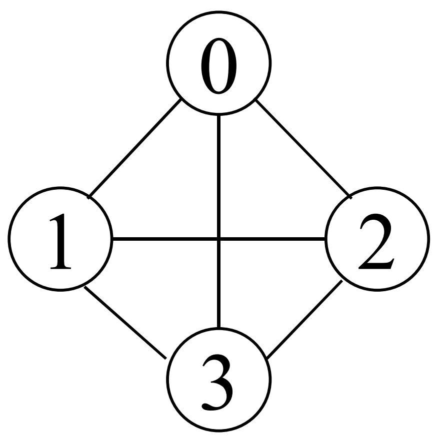
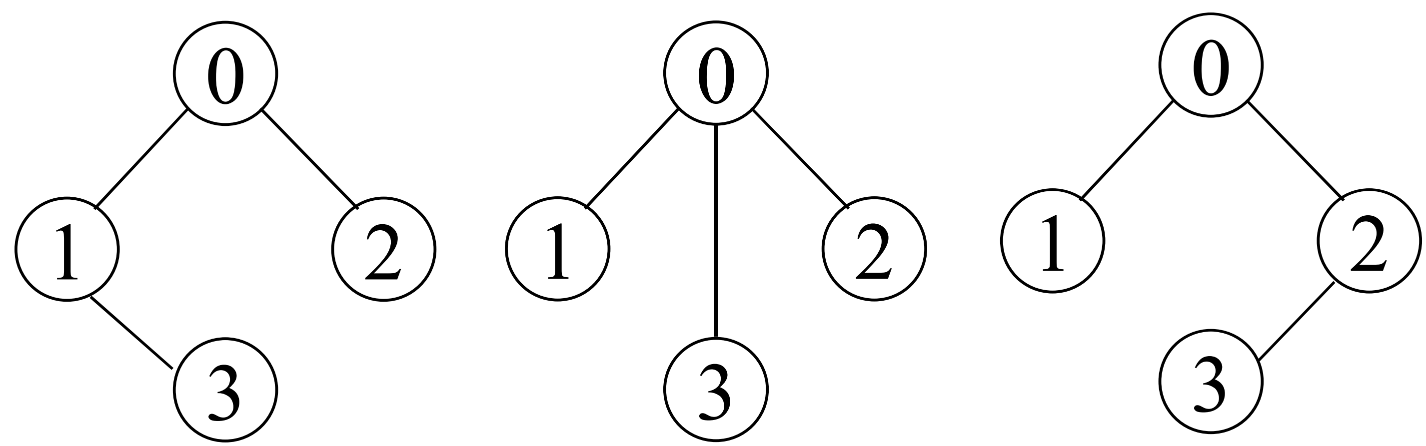
Spanning Trees

- When graph G is connected, a depth first or breadth first search starting at any vertex will visit all vertices in G
- A spanning tree is any tree that consists solely of edges in G and that includes all the vertices
- E(G): T (tree edges) + N (nontree edges)
  where T: set of edges used during search
  N: set of remaining edges

Examples of Spanning Tree



G_1



Possible spanning trees

Spanning Trees

- Either dfs or bfs can be used to create a spanning tree
  - When dfs is used, the resulting spanning tree is known as a depth first spanning tree
  - When bfs is used, the resulting spanning tree is known as a breadth first spanning tree
- While adding a nontree edge into any spanning tree, this will create a cycle

Minimum Cost Spanning Tree

- The cost of a spanning tree of a weighted undirected graph is the sum of the costs of the edges in the spanning tree
- A minimum cost spanning tree is a spanning tree of least cost
- Three different algorithms can be used
  - Kruskal
  - Prim
  - Sollin

Select n-1 edges from a weighted graph of n vertices with minimum cost.

Greedy Strategy

- An optimal solution is constructed in stages
- At each stage, the best decision is made at this time
- Since this decision cannot be changed later, we make sure that the decision will result in a feasible solution
- Typically, the selection of an item at each stage is based on a least cost or a highest profit criterion

Kruskal’s Idea

- Build a minimum cost spanning tree T by adding edges to T one at a time
- Select the edges for inclusion in T in nondecreasing order of the cost
- An edge is added to T if it does not form a cycle
- Since G is connected and has $n > 0$ vertices, exactly $n-1$ edges will be selected

Examples for Kruskal's Algorithm

$0 \xrightarrow{10} 5$

$2 \xrightarrow{12} 3$

$1 \xrightarrow{14} 6$

$1 \xrightarrow{16} 2$

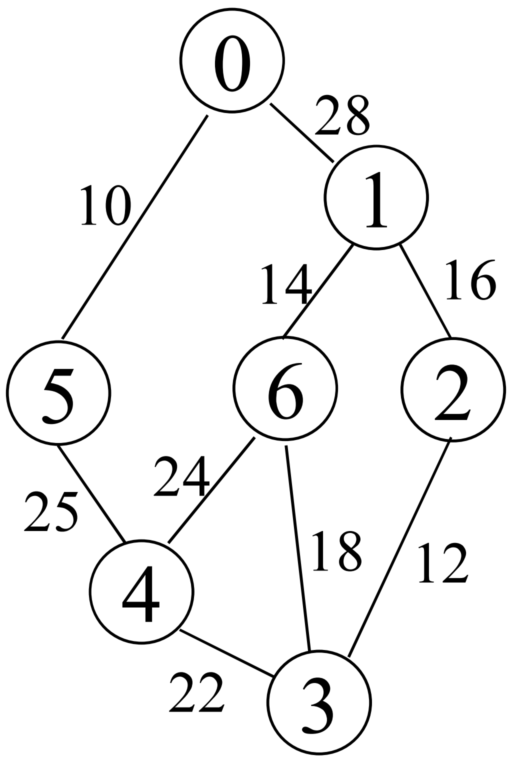
$3 \xrightarrow{18} 6$

$3 \xrightarrow{22} 4$

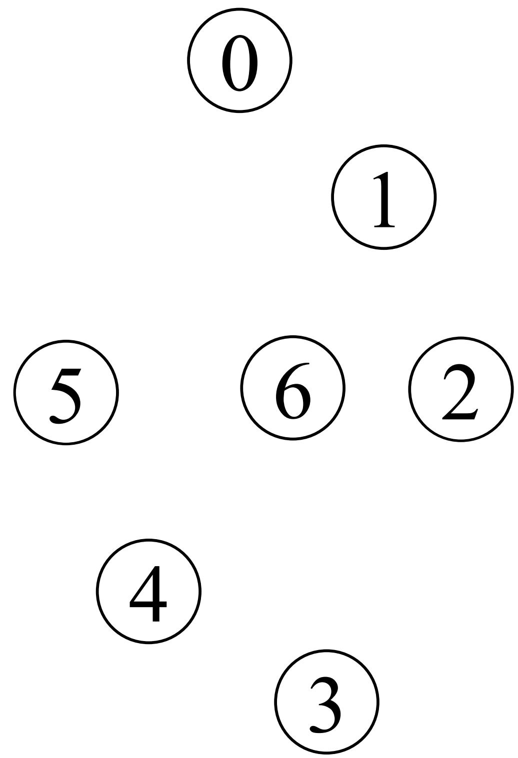
$4 \xrightarrow{24} 6$

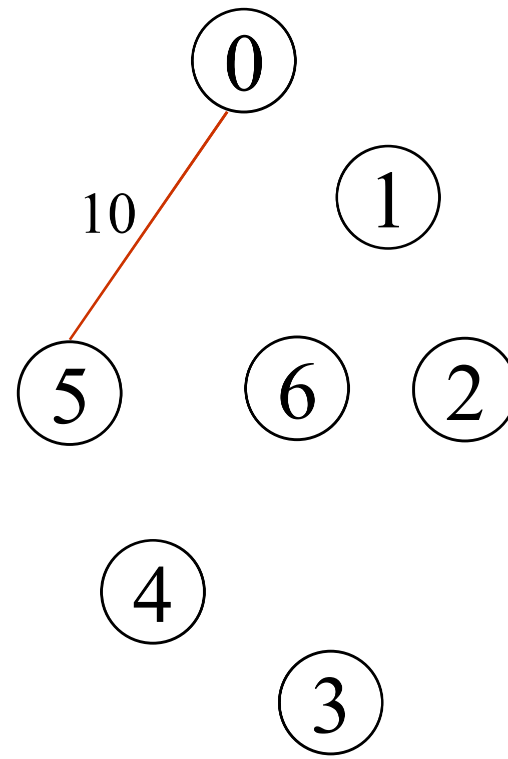
$4 \xrightarrow{25} 5$

$0 \xrightarrow{28} 1$



6/9





0 —~~10~~— 5

2 —~~12~~— 3

1 —~~14~~— 6

1 —~~16~~— 2

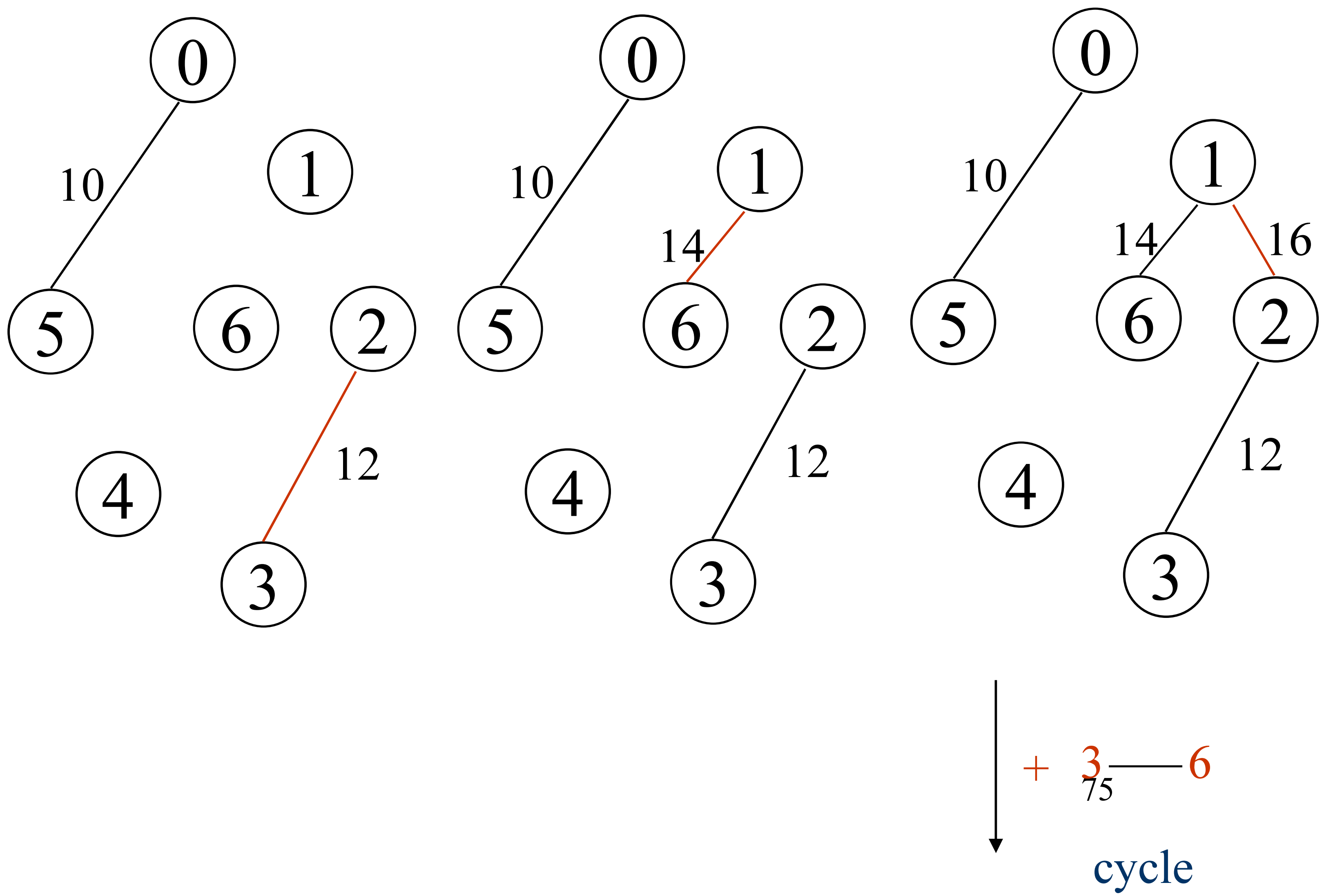
3 —~~18~~— 6

3 —~~22~~— 4

4 —~~24~~— 6

4 —~~25~~— 5

0 —~~28~~— 1



0 —10— 5

2 —12— 3

1 —14— 6

1 —16— 2

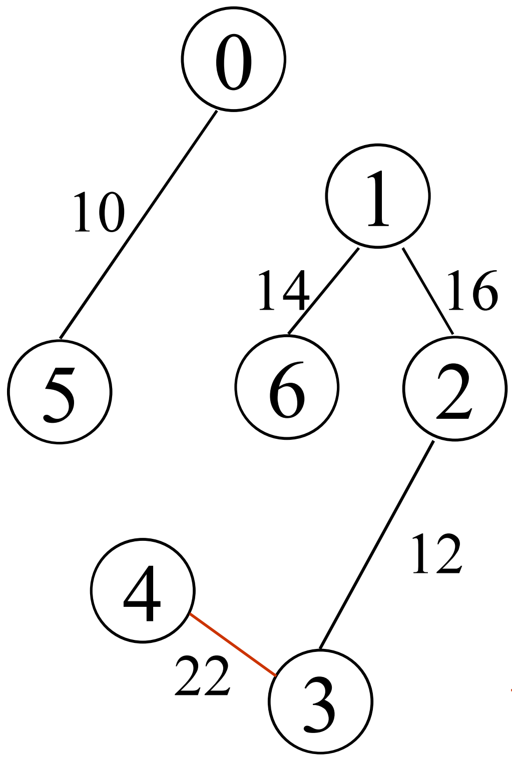
3 —18— 6

3 —22— 4

4 —24— 6

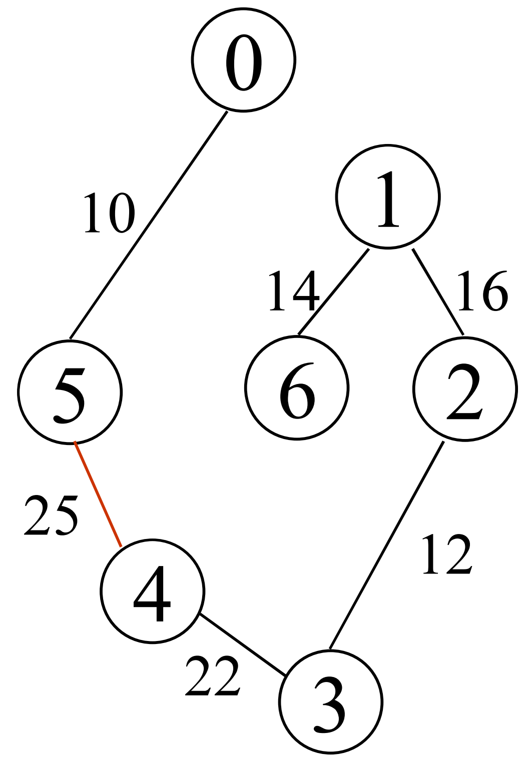
4 —25— 5

0 —28— 1



+ 4 —— 6

cycle



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

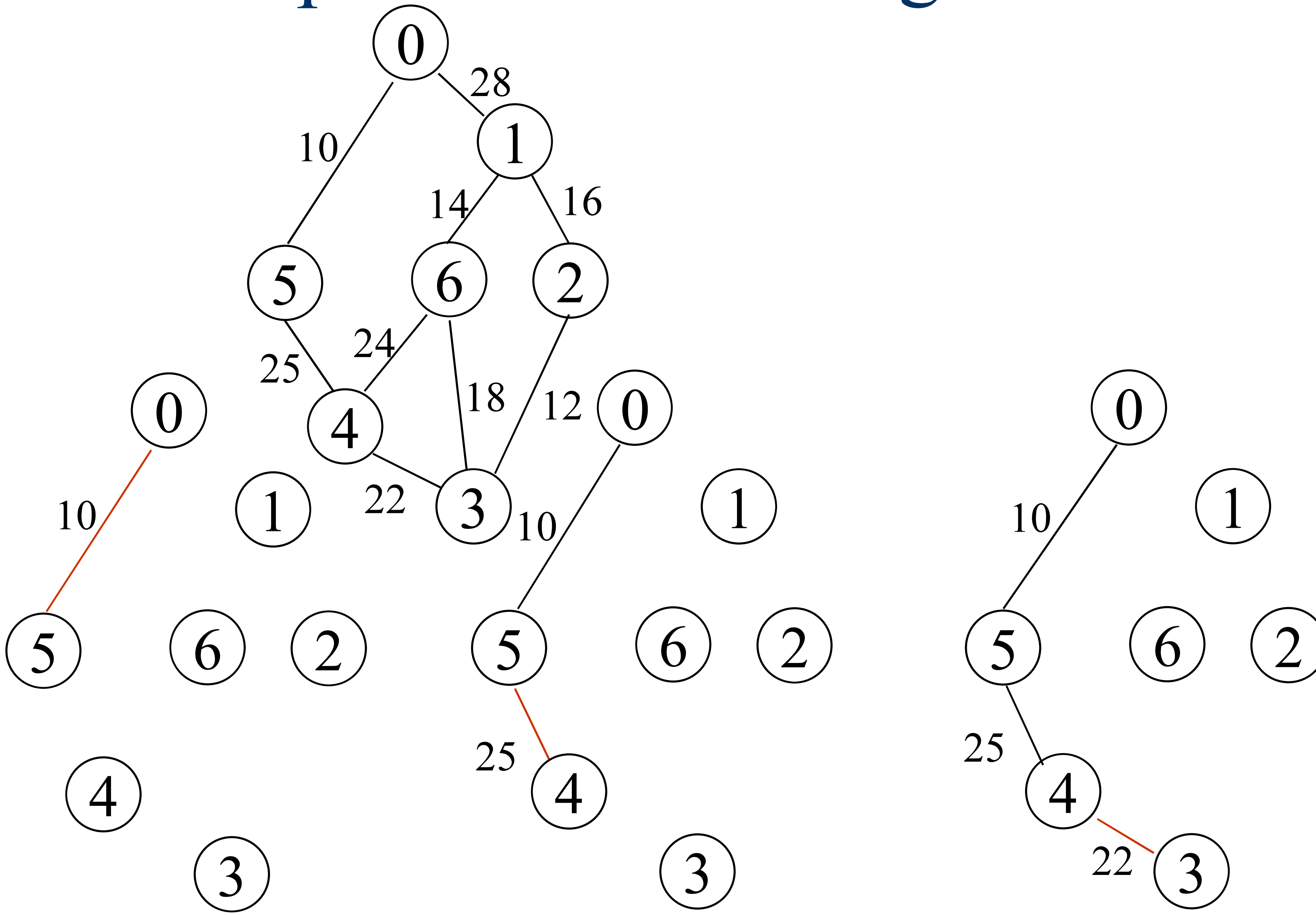
Kruskal Algorithm for MST

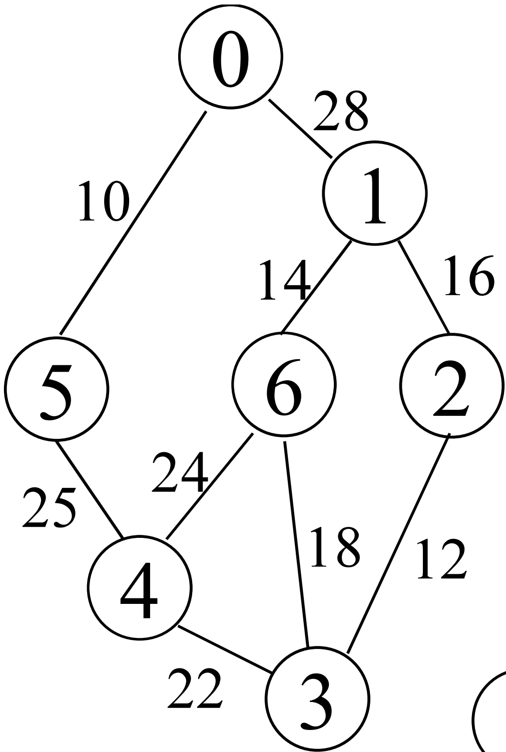
```
MST-KRUSKAL(G, w)
A ← Ø
for each vertex v ∈ V[G]
      do MAKE-SET(v)
sort the edges of E into nondecreasing order by weight w
for each edge (u, v) ∈ E, taken in nondecreasing order by weight

       do if FIND-SET(u) ≠ FIND-SET(v)
             then A ← A ∈ {(u, v)}
                   UNION(u, v)
return A
```

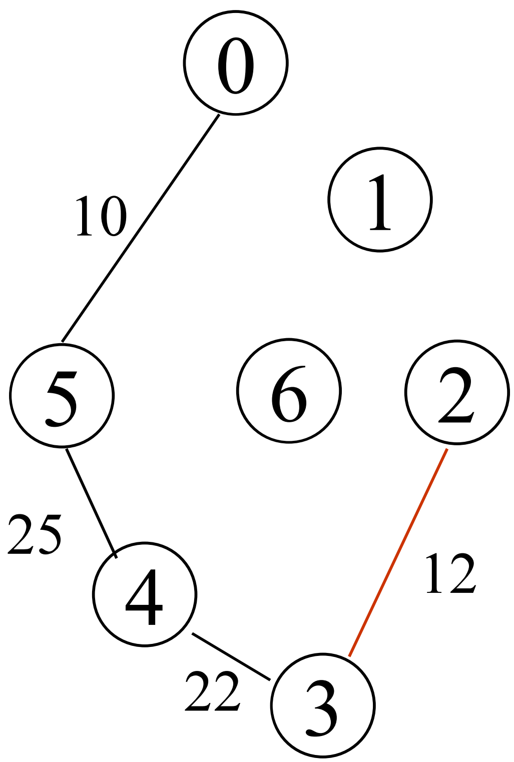
Prim’s Algorithm

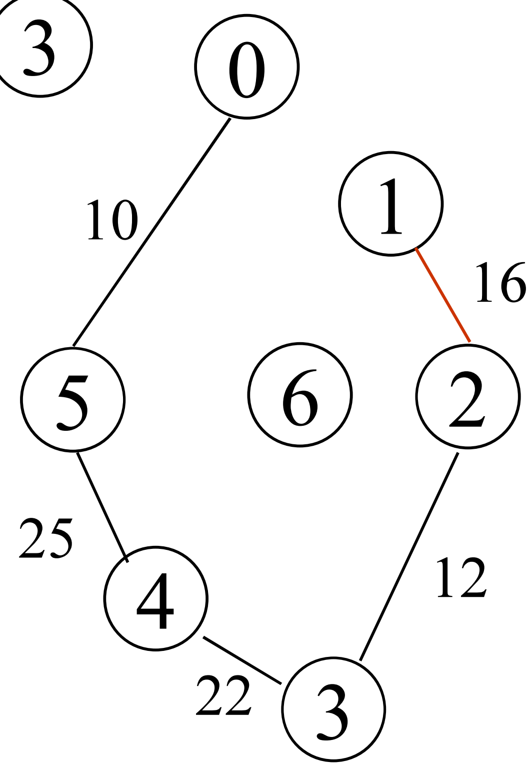
```
MST-PRIM(G, w, r)
  for each u ∈ V [G]
        do key[u] ← ∞
           π[u] ← NIL
  key[r] ← 0
   Q ← V [G]
   while Q ≠ Ø
        do u ← EXTRACT-MIN(Q)
           for each v ∈ Adj[u]
               do if v ∈ Q and w(u, v) < key[v]
                     then π[v] ← u
                          key[v]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

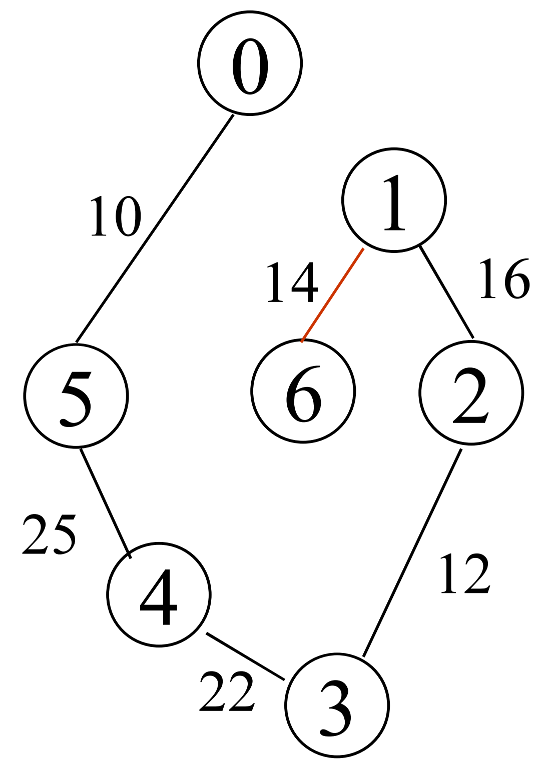
Examples for Prim's Algorithm











80