Lecture 5
Minimum Spanning Tree

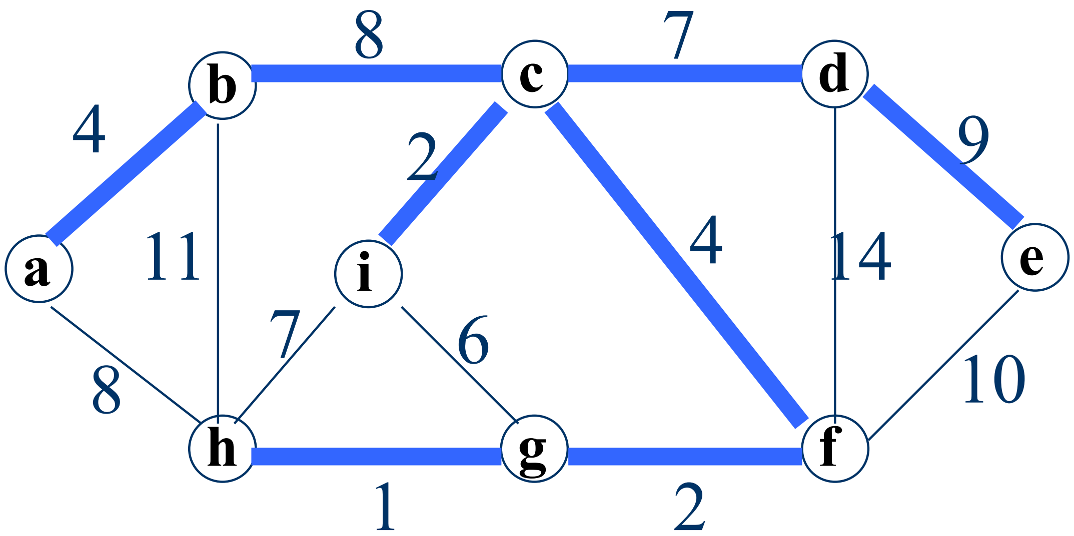
1. The Minimum Spanning Tree Problem
2. A Generic Algorithm
3. Kruskal's Algorithm
4. Prim's Algorithm

Tree and Its Properties

- Tree is an acyclic, connected graph.
- A tree of $|V|$ vertices has $|V|$-1 edges.
- There exists a unique path between any two vertices of a tree.
- Adding any edge to a tree creates a unique cycle. Breaking any edge on this cycle restores a tree.
- Deleting any edge on a tree increases the number of connected components by 1.

An Application

- In the design of networking
- Given n computers, we want to connect them so that each pair of them can communicate with each other.
- Price of cable: \$1/foot
- We want the cheapest possible network.



The Definition of Minimum Spanning Tree

Spanning tree: Given a connected undirected graph $G = (V, E)$, a spanning tree of G is *an acyclic subgraph* that connects all vertices of G.

Minimum spanning tree: Given a connected undirected graph $G = (V, E)$ and an assignment of weights $w(e)$ to the edges of G, a minimum spanning tree T of G is a spanning tree with minimum total edge weight $w(T) = \sum_{e \in T} w(e)$.

Note:

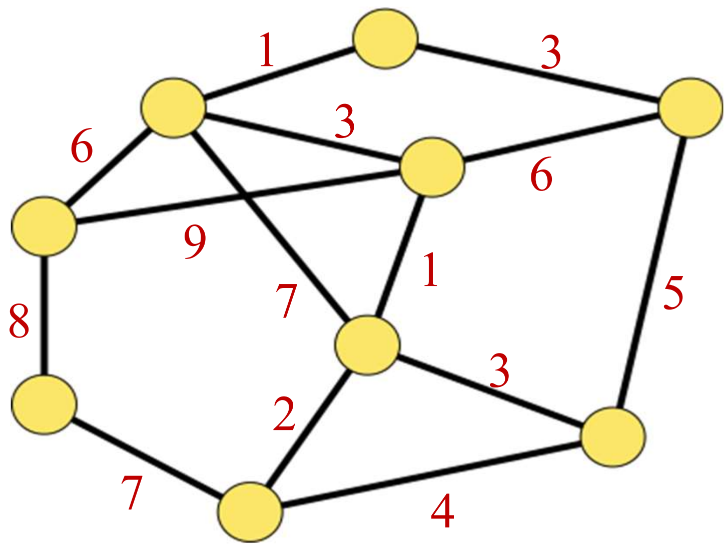
(1). A connected undirected graph may have many different spanning trees

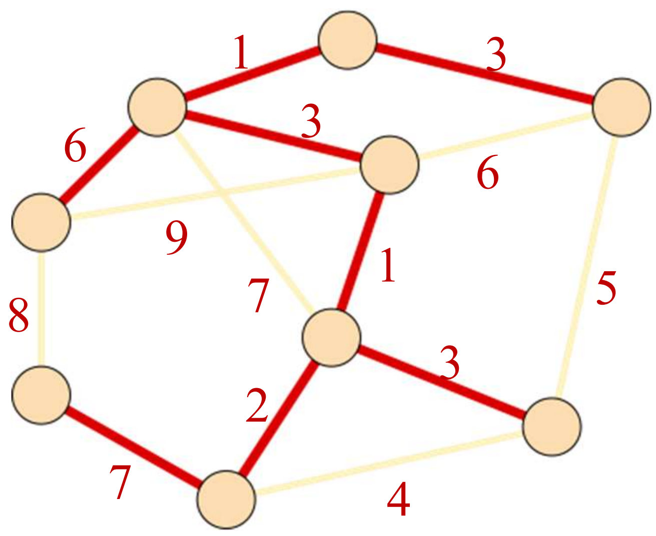
(2). The minimum spanning tree may not be unique

Minimum Spanning Tree Problem

The Problem:

- Input: A connected, weighted, undirected graph G = (V, E; W).
- Output: A minimum spanning tree T for G.





Idea of the Generic Algorithm

- It grows the minimum spanning tree *one edge at a time*.
  - What's the foundation?: An edge set A that is a subset of some minimum spanning tree T;
  - Which edge?: An edge (u, v) satisfying that $A \cup \{(u, v)\}$ is also a subset of some minimum spanning tree T'.
- In other words, it maintains the following loop invariant:
  - Prior to each iteration, A is a subset of some minimum spanning tree.

- The edge (u, v) is called a safe edge **for A** if $A \cup \{(u, v)\}$ is also a subset of a minimum spanning tree, that is,
  - (u, v) can be safely added to A without violating the above invariant

A Generic Algorithm

GENERIC-MST(G, w)

$A \leftarrow \emptyset$

while A does not form a spanning tree

  do find an edge (u, v) that is safe for A

    $A \leftarrow A \cup \{(u, v)\}$

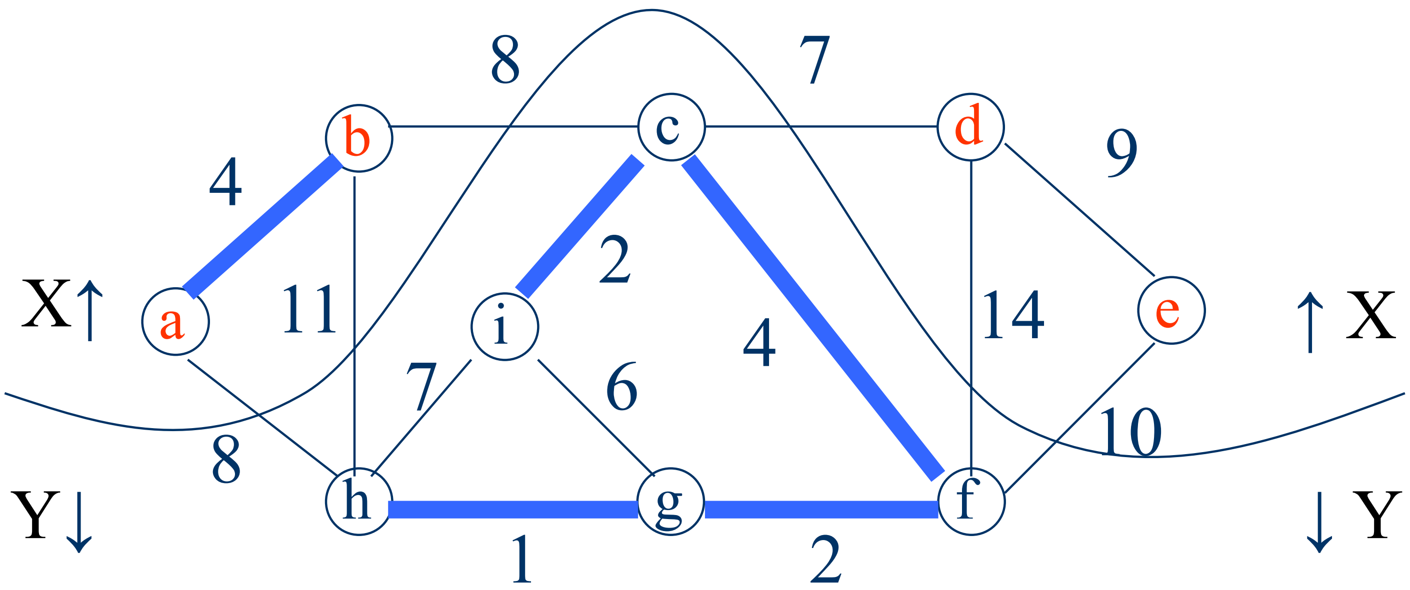
return A

- Kruskal's and Prim's algorithms are implementations of the generic algorithm on how to maintain A and find the safe edge (u, v) for A.

How to recognize Safe Edges for *A*

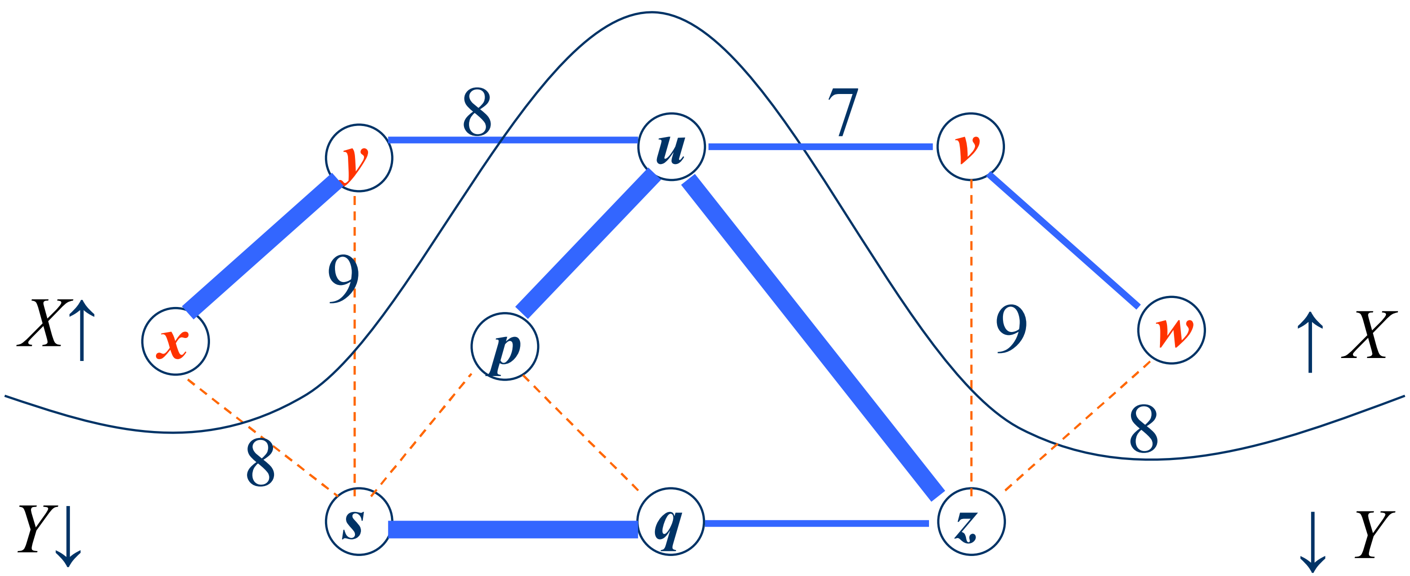
Related Notions

1. A ***cut*** (X, Y) of a graph $G = (V, E)$ is a partition of the vertex set V into two sets X and $Y = V - X$.
2. An edge $(u, v) \in E$ is said to ***cross*** the cut (X, Y) if $u \in X$ and $v \in Y$.
3. A cut (X, Y) ***respects*** a set A of edges if no edge in A crosses the cut.
4. An edge is a ***light edge*** crossing a cut if its weight is the minimum of any edge crossing the cut.

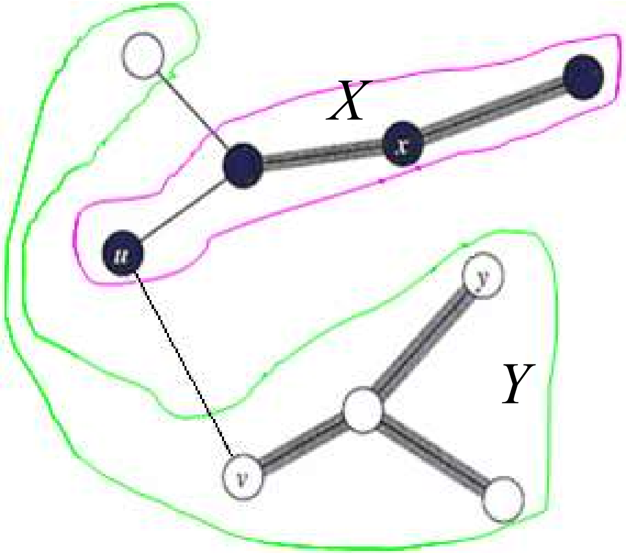


How to recognize Safe Edges for A

Theorem 23.1 Let $G = (V, E)$ be a connected, undirected graph with real-valued weight function w defined on E. Let A be a subset of E that is included in some minimum spanning tree for G, let (X, Y) be any cut of G that respects A, and let (u, v) be a light edge crossing (X, Y). Then, edge (u, v) is safe for A.



Proof of Theorem 23.1



Proof: Let T be a MST including A. It suffices to consider two cases:

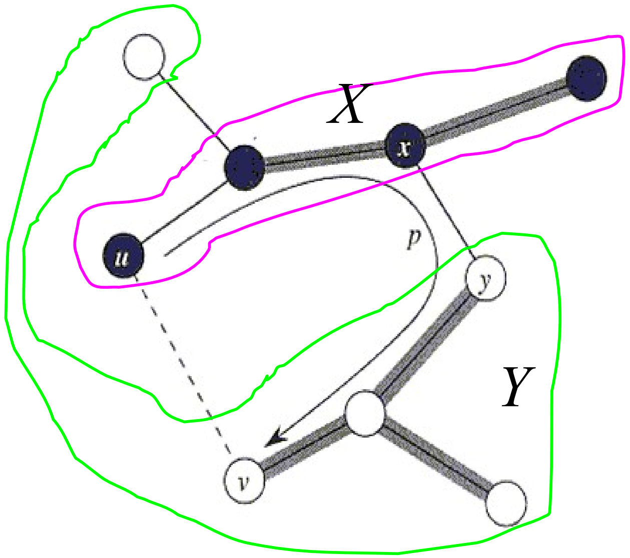
Case 1: T contains the light edge (u, v). We are done;

Case 2: T does not contain the light edge (u, v). In order to show that (u, v) is a safe edge for A, we shall construct another MST T' that includes $A \cup \{(u, v)\}$.

The edge (u, v) forms a cycle with the edges on the path p from u to v in T. Since u and v are on opposite sides of the cut (X,Y), there is at least one edge in T on the path p that also crosses the cut. Let (x, y) be any such edge. Since (x, y) is on the unique path from u to v in T, removing (x, y) breaks T into two components. Adding (u, v) reconnects them to form a new *spanning tree* $T' = T - \{(x, y)\} \cup \{(u, v)\}$.

It is easy to show that T' is a *minimum spanning tree*.

It also is easy to show that T' includes $A \cup \{(u, v)\}$.



A better understanding

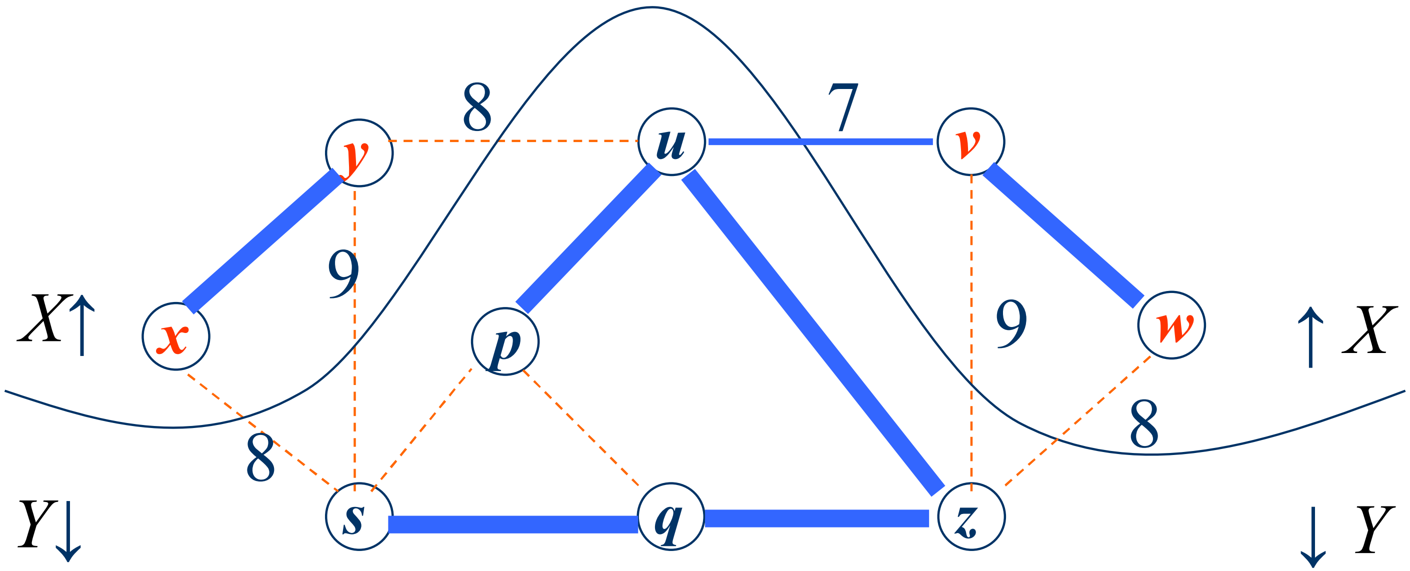
```
GENERIC-MST(G, w)
A ← Ø
while A does not form a spanning tree
      do find an edge (u, v) that is safe for A
            A ← A ∪ {(u, v)}
return A
```

- *A* is always acyclic. Why?
- So, $G_A=(V,A)$ is a forest, and each connected component of which is a tree.
- Any safe edge (u, v) for *A* connects distinct connected component of G_A. Why?
- The while-loop is executed $|V|$-1 times. Why?

A Corollary

Corollary 23.2: Let $G = (V, E)$ be a connected, undirected graph with real-valued weight function w defined on E. Let A be a subset of E that is included in some minimum spanning tree for G, and let $C = (V_C, E_C)$ be a connected component in the forest $G_A = (V, A)$. If (u, v) is a light edge connecting C to some other component in G_A, then (u, v) is safe for A.

Proof: The cut $(V_C, V-V_C)$ respects A, and (u, v) is a light edge for this cut. Therefore, (u, v) is safe for A.



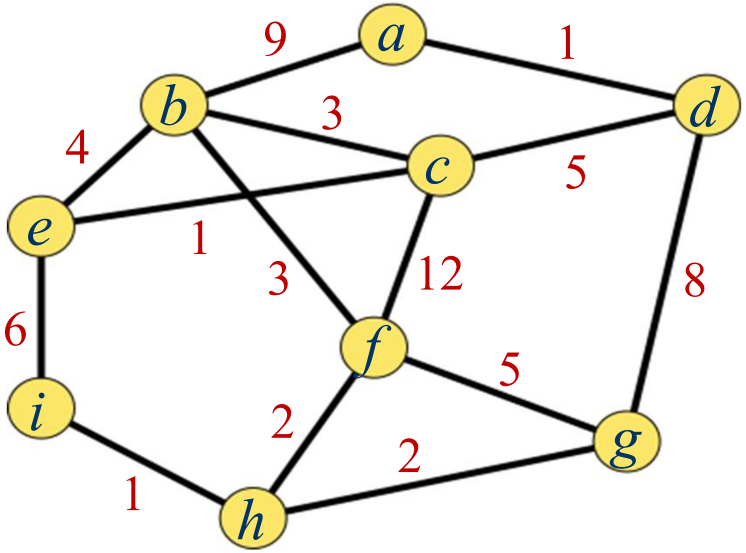
Note: Kruskal and Prim's algorithms are based on the corollary.

Idea of the Kruskal

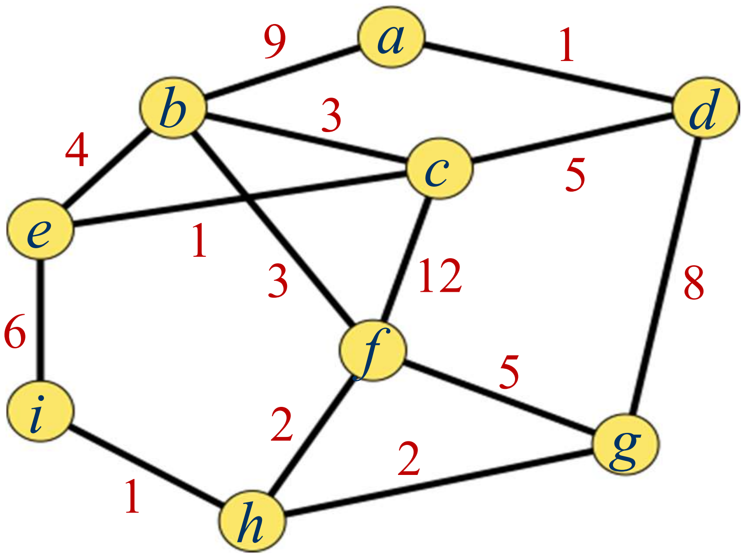
1. Initialize the forest, each vertex as a tree, $A \leftarrow \Phi$.
2. Find the least weight edge (u, v) that connects any two trees in the forest.
3. Add (u, v) to A and union the two tree into one tree, number of trees in the forest decreases 1.
4. Repeat step 2 and 3 until A forms a spanning tree.

Kruskal's Algorithm

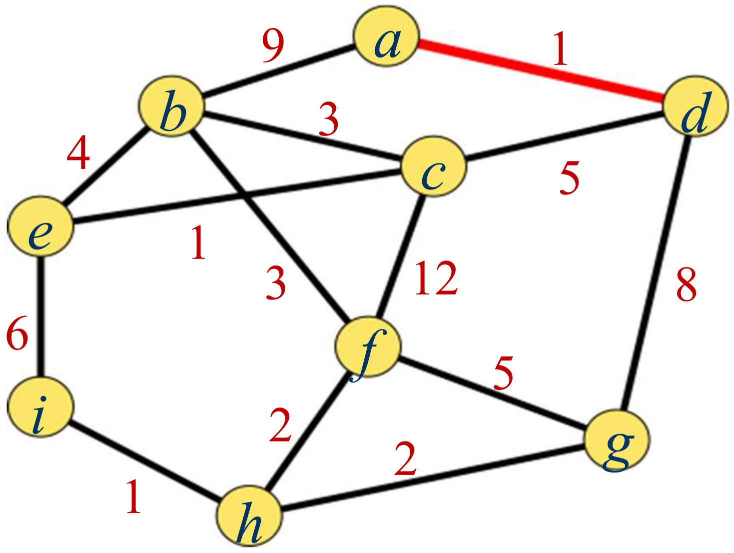
```
MST-KRUSKAL(G, w)
A ← Ø
for each vertex v ∈ V[G]
      do MAKE-SET(v)
sort the edges of E into nondecreasing order by weight w
for each edge (u, v) ∈ E, taken in nondecreasing order by weight
      do if FIND-SET(u) ≠ FIND-SET(v)
            then A ← A ∪ {(u, v)}
                  UNION(u, v)
return A
```



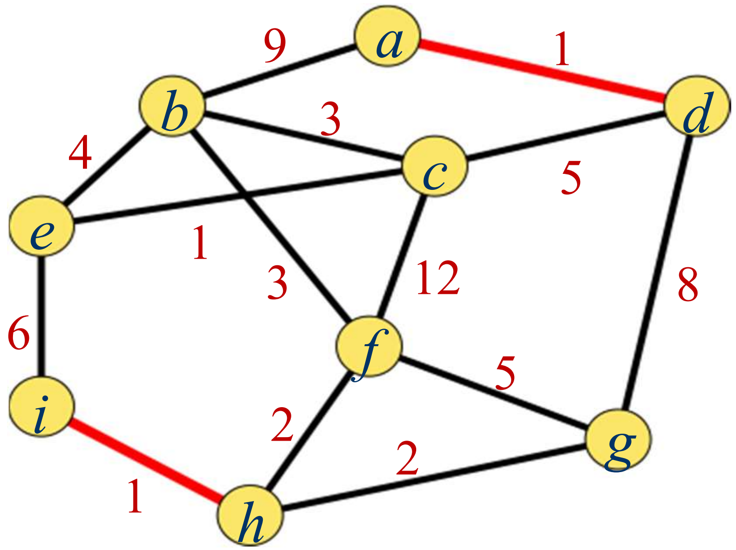
(a, d):1 (h, i):1 (c, e):1 (f, h):2 (g, h):2
(b, c):3 (b, f):3 (b, e):4 (c, d):5 (f, g):5
(e, i):6 (d, g):8 (a, b):9 (c, f):12



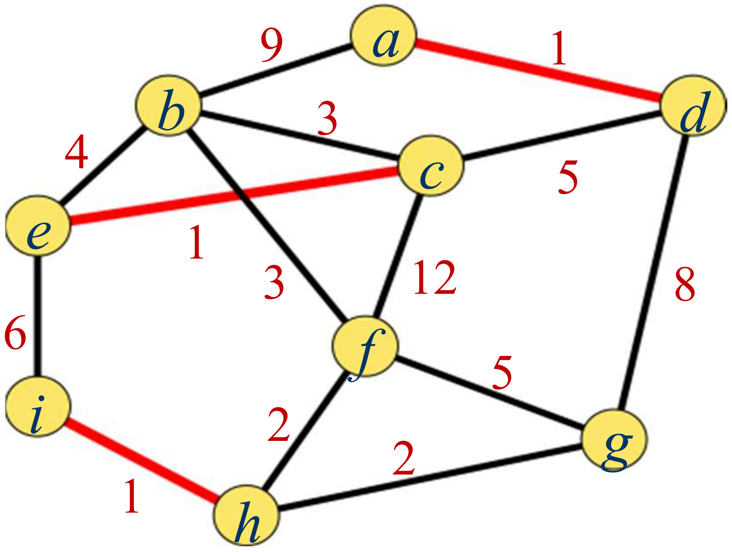
(a, d):1 (h, i):1 (c, e):1 (f, h):2 (g, h):2
(b, c):3 (b, f):3 (b, e):4 (c, d):5 (f, g):5
(e, i):6 (d, g):8 (a, b):9 (c, f):12



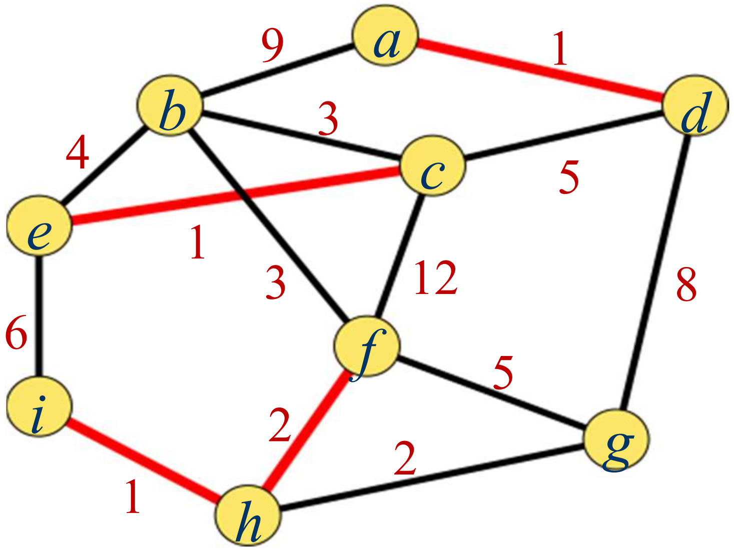
(a, d):1 **(h, i):1** (c, e):1 (f, h):2 (g, h):2
(b, c):3 (b, f):3 (b, e):4 (c, d):5 (f, g):5
(e, i):6 (d, g):8 (a, b):9 (c, f):12



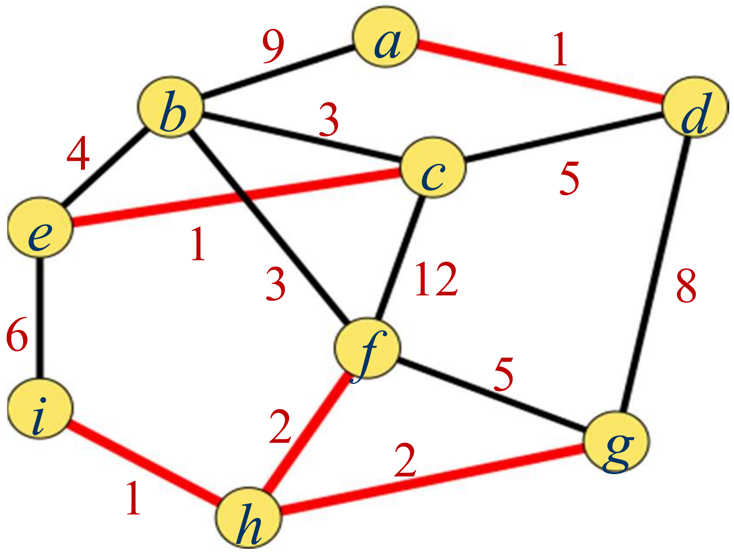
(a, d):1 **(h, i):1** **(c, e):1** (f, h):2 (g, h):2
(b, c):3 (b, f):3 (b, e):4 (c, d):5 (f, g):5
(e, i):6 (d, g):8 (a, b):9 (c, f):12



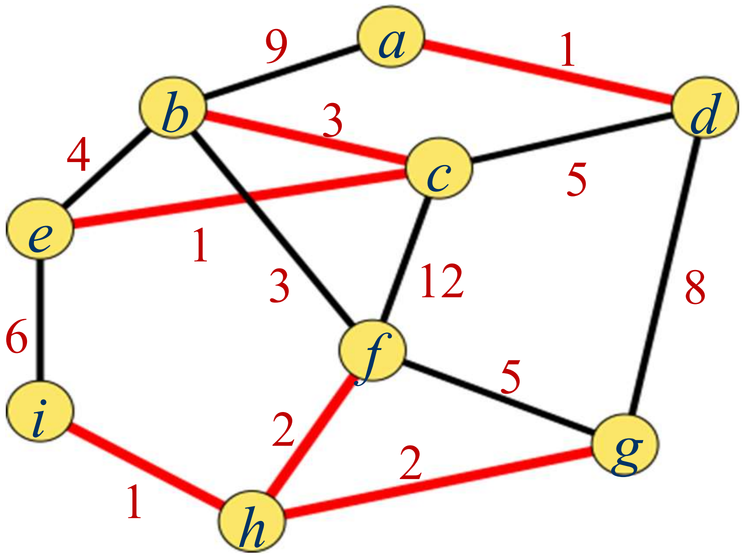
(a, d):1 (h, i):1 (c, e):1 (f, h):2 (g, h):2
(b, c):3 (b, f):3 (b, e):4 (c, d):5 (f, g):5
(e, i):6 (d, g):8 (a, b):9 (c, f):12



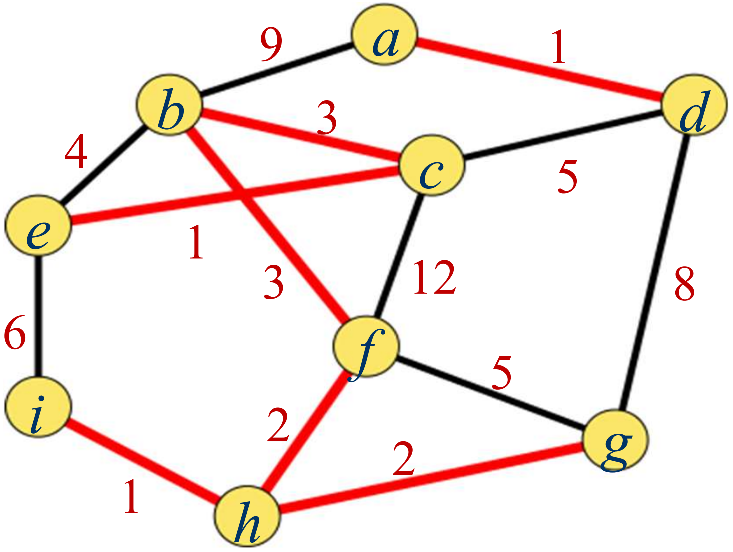
(a, d):1 (h, i):1 (c, e):1 (f, h):2 (g, h):2
(b, c):3 (b, f):3 (b, e):4 (c, d):5 (f, g):5
(e, i):6 (d, g):8 (a, b):9 (c, f):12



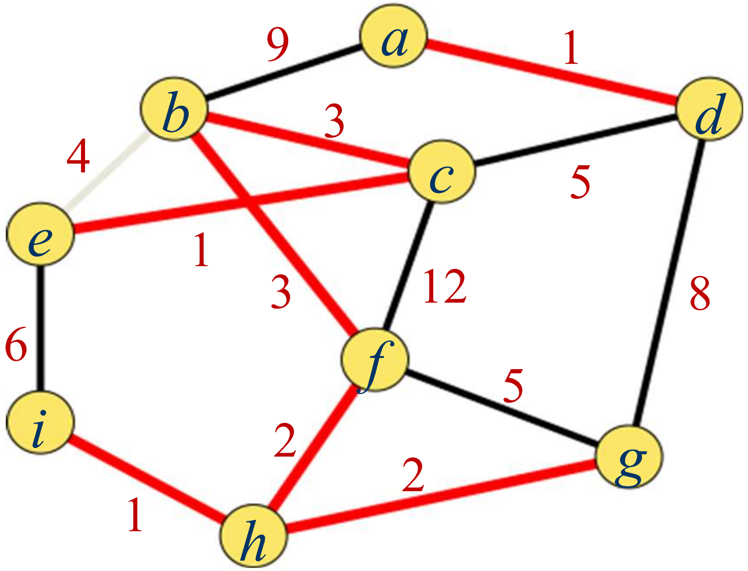
(*a*, *d*):1 (*h*, *i*):1 (*c*, *e*):1 (*f*, *h*):2 (*g*, *h*):2
(*b*, *c*):3 (*b*, *f*):3 (*b*, *e*):4 (*c*, *d*):5 (*f*, *g*):5
(*e*, *i*):6 (*d*, *g*):8 (*a*, *b*):9 (*c*, *f*):12



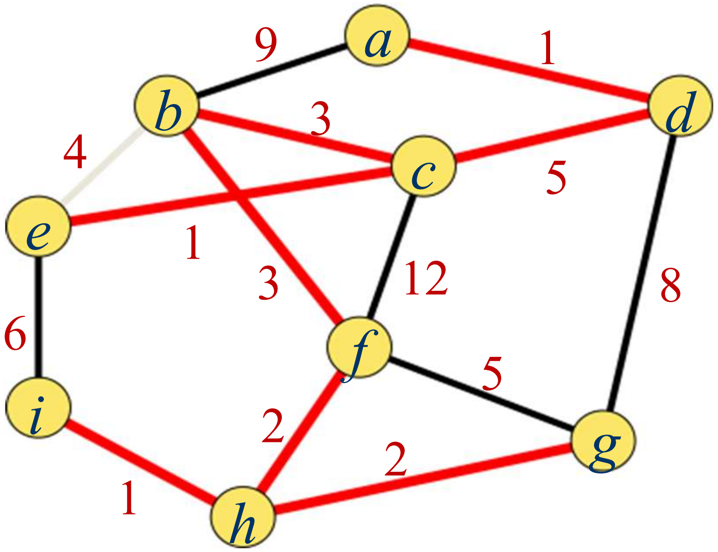
(*a*, *d*):1 (*h*, *i*):1 (*c*, *e*):1 (*f*, *h*):2 (*g*, *h*):2
(*b*, *c*):3 (*b*, *f*):3 (*b*, *e*):4 (*c*, *d*):5 (*f*, *g*):5
(*e*, *i*):6 (*d*, *g*):8 (*a*, *b*):9 (*c*, *f*):12



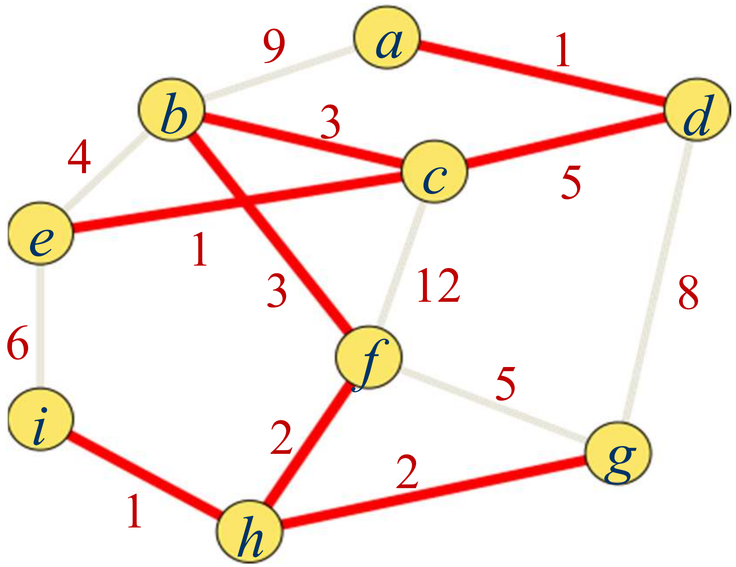
(a, d):1 (h, i):1 (c, e):1 (f, h):2 (g, h):2
(b, c):3 (b, f):3 **(b, e):4** (c, d):5 (f, g):5
(e, i):6 (d, g):8 (a, b):9 (c, f):12



(a, d):1 (h, i):1 (c, e):1 (f, h):2 (g, h):2
(b, c):3 (b, f):3 ~~(b, e)~~:4 **(c, d):5** (f, g):5
(e, i):6 (d, g):8 (a, b):9 (c, f):12



(*a*, *d*):1 (*h*, *i*):1 (*c*, *e*):1 (*f*, *h*):2 (*g*, *h*):2
(*b*, *c*):3 (*b*, *f*):3 ~~(*b*, *e*)~~:4 **(*c*, *d*):5** **(*f*, *g*):5**
(*e*, *i*):6 (*d*, *g*):8 (*a*, *b*):9 (*c*, *f*):12



(*a*, *d*):1 (*h*, *i*):1 (*c*, *e*):1 (*f*, *h*):2 (*g*, *h*):2
(*b*, *c*):3 (*b*, *f*):3 ~~(*b*, *e*):4~~ **(*c*, *d*):5** ~~(*f*, *g*):5~~
~~(*e*, *i*):6~~ ~~(*d*, *g*):8~~ ~~(*a*, *b*):9~~ ~~(*c*, *f*):12~~

Correctness Proof of Kruskal

1. At any time, A is a subset of a MST
2. In the end, A is a MST

Proof of 1. (By induction on $|A|$)

- Basic step: $|A|=0$, A_0 (= $\varnothing$) of course is a subset of a MST.
- Inductive hypothesis: A_k is a subset of a MST.
- Inductive step: Let's consider A_{k+1}. Suppose the $(k+1)th$ edge added to A_k is $e_{k+1}=(u,v)$. Without loss of generality, suppose $u \in C$, here C is a connected component of $G_k=(V, A_k)$. Since (u,v) is the smallest remaining edge, it must be a light edge connecting C to some other component of $G_k=(V, A_k)$, so it is safe for A_k. Hence A_{k+1} is subset of a MST.

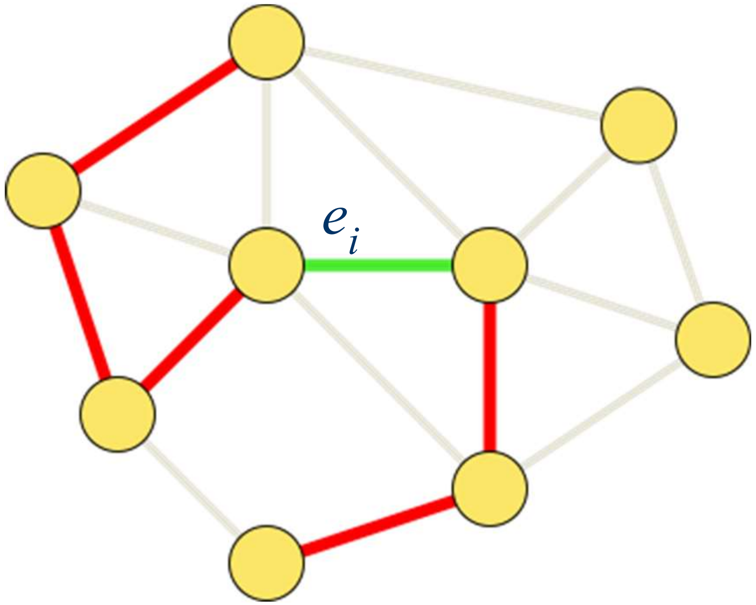
Correctness Proof (Continued)

Proof of 2. By 1, in the end, A is a subset of a MST T.

By contradiction suppose that A is not a MST. Then, $T\text{-}A \neq \varnothing$.

Suppose $e_i \in T\text{-}A$. e_i must have larger weight than each edge in A, since otherwise e_i should have been added to A. Then, after all the edges in A have been chosen , since $e_i \cup A$ do not contain a cycle, there must be a time that e_i is considered. e_i should be added to A at that time.

A contradiction.

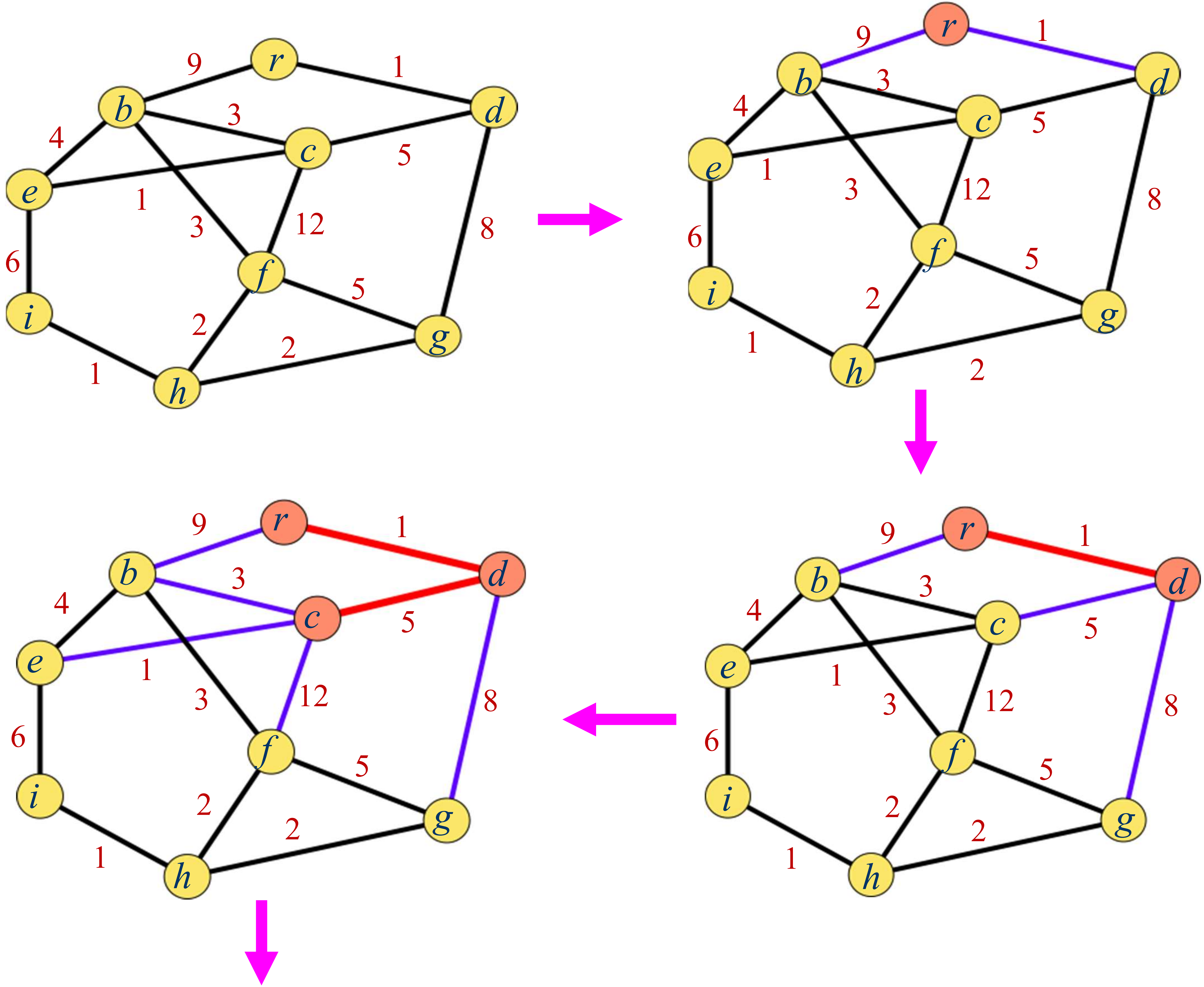


Idea of the Prim

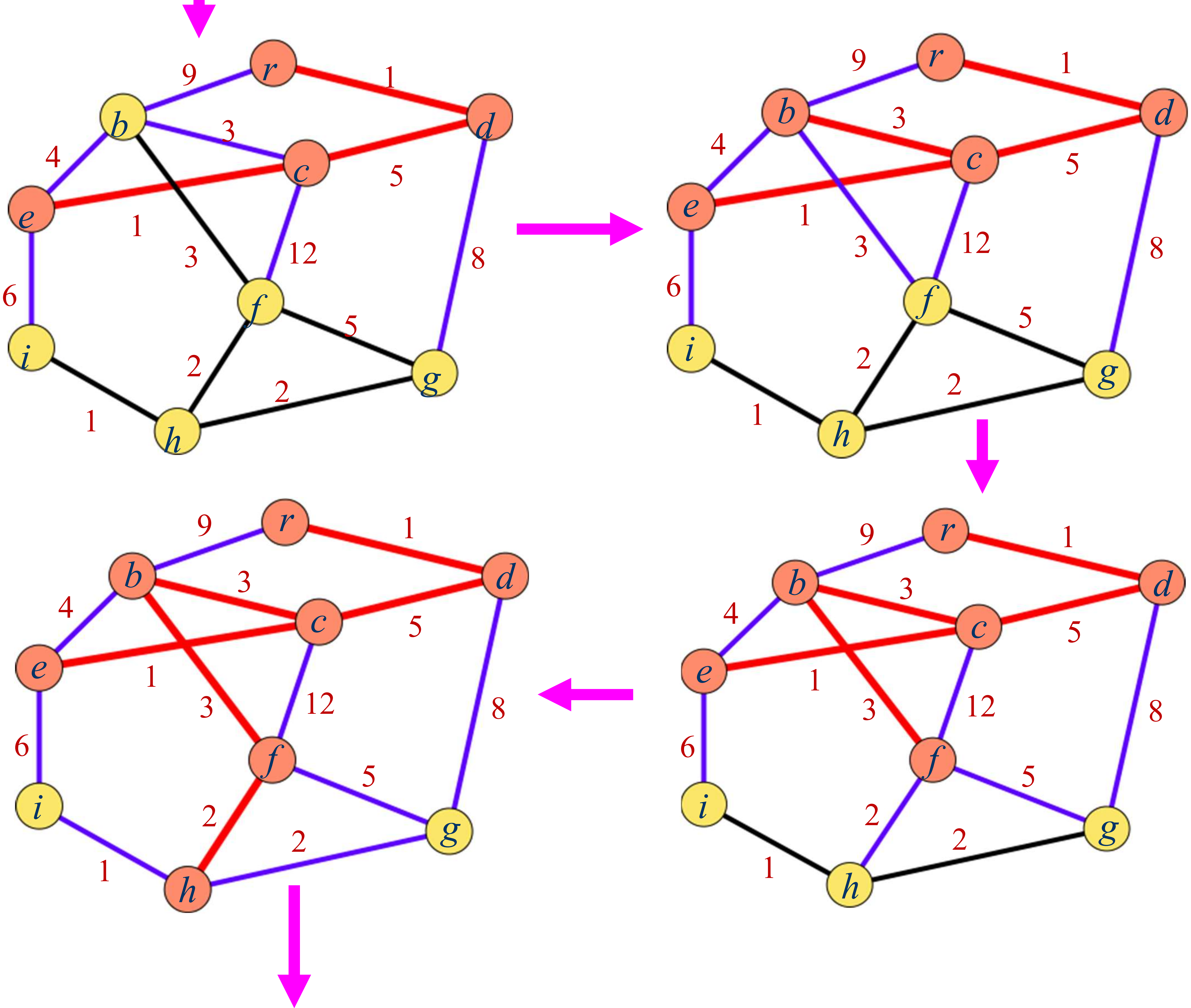
1. Start from a vertex r, add r to a vertex set U which is initialized to empty.
2. Find the least weight edge (u,v), $u \in U$, $v \in V-U$, add (u,v) to A, and add v to U.
3. Repeat 2 until A forms a spanning tree or $U = V$.

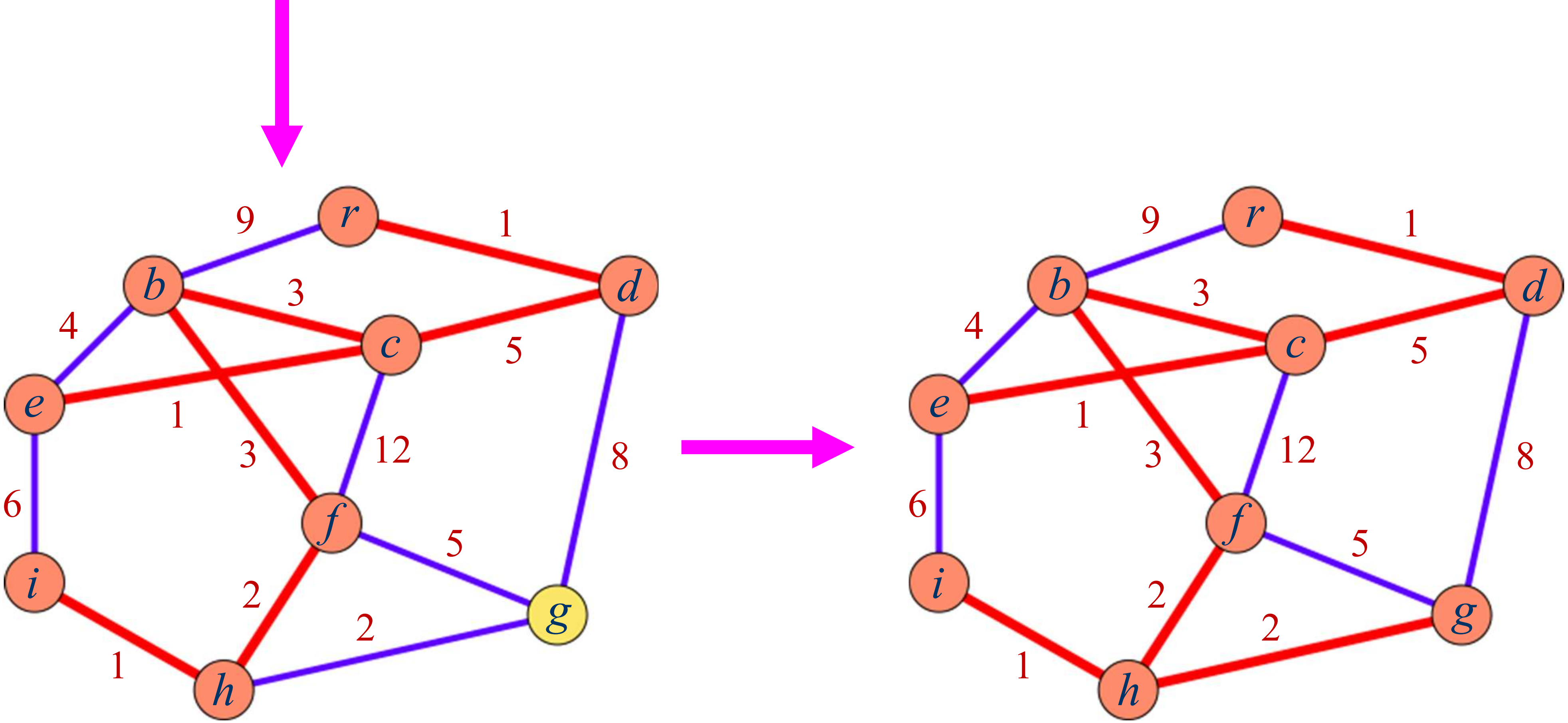
Prim's Algorithm

```
MST-PRIM(G, w, r)
for each u ∈ V[G]
    do key[u] ← ∞
       π[u] ← NIL
key[r] ← 0
Q ← V[G]
while Q ≠ ∅
    do u ← EXTRACT-MIN(Q)
       for each v ∈ Adj[u]
           do if v ∈ Q and w(u, v) < key[v]
                 then π[v] ← u
                      key[v] ← w(u, v)
```



r b c d e f g h i
9 1 3 4 5 1 3 12 8 6 5 2 2 1
r b c d e f g h i
9 1 3 4 5 1 3 12 8 6 5 2 1 2
r b c d e f g h i
9 1 3 4 5 1 3 12 8 6 5 2 2 1
r b c d e f g h i
9 1 3 4 5 1 3 12 8 6 5 2 2 1


r b c d e f g h i
9 1 3 4 5 1 3 12 8 6 5 2 2 1
r b c d e f g h i
9 1 3 4 5 1 3 12 8 6 5 2 2 1
r b c d e f g h i
9 1 3 4 5 1 3 12 8 6 5 2 2 1
r b c d e f g h i
9 1 3 4 5 1 3 12 8 6 5 2 2 1



r
b
c
d
e
f
g
h
i
9
1
3
4
5
1
3
12
8
6
5
2
2
1

Loop invariant

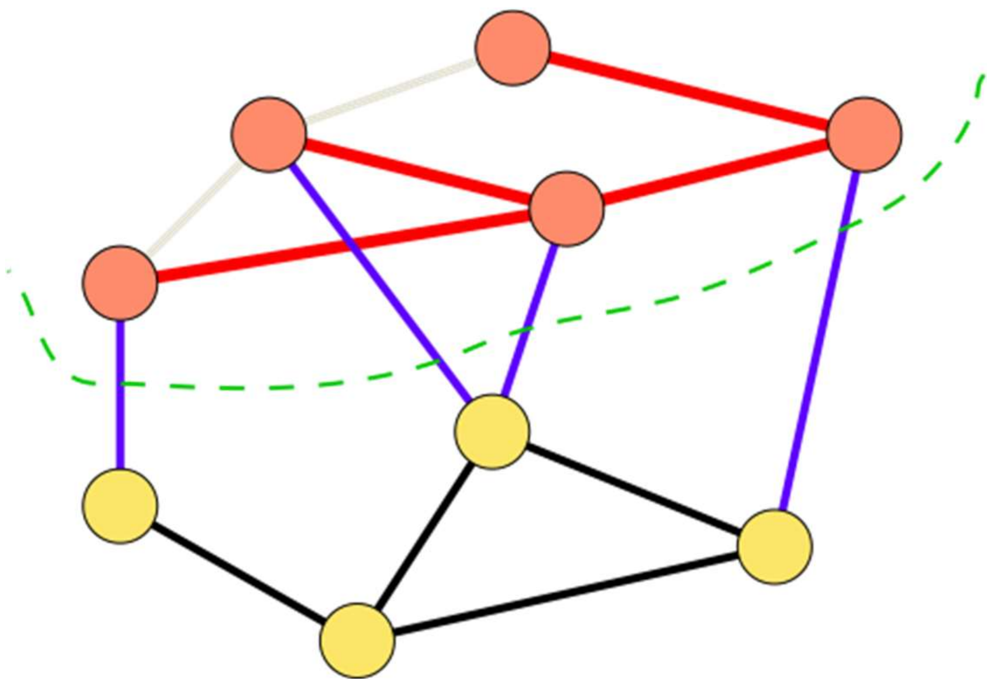
- *K*ey[u] is the minimum weight of the edges connect a vertex u in Q to vertices in U=V-Q.
- π[u] is right such a vertex in U.
- The three loop invariant:
  - $A=\{(v,\pi[v]) \mid v\in V-\{r\}-Q\}$
  - The vertices already placed into the MST are those in U=V-Q
  - For all vertices $v \in$Q, if $\pi[v]\neq$NIL, then key[v]<∞ and key[v] is the weight of a light edge (v,π[v]) connecting v to some vertex already placed into the MST.

Correctness Proof

Similar to that of Kruskal's.

1: each time we add a safe edge (a light edge crossing (U, Q)).

2: in the end, if A is not a MST, then Q is not empty, and (U, Q) is a cut that respects A. Since G is connected, we can find a light edge crossing (U, Q), which is safe for A. A contradiction.



Conclusion

- The Minimum Spanning Tree Problem
- A Generic Algorithm
- Kruskal’s Algorithm
- Prim’s Algorithm

Homework

- 23.1-5, 23.2-1, 23.2-8.
- Problem 23-3, 23-4(not compulsory)