INT102
Algorithmic Foundations and Problem Solving
Greedy Methods

Dr Pengfei Fan
Department of Intelligent Science

西交利物浦大學
Xi'an Jiaotong-Liverpool University

Greedy methods …

Textbook:

Chapter 9, INTRODUCTION TO THE DESIGN AND ANALYSIS OF ALGORITHMS A. V. LEVITIN

Learning outcomes

- Understand what greedy method is
- Able to apply Prim's algorithm to find minimum spanning tree
- Able to apply Kruskal's algorithm to find minimum spanning tree
- Able to apply Dijkstra's algorithm to find single-source shortest-paths

Learning outcomes

- Understand what greedy method is
- Able to apply Prim's algorithm to find minimum spanning tree
- Able to apply Kruskal's algorithm to find minimum spanning tree
- Able to apply Dijkstra's algorithm to find single-source shortest-paths

Greedy methods

- How to be greedy?
  - At every step, make the best move you can make.
  - Keep going until you're done
- Advantages
  - Don't need to pay much effort at each step.
  - Usually finds a solution very quickly.
  - The solution found is usually not bad.
- Possible problem
  - The solution found may NOT be the best one

Design of Greedy Algorithms

- Cast the optimization problem as one in which we make a choice and remain one subproblem to solve.
- Demonstrate that, having make the greedy choice, what remains is a subproblem with the property that <u>if we combine an optimal solution to the subproblem with the greedy choice we have made, we arrive at an optimal solution to the original problem.</u>

Greedy methods - examples

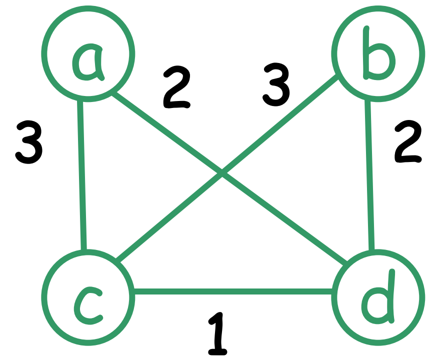
- Minimum spanning tree
  - Prim's algorithm
  - Kruskal's algorithm
- Single-source shortest-paths
  - Dijkstra's algorithm
- All above-mentioned algorithms find (one of) the BEST solution

Minimum Spanning tree (MST)

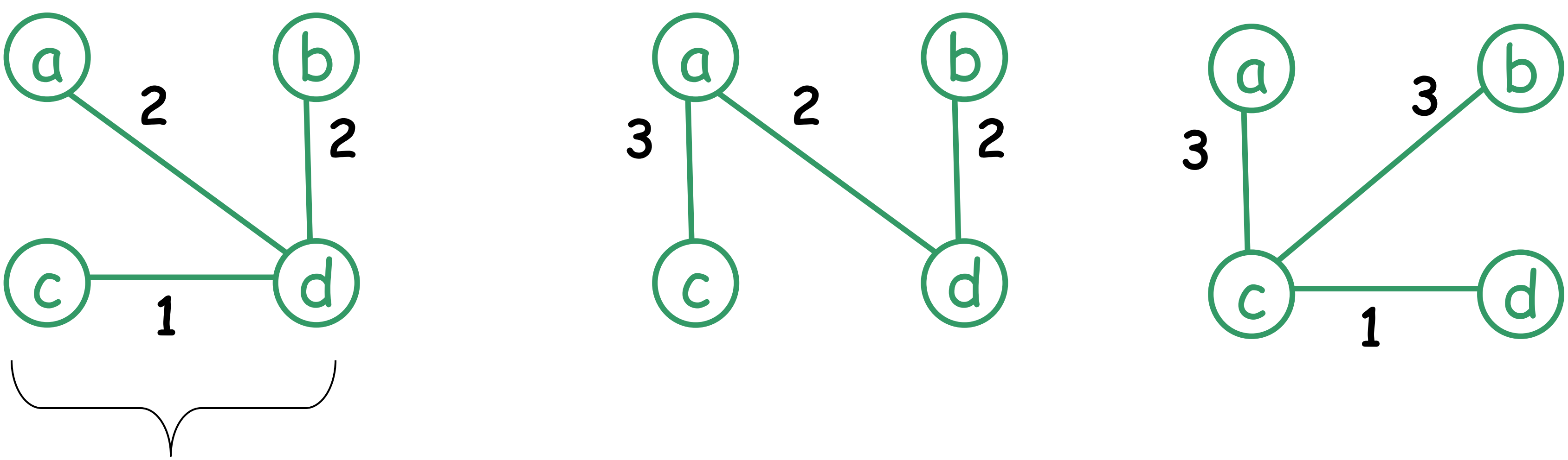
- Given an undirected connected graph *G*
  - The edges are labelled by weight
- Spanning tree of *G*
  - a tree containing all vertices in *G*
- Minimum spanning tree
  - a spanning tree of *G* with minimum weight

Examples

Graph G
(edge label is weight)



Spanning trees of G

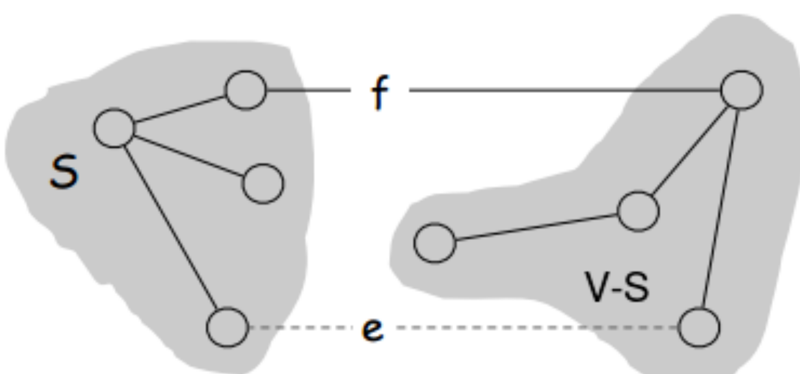


MST

A Theorem

Cut Property

Let G=(V,E) be a connected, undirected graph with real-valued weights on edges. Let *A* be a subset of E that forms a minimum spanning tree of *G*, and let S and V-S be two disjoint subsets of V. Let e be the minimum weight edge that has one endpoint in S and the other endpoint in V-S. Then, e is part of the minimum spanning tree *A* of *G*.



Learning outcomes of this lecture

– Understand what greedy method is
– Able to apply Prim's and Kruskal's algorithms to find minimum spanning tree

Prim's algorithm …

Reminder - MST

- Given an undirected connected graph *G*
  - The edges are labelled by weight
- Spanning tree of *G*
  - a tree containing all vertices in *G*
- Minimum spanning tree
  - a spanning tree of *G* with minimum weight

Pseudo code

// Given a weighted connected graph $G=(V,E)$

pick a vertex v_0 in V

$V_T = \{ v_0 \}$

$E_T = \varnothing$

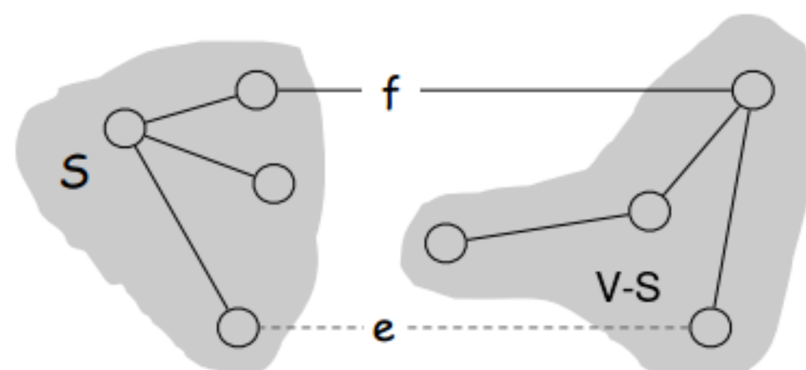
For i=1 to |V|-1 do

pick an edge $e = (v^*, u^*)$ with minimum weight among all the edges (v, u) such that v is in V_T and u is in $V - V_T$

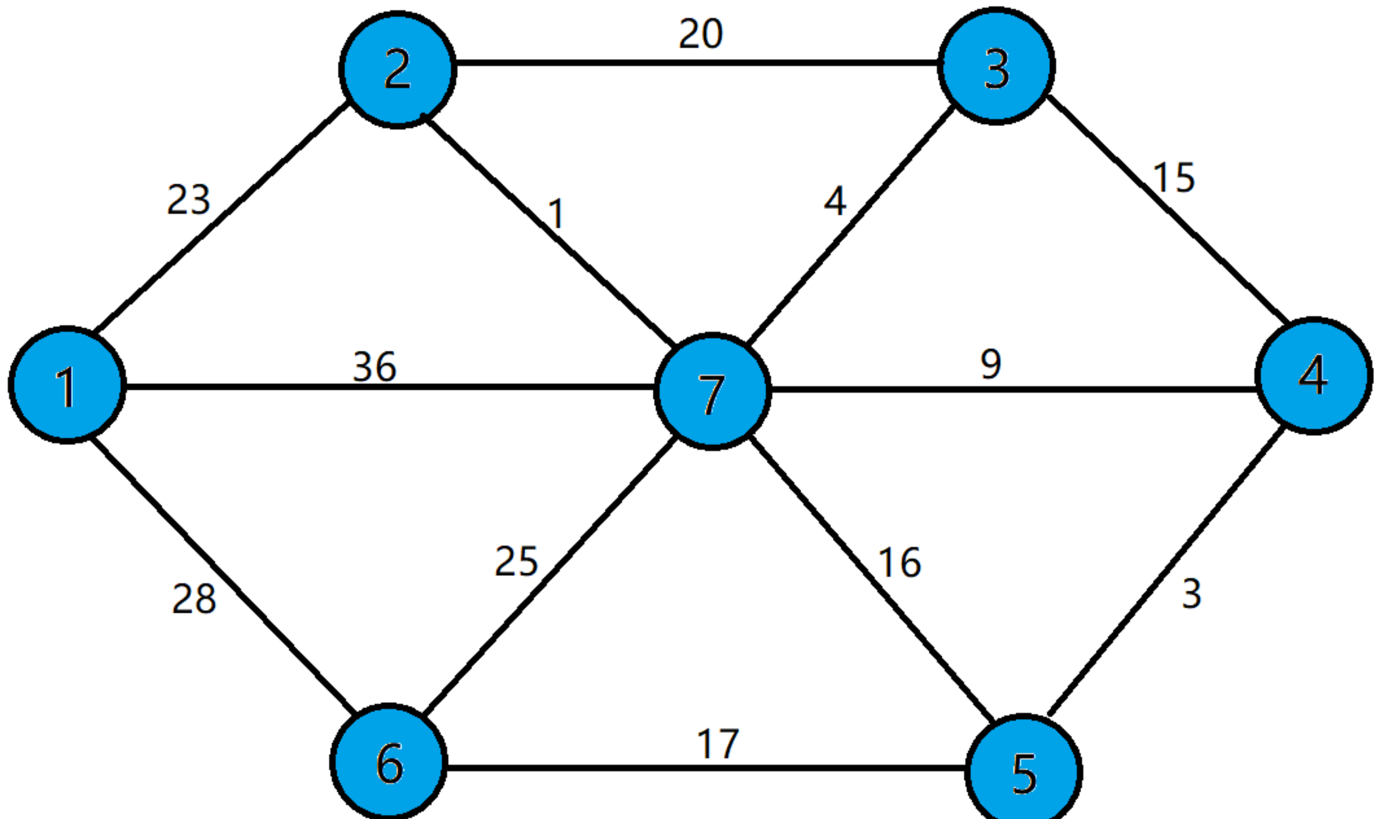
$V_T = V_T \cup \{ u^* \}$

$E_T = E_T \cup \{ e^* \}$

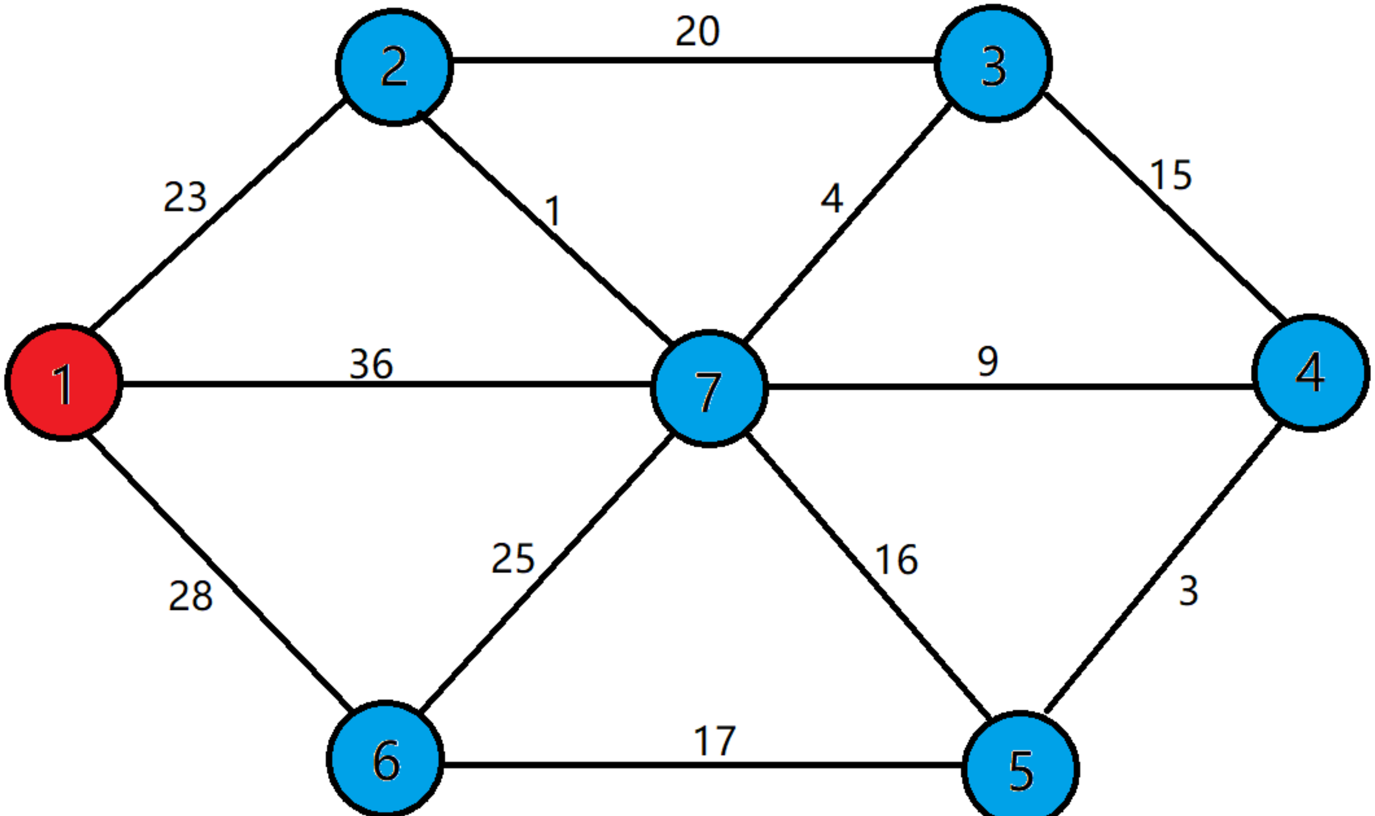
Return E_T



Example of Prim's Algorithm

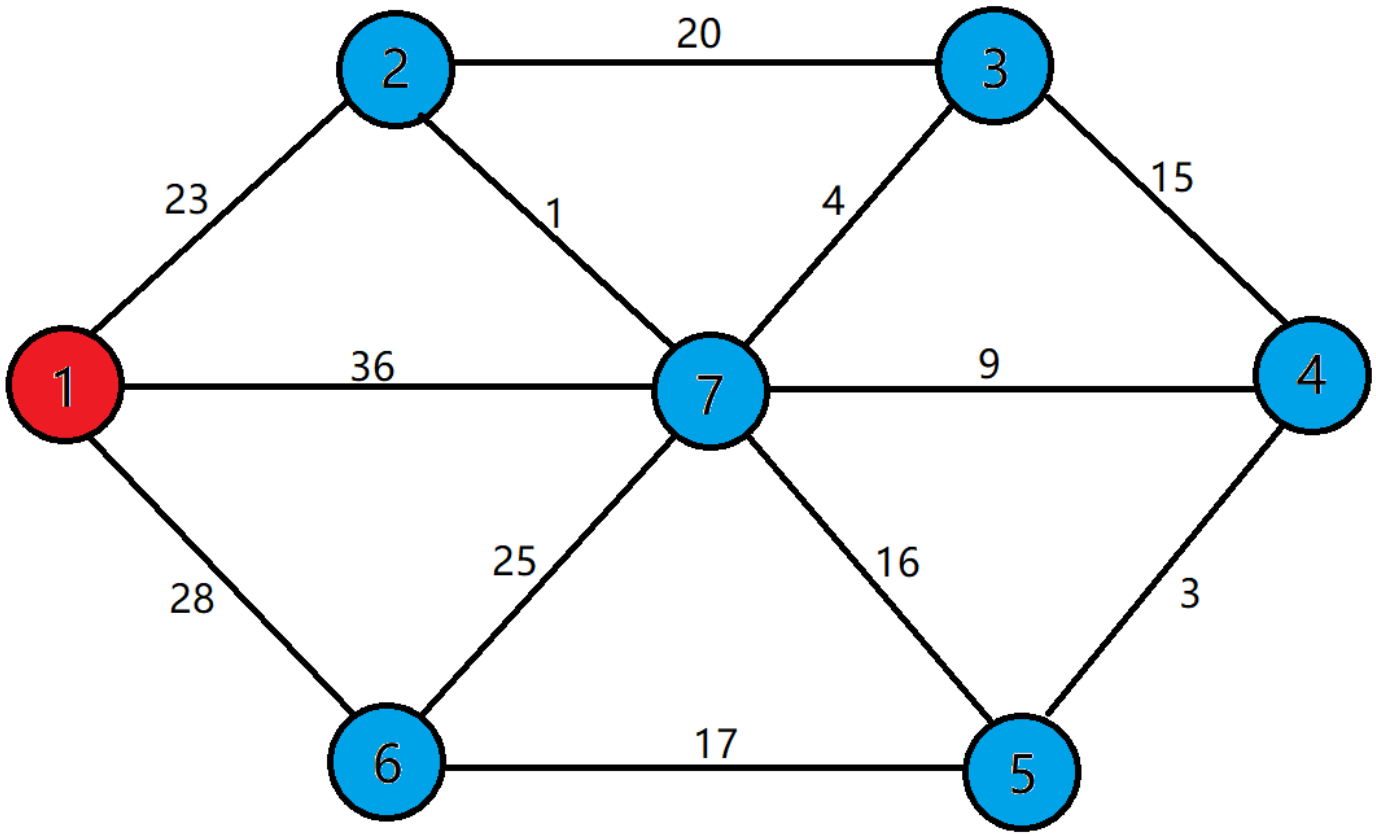


Example of Prim's Algorithm



dist	0	∞	∞	∞	∞	∞	∞
	1	2	3	4	5	6	7

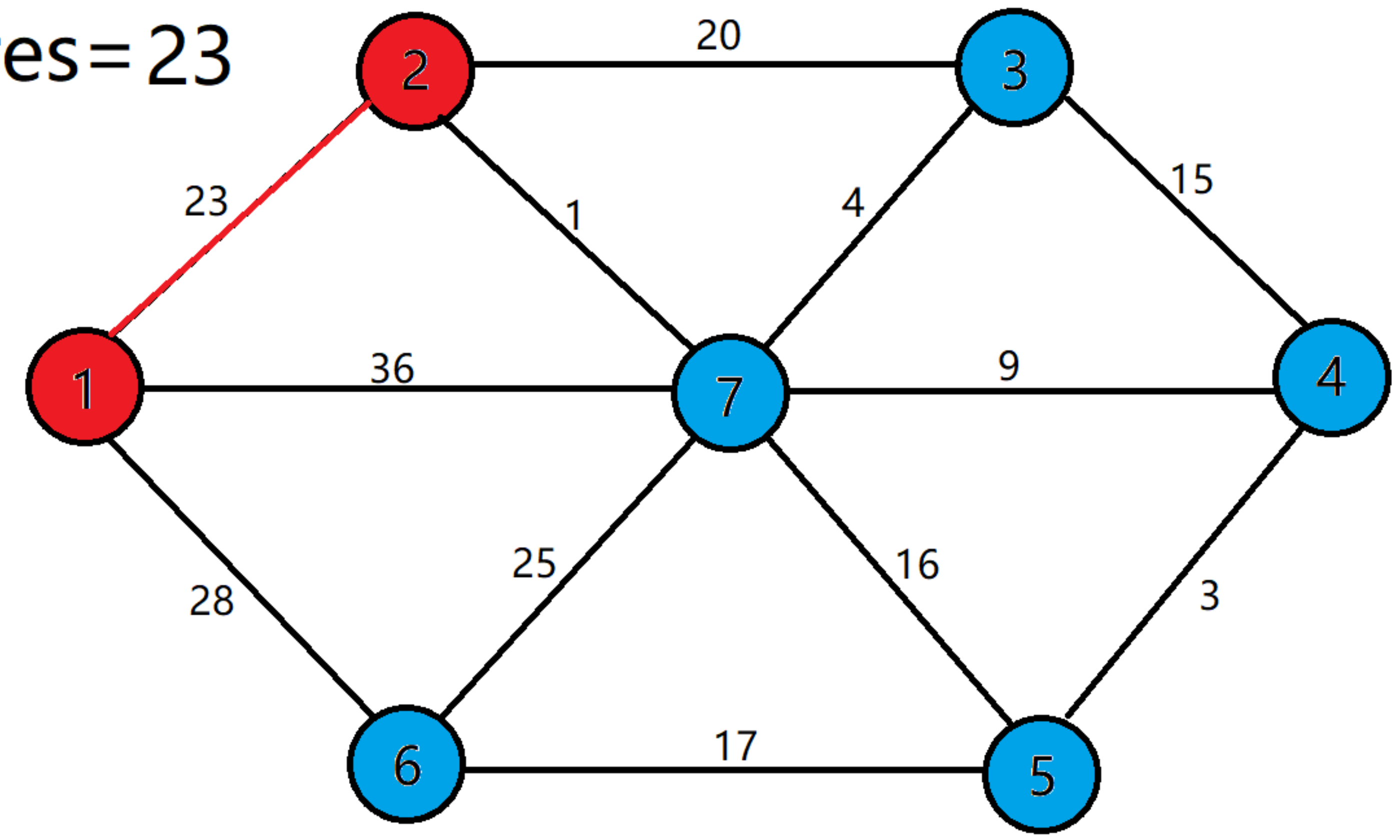
Example of Prim’s Algorithm



dist	0	23	∞	∞	∞	28	36
	1	2	3	4	5	6	7

Example of Prim’s Algorithm

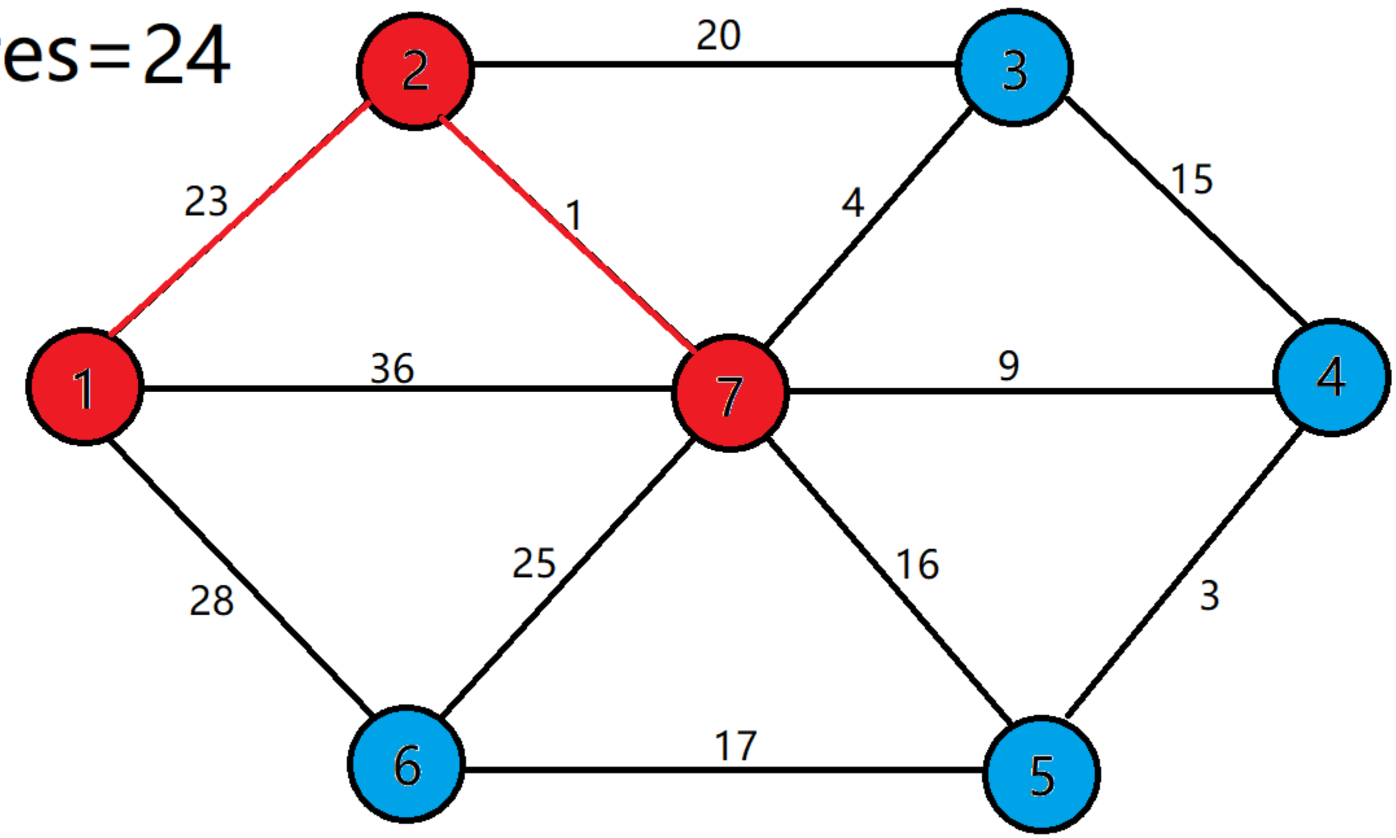
res=23



dist							
	0	23	20	∞	∞	28	1
	1	2	3	4	5	6	7

Example of Prim's Algorithm

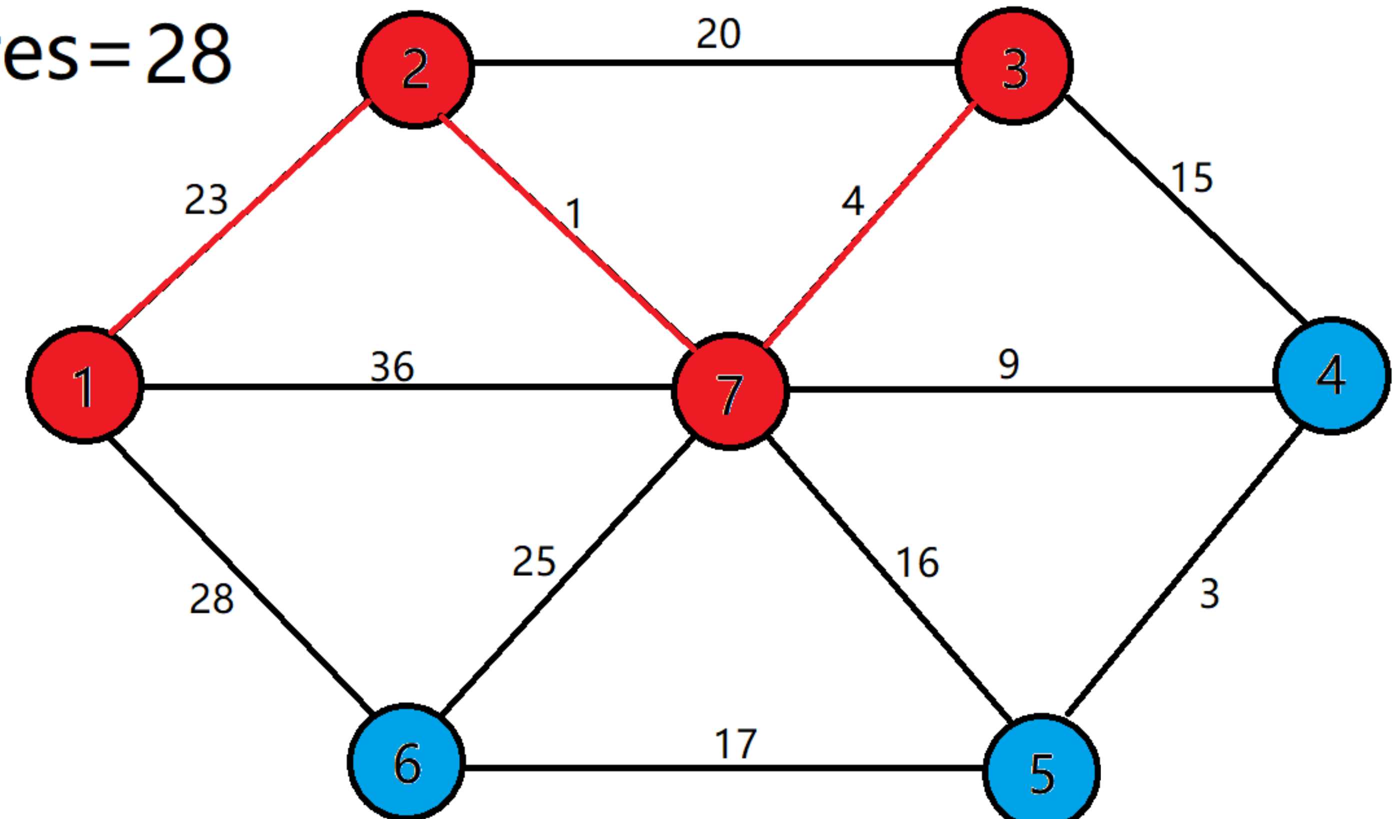
res=24



dist							
	0	23	4	9	16	25	1
	1	2	3	4	5	6	7

Example of Prim’s Algorithm

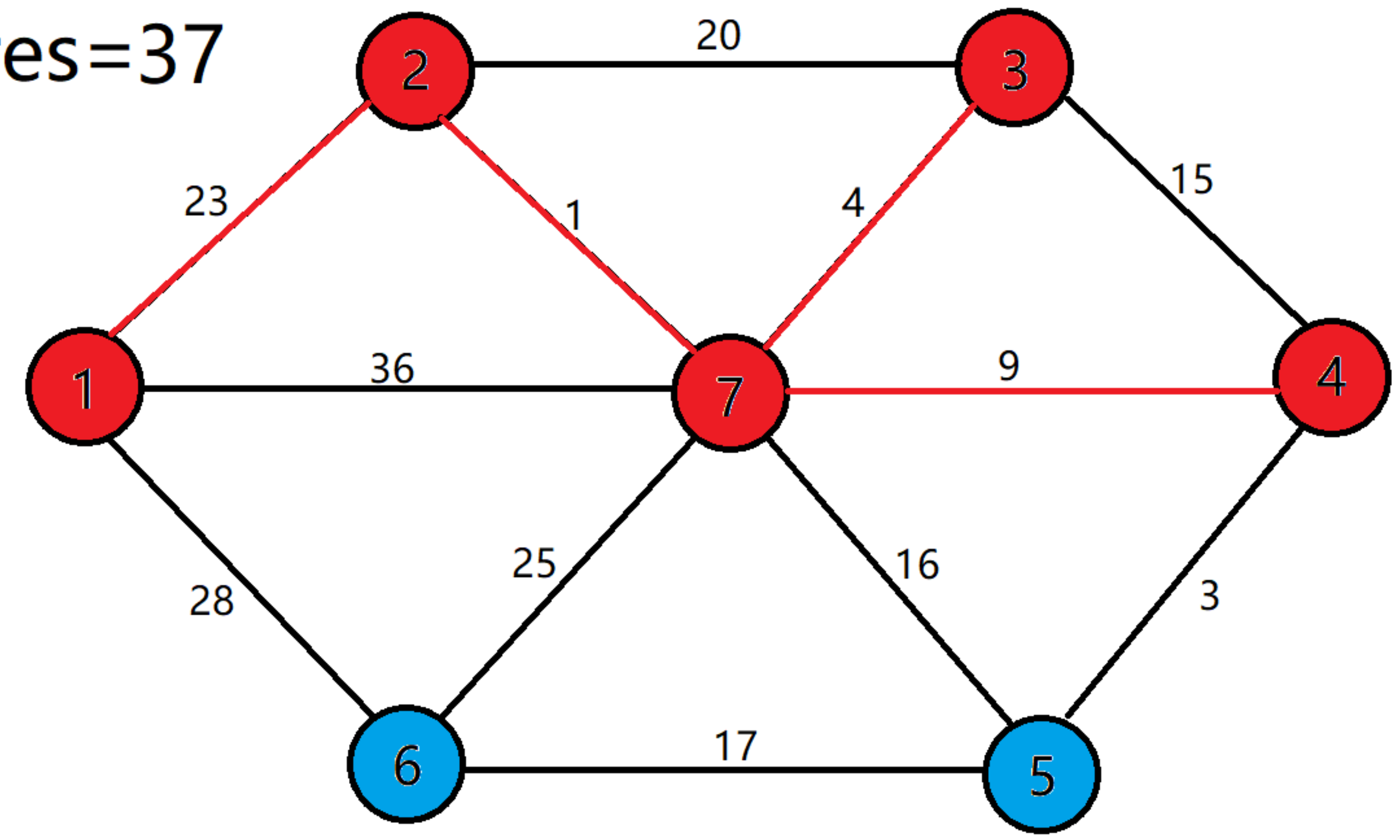
res=28



dist	0	23	4	9	16	25	1
	1	2	3	4	5	6	7

Example of Prim's Algorithm

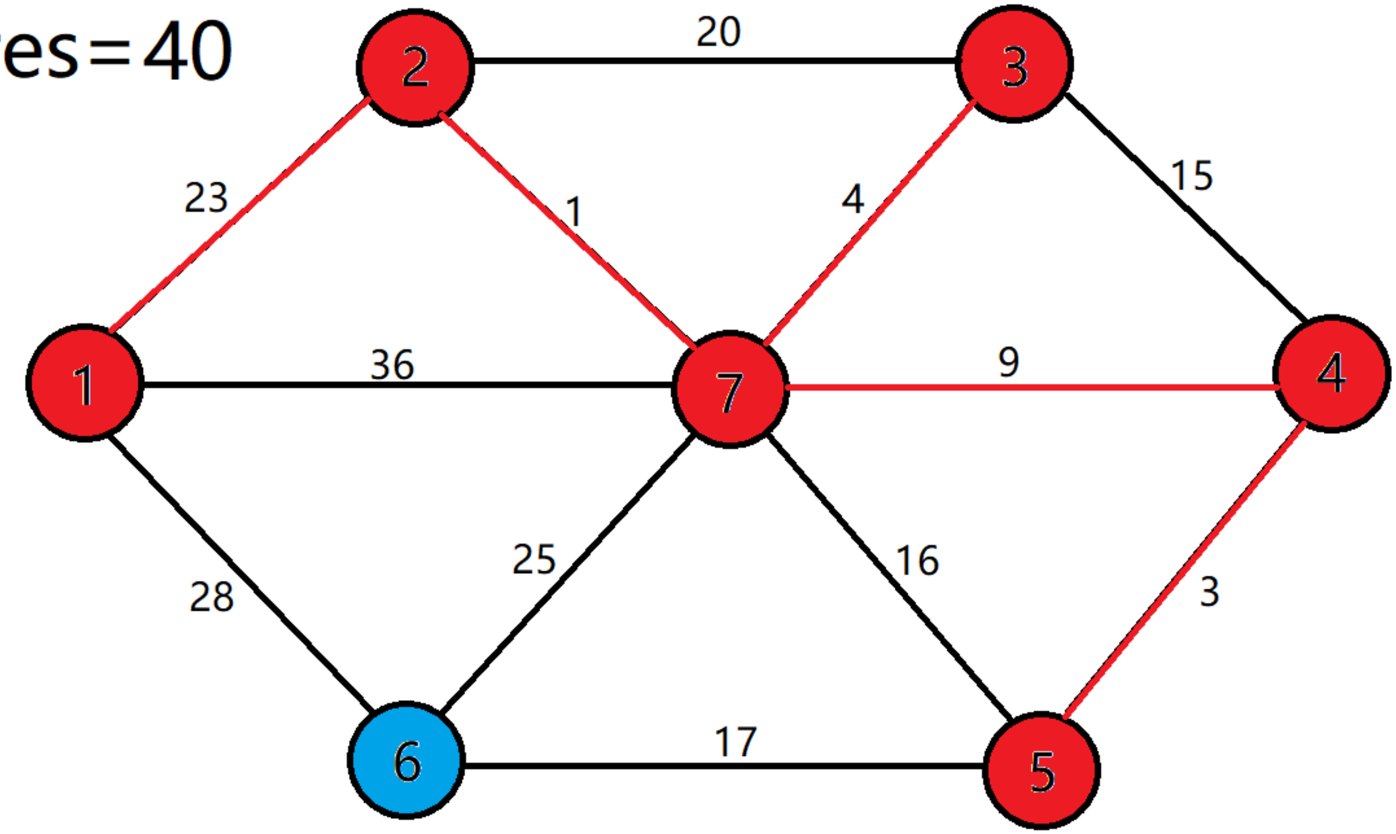
res=37



dist	0	23	4	9	3	25	1
	1	2	3	4	5	6	7

Example of Prim's Algorithm

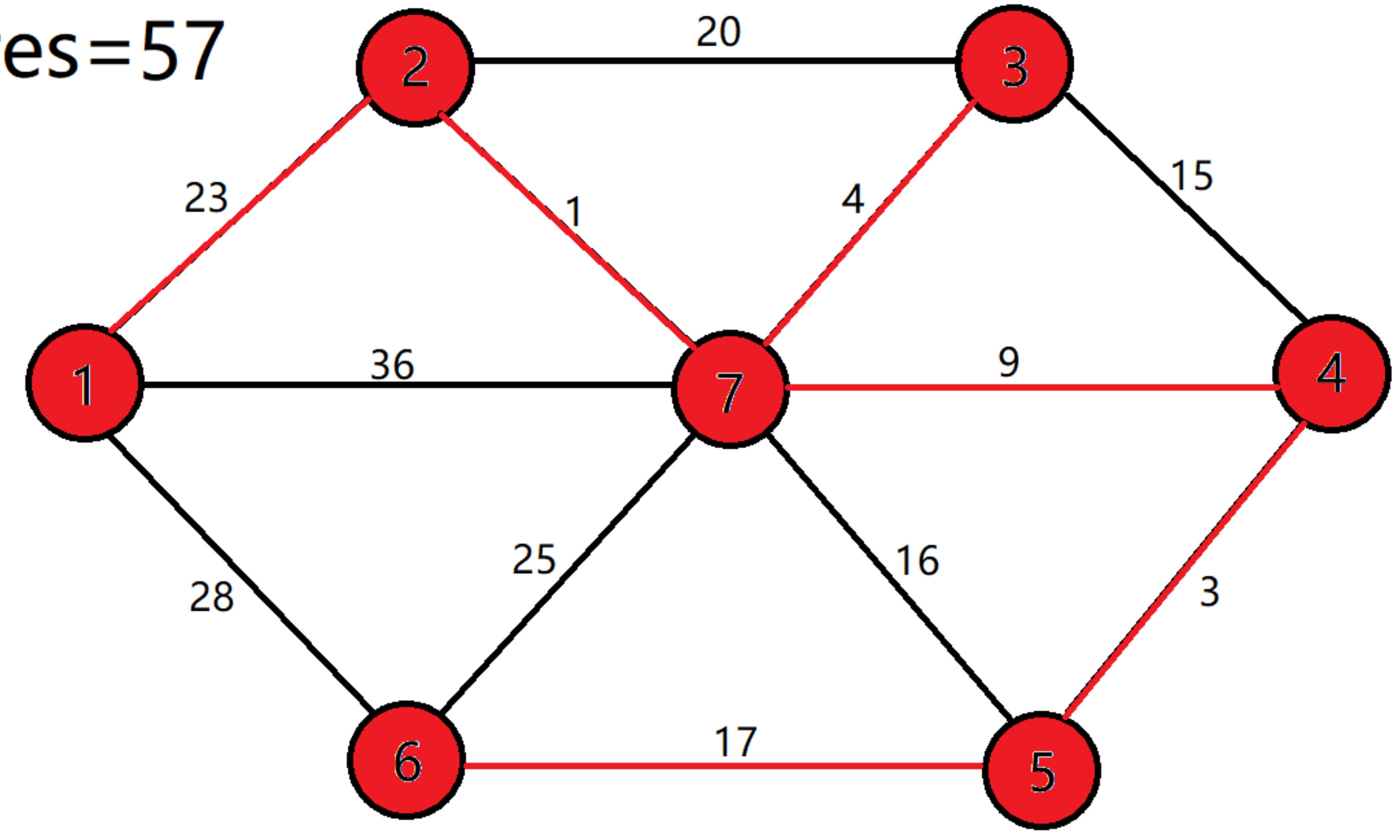
res=40



	1	2	3	4	5	6	7
dist	0	23	4	9	3	17	1

Example of Prim's Algorithm

res=57



Questions:

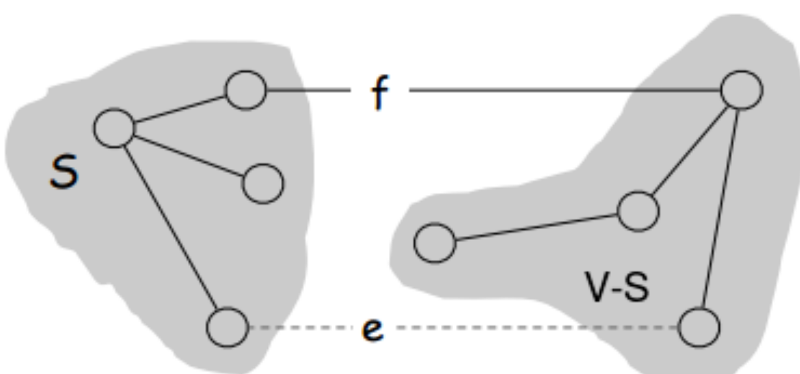
- Correctness: Does Prim's algorithm always yield a minimum spanning tree?
- Complexity: what is the time complexity of Prim's algorithm?

Correctness of Prim's Algorithm

Observation (loop invariant):

Prior to each loop, E_T is a subset of a minimum spanning tree.

This Can be proved by using the cut property.



Complexity of Prim's Algorithm (optional)

Given $G=(V, E)$ and if the following part

"pick an edge e =(v*, u*) with minimum weight among all the edges (v, u) such that v is in V_T and u is in $V-V_T$"

is implemented by min-heap (for priority queue) , then the answer is

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

Pseudo code with data structure

```
MST-PRIM(G, w, root)    //G=(V,E) and a root vertex
for each u in V
    do key[u] <- ∞
        π[u] <- NIL
key[root] <- 0
Q <- V   //implement by min-heap, O(|V|)
while Q is not empty
        do u <- Extract-min(Q)    // O(lg(|V|) by heap
            for each v in Adj[u]
            do if v in Q and w(u,v) <key[v]
                then π[v] <- u
                    key[v] = w(u,v)
```

https://www.youtube.com/watch?v=z1L3rMzG1_A&ab_channel=BoQian

Example: Question 3 in Week6 Tutorial

Worst case time complexity for heaps (optional)

- Build heap with *n* items - O(n)
- *insert*() into a heap with n items – O(lg n)
- *deleteMin*() from a heap with n items- O(lg n)
- *FindMin*() - O(1)
- Extract-min= findMin() + deleteMin() O(1)+O(lg n)

Kruskal's algorithm …

Kruskal's algorithm - MST

- Kruskal's algorithm is greedy in the sense that it always attempt to select the smallest weight edge to be included in the MST

Pseudo code

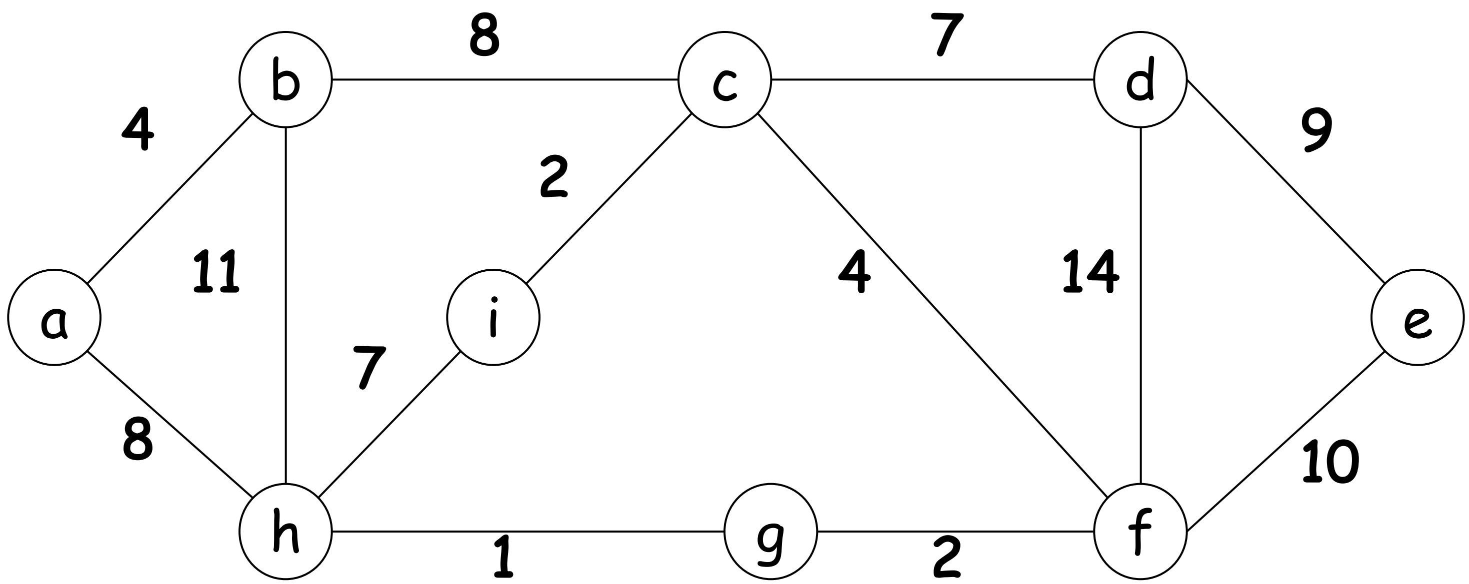
```
// Given an undirected connected graph G=(V,E)
pick an edge e in E with minimum weight
T = { e } and E' = E - { e }
while E' ≠ ∅ do
begin
    pick an edge e in E' with minimum weight     O(nm)
    if adding e to T does not form cycle then
        T = T ∪ { e }
    E' = E' - { e }
end
```

Time complexity?

Can be tested by marking vertices

Kruskal’s algorithm - MST

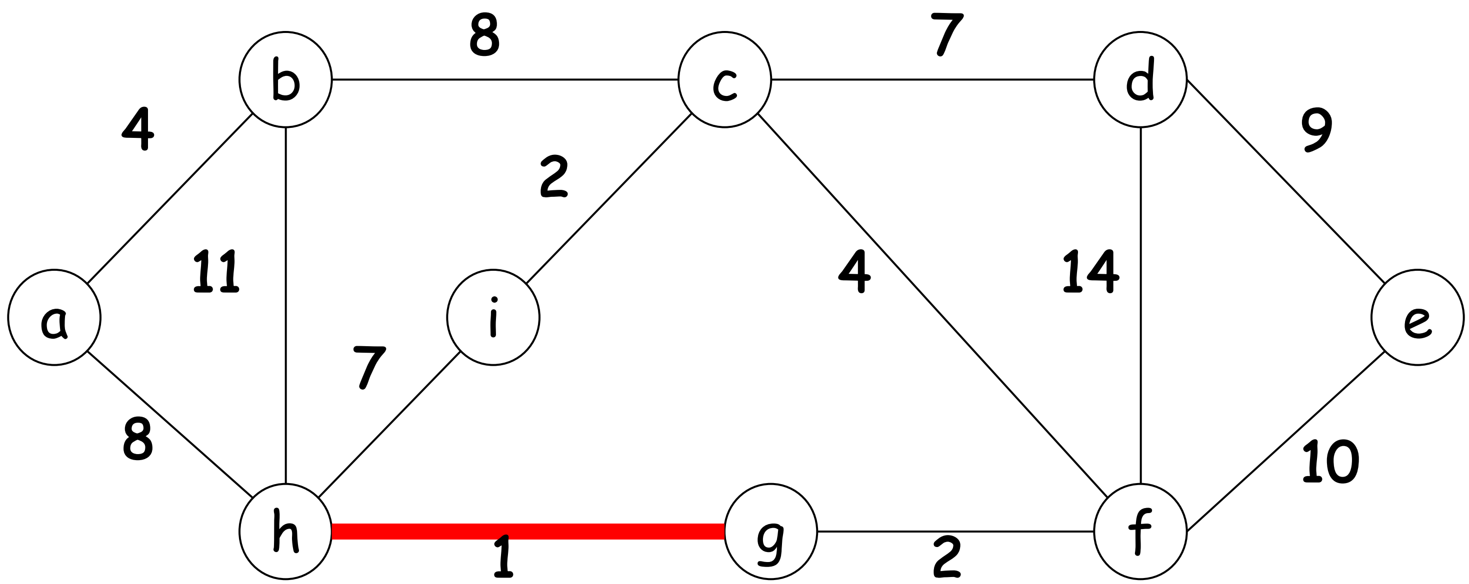
(h,g)	1
(i,c)	2
(g,f)	2
(a,b)	4
(c,f)	4
(c,d)	7
(h,i)	7
(b,c)	8
(a,h)	8
(d,e)	9
(f,e)	10
(b,h)	11
(d,f)	14



Arrange the edges from smallest to largest weight

Kruskal's algorithm - MST

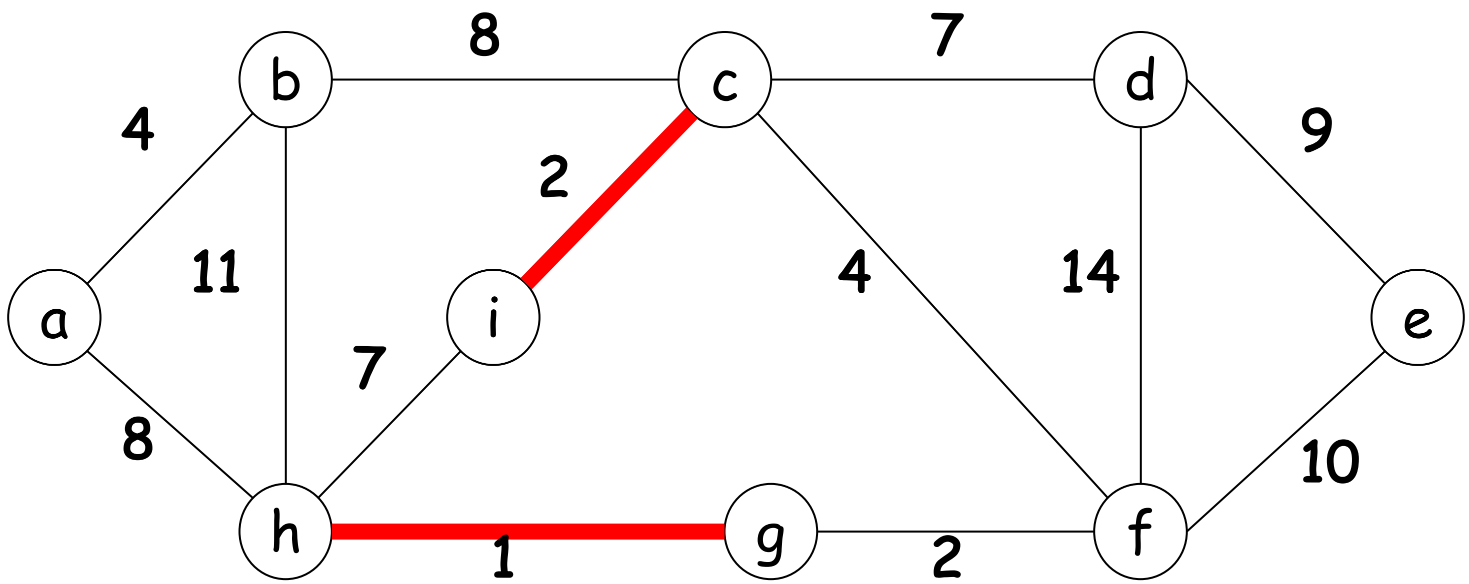
(h,g)	*1*
(i,c)	2
(g,f)	2
(a,b)	4
(c,f)	4
(c,d)	7
(h,i)	7
(b,c)	8
(a,h)	8
(d,e)	9
(f,e)	10
(b,h)	11
(d,f)	14



Choose the minimum weight edge

Kruskal's algorithm - MST

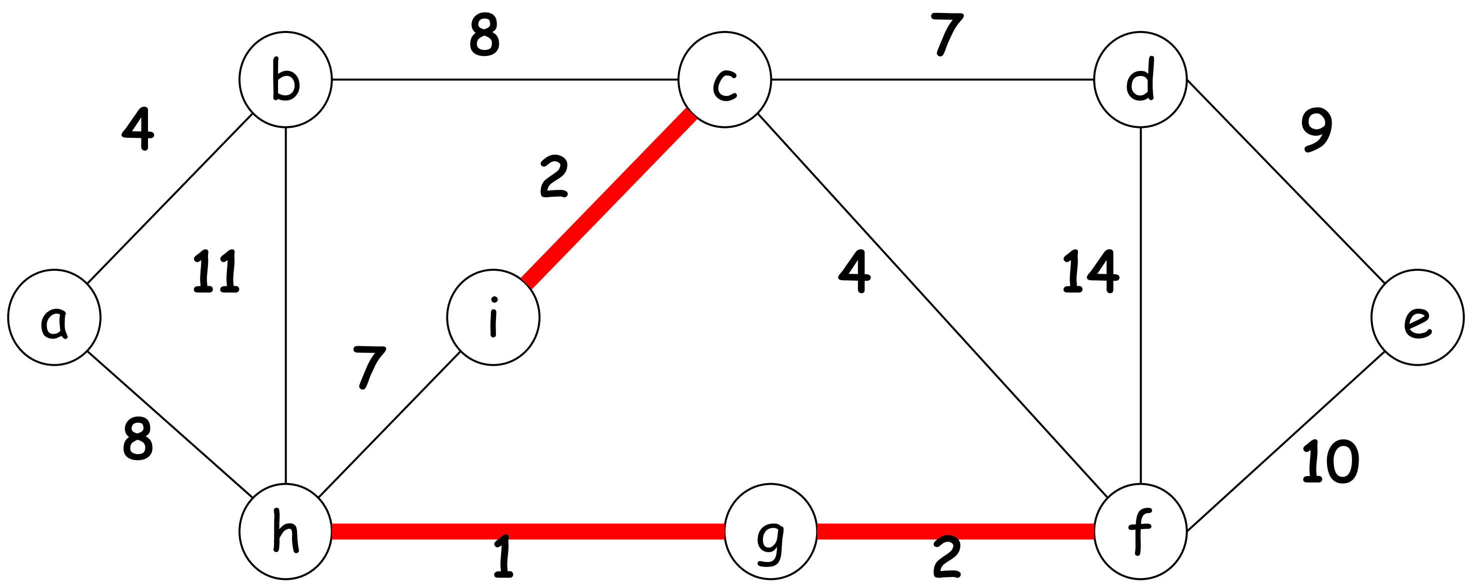
(h,g)	*1*
(i,c)	*2*
(g,f)	2
(a,b)	4
(c,f)	4
(c,d)	7
(h,i)	7
(b,c)	8
(a,h)	8
(d,e)	9
(f,e)	10
(b,h)	11
(d,f)	14



Choose the next minimum weight edge

Kruskal’s algorithm - MST

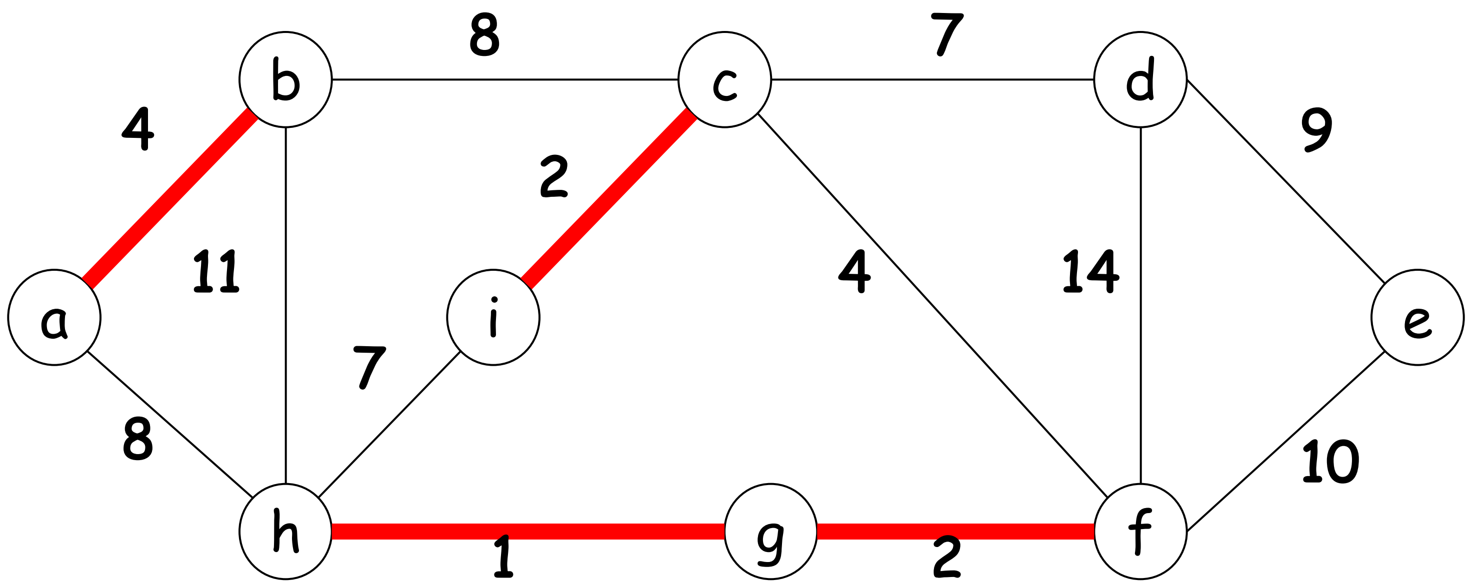
(h,g)	*1*
(i,c)	*2*
(g,f)	*2*
(a,b)	4
(c,f)	4
(c,d)	7
(h,i)	7
(b,c)	8
(a,h)	8
(d,e)	9
(f,e)	10
(b,h)	11
(d,f)	14



Continue as long as no cycle forms

Kruskal's algorithm - MST

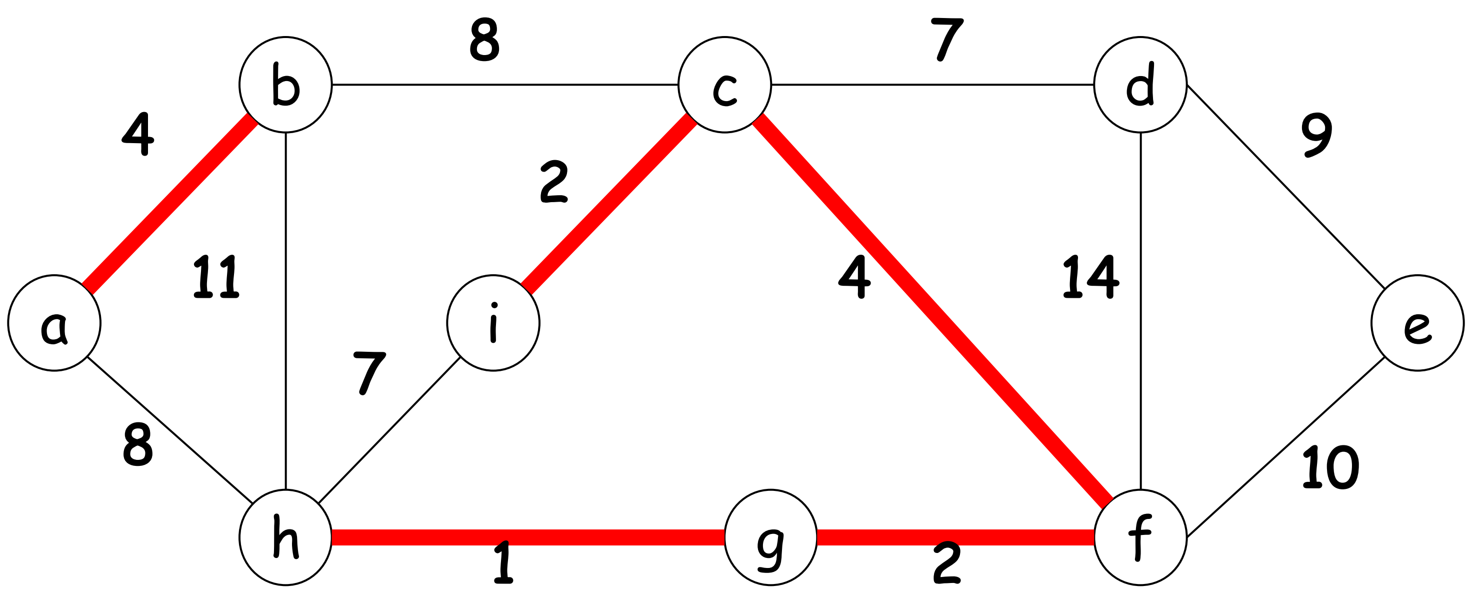
(h,g)	*1*
(i,c)	*2*
(g,f)	*2*
(a,b)	*4*
(c,f)	4
(c,d)	7
(h,i)	7
(b,c)	8
(a,h)	8
(d,e)	9
(f,e)	10
(b,h)	11
(d,f)	14



Continue as long as no cycle forms

Kruskal's algorithm - MST

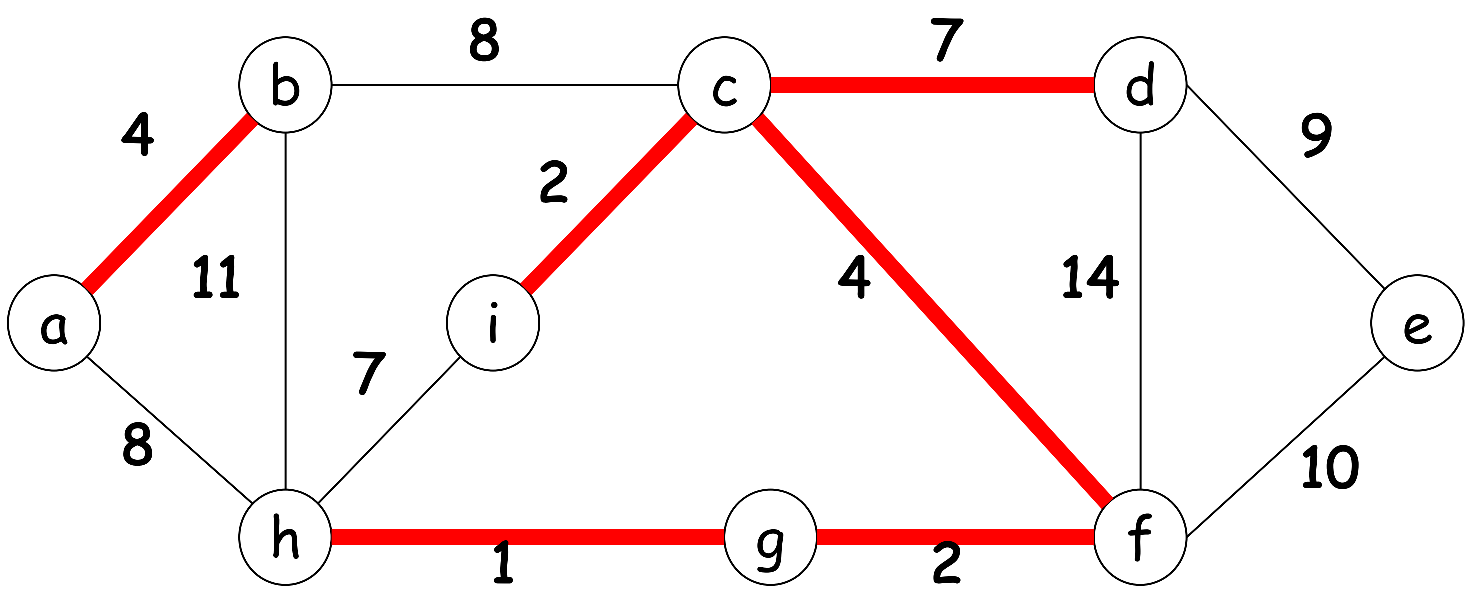
(h,g)	*1*
(i,c)	*2*
(g,f)	*2*
(a,b)	*4*
(c,f)	*4*
(c,d)	7
(h,i)	7
(b,c)	8
(a,h)	8
(d,e)	9
(f,e)	10
(b,h)	11
(d,f)	14



Continue as long as no cycle forms

Kruskal's algorithm - MST

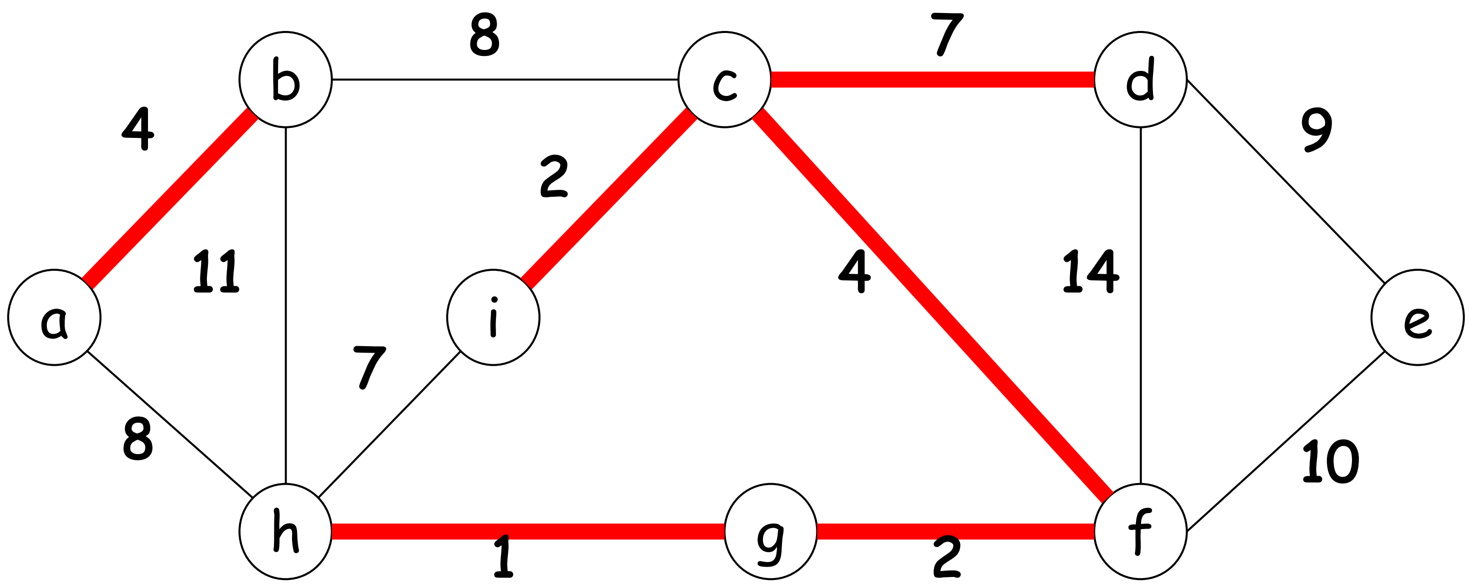
(h,g)	*1*
(i,c)	*2*
(g,f)	*2*
(a,b)	*4*
(c,f)	*4*
(c,d)	*7*
(h,i)	7
(b,c)	8
(a,h)	8
(d,e)	9
(f,e)	10
(b,h)	11
(d,f)	14



Continue as long as no cycle forms

Kruskal’s algorithm - MST

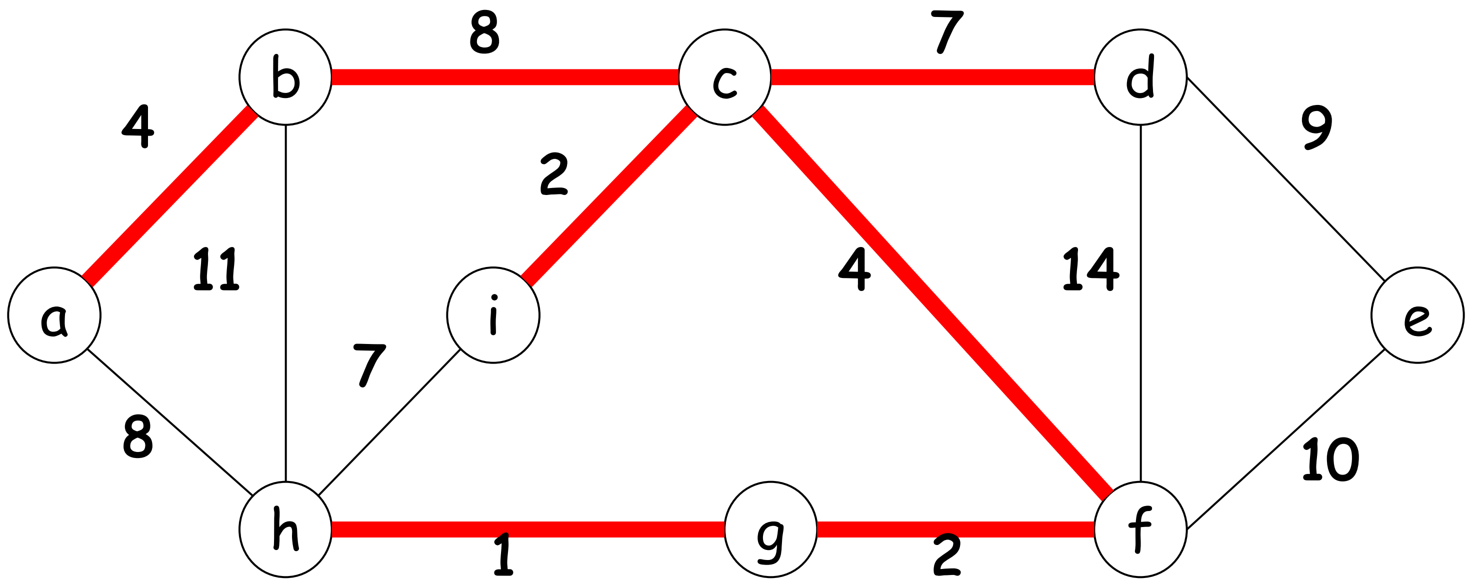
(h,g)	1
(i,c)	2
(g,f)	2
(a,b)	4
(c,f)	4
(c,d)	7
~~(h,i)~~	~~7~~
(b,c)	8
(a,h)	8
(d,e)	9
(f,e)	10
(b,h)	11
(d,f)	14



(h,i) cannot be included, otherwise, a cycle is formed

Kruskal’s algorithm - MST

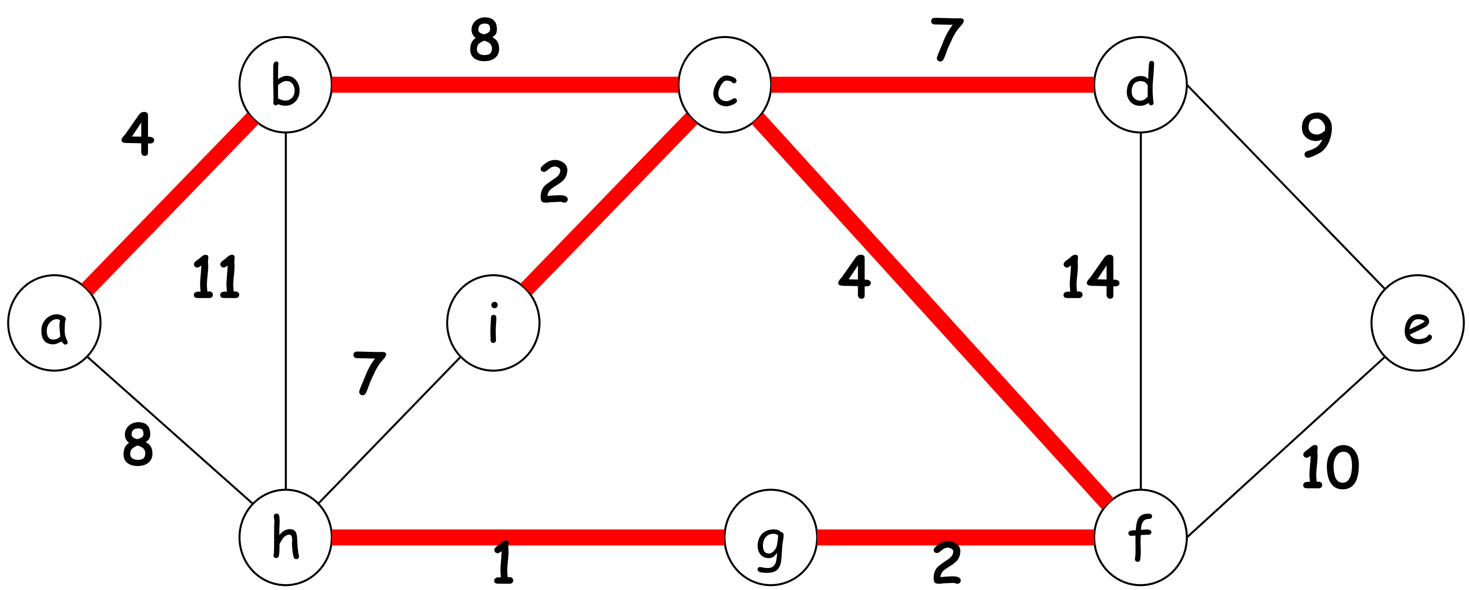
(h,g)	1
(i,c)	2
(g,f)	2
(a,b)	4
(c,f)	4
(c,d)	7
~~(h,i)~~	~~7~~
(b,c)	8
(a,h)	8
(d,e)	9
(f,e)	10
(b,h)	11
(d,f)	14



Choose the next minimum weight edge

Kruskal's algorithm - MST

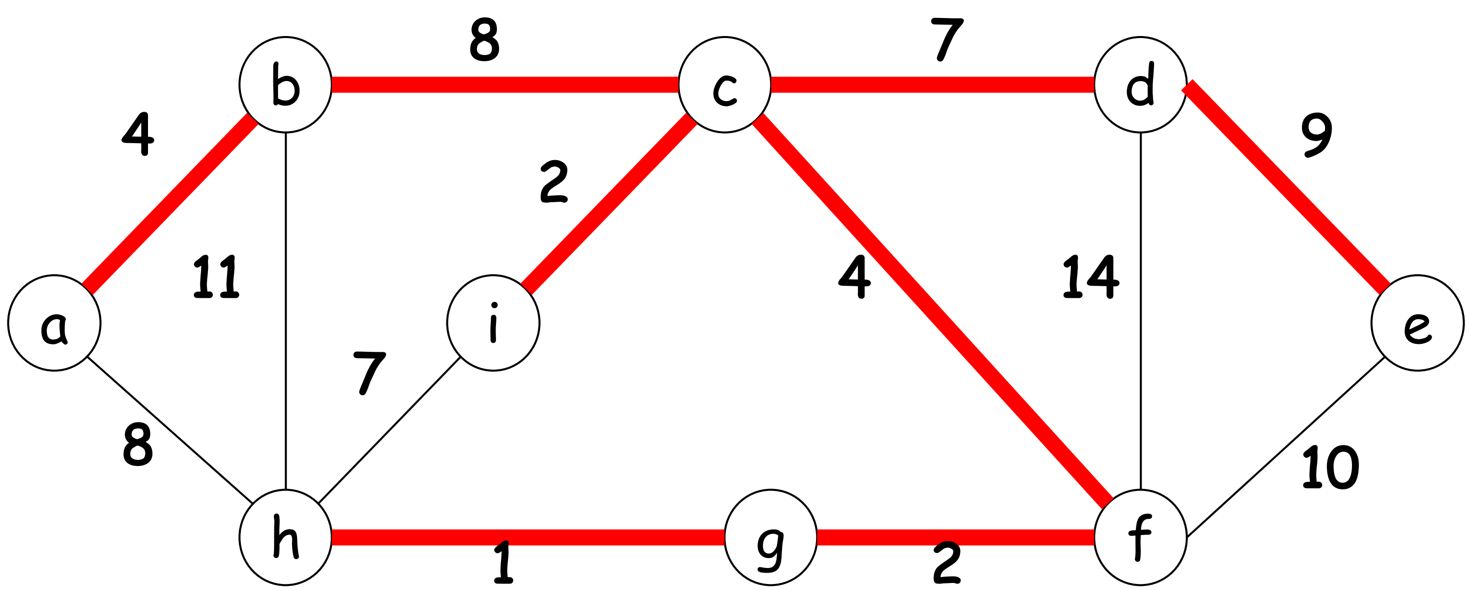
(h,g)	*1*
(i,c)	*2*
(g,f)	*2*
(a,b)	*4*
(c,f)	*4*
(c,d)	*7*
~~(h,i)~~	~~7~~
(b,c)	*8*
~~(a,h)~~	~~8~~
(d,e)	9
(f,e)	10
(b,h)	11
(d,f)	14



(a,h) cannot be included, otherwise, a cycle is formed

Kruskal's algorithm - MST

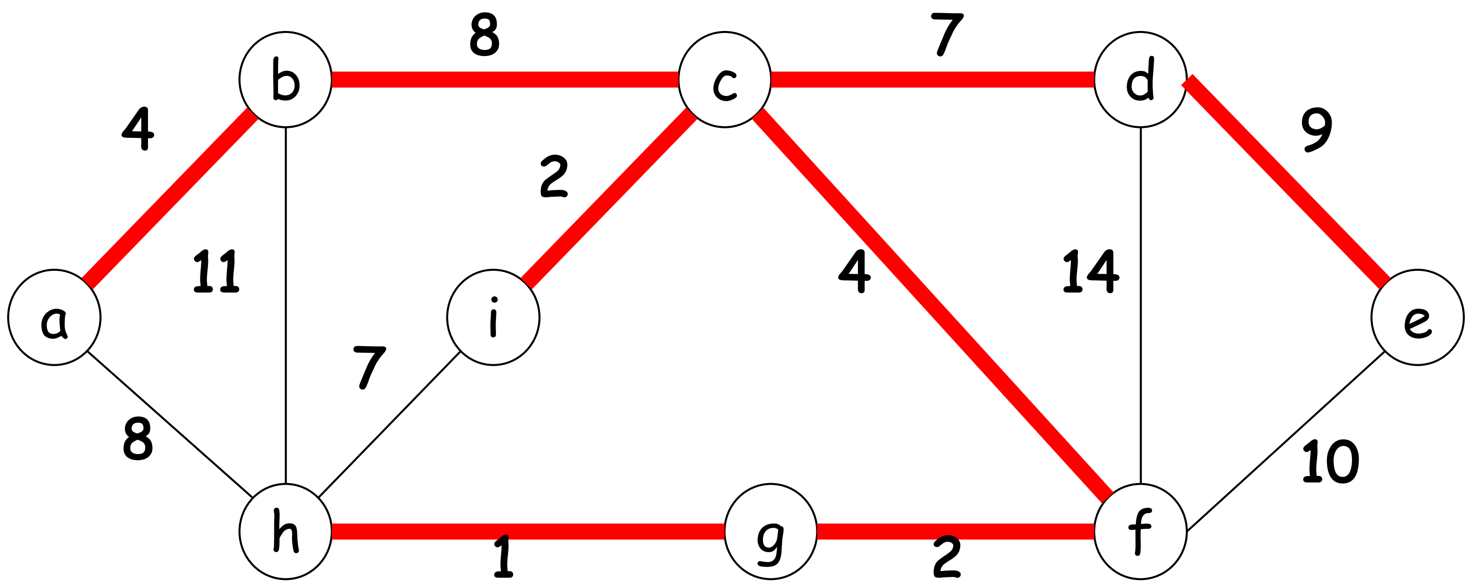
(h,g)	1
(i,c)	2
(g,f)	2
(a,b)	4
(c,f)	4
(c,d)	7
~~(h,i)~~	~~7~~
(b,c)	8
~~(a,h)~~	~~8~~
(d,e)	9
(f,e)	10
(b,h)	11
(d,f)	14



Choose the next minimum weight edge

Kruskal's algorithm - MST

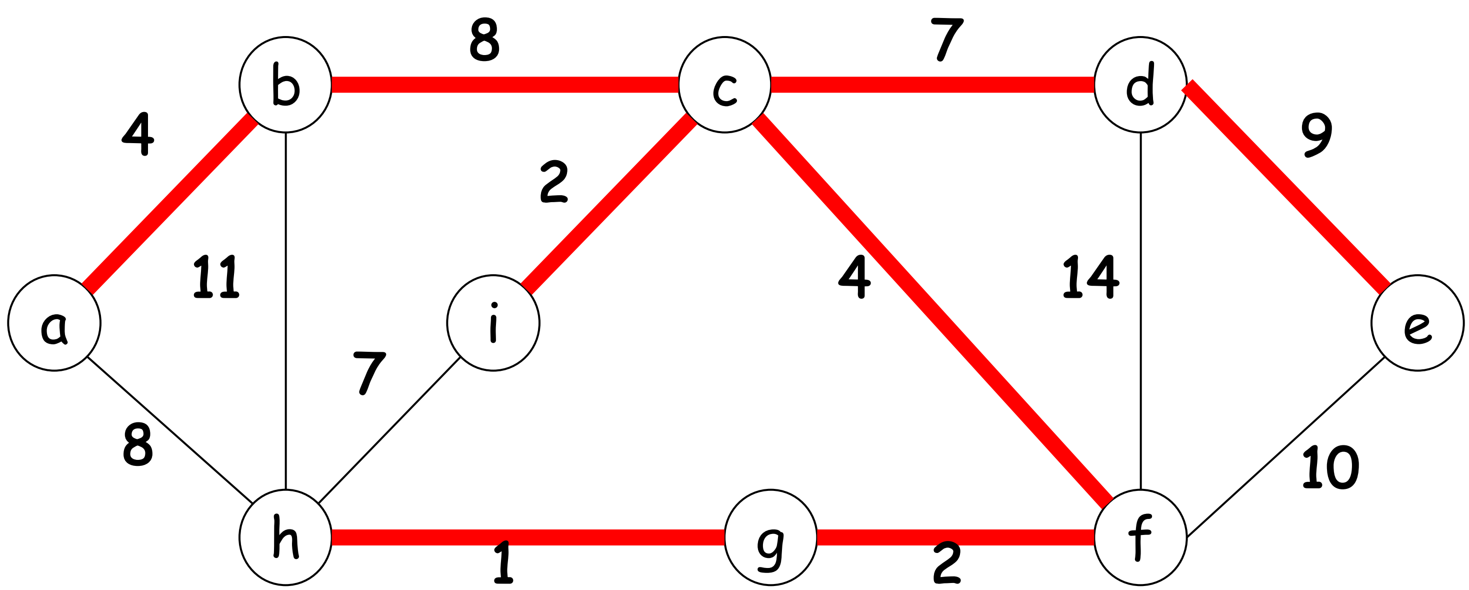
(h,g)	1
(i,c)	2
(g,f)	2
(a,b)	4
(c,f)	4
(c,d)	7
~~(h,i)~~	~~7~~
(b,c)	8
~~(a,h)~~	~~8~~
(d,e)	9
~~(f,e)~~	~~10~~
(b,h)	11
(d,f)	14



(f,e) cannot be included, otherwise, a cycle is formed

Kruskal's algorithm - MST

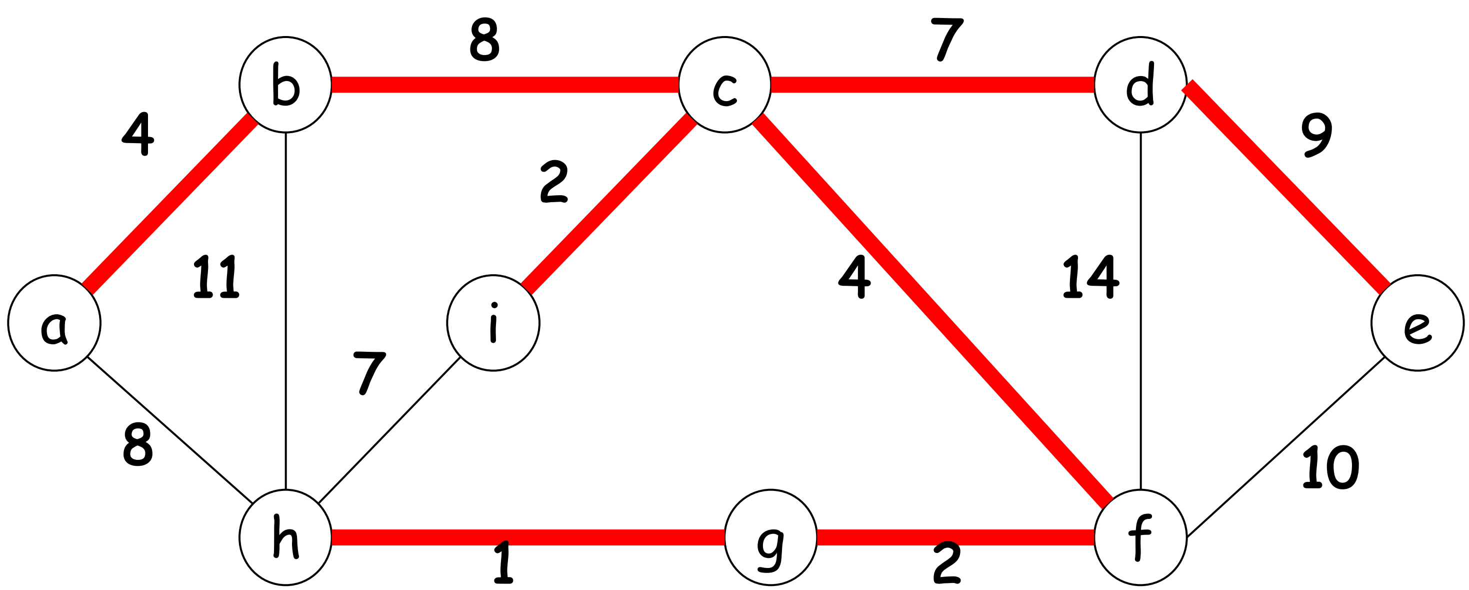
(h,g)	*1*
(i,c)	*2*
(g,f)	*2*
(a,b)	*4*
(c,f)	*4*
(c,d)	*7*
~~(h,i)~~	~~7~~
(b,c)	*8*
~~(a,h)~~	~~8~~
(d,e)	*9*
~~(f,e)~~	~~10~~
~~(b,h)~~	~~11~~
(d,f)	14



(b,h) cannot be included, otherwise, a cycle is formed

Kruskal's algorithm - MST

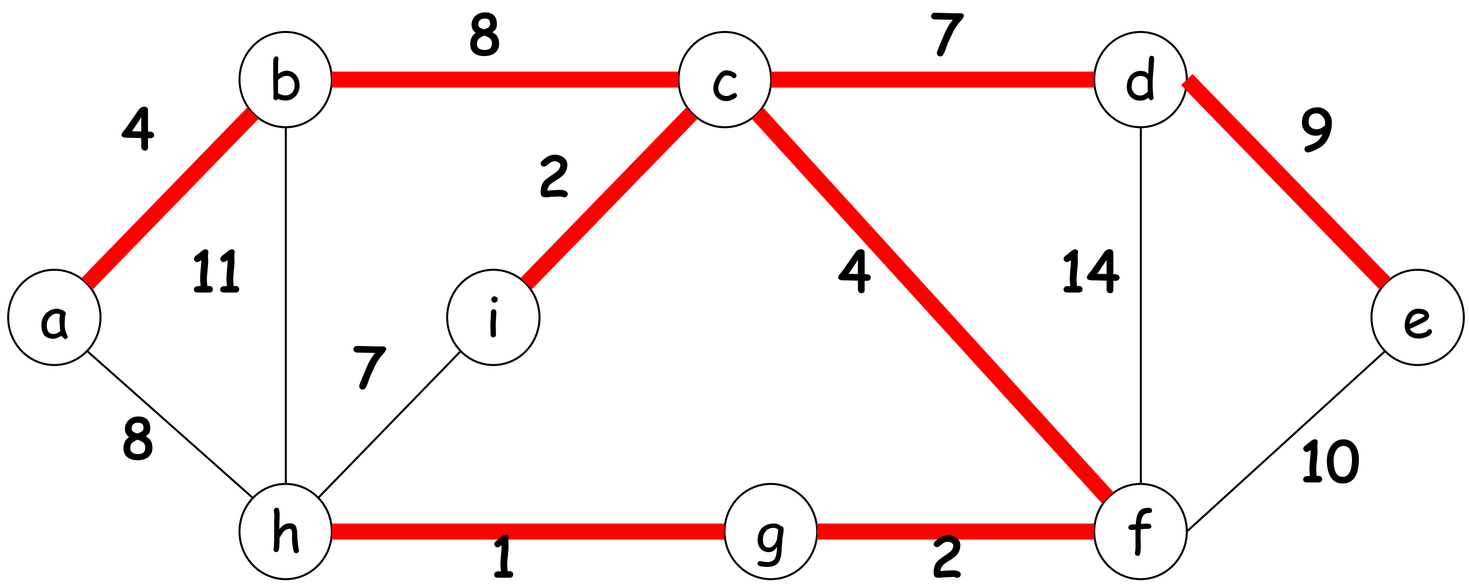
(h,g)	1
(i,c)	2
(g,f)	2
(a,b)	4
(c,f)	4
(c,d)	7
~~(h,i)~~	~~7~~
(b,c)	8
~~(a,h)~~	~~8~~
(d,e)	9
~~(f,e)~~	~~10~~
~~(b,h)~~	~~11~~
~~(d,f)~~	~~14~~



(d,f) cannot be included, otherwise, a cycle is formed

Kruskal's algorithm - MST

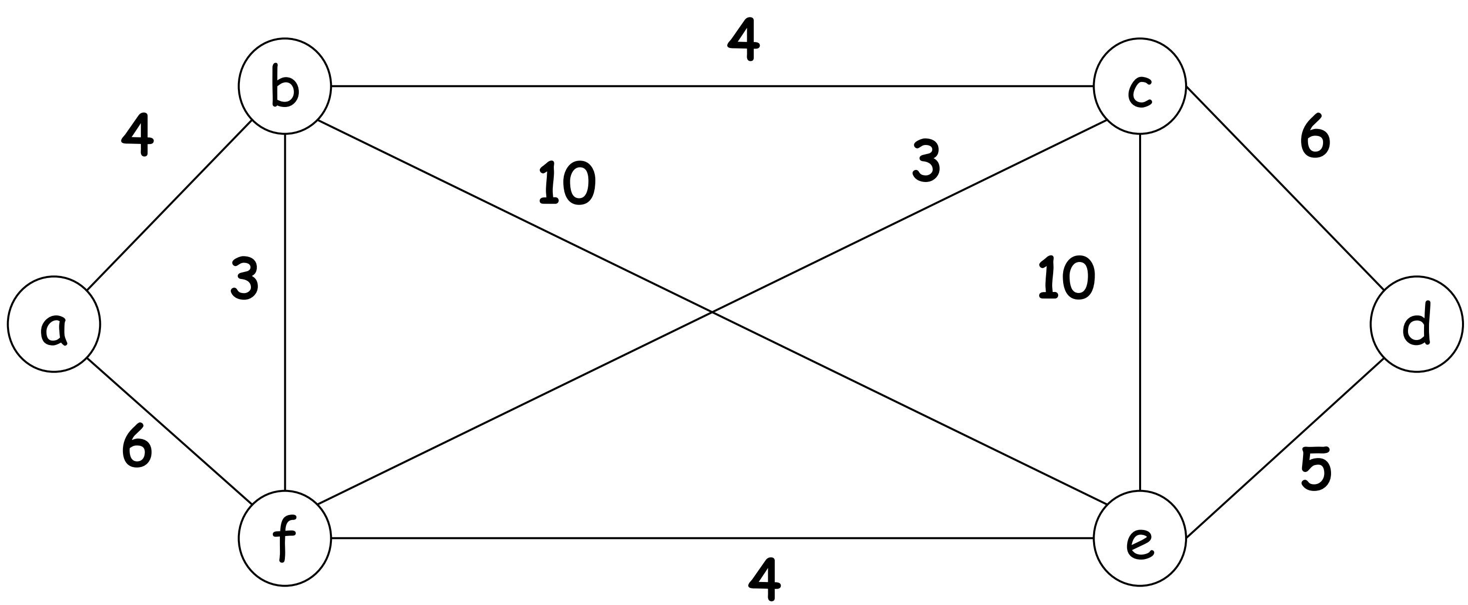
(h,g)	*1*
(i,c)	*2*
(g,f)	*2*
(a,b)	*4*
(c,f)	*4*
(c,d)	*7*
~~(h,i)~~	~~7~~
(b,c)	*8*
~~(a,h)~~	~~8~~
(d,e)	*9*
~~(f,e)~~	~~10~~
~~(b,h)~~	~~11~~
~~(d,f)~~	~~14~~



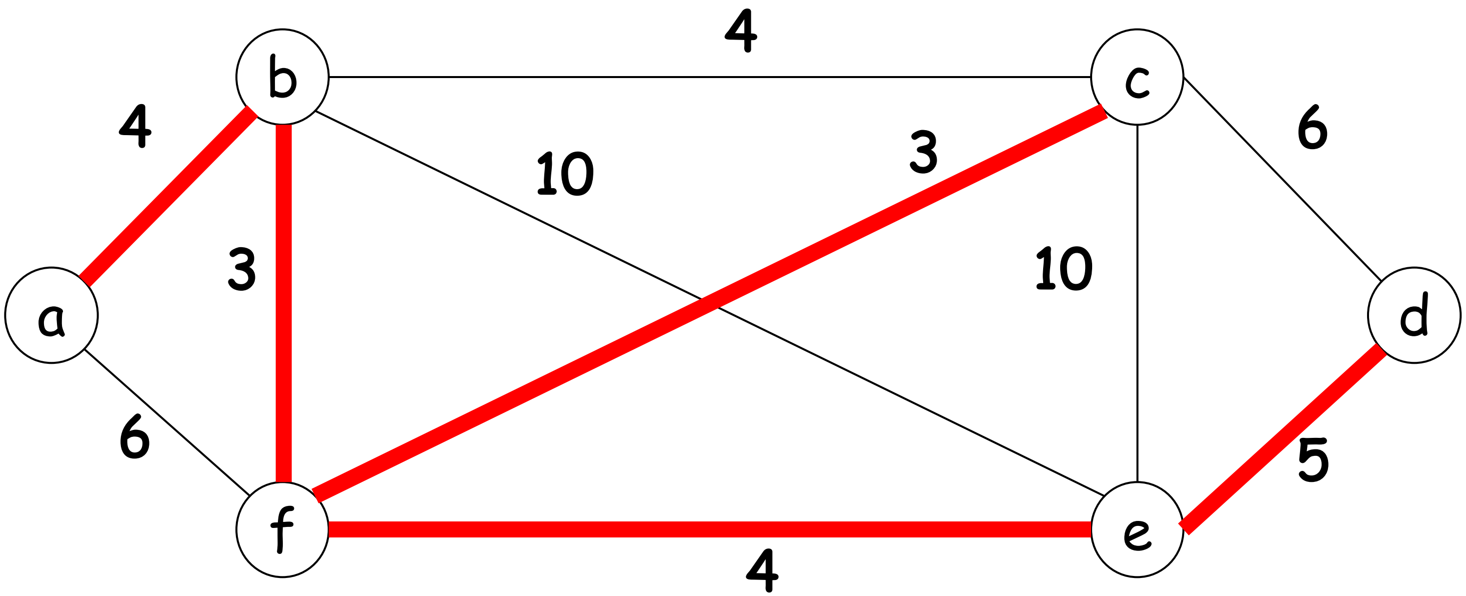
MST is found when all edges are examined

Exercise

1. Find an MST for this graph



Exercise - MST



order of selection: (b,f), (c,f), (a,b), (f,e), (e,d)

Questions:

- Correctness: Does Kruskal's algorithm always yield a minimum spanning tree?
- Complexity: what is the time complexity of Kruskal's algorithm?

Proof of Correctness (self study, optional)

- The proof consists of two parts.
  - First, it is proved that the algorithm produces a spanning tree.
  - Second, it is proved that the constructed spanning tree is of minimal weight.
- the algorithm produces a spanning tree:
  - Let be a connected, weighted graph and let be the subgraph of produced by the algorithm. cannot have a cycle, since the last edge added to that cycle would have been within one subtree and not between two different trees. cannot be disconnected, since the first encountered edge that joins two components of would have been added by the algorithm. Thus, is a spanning tree of .
- The constructed spanning tree is of minimal weight:
  - We show that the following proposition ***P*** is true by induction: If *F* is the set of edges chosen at any stage of the algorithm, then there is some minimum spanning tree that contains *F*.
  - Clearly ***P*** is true at the beginning, when *F* is empty: any minimum spanning tree will do, and there exists one because a weighted connected graph always has a minimum spanning tree.

Proof of Correctness (contd.) (optional)

– Now assume ***P*** is true for some non-final edge set F and let T be a minimum spanning tree that contains F. If the next chosen edge e is also in T, then ***P*** is true for $F + e$. Otherwise, $T + e$ has a cycle C and there is another edge f that is in C but not F. (If there were no such edge f, then e could not have been added to F, since doing so would have created the cycle C.) Then $T - f + e$ is a tree, and it has the same weight as T, since T has minimum weight and the weight of f cannot be less than the weight of e, otherwise the algorithm would have chosen f instead of e. So $T - f + e$ is a minimum spanning tree containing $F + e$ and again ***P*** holds.

– Therefore, by the principle of induction, ***P*** holds when F has become a spanning tree, which is only possible if F is a minimum spanning tree itself

Prim's algorithm vs. Kruskal's algorithm

- Approach:

  vertex, edge
- Data structures used:

  priority queue data structure

  disjoint set data structure
- Running time (optional):

  O(E log V) using a binary heap

  O(E log E) using a sorting algorithm like merge sort
- Graph type:

  connected graphs

  both connected and disconnected

Learning outcomes

- ✓ Understand what greedy method is
- ✓ Able to apply Prim's algorithm to find minimum spanning tree
- ✓ Able to apply Kruskal's algorithm to find minimum spanning tree
- Able to apply Dijkstra's algorithm to find single-source shortest-paths