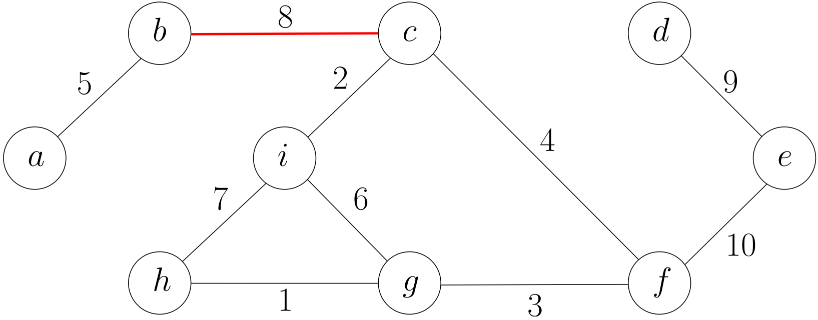
Reverse Kruskal's Algorithm: Example



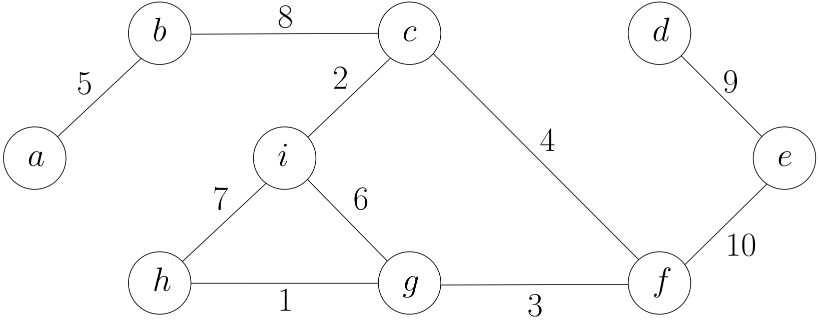
Reverse Kruskal's Algorithm: Example



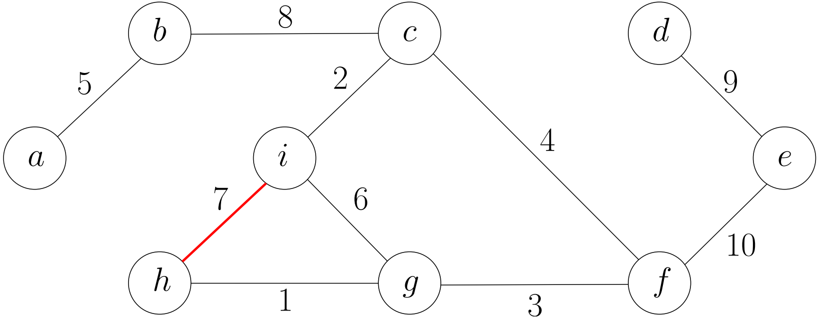
Reverse Kruskal's Algorithm: Example



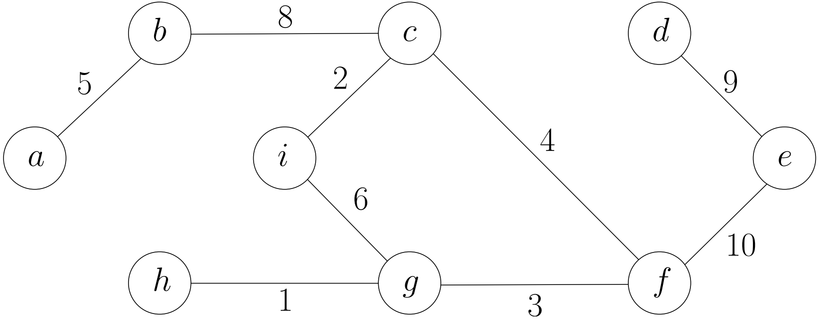
Reverse Kruskal's Algorithm: Example



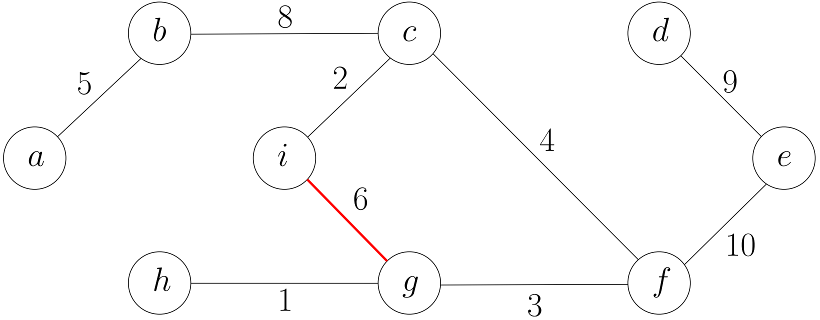
Reverse Kruskal's Algorithm: Example



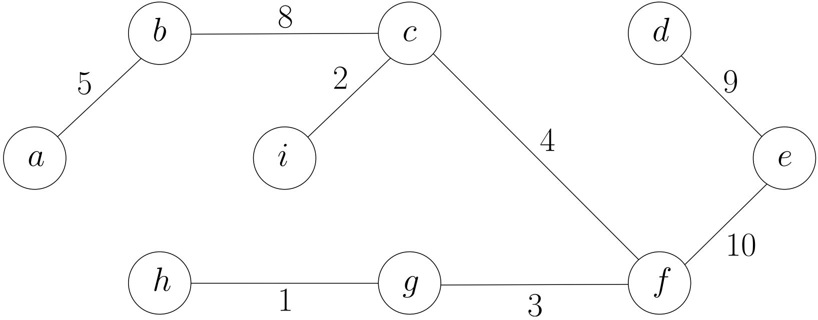
Reverse Kruskal's Algorithm: Example



Reverse Kruskal's Algorithm: Example



Reverse Kruskal's Algorithm: Example

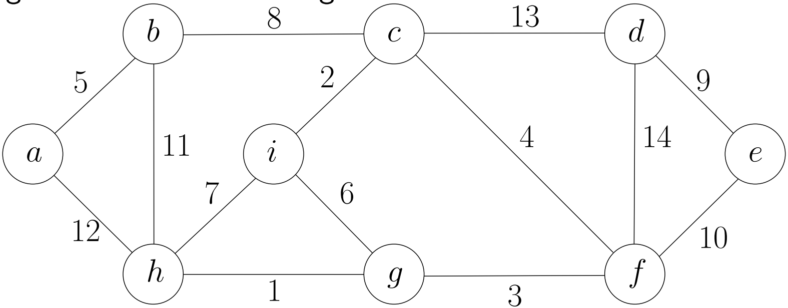


Outline

1. Minimum Spanning Tree
   - Kruskal's Algorithm
   - Reverse-Kruskal's Algorithm
   - Prim's Algorithm
2. Single Source Shortest Paths
   - Dijkstra's Algorithm
3. Shortest Paths in Graphs with Negative Weights
4. All-Pair Shortest Paths and Floyd-Warshall

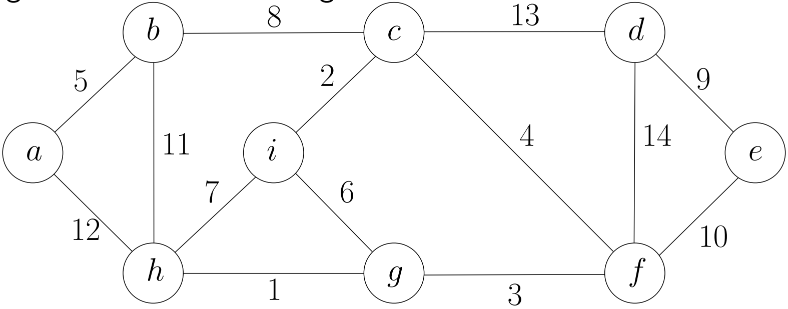
Design Greedy Strategy for MST

- Recall the greedy strategy for Kruskal's algorithm: choose the edge with the smallest weight.



Design Greedy Strategy for MST

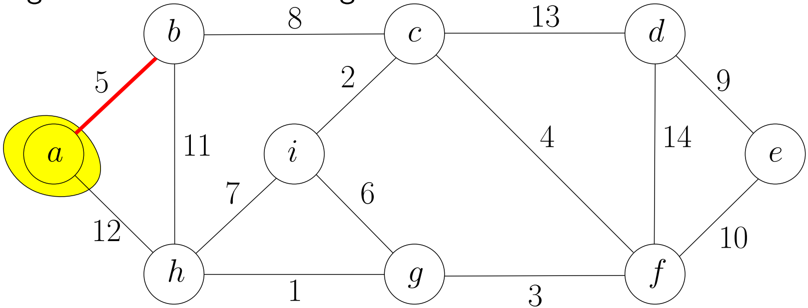
- Recall the greedy strategy for Kruskal's algorithm: choose the edge with the smallest weight.



- Greedy strategy for Prim's algorithm: choose the lightest edge incident to a.

Design Greedy Strategy for MST

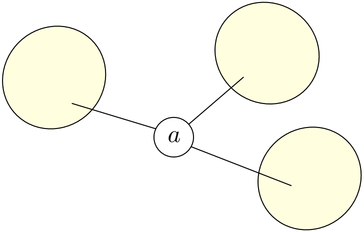
- Recall the greedy strategy for Kruskal's algorithm: choose the edge with the smallest weight.



- Greedy strategy for Prim's algorithm: choose the lightest edge incident to a.

Lemma It is safe to include the lightest edge incident to a.

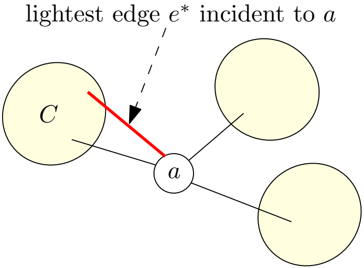
Lemma It is safe to include the lightest edge incident to a.



Proof.

- Let T be a MST
- Consider all components obtained by removing a from T

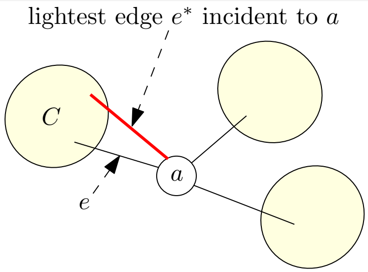
Lemma It is safe to include the lightest edge incident to a.



Proof.

- Let T be a MST
- Consider all components obtained by removing a from T
- Let e^* be the lightest edge incident to a and e^* connects a to component C

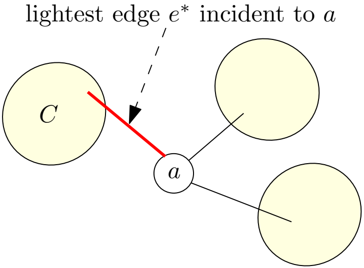
Lemma It is safe to include the lightest edge incident to a.



Proof.

- Let T be a MST
- Consider all components obtained by removing a from T
- Let e^* be the lightest edge incident to a and e^* connects a to component C
- Let e be the edge in T connecting a to C

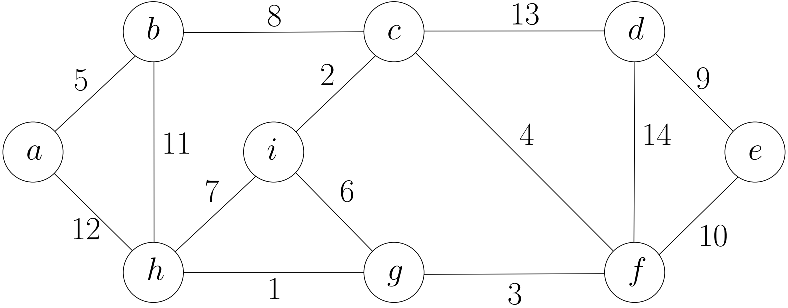
Lemma It is safe to include the lightest edge incident to a.



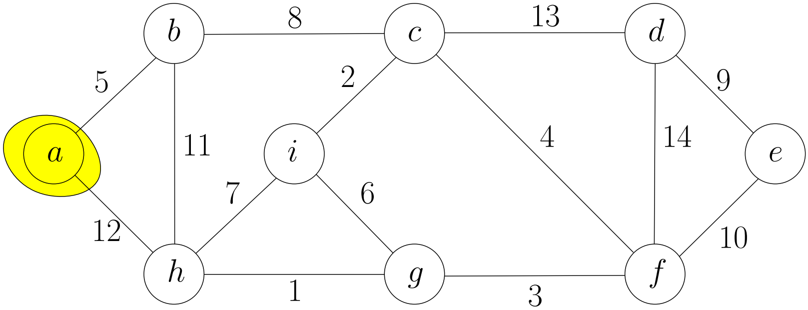
Proof.

- Let T be a MST
- Consider all components obtained by removing a from T
- Let e^* be the lightest edge incident to a and e^* connects a to component C
- Let e be the edge in T connecting a to C
- $T' = T \setminus \{e\} \cup \{e^*\}$ is a spanning tree with $w(T') \leq w(T)$ $\square$

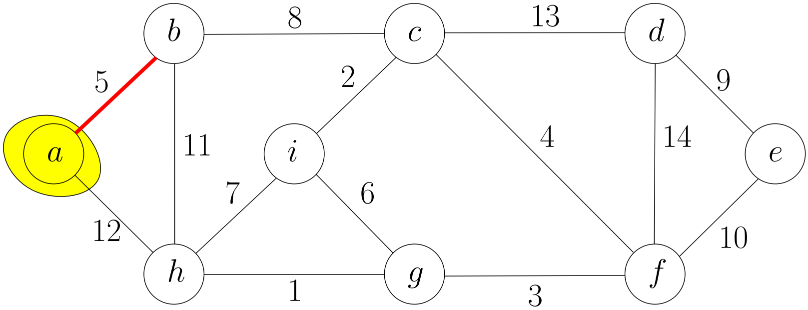
Prim's Algorithm: Example



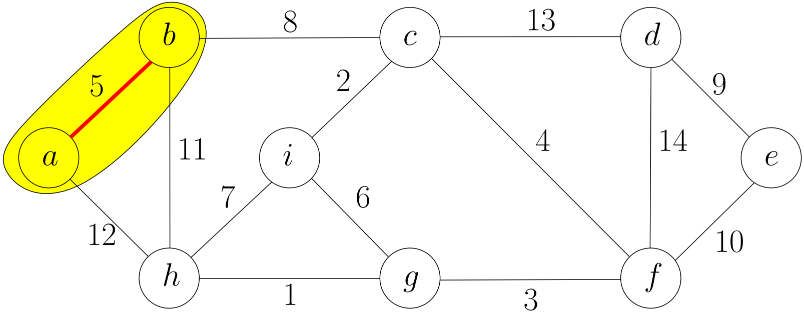
Prim's Algorithm: Example



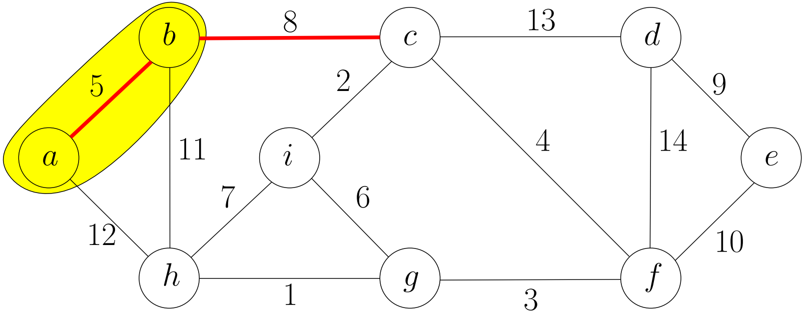
Prim's Algorithm: Example



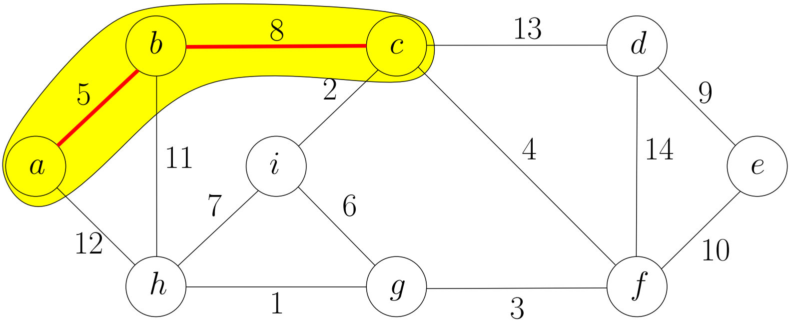
Prim's Algorithm: Example



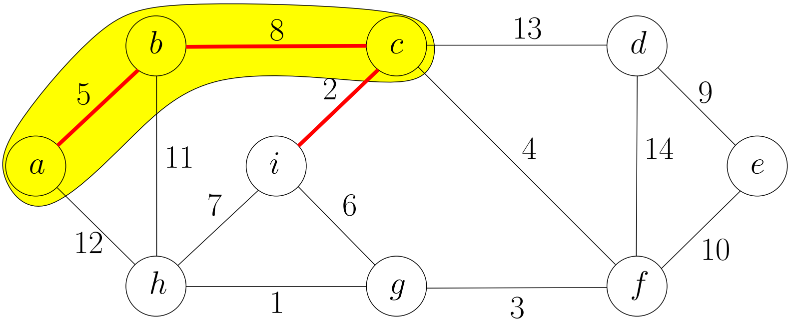
Prim's Algorithm: Example



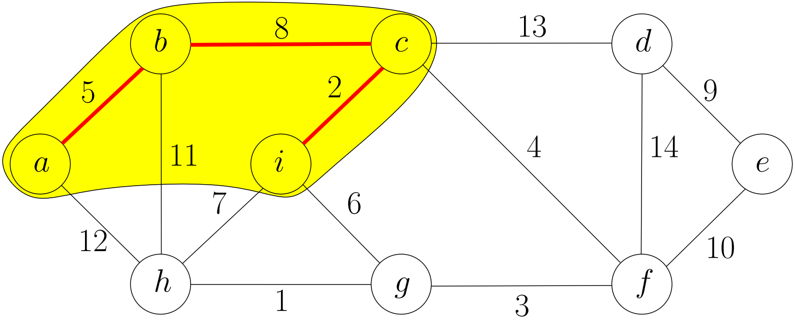
Prim's Algorithm: Example



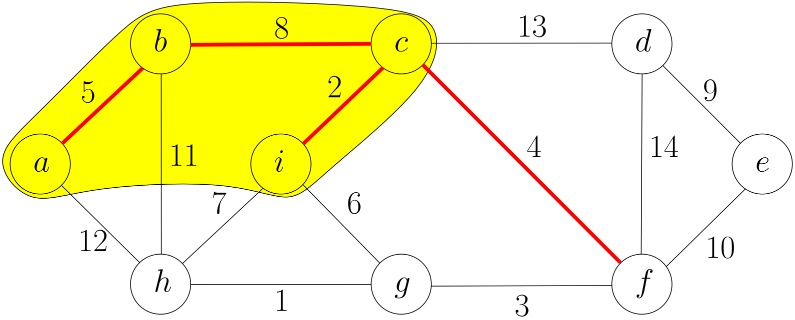
Prim's Algorithm: Example



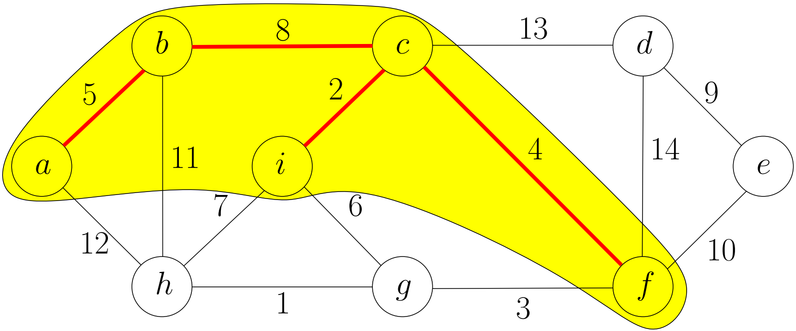
Prim's Algorithm: Example



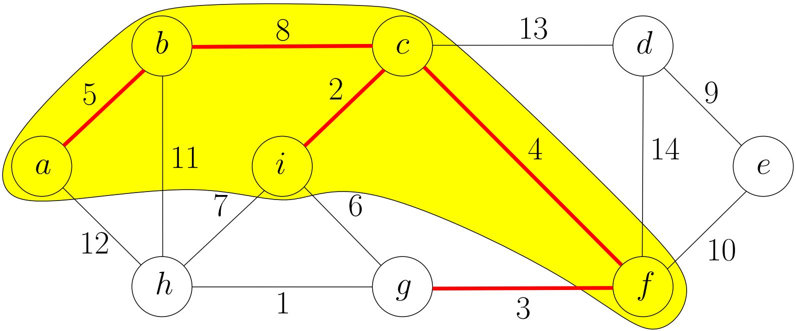
Prim's Algorithm: Example



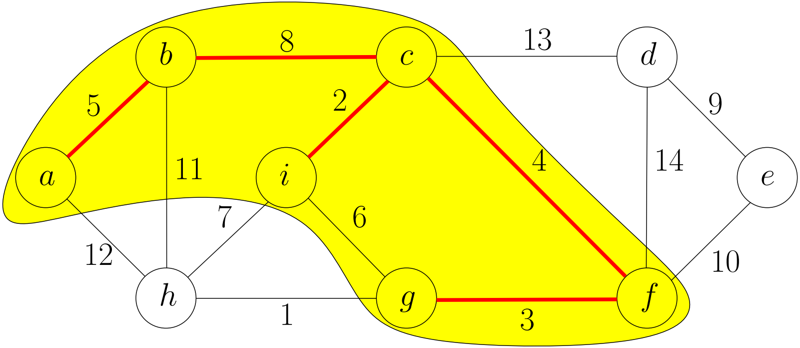
Prim's Algorithm: Example



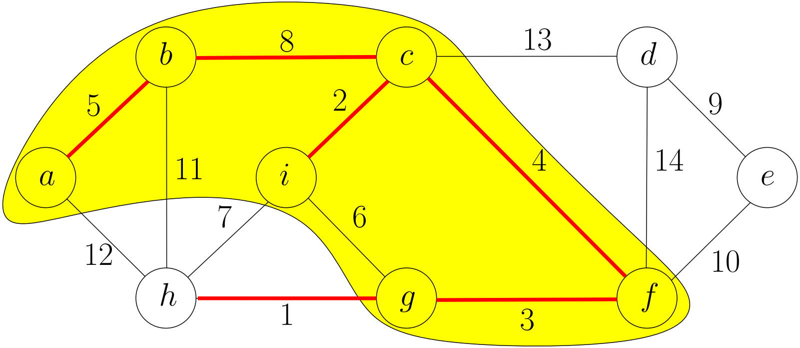
Prim's Algorithm: Example



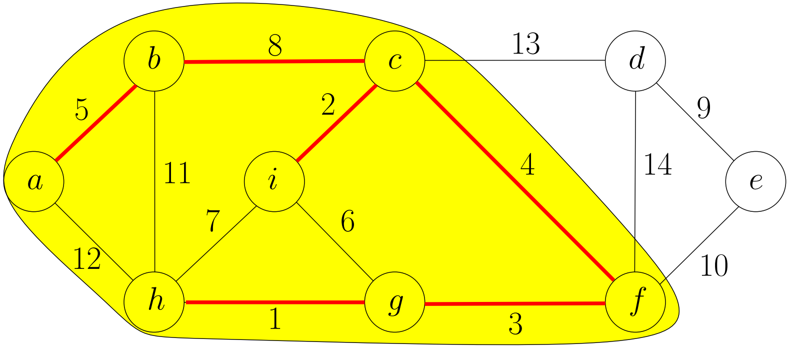
Prim's Algorithm: Example



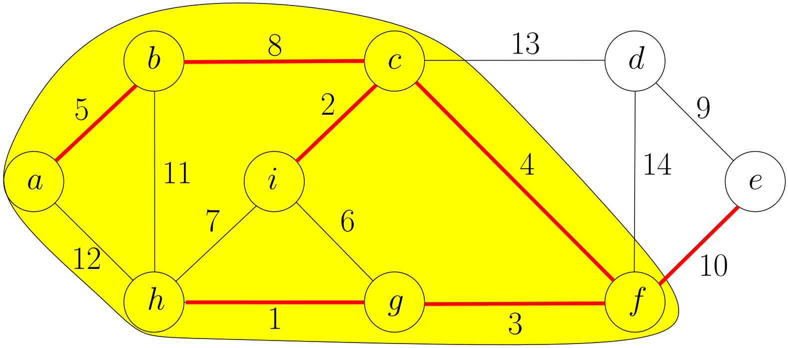
Prim's Algorithm: Example



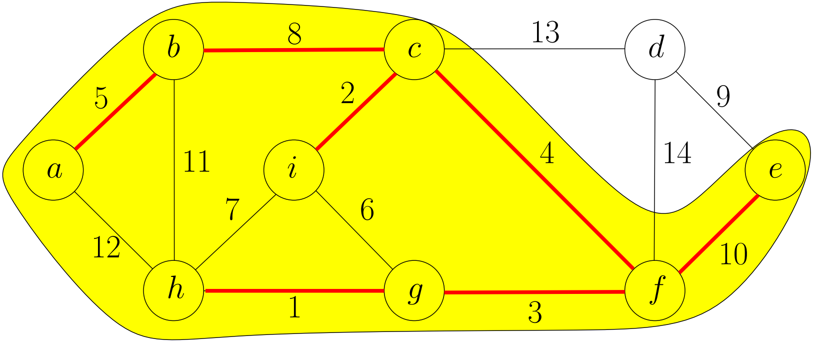
Prim's Algorithm: Example



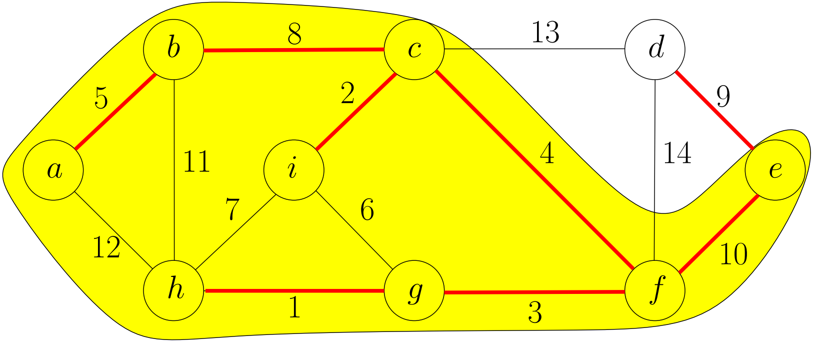
Prim's Algorithm: Example



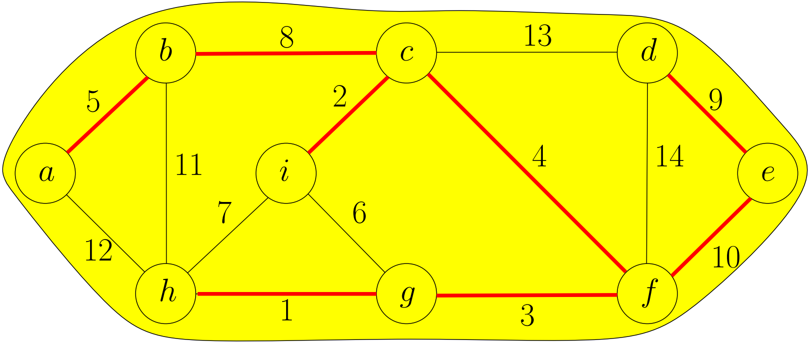
Prim's Algorithm: Example



Prim's Algorithm: Example



Prim's Algorithm: Example



Greedy Algorithm

MST-Greedy1(G, w)

$S \leftarrow \{s\}$, where s is arbitrary vertex in V
$F \leftarrow \emptyset$
while $S \neq V$ **do**
    $(u, v) \leftarrow$ lightest edge between S and $V \setminus S$,
        where $u \in S$ and $v \in V \setminus S$
    $S \leftarrow S \cup \{v\}$
    $F \leftarrow F \cup \{(u, v)\}$
return (V, F)

Greedy Algorithm

MST-Greedy1(G, w)

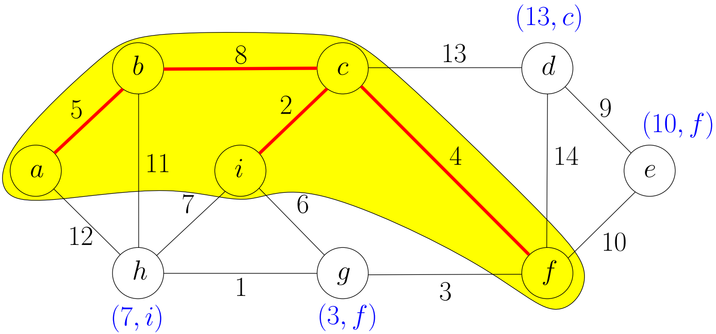
```
S ← {s}, where s is arbitrary vertex in V
F ← ∅
while S ≠ V do
    (u, v) ← lightest edge between S and V \ S,
                         where u ∈ S and v ∈ V \ S
    S ← S ∪ {v}
    F ← F ∪ {(u, v)}
return (V, F)
```

- Running time of naive implementation: $O(nm)$

Prim's Algorithm: Efficient Implementation of Greedy Algorithm

For every $v \in V \setminus S$ maintain

- $d[v] = \min_{u \in S:(u,v) \in E} w(u,v)$:
  the weight of the lightest edge between v and S
- $\pi[v] = \arg\min_{u \in S:(u,v) \in E} w(u,v)$:
  $(\pi[v], v)$ is the lightest edge between v and S



Prim's Algorithm: Efficient Implementation of Greedy Algorithm

For every $v \in V \setminus S$ maintain

- $d[v] = \min_{u \in S:(u,v) \in E} w(u, v)$:
  the weight of the lightest edge between v and S
- $\pi[v] = \arg\min_{u \in S:(u,v) \in E} w(u, v)$:
  $(\pi[v], v)$ is the lightest edge between v and S

In every iteration

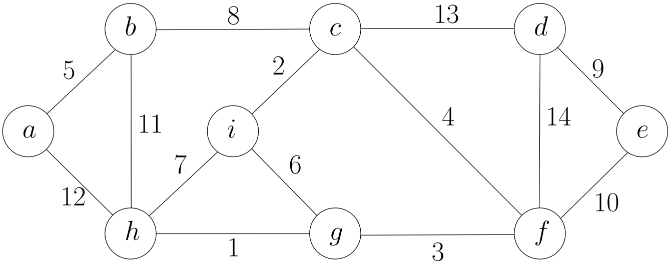
- Pick $u \in V \setminus S$ with the smallest $d[u]$ value
- Add $(\pi[u], u)$ to F
- Add u to S, update d and π values.

Prim's Algorithm

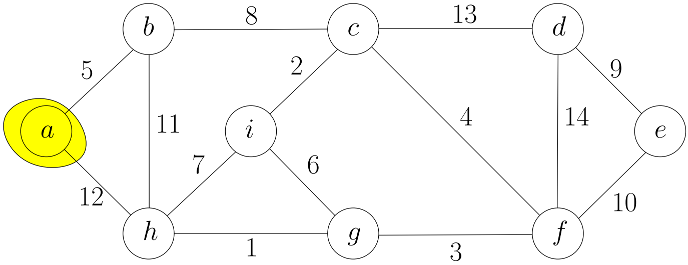
MST-Prim(G, w)

```
s ← arbitrary vertex in G
S ← ∅, d(s) ← 0 and d[v] ← ∞ for every v ∈ V \ {s}
while S ≠ V do
    u ← vertex in V \ S with the minimum d[u]
    S ← S ∪ {u}
    for each v ∈ V \ S such that (u, v) ∈ E do
        if w(u, v) < d[v] then
            d[v] ← w(u, v)
            π[v] ← u
return {(u, π[u]) | u ∈ V \ {s}}
```

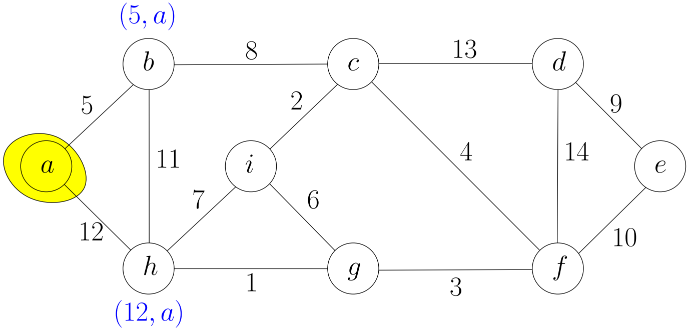
Example



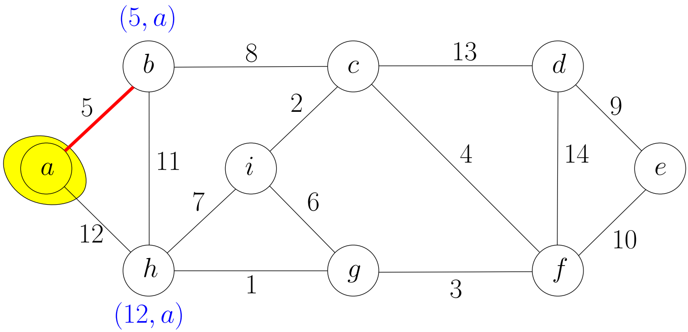
Example



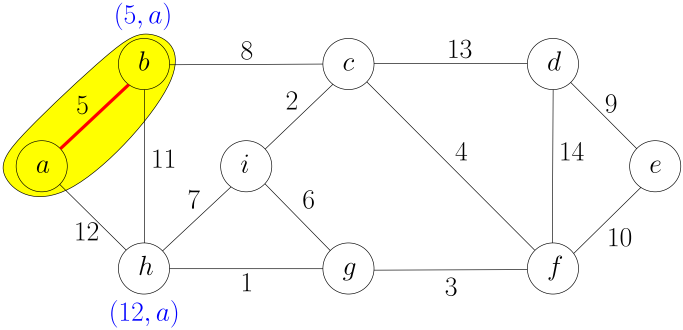
Example



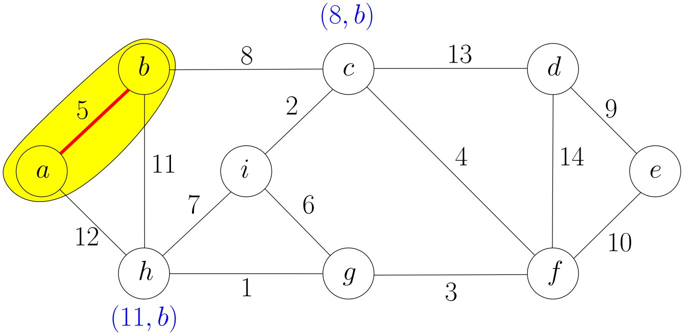
Example



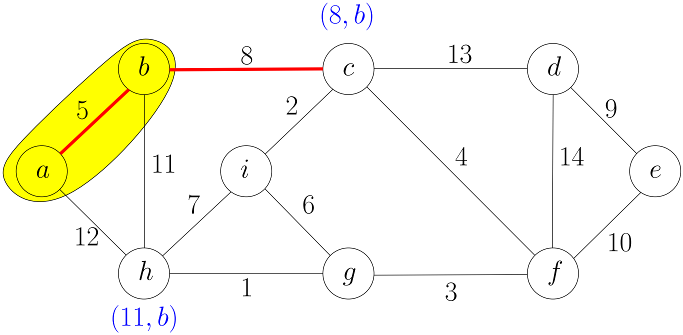
Example



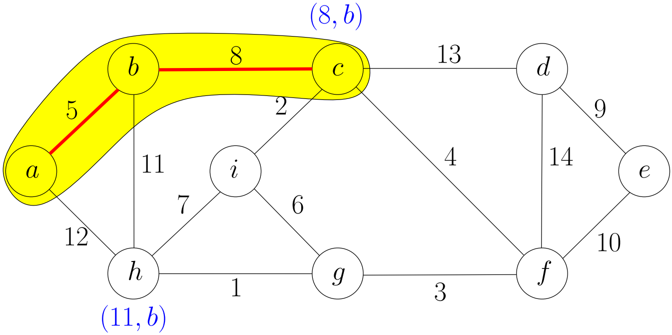
Example



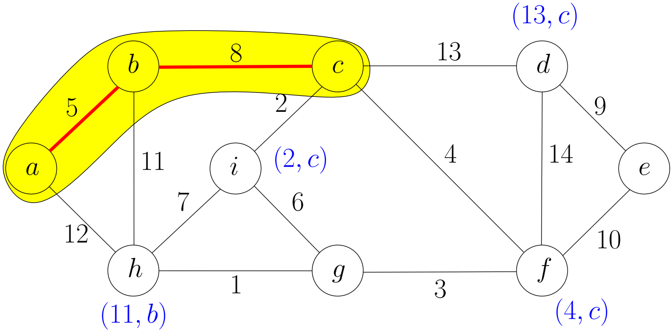
Example



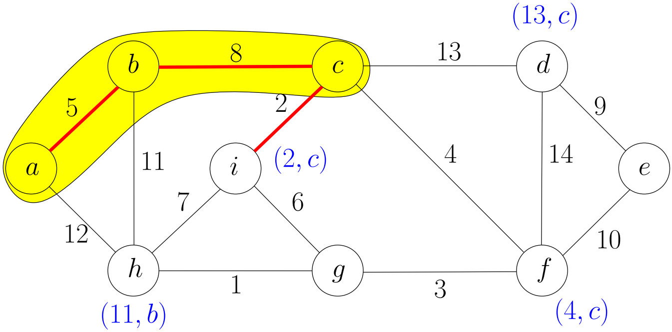
Example



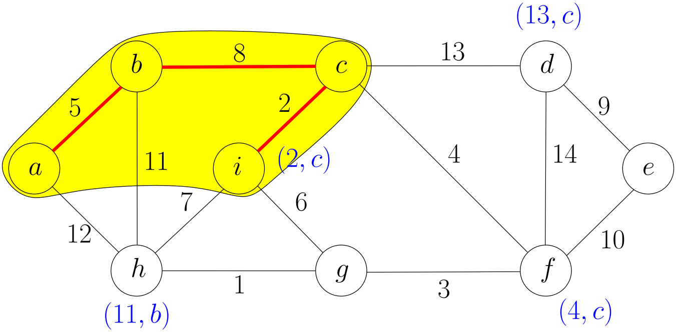
Example



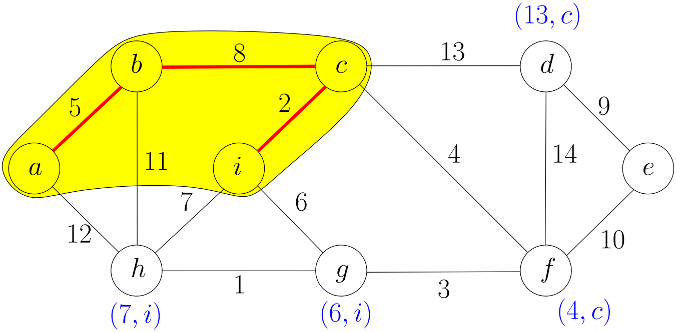
Example



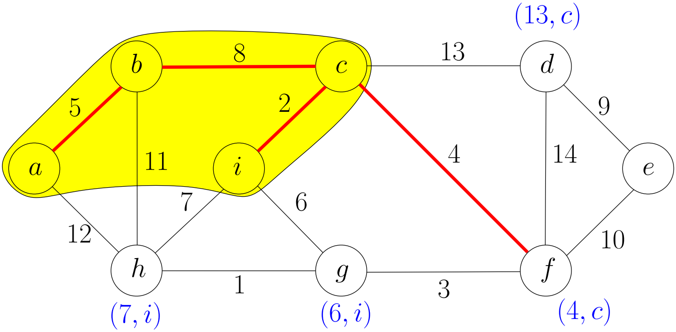
Example



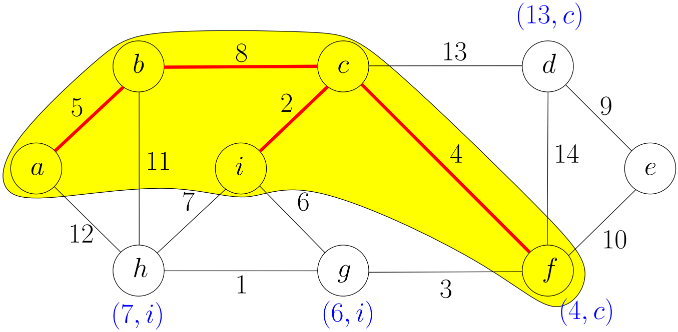
Example



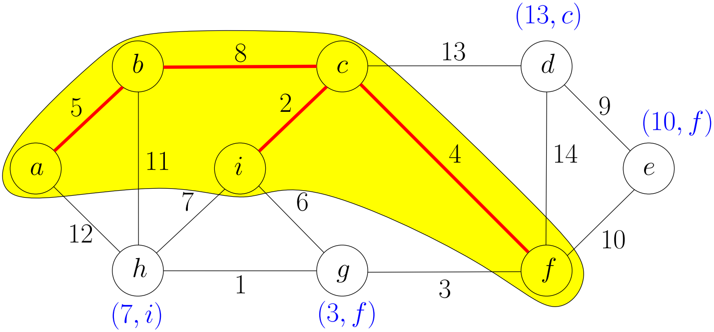
Example



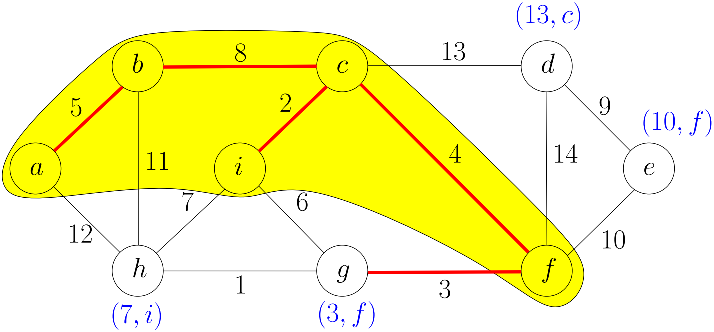
Example



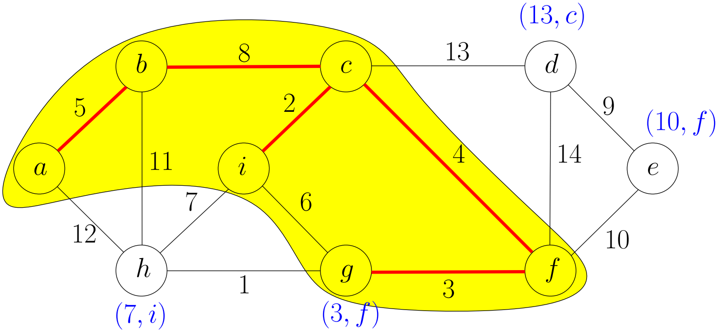
Example



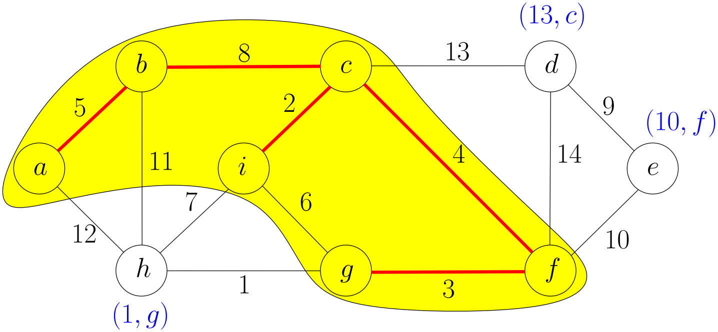
Example



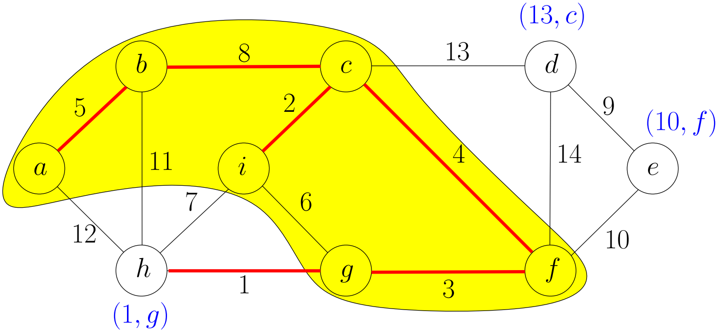
Example



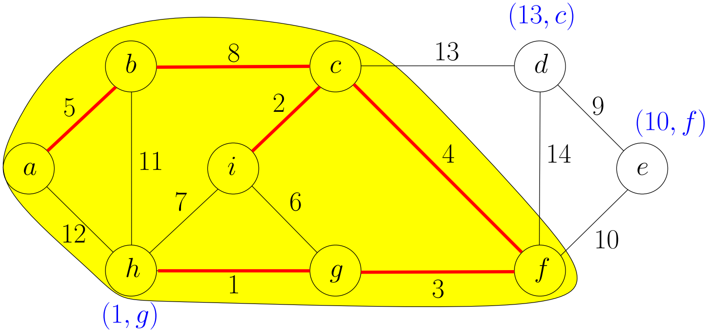
Example



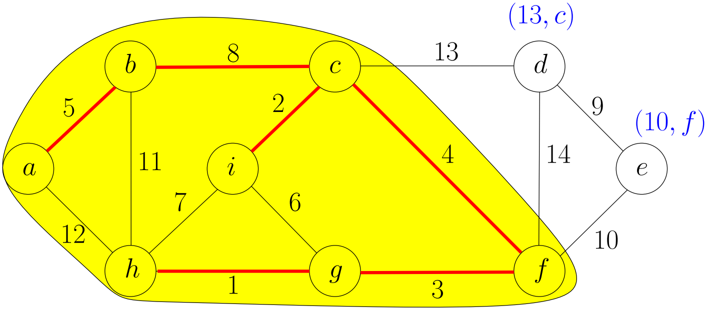
Example



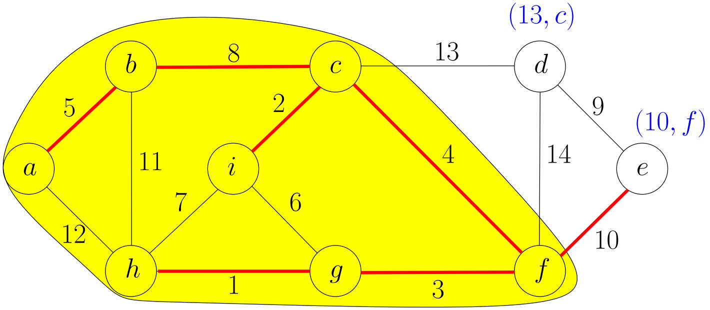
Example



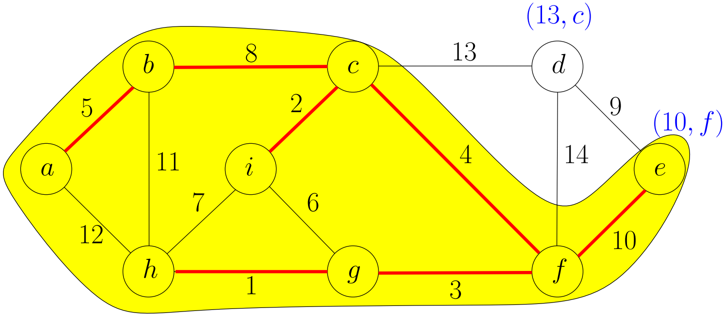
Example



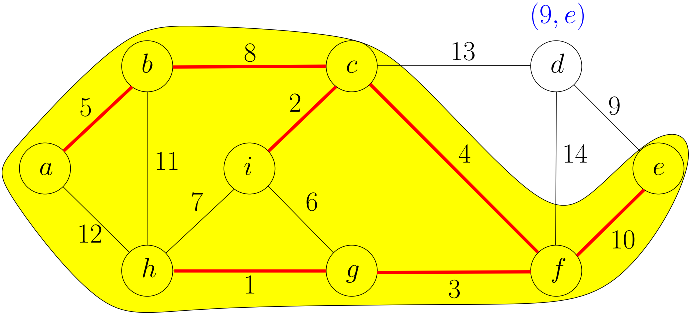
Example



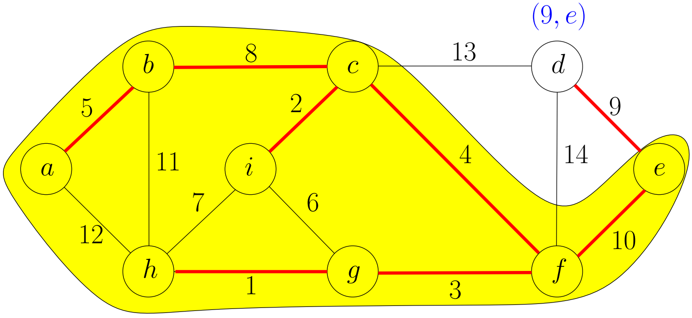
Example



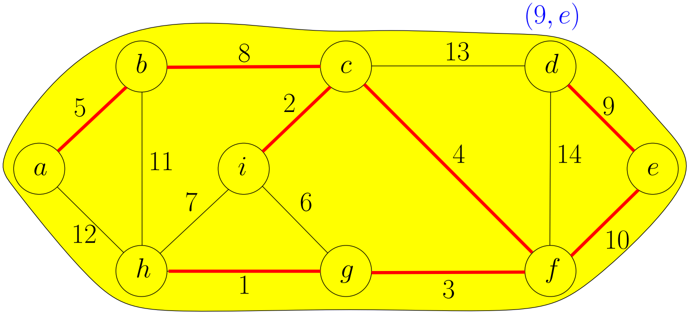
Example



Example



Example



Example