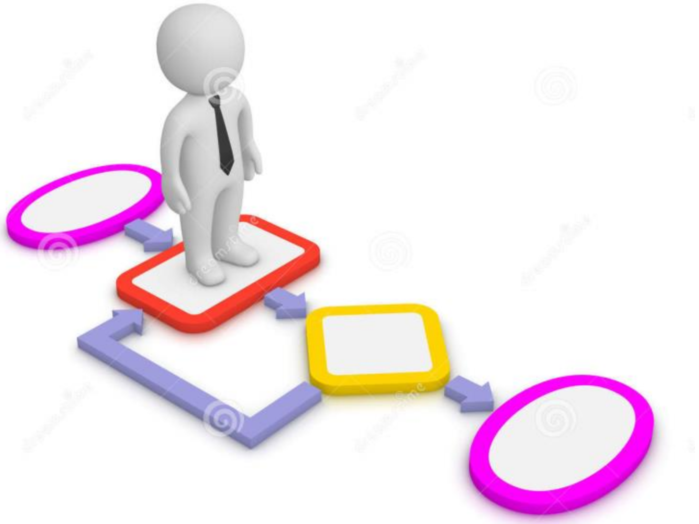
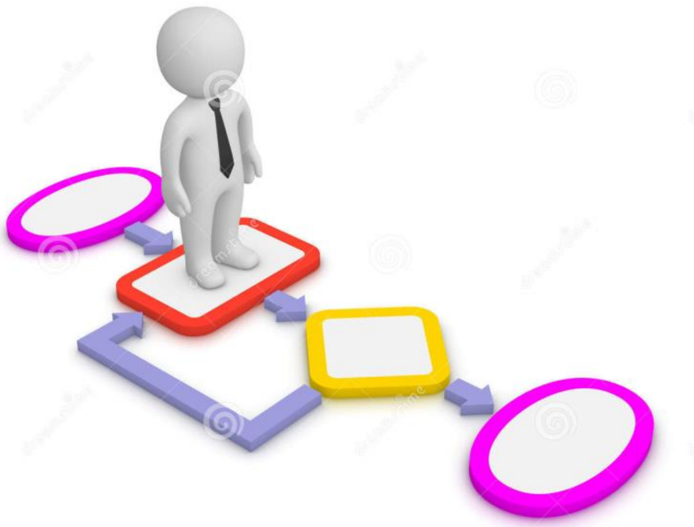


# Design & Analysis of Algorithms

**Saman Riaz (PhD)**
**Associate Professor**
**RSCI, Lahore**

**Lecture # 13**



# ELEMENTARY GRAPH ALGORITHMS

# What is Tree ?

- Definition: **A tree is a connected undirected graph with** no simple circuits.

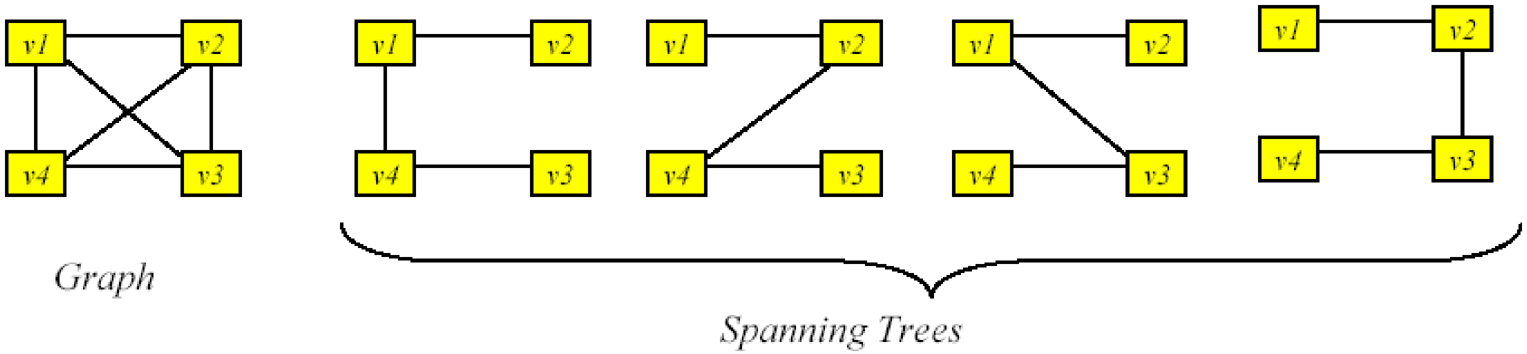
- Since a tree cannot have a simple circuit, a tree cannot contain multiple edges or loops.

- Therefore, any tree must be a **simple graph.**

- **Theorem: An undirected graph is a tree if and only if t**here is a **unique simple path between any of its** vertices.

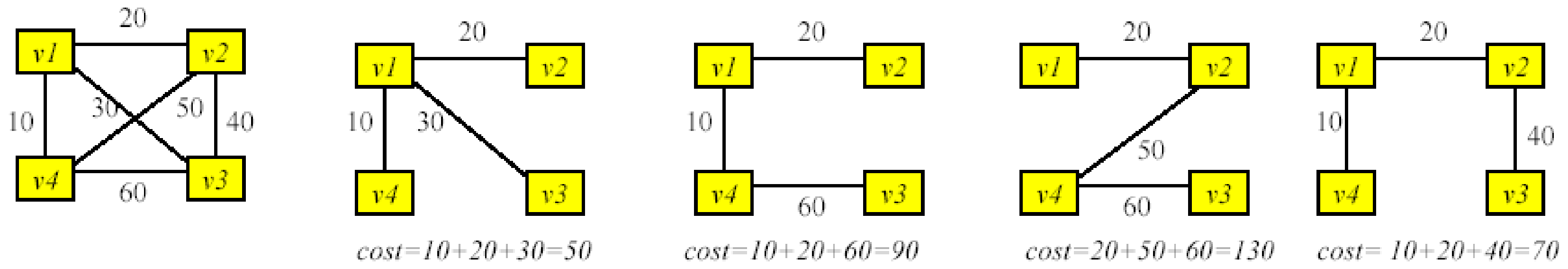
# Spanning Trees

- A *spanning tree* for an undirected graph is a sub-graph which includes *all vertices* but has *no cycles.*
- *There can be several spanning trees for a graph.* Figure shows some of the trees for the graph with vertices v1,v2,v3,v4
- Each tree has *same number of edges*



# Minimum Spanning Trees

- A *weighted undirected graph* can have several spanning trees
- One of the spanning trees has *smallest sum of all the weights* associated with the edges. This tree is called *minimum spanning tree* (MST).
- Figure shows a sample weighted graph, some of the *spanning trees*, and the *minimum spanning tree*




*(a) Weighted graph* *(b) Minimum Spanning Tree* *(c) Spanning Trees*

# Why do we need MST?

- Minimum spanning trees have many practical applications. Some typical applications are:
  - *A telephone network can be configured, using minimum spanning tree, to have minimum cable length.*
  - *The air travel routes can be selected so that the travel time or travel cost is least.*
  - *A computer network can be set up with minimum routing distance*
  - *Linking a group of island with bridges so that total bridge span length is minimum*
  - However, MST is not necessary the shortest path and it does not apply to cycle

# MST don’t solve TSP

- Travel salesman problem (TSP) can not be solved by MST :
  - salesman needs to go home (what’s the cost going home?)
  - TSP is a cycle
  - use MST to approximate
  - solve TSP by exhaustive approach try every permutation on cyclic graph

Two important algorithms for creating a minimum spanning tree for a graph, named after their inventors, are ***Kruskal’s algorithm*** and ***Prim’s algorithm.***

# Kruskal’s Algorithm

- The *Kruskal’s algorithm* works as follows:
  - *Step #: 1 Remove all edges of the graph*
  - *Step #:2 Arrange edges according to their weights*
  - *Step # 3: Select a edge with least weight*
  - *Step # 4: Attach the edge to the corresponding vertices if it does not form cycle; otherwise, drop the edge*
  - *Step # 5: Repeat steps 3 to 4 until all the edges are processed (added or dropped)*
- Kruskal’s algorithm is categorized as ***greedy***, because at each step it picks an edge with least weight.

# Examples for Kruskal's Algorithm

0 —10— 5

2 —12— 3

1 —14— 6

1 —16— 2

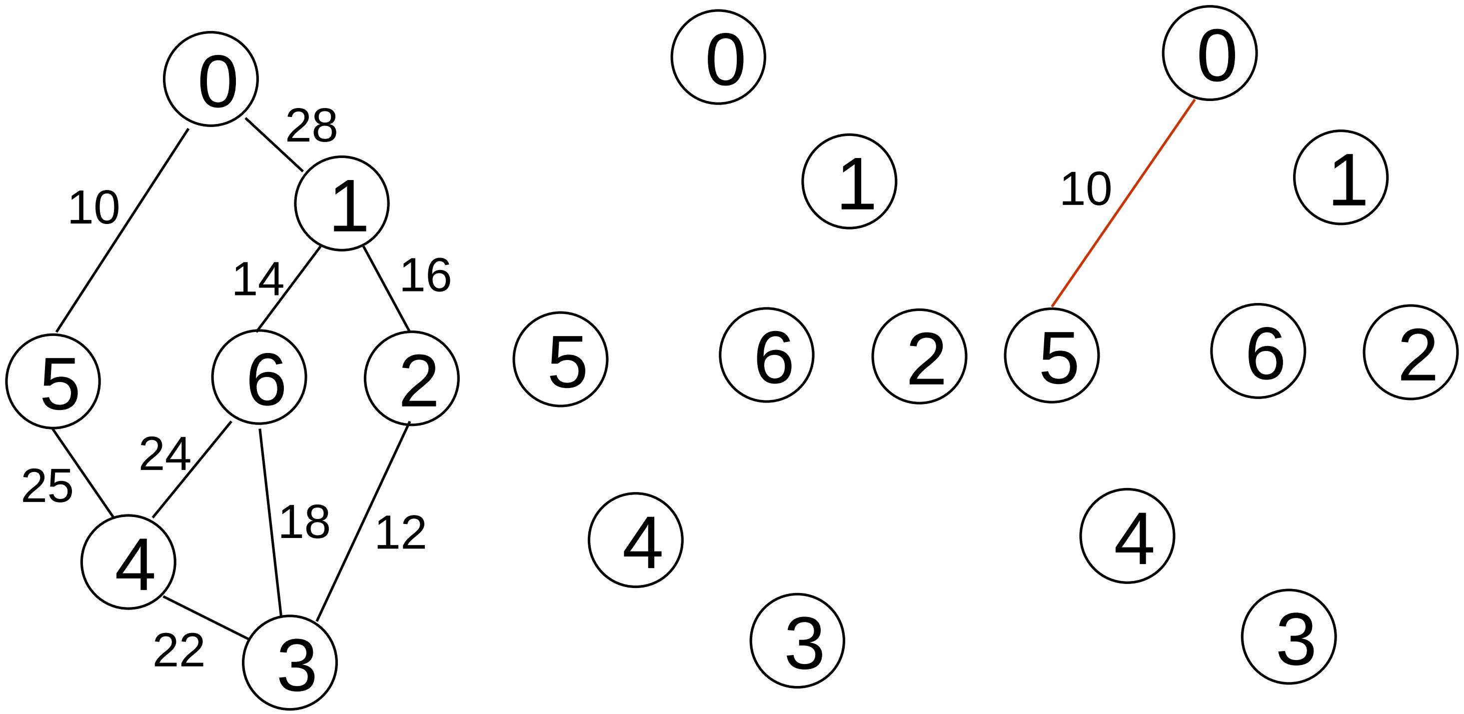
3 —18— 6

3 —22— 4

4 —24— 6

4 —25— 5

0 —28— 1



0 10 5

2 12 3

1 14 6

1 16 2

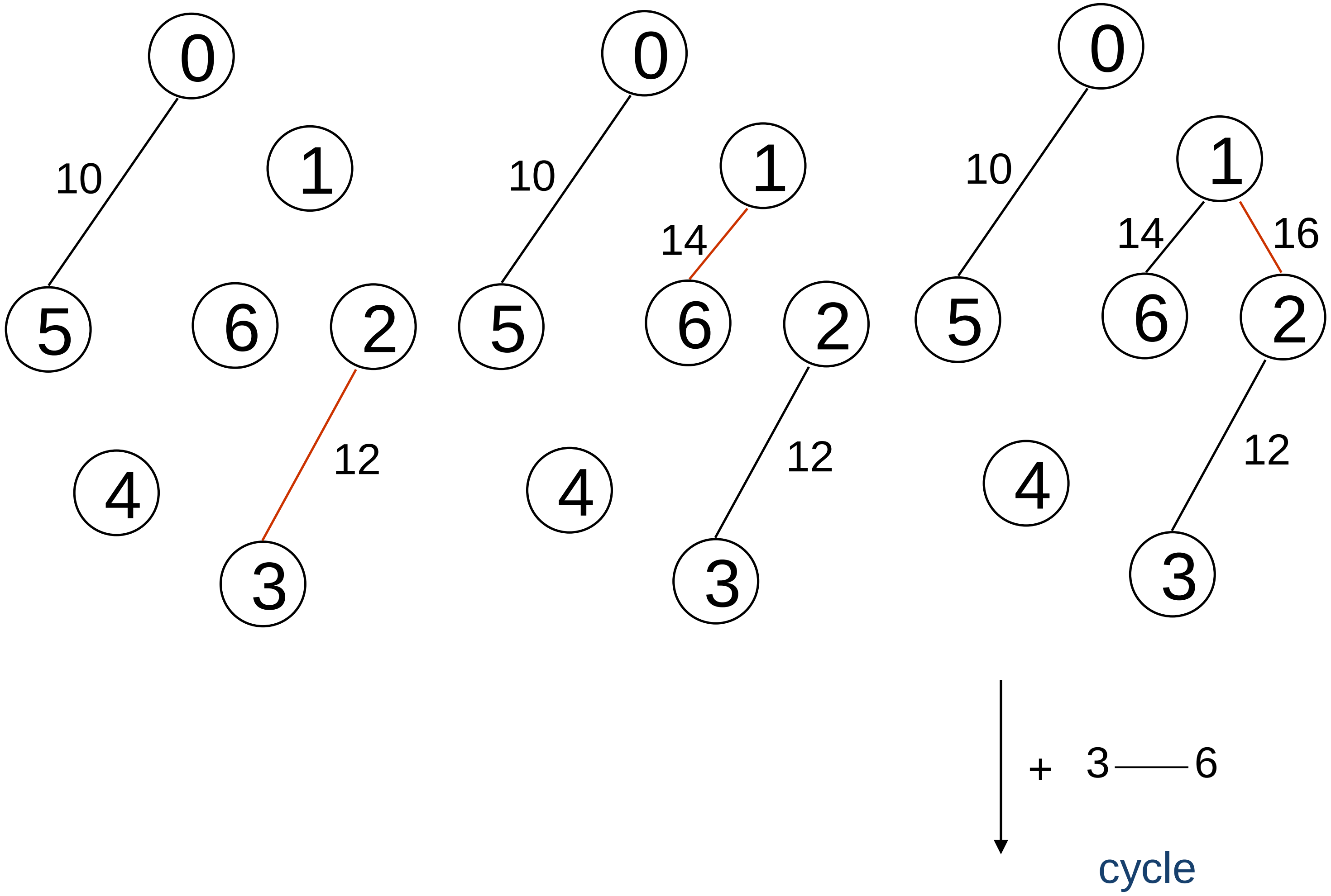
3 18 6

3 22 4

4 24 6

4 25 5

0 28 1



0 —10— 5

2 —12— 3

1 —14— 6

1 —16— 2

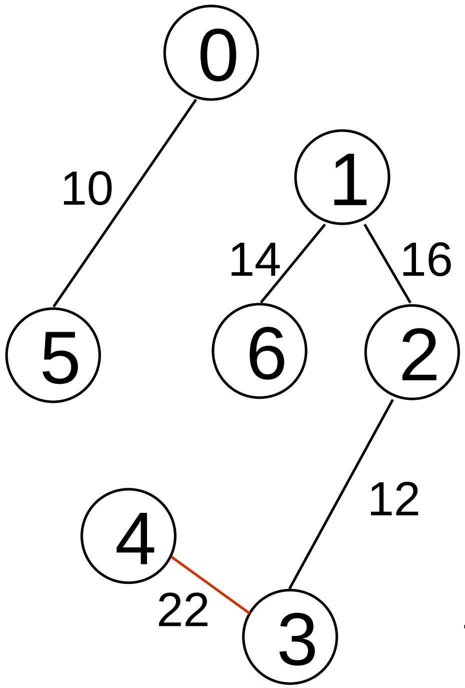
3 —18— 6

3 —22— 4

4 —24— 6

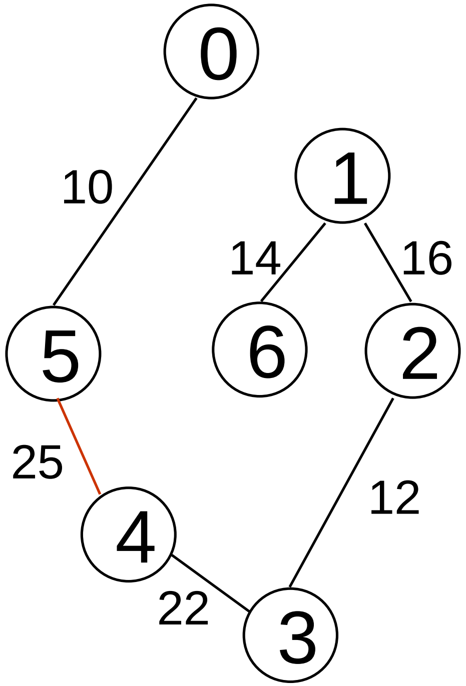
4 —25— 5

0 —28— 1




+ 4 —— 6

cycle




cost = 10 +25+22+12+16+14

# 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 non-decreasing order by weight w
for each edge (u, v) ∈ E, taken in non-decreasing order by weight
    do if FIND-SET(u) ≠ FIND-SET(v)
        then A ← A U {(u, v)}
            UNION(u, v)
Return A
```

# Kruskal’s Algorithm

- MAKE-SET(x) creates a new set whose only member (and thus representative) is x. Since sets are disjoint, x should not be in some other set
- FIND-SET(x) returns a pointer to the representative of the (unique set) containing x
- UNION(x,y) unites the sets into a new set. The original sets are removed from the collection

# Kruskal's Algorithm - Analysis

1. Initialize A: $O(1)$
2. First **For** loop: $|V|$ Make-Set operations
3. Sort the edges E: $O(E \lg E)$
4. Second **For** loop: $|E|$ Find-Set and Union operations

- 2 & 4 together take $O((V+E)\alpha(V))$, where $\alpha$ is a very slow growing function

- Since $|E| >= |V|-1$, $O((V+E)\alpha(V)) = O((E)\alpha(V)) = O((E)O(\lg V) = O(E)O(\lg E)$

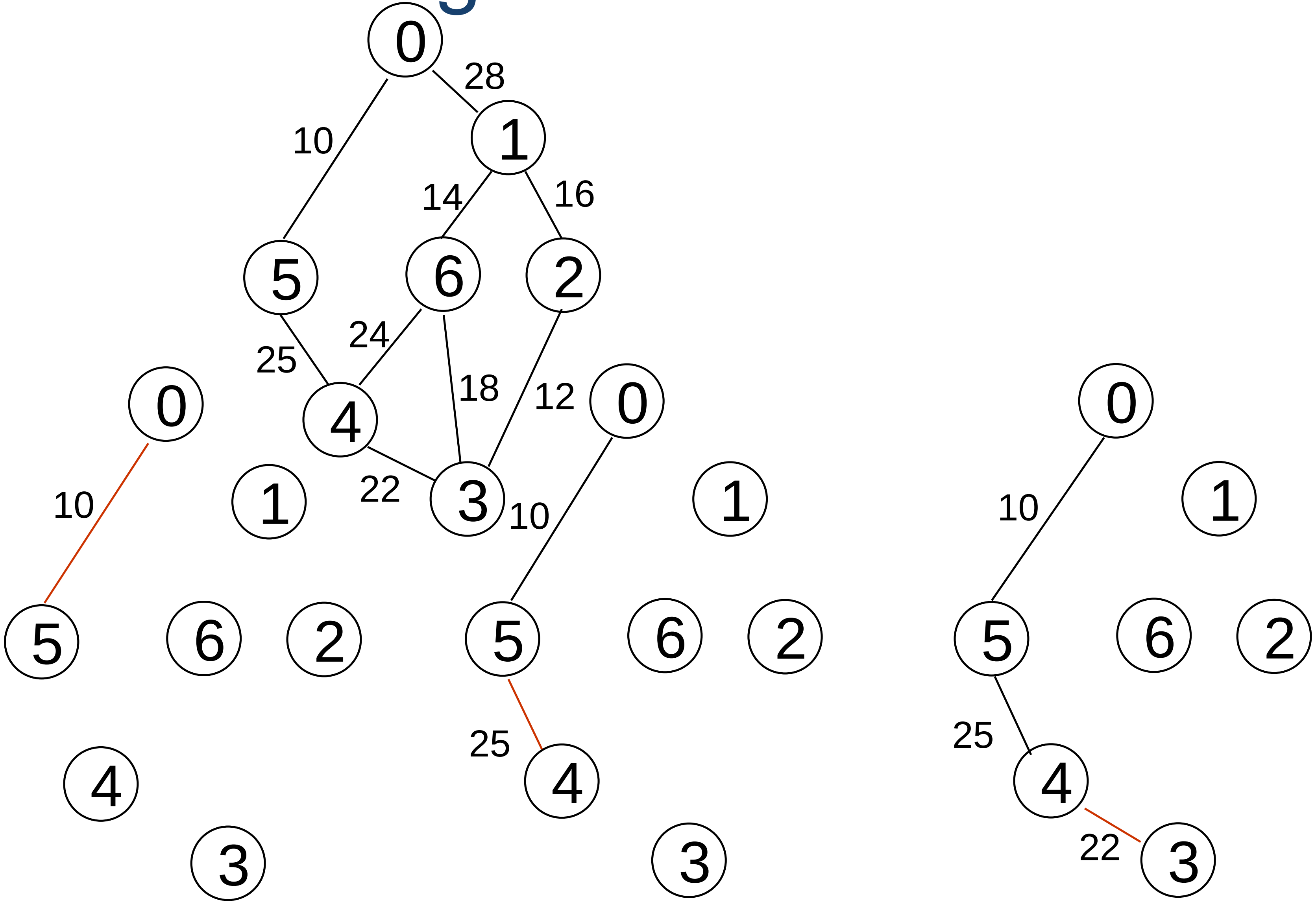
- Therefore Total running time = $O(E \lg E)$

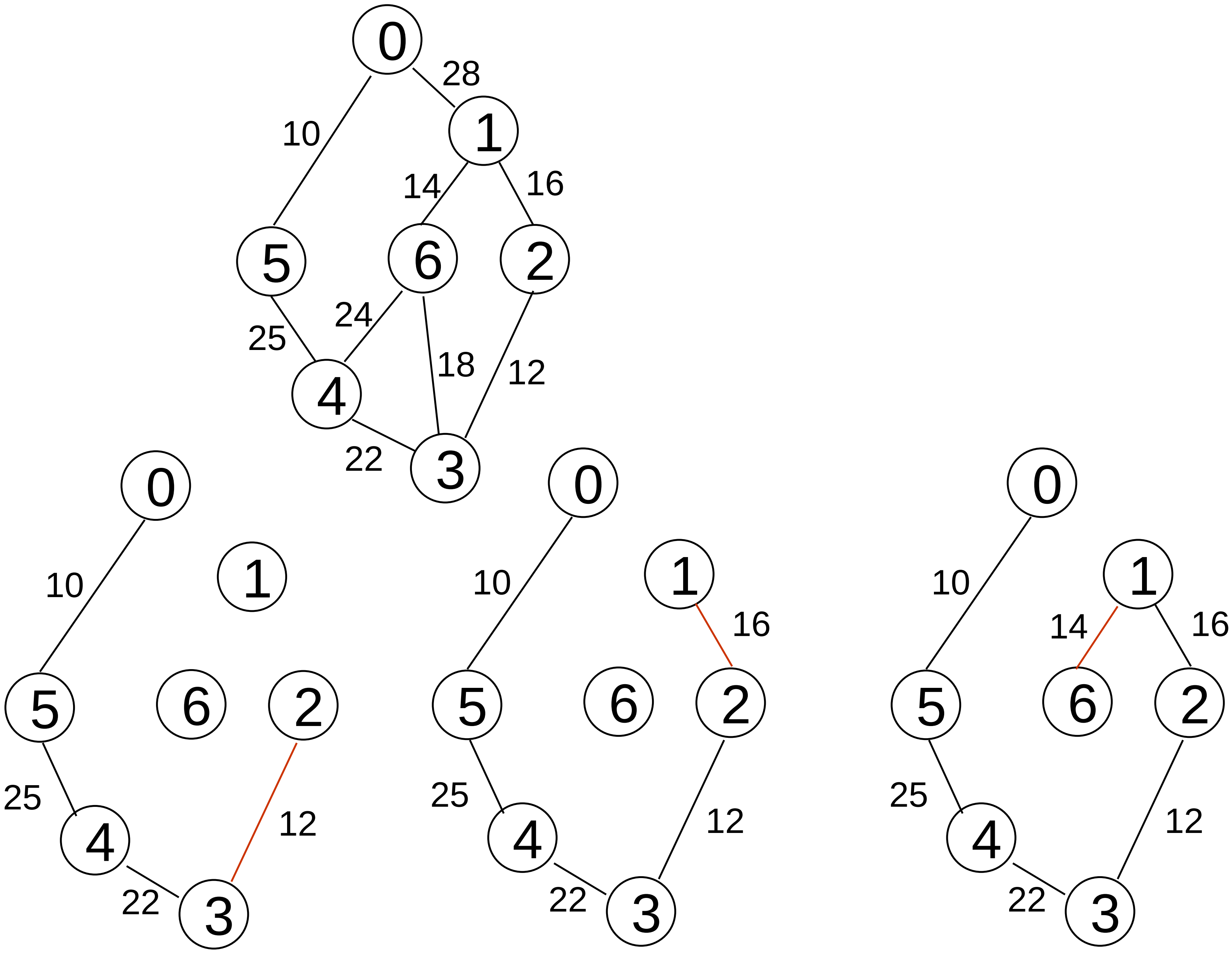
# Prim's Algorithm

- Let V be the vertex set for a  graph G. Let T be the minimum spanning tree for G. The  Prim's algorithm proceeds as follows:

- *Step #:1 Select some vertex s in V as the starting vertex.*
- *Step # 2: Add vertex **s** to an empty set S. Remove **s** from V.*
- *Step # 3: Repeat Step #:4 through Step #:6 until the set V is empty.*
- *Step # 4: Examine all vertices in S which are linked to vertices in V.*
- *Step # 5: Choose the vertex **u**  in V which has the minimum distance from vertex **v** in S.*
- *Step # 6: Remove vertex **u** from V and add it to S. Move edge **(v , u)** to T.*

# Examples for Prim's Algorithm




0
1
2
3
4
5
6
28
10
14
16
24
25
18
12
22
0
1
5
6
2
4
3
10
25
12
22
0
1
5
6
2
4
3
10
16
25
12
22
0
1
5
6
2
4
3
10
14
16
25
12
22

# Prim's Algorithm

```
MST_PRIM (G, w, r)
for each u in V[G]
    do key [u] ← ∞
        π[r] ← NIL
key [r] ← 0
Q ← V[G]
while Q is not empty
    do u ← EXTRACT_MIN (Q)
        for each v in Adj[u] do
            do if v is in Q and w(u, v) < key [v]
                then π[v] ← u
                    key [v] ← w(u, v)
```

# Prim’s Algorithm - Analysis

- Depends on how minimum priority queue is implemented
- Suppose Q is a binary heap
- Initialization and first for loop O(V)
- The 2nd for loop is executed O(E) times
- Last line involves a decrease key operation, which takes O(lgV) time
- So time for Prim’s algorithm is O(ElgV)
- This can be improved to O(E+VlgV) with Fibonacci heaps