Presentation
On
Prim's Algorithm

Submitted to:
Mrs. Shano Solanki
Assistant Professor
Department of CSE
NITTTR Chandigarh

Submitted by:
Pankaj Thakur
ME(CSE) Modular
RN 171408

Sub Topics to be covered:

- Introduction
- Basics concepts
- Minimum Spanning Tree Problem
- Solution to Minimum Spanning Tree Problem
- Generic-MST
- Prim’s algorithm details
- Complexity Analysis of Prim’s Algorithm.
- Applications and latest research work based on Prim’s Algorithm
- Summary
- Queries.

Introduction

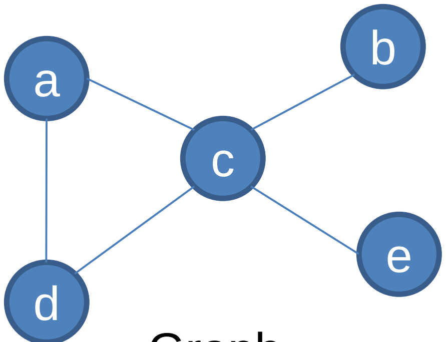
•Prim's algorithm is a greedy algorithm that finds a minimum spanning tree for a weighted undirected graph.

•Developed in 1930 by Czech mathematician Vojtěch Jarník and later rediscovered and republished by computer scientists Robert C. Prim in 1957 and E. W. Dijkstra in 1959.

•Therefore, it is also sometimes called the **DJP algorithm**, **Jarník's algorithm**, the **Prim–Jarník algorithm**,or the **Prim–Dijkstra algorithm**.

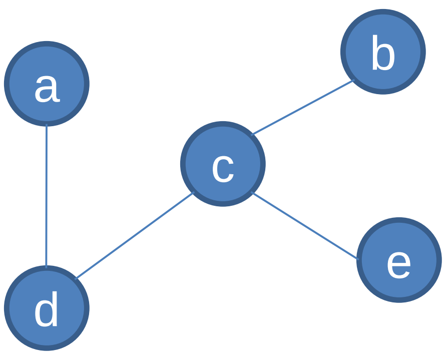
Basic Concepts

Spanning Trees: A subgraph T of a undirected graph G = (V, E) is a spanning tree of G if it is a tree and contains every vertex of G.

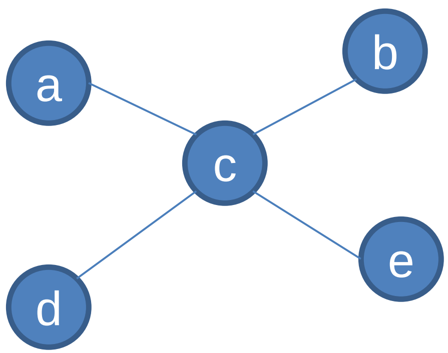


Graph

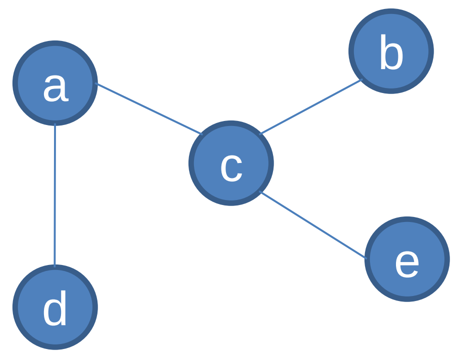
- Every connected graph has a spanning tree.
- May have multiple spanning tree.
- For example see this graph.



Spanning Tree 1

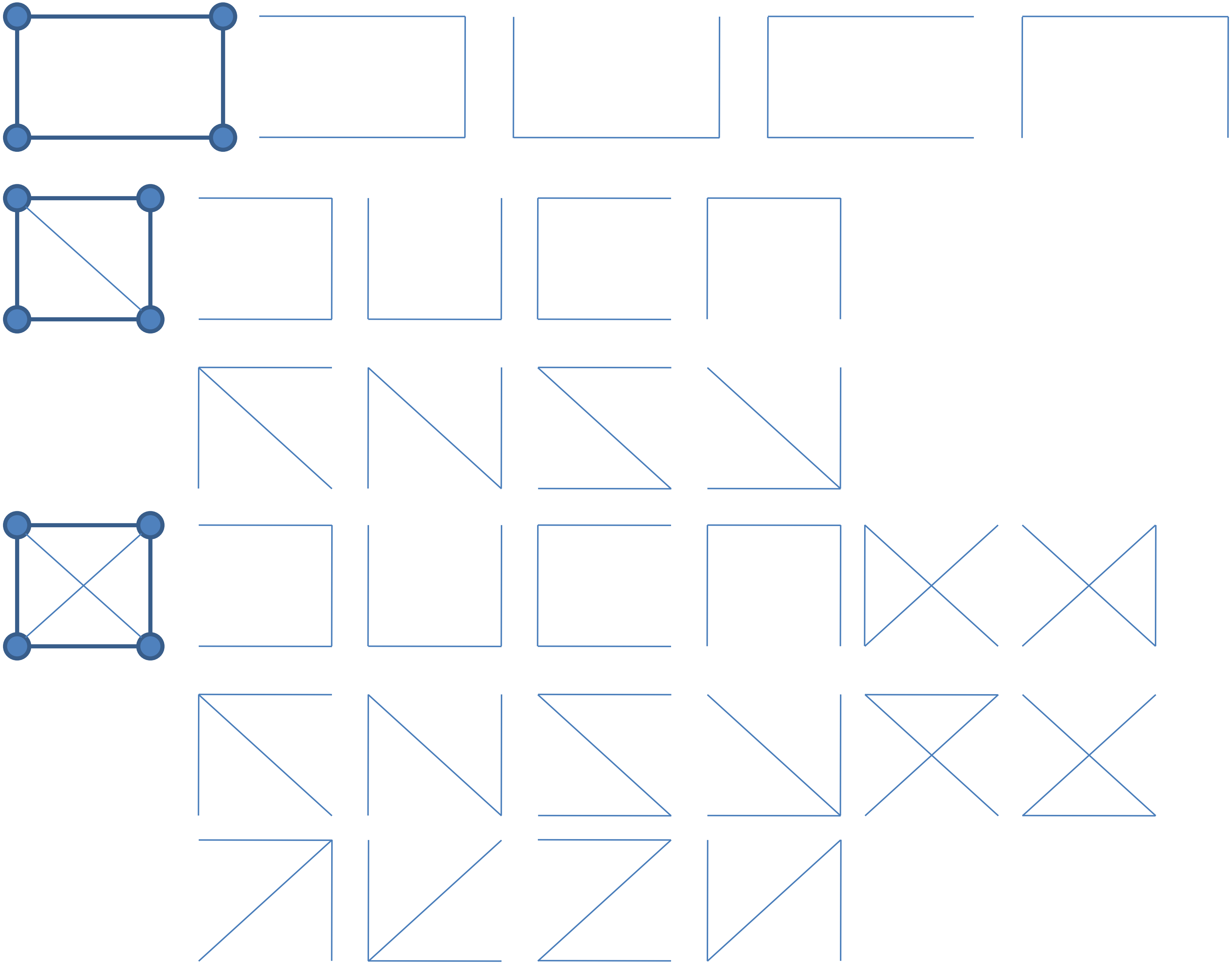


Spanning Tree 2



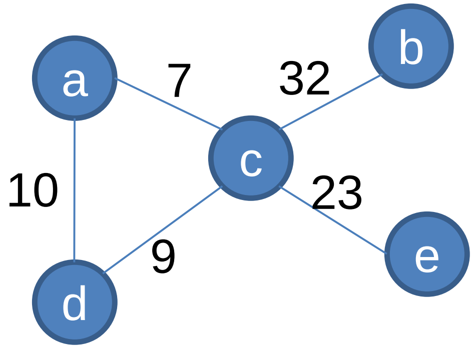
Spanning Tree 3

Spanning Tree Cont…



Basic Concepts Cont…..

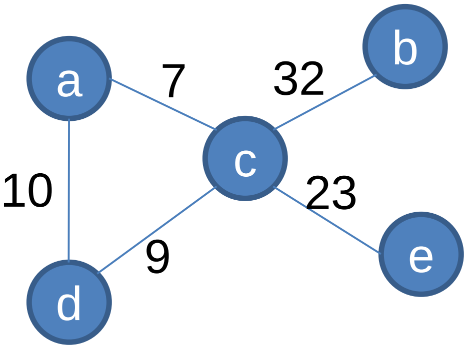
Weighted Graph: A weighted graph is a graph, in which each edge has a weight (some real number) Example:



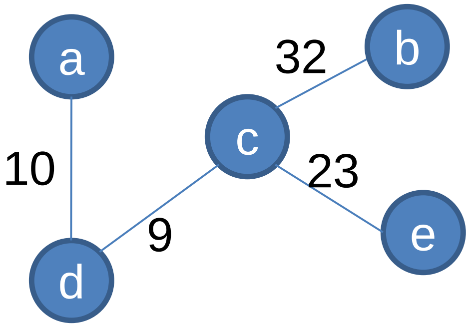
Weighted Graph

Basic Concepts Cont….

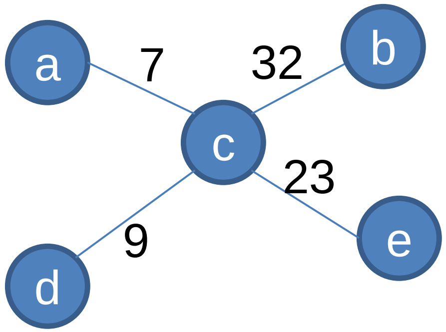
Minimum Spanning Tree in an undirected connected weighted graph is a spanning tree of minimum weight. Example:



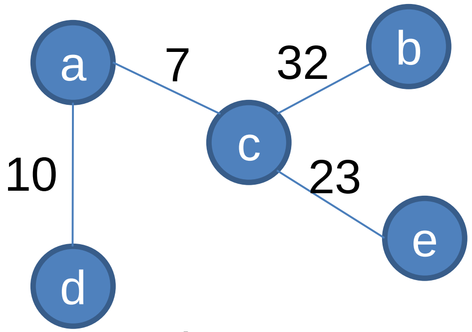
Weighted Graph



Spanning Tree 1,
w=74



Spanning Tree 2,
w=71
(Minimum Spanning Tree)



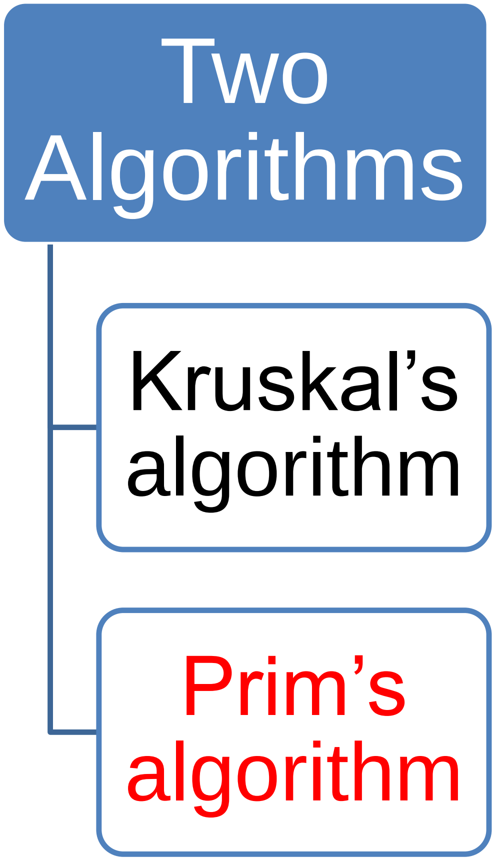
Spanning Tree 3,
w=72

Minimum Spanning Tree Problem

MST Problem : Given a connected weighted undirected graph G, design an algorithm that outputs a minimum spanning tree (MST) of graph G.

- **How to find Minimum Spanning Tree ?**

- **Generic solution to MST**



GENERIC-MST Algorithm

GENERIC-MST (G, w)

```
A = Ɵ
while A does not form a spanning tree
        find an edge ( u, v ) that is safe for A
        A = A U { ( u, v ) }
return A
```

- The idea is to start with an empty graph and try to add edges one at a time, always making sure that what is built remains acyclic.
- Gives us an idea how to grow a MST.
- An edge (u, v) is safe for A if and only if $A \cup \{(u, v)\}$ is also a subset of some MST

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 is not Empty
          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 )
```

- A special case of generic minimum-spanning-tree algorithm and operates much like Dijkstra's algorithm.
- Edges in the set A always form a single tree.
- Greedy algorithm since at each step it adds to the tree an edge that contributes the minimum amount possible to the tree's weight.
- Connected graph G and the root r of the MST to be drawn are inputs.
- During execution of the algorithm, all vertices that are not in the MST reside in a min-priority queue Q based on a key attribute.
- For each vertex v, the attribute v.key is the minimum weight of any edge connecting v to a vertex in the tree.
- The attribute v.Π names parent of v in the tree.
- Maintains the set A from GENERIC-MST as

$A = \{ (v, v.\Pi) : v \in V - \{r\} - Q \}.$

When the algorithm terminates, the min-priority queue Q is empty.

The MST A for G is thus

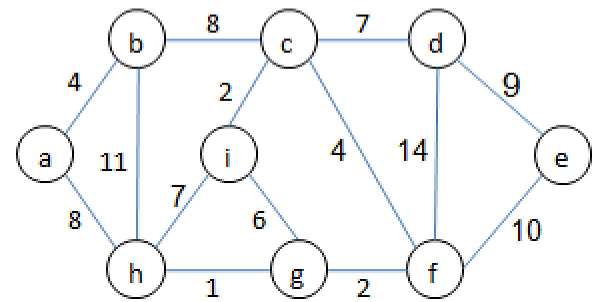
$A = \{ (v, v.\Pi) : v \in V - \{r\} \}.$

PRIM's Algorithm (Steps 1-5 : Initialization)

MST-PRIM(G,w,r)

1	for each u∈V[G]	Initialization
2	do key[u]←∞	
3	Π[u]←NIL	
4	key[r]←0	
5	Q←V[G]	
6	**while** Q is not Empty	
7	**do** u←EXTRACT-MIN(Q)	
8	**for** each v∈Adj[u]	
9	**do if** v∈Q and w(u,v)<key[v]	
10	**then** Π[v]←u	
11	key[v]←w(u, v)	

Example Graph



u	a	b	c	d	e	f	g	h	i
key[u]	∞	∞	∞	∞	∞	∞	∞	∞	∞
Π[u]	NIL	NIL	NIL	NIL	NIL	NIL	NIL	NIL	NIL

After Steps 1-3

u	a	b	c	d	e	f	G	H	i
key[u]	0	∞	∞	∞	∞	∞	∞	∞	∞
Π[u]	NIL	NIL	NIL	NIL	NIL	NIL	NIL	NIL	NIL

After Step 4

Q	A	b	c	d	e	f	g	h	i
A	EMPTY								

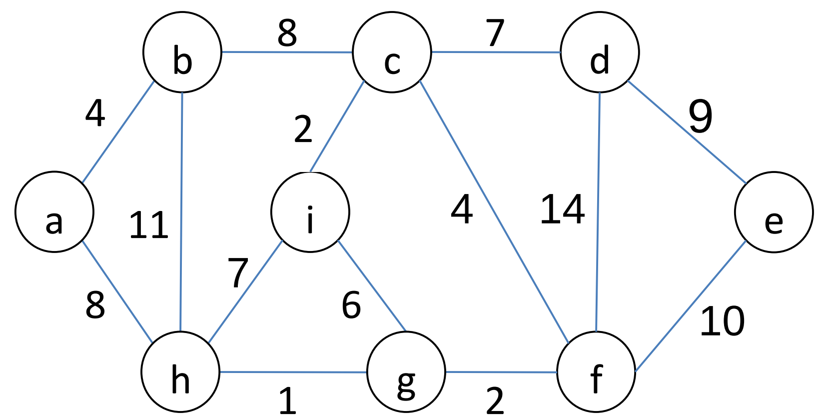
After Step 5

PRIM's Algorithm (Steps 6 to 11) ……

MST-PRIM(G,w,r)

1	for each u∈V[G]
2	do key[u]←∞
3	Π[u]←NIL
4	key[r]←0
5	Q←V[G]
6	**while** Q is not Empty
7	**do** u←EXTRACT-MIN(Q)
8	**for** each v∈Adj[u]
9	**do if** v∈Q and w(u,v)<key[v]
10	**then** Π[v]←u
11	key[v]←w(u,v)

Example Graph



Before Step 6

Q	a	b	c	d	e	f	g	h	i
key[u]	0	∞	∞	∞	∞	∞	∞	∞	∞
Π[u]	NIL	NIL	NIL	NIL	NIL	NIL	NIL	NIL	NIL

Q	a	b	c	d	E	f	g	h	i
A	EMPTY								

Steps 6-11 (for u=a)

u	v	v∈Q AND w(u,v) < Key[v]	Π[v]←u, Key[v]←w(u,v)
a	b	YES	Π[b]←a, Key[b]←4
a	h	YES	Π[h]←a, Key[h]←8

After Step 6-11 (for u=a)

Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	∞	∞	∞	∞	∞	8	∞
Π[u]	NIL	a	NIL	NIL	NIL	NIL	NIL	a	NIL

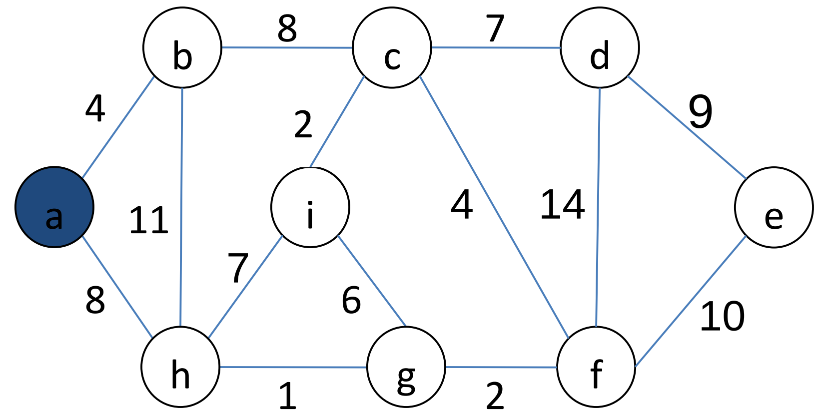
Q	b	c	d	e	f	g	h	i	
A	a								

PRIM’s Algorithm (Steps 6 to 11) ……

MST-PRIM(G,w,r)

1	for each u∈V[G]
2	do key[u]←∞
3	Π[u]←NIL
4	key[r]←0
5	Q←V[G]
6	**while** Q is not Empty
7	**do** u←EXTRACT-MIN(Q)
8	**for** each v∈Adj[u]
9	**do if** v∈Q and w(u,v)<key[v]
10	**then** Π[v]←u
11	key[v]←w(u,v)

Example Graph



Before Step 6

Q	a	b	c	d	e	f	g	h	i
key[u]	0	∞	∞	∞	∞	∞	∞	∞	∞
Π[u]	NIL	NIL	NIL	NIL	NIL	NIL	NIL	NIL	NIL

Q	a	b	c	d	e	f	g	h	i
A									

Steps 6-11 (for u=a)

u	v	v∈Q AND w(u,v) < Key[v]	Π[v]←u, Key[v]←w(u,v)
a	b	YES	Π[b]←a, Key[b]←4
a	h	YES	Π[h]←a, Key[h]←8

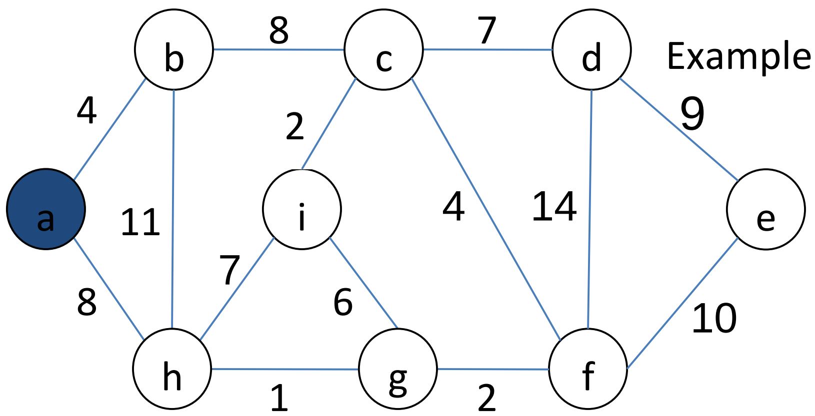
After Step 6-11 (for u=a)

Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	∞	∞	∞	∞	∞	8	∞
Π[u]	NIL	a	NIL	NIL	NIL	NIL	NIL	a	NIL

Q	b	c	d	e	f	g	h	i	
A	a								

PRIM's Algorithm (Steps 6 to 11, for u=b)

6	**while** Q is not Empty
7	**do** u←EXTRACT-MIN(Q)
8	**for** each v∈Adj[u]
9	**do if** v∈Q and w(u,v)<key[v]
10	**then** Π[v]←u
11	key[v]←w(u,v)



Example Graph

Status of Q before using u=b

Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	∞	∞	∞	∞	∞	8	∞
Π[u]	NIL	a	NIL	NIL	NIL	NIL	NIL	a	NIL

Q	b	c	d	e	f	g	h	i	
A	a								

Steps 6-11(for u=b)

u	v	v∈Q AND w(u, v) < Key[v]	Then Π[v]←u, Key[v]←w(u,v)
b	c	YES	Π[c]←b, Key[c]←8
b	h	NO	do nothing
b	a	NO	do nothing

After Step 6-11 (for u=b)

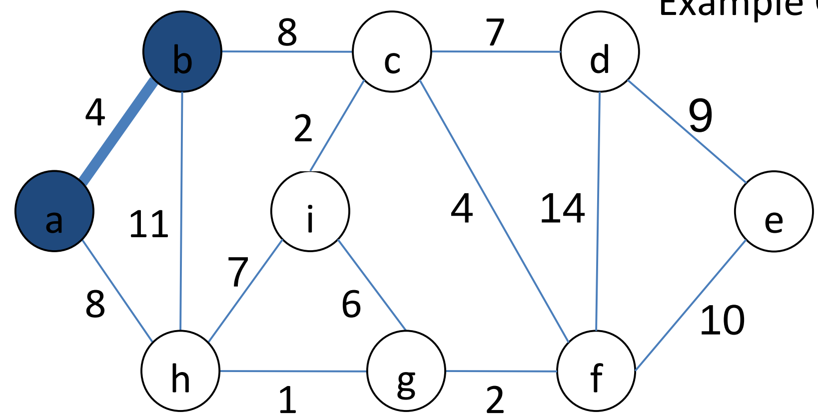
Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	8	∞	∞	∞	∞	8	∞
Π[u]	NIL	a	b	NIL	NIL	NIL	NIL	a	NIL

Q	c	d	e	f	g	h	i		
A	a	b							

PRIM's Algorithm (Steps 6 to 11, for u=b)

6	**while** Q is not Empty
7	**do** u←EXTRACT-MIN(Q)
8	**for** each v∈Adj[u]
9	**do if** v∈Q and w(u,v)<key[v]
10	**then** Π[v]←u
11	key[v]←w(u,v)

Example Graph



Status of Q before using u=b

Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	∞	∞	∞	∞	∞	8	∞
Π[u]	NIL	a	NIL	NIL	NIL	NIL	NIL	a	NIL

Q	b	c	d	e	f	g	h	i	
A	a								

Steps 6-11(for u=b)

u	v	v∈Q AND w(u, v) < Key[v]	Then Π[v]←u, Key[v]←w(u,v)
b	c	YES	Π[c]←b, Key[c]←8
b	h	NO	do nothing
b	a	NO	do nothing

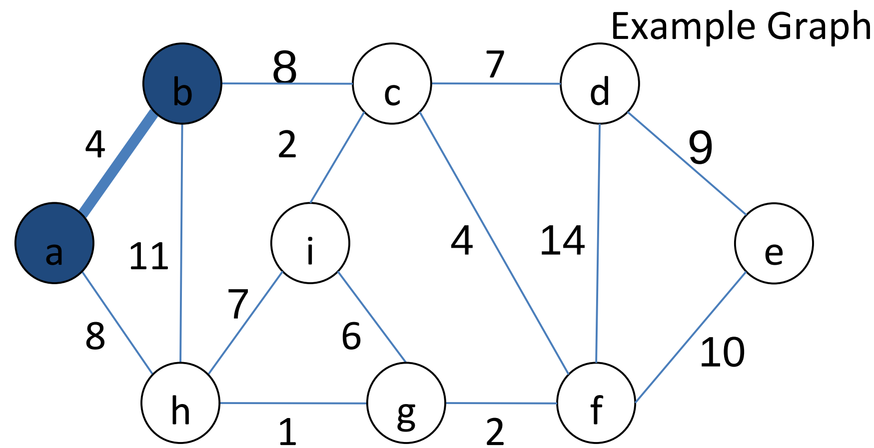
After Step 6-11 (for u=b)

Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	8	∞	∞	∞	∞	8	∞
Π[u]	NIL	a	b	NIL	NIL	NIL	NIL	a	NIL

Q	c	d	e	f	g	h	i		
A	a	b							

PRIM's Algorithm (Steps 6 to 11, for u=c)

6	**while** Q is not Empty
7	**do** u←EXTRACT-MIN(Q)
8	**for** each v∈Adj[u]
9	**do if** v∈Q and w(u,v)<key[v]
10	**then** Π[v]←u
11	key[v]←w(u,v)



Status of Q before using u=c

Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	8	∞	∞	∞	∞	8	∞
Π[u]	NIL	a	b	NIL	NIL	NIL	NIL	a	NIL

Q	c	d	e	f	g	h	i		
A	a	b							

Steps 6-11(for u=c)

u	v	v∈Q AND w(u, v) < Key[v]	Then Π[v]←u, Key[v]←w(u,v)
c	i	YES	Π[i]←c, Key[i]←2
c	f	YES	Π[f]←c, Key[f]←4
c	d	YES	Π[d]←c, Key[d]←7
c	b	NO	Do Nothing

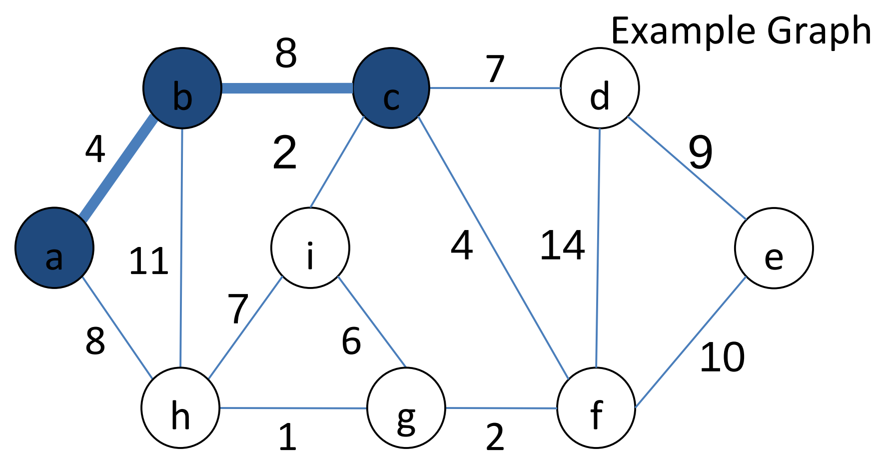
After Step 6-11 (for u=c)

Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	8	7	∞	4	∞	8	2
Π[u]	NIL	a	b	c	NIL	c	NIL	a	c

Q	d	e	f	g	h	i			
A	a	b	c						

PRIM's Algorithm (Steps 6 to 11, for u=c)

6	**while** Q is not Empty
7	**do** u←EXTRACT-MIN(Q)
8	**for** each v∈Adj[u]
9	**do if** v∈Q and w(u,v)<key[v]
10	**then** Π[v]←u
11	key[v]←w(u,v)



Status of Q before using u=c

Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	8	∞	∞	∞	∞	8	∞
Π[u]	NIL	a	b	NIL	NIL	NIL	NIL	a	NIL

Q	c	d	e	f	g	h	i		
A	a	b							

Steps 6-11(for u=c)

u	v	v∈Q AND w(u, v) < Key[v]	Then Π[v]←u, Key[v]←w(u,v)
c	i	YES	Π[i]←c, Key[i]←2
c	f	YES	Π[f]←c, Key[f]←4
c	d	YES	Π[d]←c, Key[d]←7
c	b	NO	Do Nothing

After Step 6-11 (for u=c)

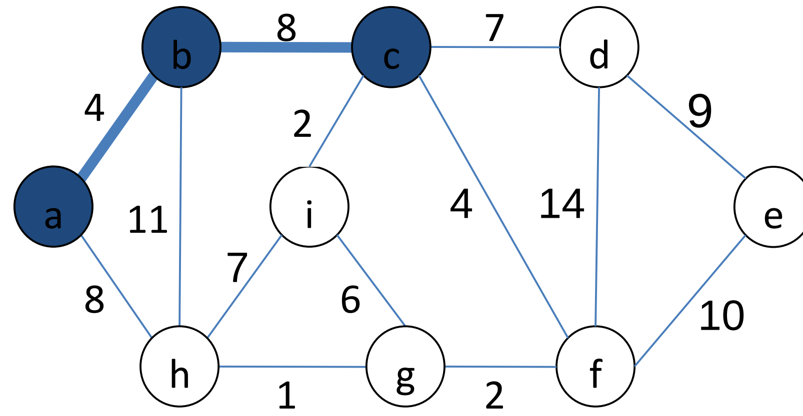
Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	8	7	∞	4	∞	8	2
Π[u]	NIL	a	b	c	NIL	c	NIL	a	c

Q	d	e	f	g	h	i			
A	a	b	c						

PRIM's Algorithm (Steps 6 to 11, for u=i)

6	**while** Q is not Empty
7	**do** u←EXTRACT-MIN(Q)
8	**for** each v∈Adj[u]
9	**do if** v∈Q and w(u,v)<key[v]
10	**then** Π[v]←u
11	key[v]←w(u,v)

Example Graph



Status of Q before using u=i

Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	8	7	∞	4	∞	8	2
Π[u]	NIL	a	b	c	NIL	c	NIL	a	c

Q	d	e	f	g	h	i			
A	a	b	c						

Steps 6-11(for u=i)

u	v	v∈Q AND w(u, v) < Key[v]	Then Π[v]←u, Key[v]←w(u,v)
i	h	YES	Π[h]←i, Key[h]←7
i	g	YES	Π[g]←i, Key[g]←6
i	c	NO	Do Nothing

After Step 6-11 (for u=i)

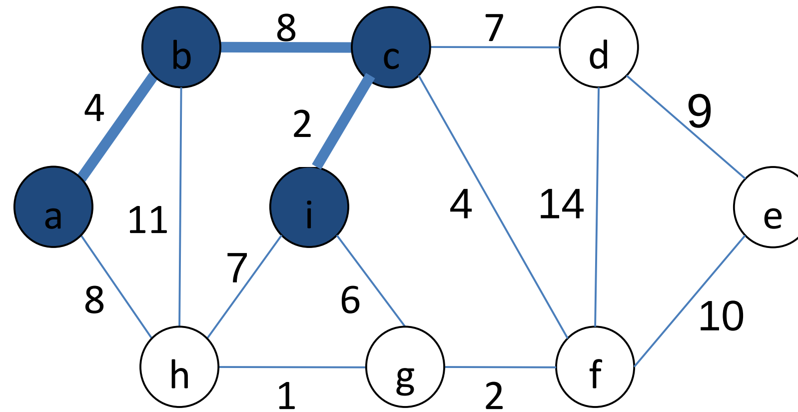
Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	8	7	∞	4	6	7	2
Π[u]	NIL	a	b	c	NIL	c	i	i	c

Q	d	e	f	g	h				
A	a	b	c	i					

PRIM's Algorithm (Steps 6 to 11, for u=i)

6	**while** Q is not Empty
7	**do** u←EXTRACT-MIN(Q)
8	**for** each v∈Adj[u]
9	**do if** v∈Q and w(u,v)<key[v]
10	**then** Π[v]←u
11	key[v]←w(u,v)

Example Graph



Status of Q before using u=i

Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	8	7	∞	4	∞	8	2
Π[u]	NIL	a	b	c	NIL	c	NIL	a	c

Q	d	e	f	g	h	i			
A	a	b	c						

Steps 6-11(for u=i)

u	v	v∈Q AND w(u, v) < Key[v]	Then Π[v]←u, Key[v]←w(u,v)
i	h	YES	Π[h]←i, Key[h]←7
i	g	YES	Π[g]←i, Key[g]←6
i	c	NO	Do Nothing

After Step 6-11 (for u=i)

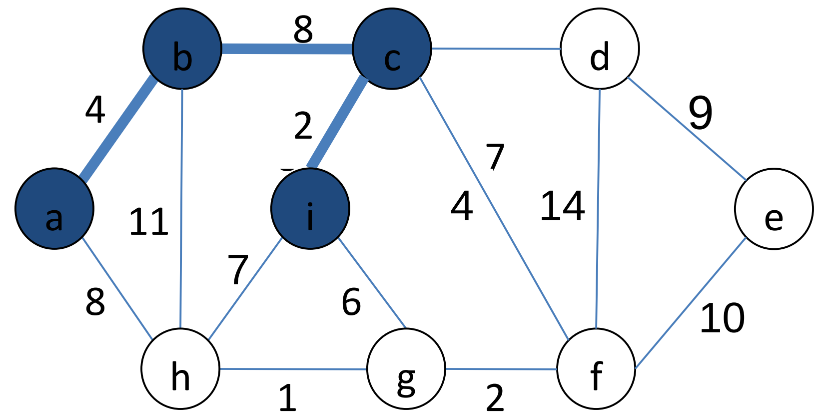
Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	8	7	∞	4	6	7	2
Π[u]	NIL	a	b	c	NIL	c	i	i	c

Q	d	e	f	g	h				
A	a	b	c	i					

PRIM's Algorithm (Steps 6 to 11, for u=f)

6	**while** Q is not Empty
7	**do** u←EXTRACT-MIN(Q)
8	**for** each v∈Adj[u]
9	**do if** v∈Q and w(u,v)<key[v]
10	**then** Π[v]←u
11	key[v]←w(u,v)

Example Graph



Status of Q before using u=f

Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	8	7	∞	4	6	7	2
Π[u]	NIL	a	b	c	NIL	c	i	i	c

Q	d	e	f	g	h				
A	a	b	c	i					

Steps 6-11(for u=f)

u	v	v∈Q AND w(u, v) < Key[v]	Then Π[v]←u, Key[v]←w(u,v)
f	d	NO	Do nothing
f	e	YES	Π[e]←f, Key[e]←10
f	g	YES	Π[g]←f, Key[g]←2
f	c	NO	Do nothing

After Step 6-11 (for u=f)

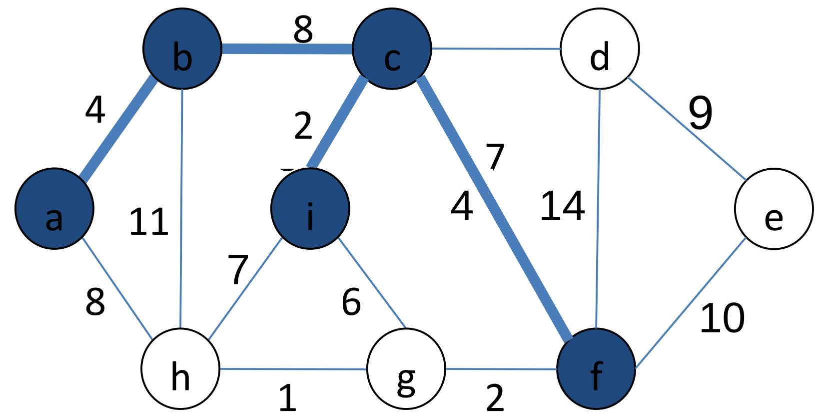
Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	8	7	10	4	2	7	2
Π[u]	NIL	a	b	c	f	c	f	i	c

Q	d	e	g	h					
A	a	b	c	i	f				

PRIM's Algorithm (Steps 6 to 11, for u=f)

6	**while** Q is not Empty
7	**do** u←EXTRACT-MIN(Q)
8	**for** each v∈Adj[u]
9	**do if** v∈Q and w(u,v)<key[v]
10	**then** Π[v]←u
11	key[v]←w(u,v)

Example Graph



Status of Q before using u=f

Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	8	7	∞	4	6	7	2
Π[u]	NIL	a	b	c	NIL	c	i	i	c

Q	d	e	f	g	h				
A	a	b	c	i					

Steps 6-11(for u=f)

u	v	v∈Q AND w(u, v) < Key[v]	Then Π[v]←u, Key[v]←w(u,v)
f	d	NO	Do nothing
f	e	YES	Π[e]←f, Key[e]←10
f	g	YES	Π[g]←f, Key[g]←2
f	c	NO	Do nothing

After Step 6-11 (for u=f)

Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	8	7	10	4	2	7	2
Π[u]	NIL	a	b	c	f	c	f	i	c

Q	d	e	g	h					
A	a	b	c	i	f				

PRIM's Algorithm (Steps 6 to 11, for u=g)

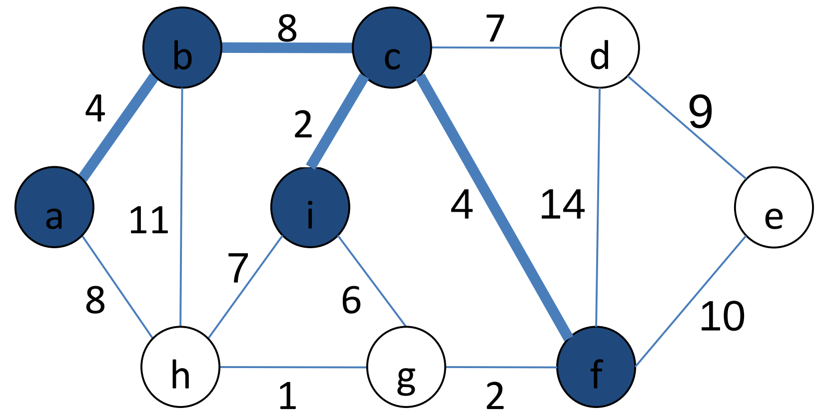
6	**while** Q is not Empty
7	**do** u←EXTRACT-MIN(Q)
8	**for** each v∈Adj[u]
9	**do if** v∈Q and w(u,v)<key[v]
10	**then** Π[v]←u
11	key[v]←w(u,v)

Status of Q before using u=g

Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	8	7	10	4	2	7	2
Π[u]	NIL	a	b	c	f	c	f	i	c

Q	d	e	g	h					
A	a	b	c	i	f				

Example Graph



After Step 6-11 (for u=g)

Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	8	7	10	4	2	1	2
Π[u]	NIL	a	b	c	f	c	f	g	c

Q	d	e	h						
A	a	b	c	i	f	g			

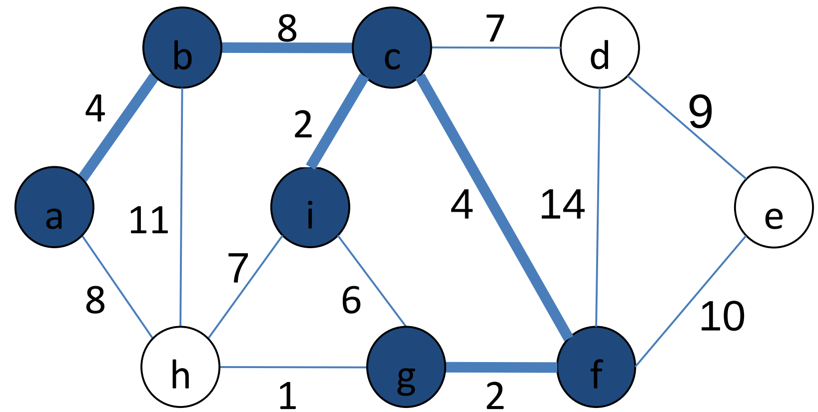
Steps 6-11(for u=g)

u	v	v∈Q AND w(u, v) < Key[v]	Then Π[v]←u, Key[v]←w(u,v)
g	h	YES	Π[h]←g, Key[h]←1
g	i	NO	Do Nothing
g	f	NO	Do Nothing

PRIM's Algorithm (Steps 6 to 11, for u=g)

6	**while** Q is not Empty
7	**do** u←EXTRACT-MIN(Q)
8	**for** each v∈Adj[u]
9	**do if** v∈Q and w(u,v)<key[v]
10	**then** Π[v]←u
11	key[v]←w(u,v)

Example Graph



Status of Q before using u=g

Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	8	7	10	4	2	7	2
Π[u]	NIL	a	b	c	f	c	f	i	c

Q	d	e	g	h					
A	a	b	c	i	f				

After Step 6-11 (for u=g)

Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	8	7	10	4	2	1	2
Π[u]	NIL	a	b	c	f	c	f	g	c

Q	d	e	h						
A	a	b	c	i	f	g			

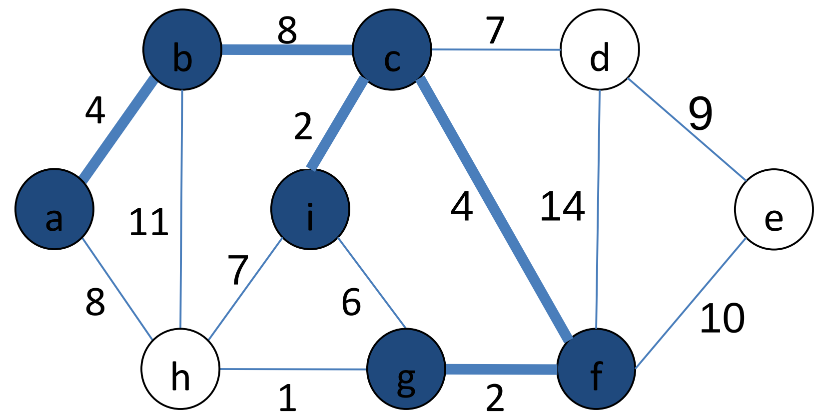
Steps 6-11(for u=g)

u	v	v∈Q AND w(u, v) < Key[v]	Then Π[v]←u, Key[v]←w(u,v)
g	h	YES	Π[h]←g, Key[h]←1
g	i	NO	Do Nothing
g	f	NO	Do Nothing

PRIM's Algorithm (Steps 6 to 11, for u=h)

6	**while** Q is not Empty
7	**do** u←EXTRACT-MIN(Q)
8	**for** each v∈Adj[u]
9	**do if** v∈Q and w(u,v)<key[v]
10	**then** Π[v]←u
11	key[v]←w(u,v)

Example Graph



Status of Q before using u=h

Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	8	7	10	4	2	7	2
Π[u]	NIL	a	b	c	f	c	f	i	c

Q	d	e	h						
A	a	b	c	i	f	g			

Steps 6-11(for u=h)

u	v	v∈Q AND w(u, v) < Key[v]	Then Π[v]←u, Key[v]←w(u,v)
h	a	NO	Do Nothing
h	b	NO	Do Nothing
h	i	NO	Do Nothing
h	g	NO	Do Nothing

After Step 6-11 (for u=h)

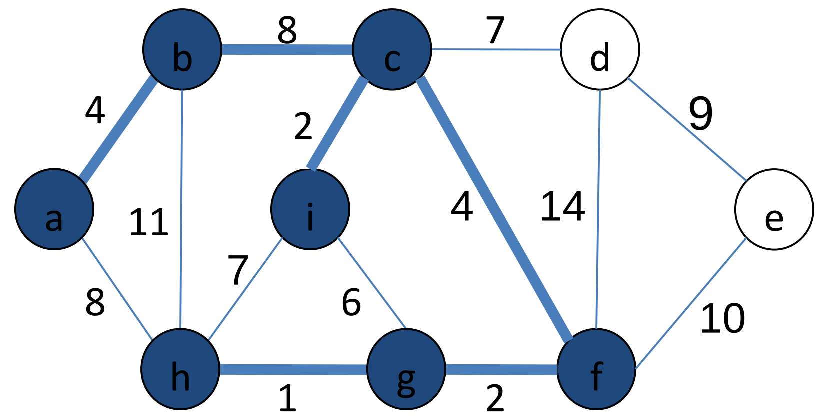
Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	8	7	10	4	2	1	2
Π[u]	NIL	a	b	c	f	c	f	g	c

Q	d	e							
A	a	b	c	i	f	g	h		

PRIM's Algorithm (Steps 6 to 11, for u=h)

6	**while** Q is not Empty
7	**do** u←EXTRACT-MIN(Q)
8	**for** each v∈Adj[u]
9	**do if** v∈Q and w(u,v)<key[v]
10	**then** Π[v]←u
11	key[v]←w(u,v)

Example Graph



Status of Q before using u=h

Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	8	7	10	4	2	7	2
Π[u]	NIL	a	b	c	f	c	f	i	c

Q	d	e	g	h					
A	a	b	c	i	f				

Steps 6-11(for u=h)

u	v	v∈Q AND w(u, v) < Key[v]	Then Π[v]←u, Key[v]←w(u,v)
h	a	NO	Do Nothing
h	b	NO	Do Nothing
h	i	NO	Do Nothing
h	g	NO	Do Nothing

After Step 6-11 (for u=h)

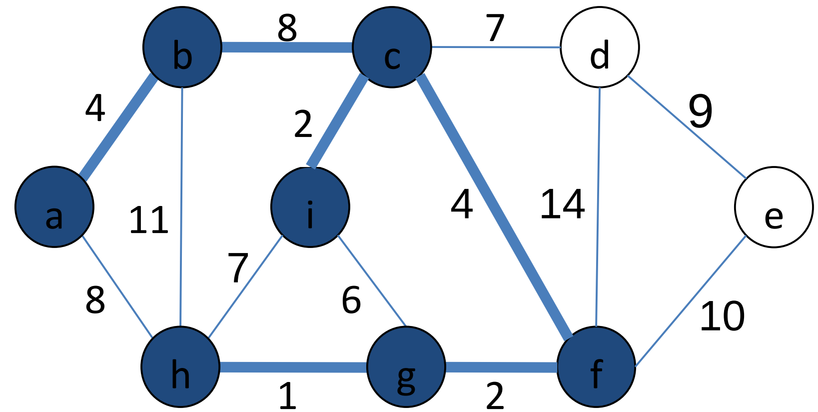
Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	8	7	10	4	2	1	2
Π[u]	NIL	a	b	c	f	c	f	g	c

Q	d	e							
A	a	b	c	i	f	g	h		

PRIM's Algorithm (Steps 6 to 11, for u=d)

6	**while** Q is not Empty
7	**do** u←EXTRACT-MIN(Q)
8	**for** each v∈Adj[u]
9	**do if** v∈Q and w(u,v)<key[v]
10	**then** Π[v]←u
11	key[v]←w(u,v)

Example Graph



Status of Q before using u=d

Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	8	7	10	4	2	7	2
Π[u]	NIL	a	b	c	f	c	f	i	c

Q	d	e							
A	a	b	c	i	f	h			

Steps 6-11(for u=d)

u	v	v∈Q AND w(u, v) < Key[v]	Then Π[v]←u, Key[v]←w(u,v)
d	c	NO	Do Nothing
d	f	NO	Do Nothing
d	e	YES	Π[e]←d, Key[e]←9

After Step 6-11 (for u=d)

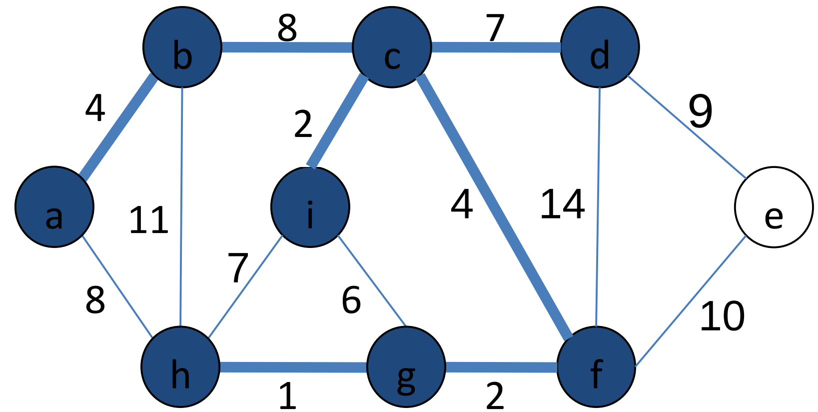
Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	8	7	9	4	2	1	2
Π[u]	NIL	a	b	c	d	c	f	g	c

Q	e								
A	a	b	c	i	f	g	h	d	

PRIM's Algorithm (Steps 6 to 11, for u=d)

6	**while** Q is not Empty
7	**do** u←EXTRACT-MIN(Q)
8	**for** each v∈Adj[u]
9	**do if** v∈Q and w(u,v)<key[v]
10	**then** Π[v]←u
11	key[v]←w(u,v)

Example Graph



Status of Q before using u=d

Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	8	7	10	4	2	7	2
Π[u]	NIL	a	b	c	f	c	f	i	c

Q	d	e							
A	a	b	c	i	f	h			

Steps 6-11(for u=d)

u	v	v∈Q AND w(u, v) < Key[v]	Then Π[v]←u, Key[v]←w(u,v)
d	c	NO	Do Nothing
d	f	NO	Do Nothing
d	e	YES	Π[e]←d, Key[e]←9

After Step 6-11 (for u=d)

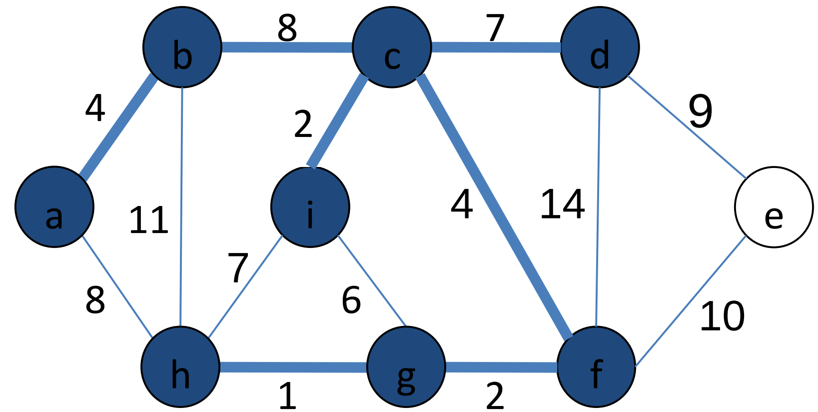
Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	8	7	9	4	2	1	2
Π[u]	NIL	a	b	c	d	c	f	g	c

Q	e								
A	a	b	c	i	f	g	h	d	

PRIM's Algorithm (Steps 6 to 11, for u=e)

6	**while** Q is not Empty
7	**do** u←EXTRACT-MIN(Q)
8	**for** each v∈Adj[u]
9	**do if** v∈Q and w(u,v)<key[v]
10	**then** Π[v]←u
11	key[v]←w(u,v)

Example Graph



Status of Q before using u=e

Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	8	7	9	4	2	1	2
Π[u]	NIL	a	b	c	d	c	f	g	c

Q	e								
A	a	b	c	i	f	g	h	d	

Steps 6-11(for u=e)

u	v	v∈Q AND w(u, v) < Key[v]	Then Π[v]←u, Key[v]←w(u,v)
e	d	NO	Do Nothing
e	f	NO	Do Nothing

After Step 6-11 (for u=e)

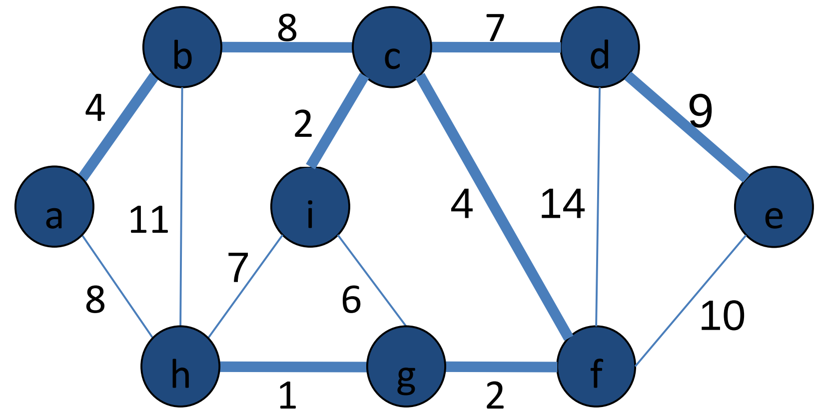
Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	8	7	9	4	2	1	2
Π[u]	NIL	a	b	c	d	c	f	g	c

Q									
A	a	b	c	i	f	g	h	d	e

PRIM's Algorithm (Steps 6 to 11, for u=e)

6	**while** Q is not Empty
7	**do** u←EXTRACT-MIN(Q)
8	**for** each v∈Adj[u]
9	**do if** v∈Q and w(u,v)<key[v]
10	**then** Π[v]←u
11	key[v]←w(u,v)

Example Graph



Status of Q before using u=e

Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	8	7	9	4	2	1	2
Π[u]	NIL	a	b	c	d	c	f	g	c

Q	e								
A	a	b	c	i	f	g	h	d	

Steps 6-11(for u=e)

u	v	v∈Q AND w(u, v) < Key[v]	Then Π[v]←u, Key[v]←w(u,v)
e	d	NO	Do Nothing
e	f	NO	Do Nothing

After Step 6-11 (for u=e)

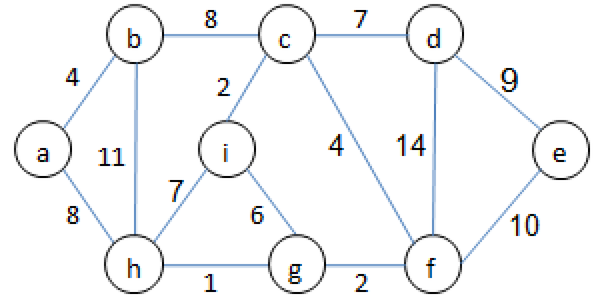
Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	8	7	9	4	2	1	2
Π[u]	NIL	a	b	c	d	c	f	g	c

Q									
A	a	b	c	i	f	g	h	d	e

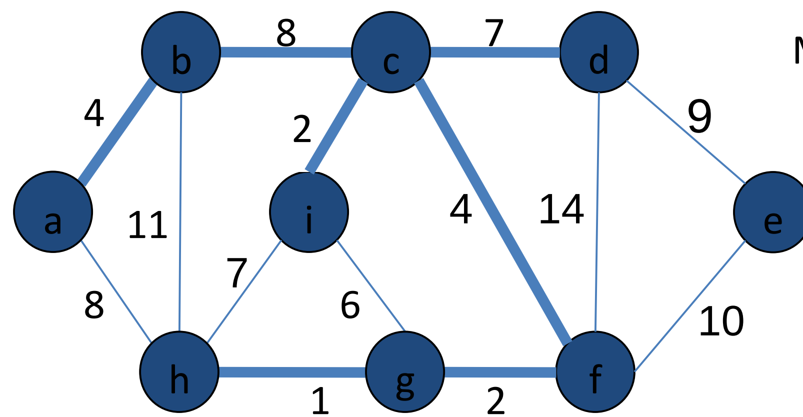
PRIM's Algorithm (Steps 6 to 11, for u=e)

MST-PRIM(G, w, r)

1	for each u∈V[G]
2	do key[u]←∞
3	Π[u]←NIL
4	key[r]←0
5	Q←V[G]
6	**while** Q is not Empty
7	**do** u←EXTRACT-MIN(Q)
8	**for** each v∈Adj[u]
9	**do if** v ∈ Q and w(u ,v) < key[v]
10	**then** Π[v]←u
11	key[v]←w(u, v)



Example Graph



MST

Q	a	b	c	d	e	f	g	h	i
key[u]	0	4	8	7	9	4	2	1	2
Π[u]	NIL	a	b	c	d	c	f	g	c

Q									
A	a	b	c	i	f	g	h	d	e

Complexity Analysis of 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 is not Empty
          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)
```

Total time: $O(V \lg V + E \lg V) = O(E \lg V)$

O(V), if Q is implemented as min-heap.

Body of while loop is executed |V| times.

Takes O(lg V) times.

Takes O(V lg V) times.

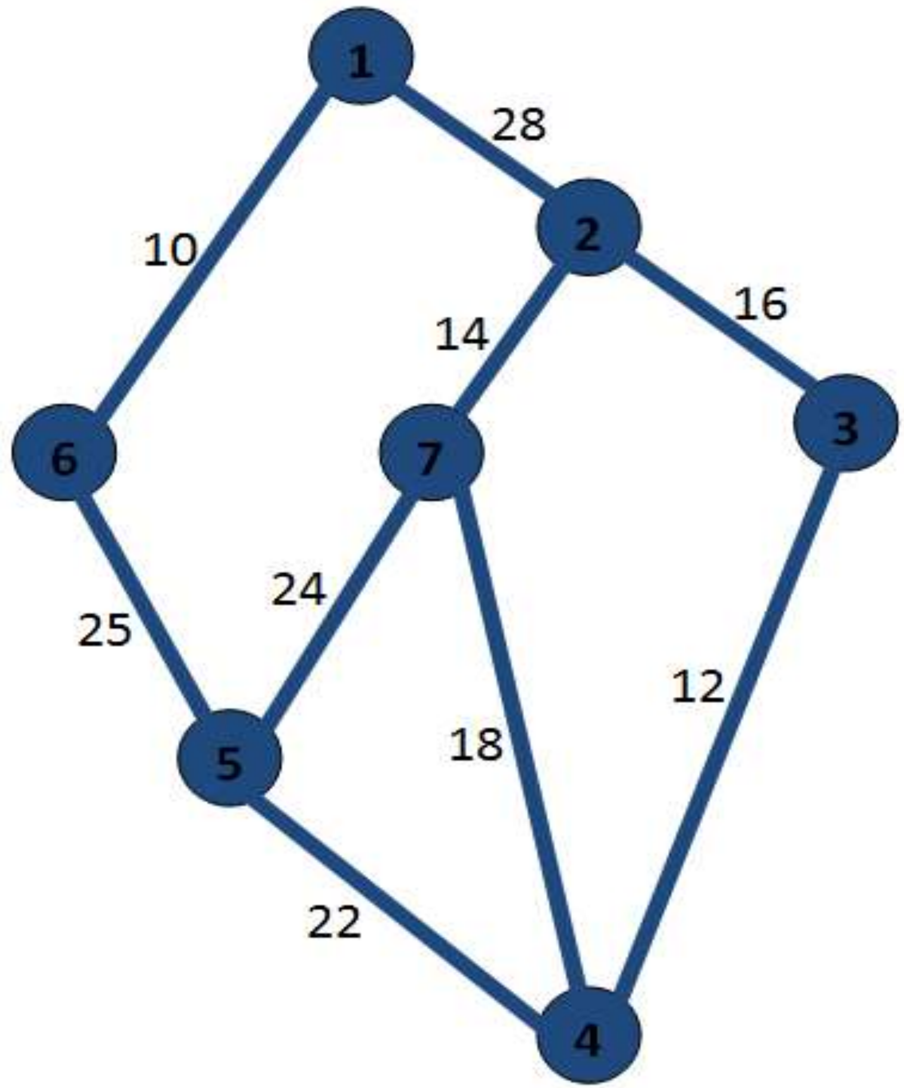
Executed O(E) times total.

Constant

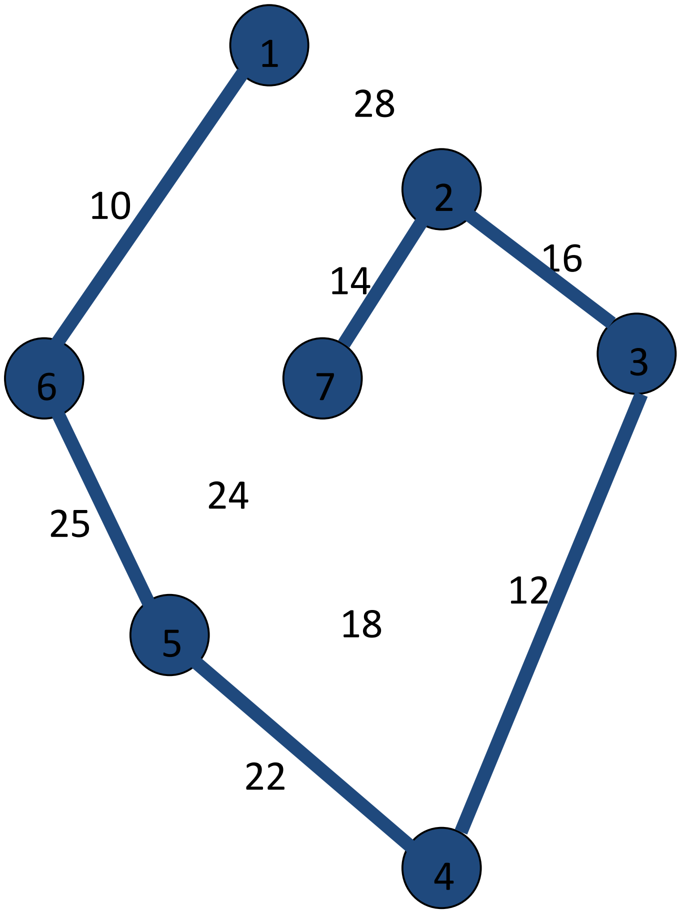
O(E lg V)

Takes O(lg V) times.

Example 2.



Example 2 Continue…..



Applications

- **Design of a network**

(telephone network, computer network, electronic circuitry, electrical wiring network, water distribution network, cable TV network)

- A less obvious application is that the minimum spanning tree can be used to approximately solve the travelling salesman problem.

- Finding airline routes.

- To create high quality mazes

- Routing algorithms

- Study of molecular bonds in Chemistry

- Cartography

- Geometry

- Clustering

- Tour/Travel Management

Latest research papers based on Prim's Algorithm

Exploring the parallel implementations of the three classical MST algorithms
(N. R. Latha; G. Shyamala; G. R. Prasad 2017 International Conference on Inventive Communication and Computational Technologies (ICICCT))
Joint reconfiguration of feeders and allocation of capacitor banks in distribution systems using a multi-start strategy
(Márcio M. Montsutsumi; Jose N. Melchor; Leonardo H. Macedo; Rubén Romero, 2017 IEEE PES Innovative Smart Grid Technologies Conference - Latin America (ISGT Latin America))
Distributed minimum spanning tree based information exchange policy for distributed systems
(Taj Alam; Zahid Raza, 2016 Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC))
Generating spanning tree of non-regular graphic sequences through a variant of Prim's algorithm
(Prantik Biswas; Abhisek Paul; Paritosh Bhattacharya, 2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015])
The transmission time analysis of IPTV multicast service in SDN/OpenFlow environments
(Pornnipa Rattanawadee; Natchaphon Ruengsakulrach; Chaiyachet Saivichit, 2015 12th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON))
Optimization of the Connection Topology of an Offshore Wind Farm Network
(Ouahid Dahmani; Salvy Bourguet; Mohamed Machmoum; Patrick Guérin; Pauline Rhein; Lionel Jossé, IEEE Systems Journal)
Multiagent-Based Distribution Automation Solution for Self-Healing Grids
(Markus Eriksson; Mikel Armendariz; Oleg O. Vasilenko; Arshad Saleem; Lars Nordström, IEEE Transactions on Industrial Electronics)
Cost-minimum network planning in large wind farm using revised prim's algorithm
Ichiro Kousaka; Daisuke Eguchi; Daiki Yamashita; Yosuke Nakanishi; Ruichi Yokoyama; Kenji Iba ISGT 2014
Prime Object Proposals with Randomized Prim's Algorithm
Santiago Manen; Matthieu Guillaumin; Luc Van Gool, 2013 IEEE International Conference on Computer Vision
Prim's algorithm based P2MP energy-saving routing design for MiDORi
Akiko Hirao; Yuki Nomura; Haruka Yonezu; Hidetoshi Takeshita; Daisuke Ishii; Satoru Okamoto; Naoaki Yamanaka, The 10th International Conference on Optical Internet (COIN2012)

References

- Coremen, Leiserson, Rivest and Stein: Introduction to algorthms, PHI

- Horowitz, Sahni and Rajsekaran: Fundamentals of Computer Algorithms, Galgotia.

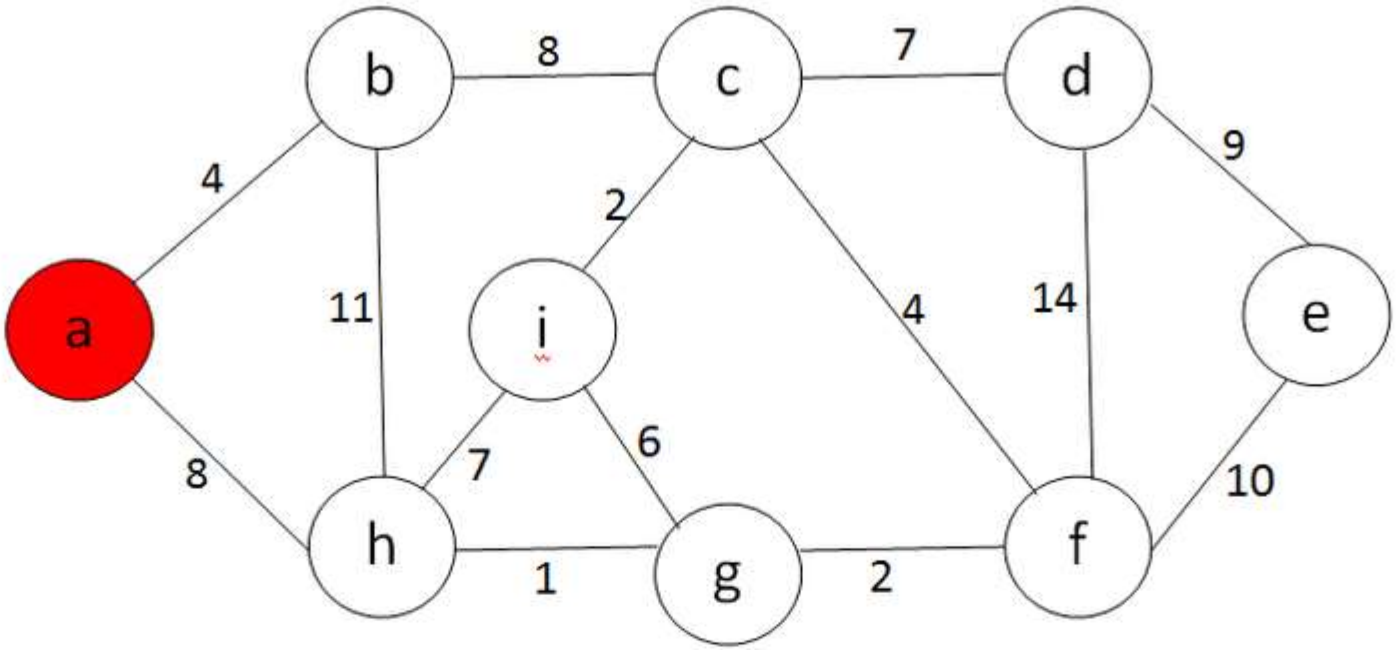
- www.mathworld.wolfram.com

- IEEE xplore

Summary

Prim's algorithm is a greedy algorithm, and is a special case of generic minimum-spanning-tree algorithm and operates much like Dijkstra's algorithm, that finds a minimum spanning tree for a weighted undirected graph and is mainly used for a dense graph i.e. a graph with lots of edges.

Question?



(a)

Thanks!