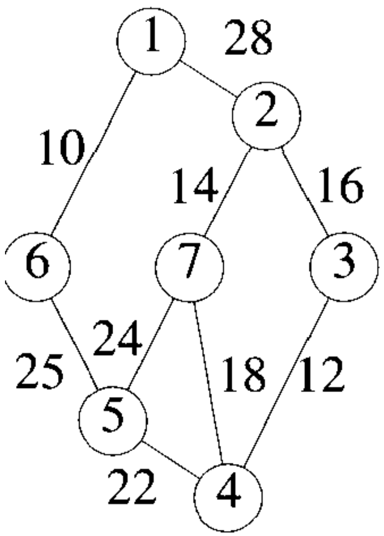
Kruskal's Algorithm

Minimum Spanning Tree

Kruskal's Algorithm

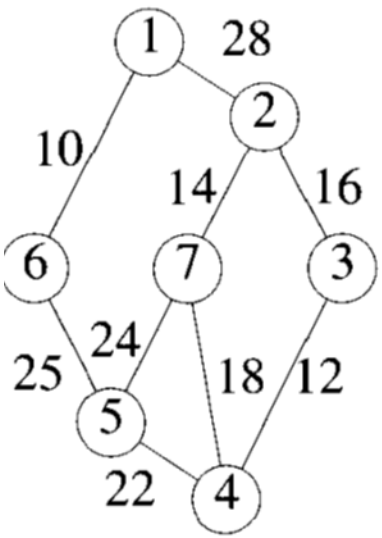
- Kruskal's approach to the construction of minimum spanning tree starts with an empty spanning tree and adds the minimum cost edge to the tree if it does not form a cycle.

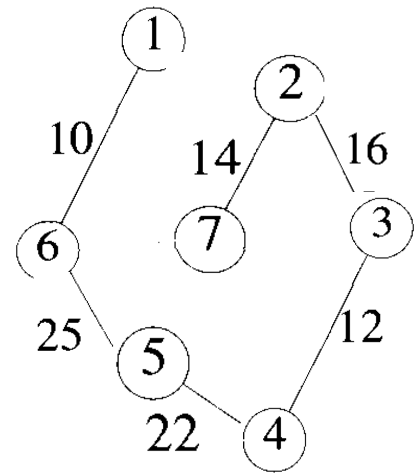


Kruskal's Algorithm

- Greedy Principle/Criterion
- Selection: Minimum Cost Edge
- Feasibility: Does not form a cycle, if added to the spanning tree.

Kruskal's Algorithm





Minimum Cost Spanning Tree

Kruskal's Algorithm

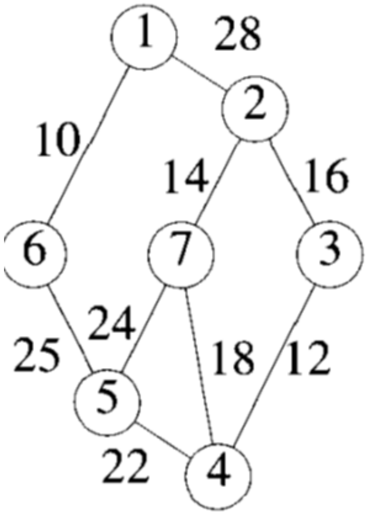
- A min-heap of edges and their costs is maintained for obtaining the next least cost edge in $log|E|$ time.
- To find out whether a cycle is formed by adding an edge, we maintain Disjoint set ADT of vertices.
- Initially, each vertex is in its own set.

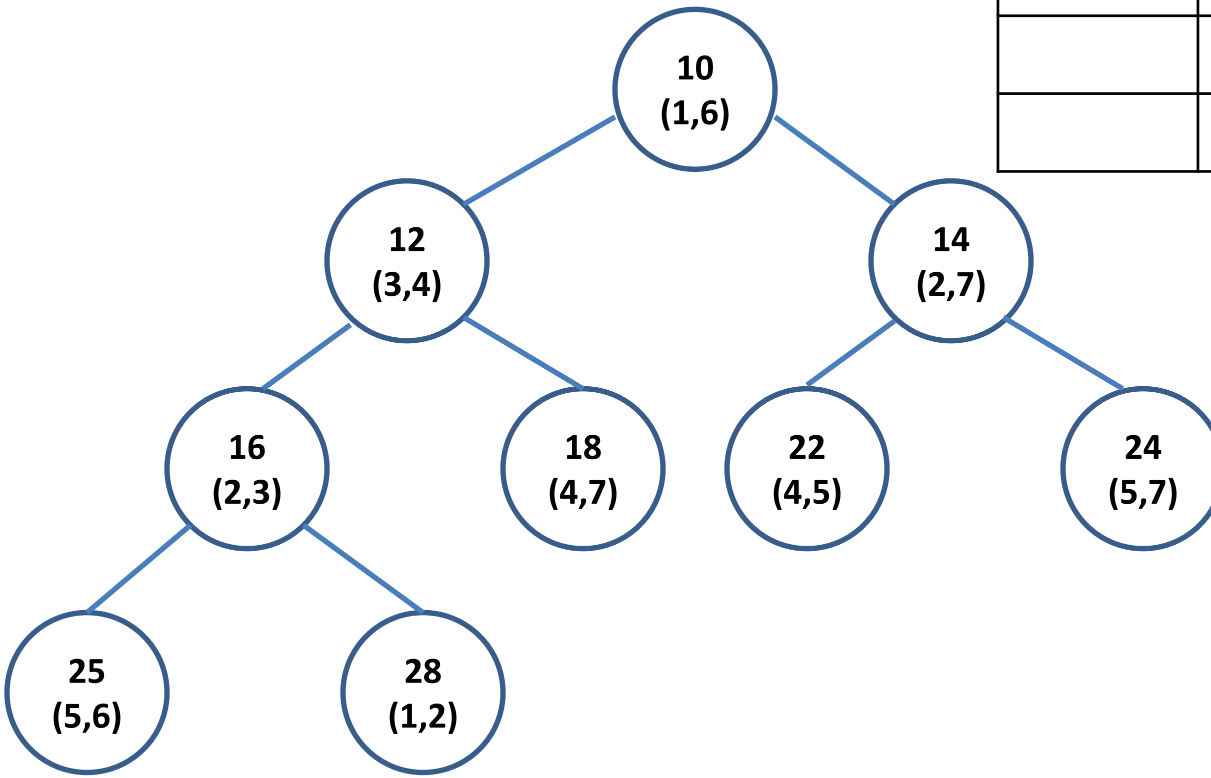
Kruskal's Algorithm

- An edge is discarded if both of its vertices are in the same set.
- Whenever an edge is added, the sets containing its vertices are merged.
- A matrix with two columns is used to store the edges selected.

Kruskal's Algorithm

t





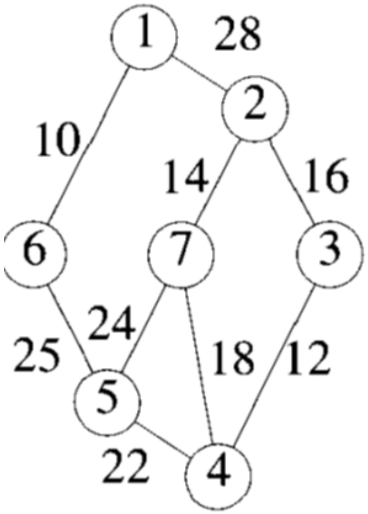
Cost=0

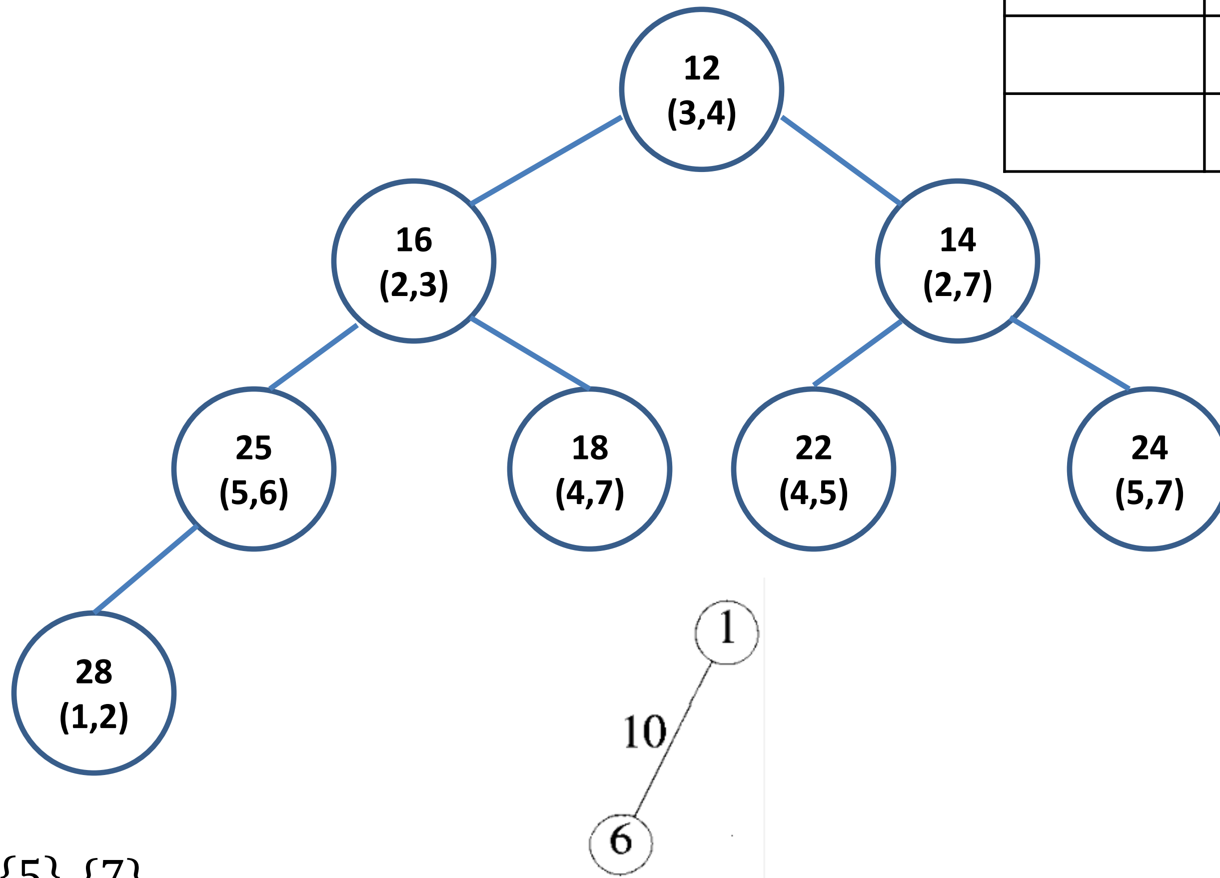
$\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}$

Kruskal's Algorithm

t

1	6





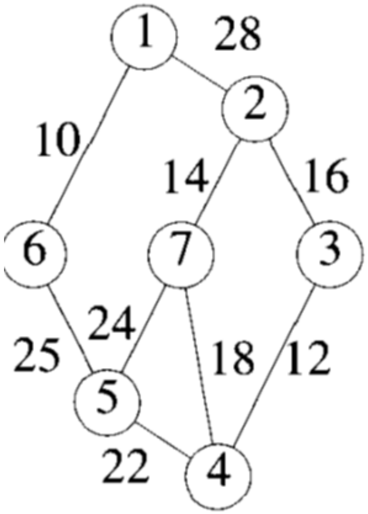
Cost=10

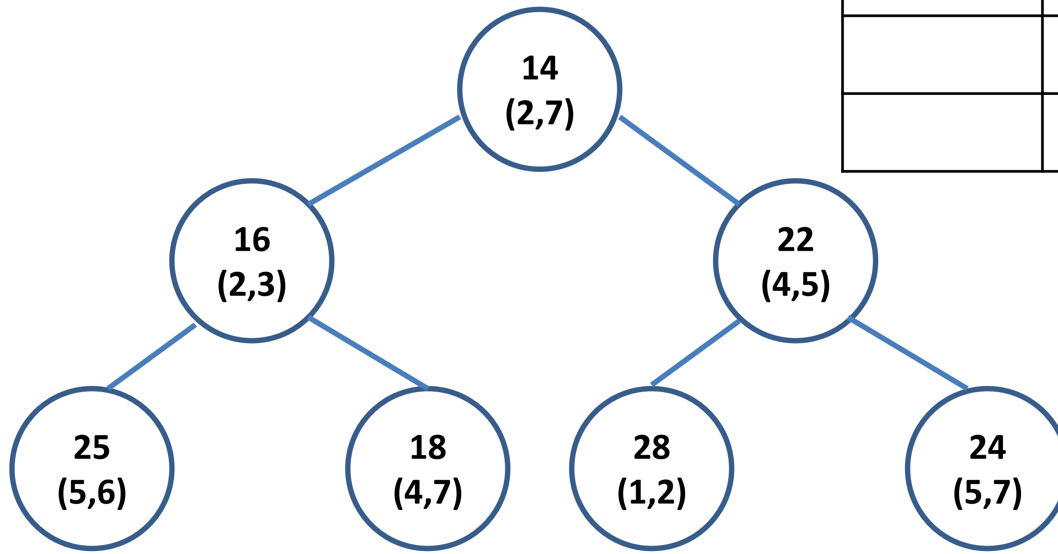
{1,6}, {2}, {3}, {4}, {5}, {7}

Kruskal's Algorithm

t

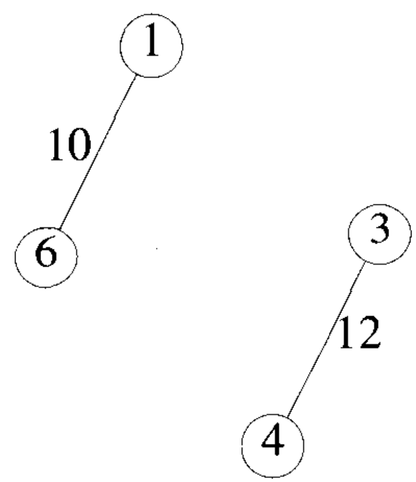
1	6
3	4





Cost=22

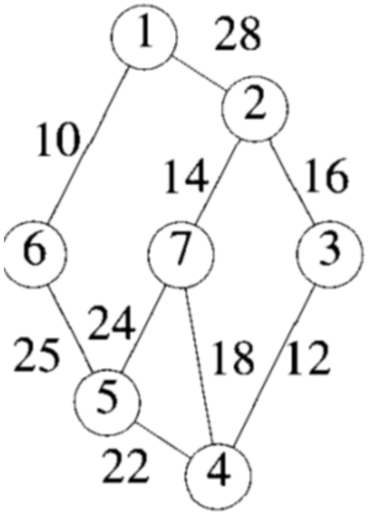
{1,6}, {2}, {3,4}, {5}, {7}

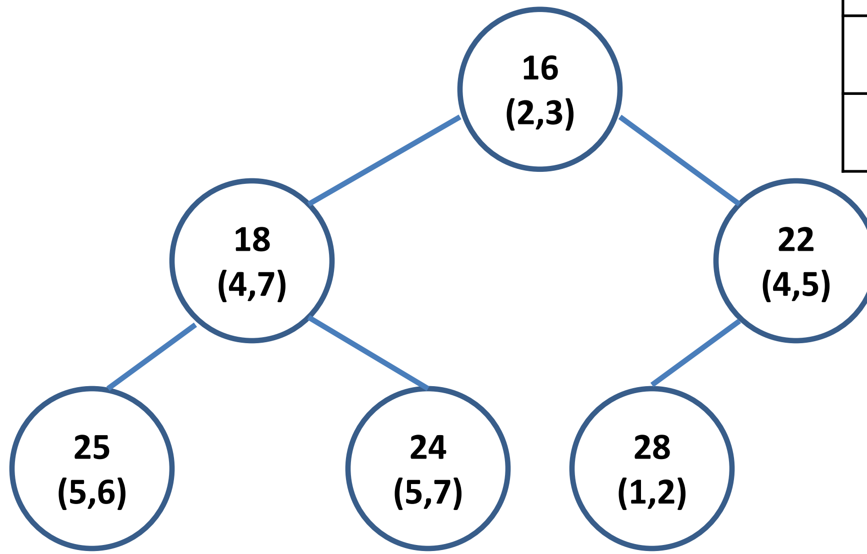


Kruskal's Algorithm

t

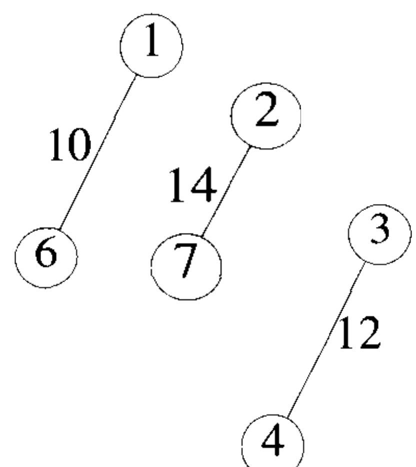
1	6
3	4
2	7





Cost=36

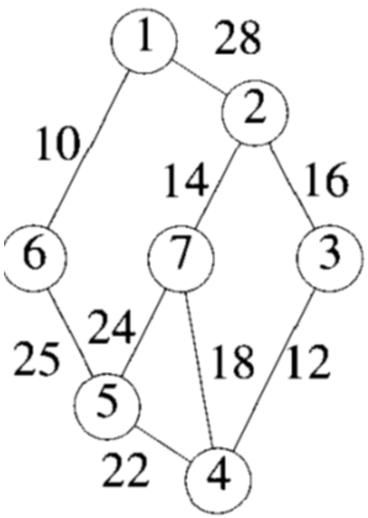
{1,6}, {2,7}, {3,4}, {5}

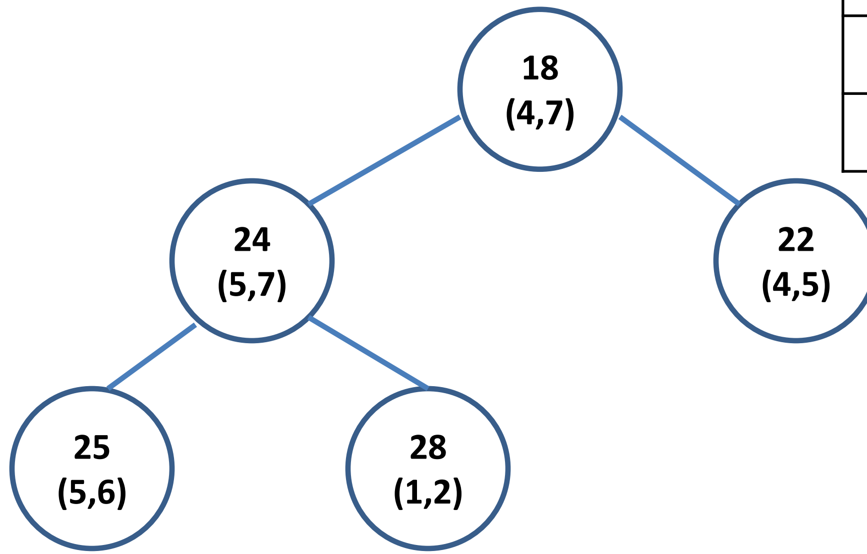


Kruskal's Algorithm

t

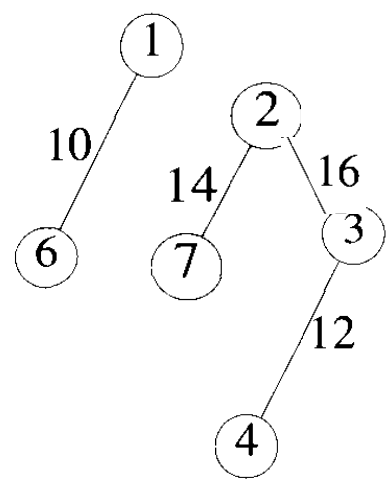
1	6
3	4
2	7
2	3





Cost=52

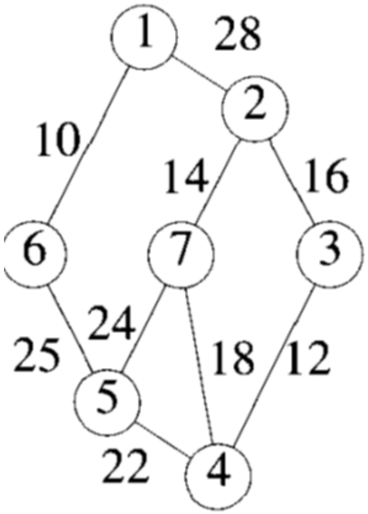
{1,6}, {2,7,3,4}, {5}

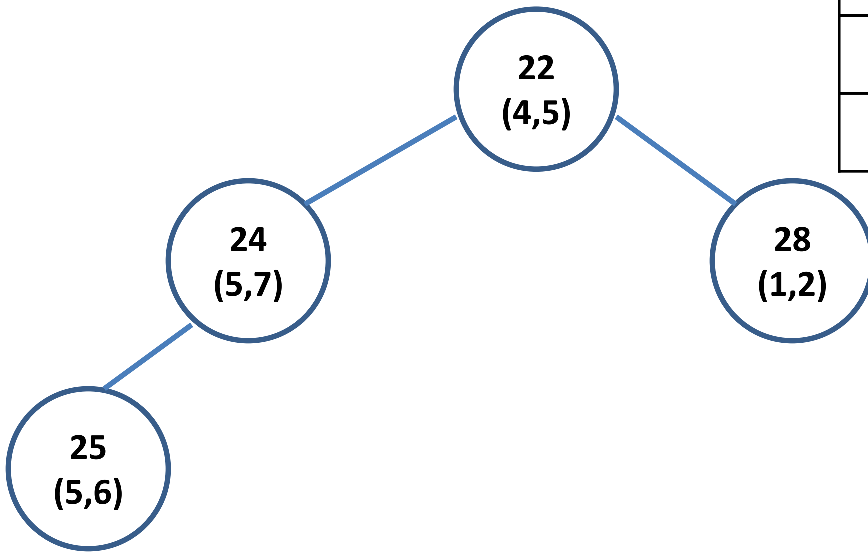


Kruskal's Algorithm

t

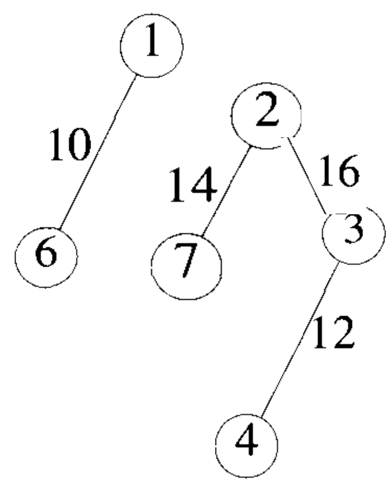
1	6
3	4
2	7
2	3





Cost=52

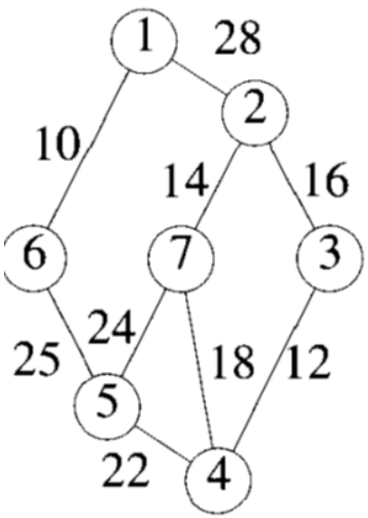
$\{1,6\}, \{2,7,3,4\}, \{5\}$

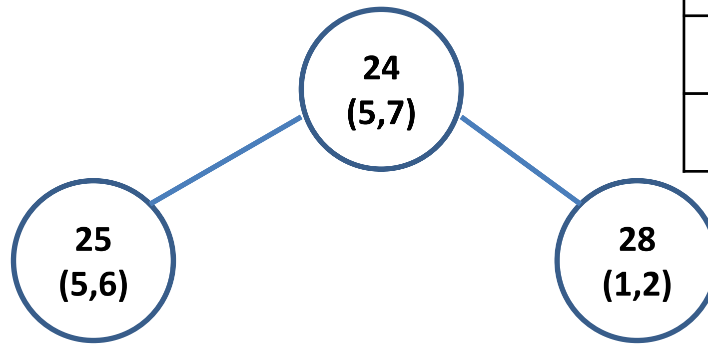


Kruskal's Algorithm

t

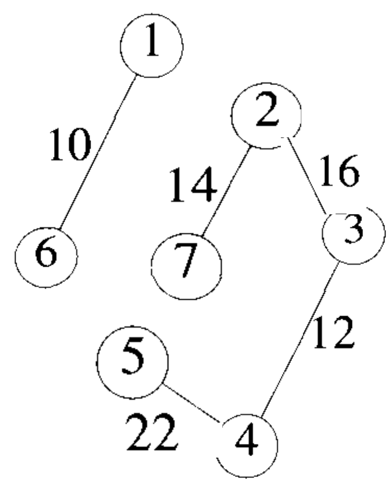
1	6
3	4
2	7
3	4
4	5



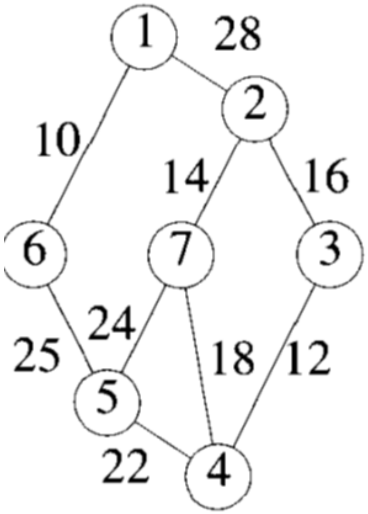


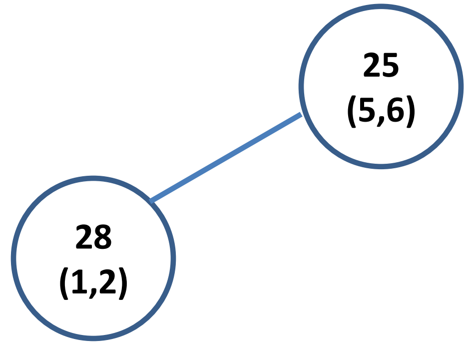
Cost=74

$\{1,6\}, \{2,7,3,4,5\}$



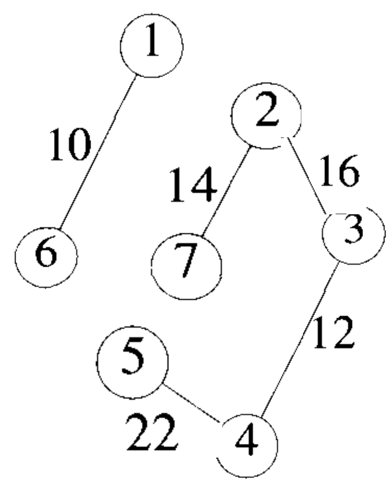
Kruskal's Algorithm





Cost=74

{1,6}, {2,7,3,4,5}

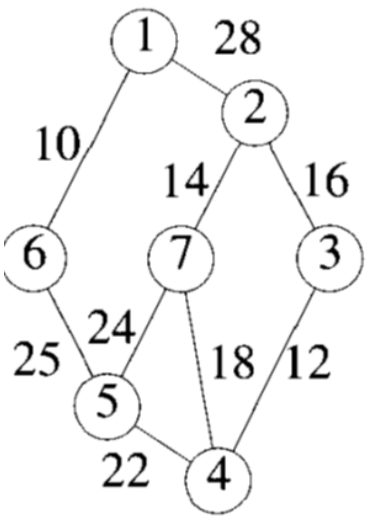


Kruskal's Algorithm

t

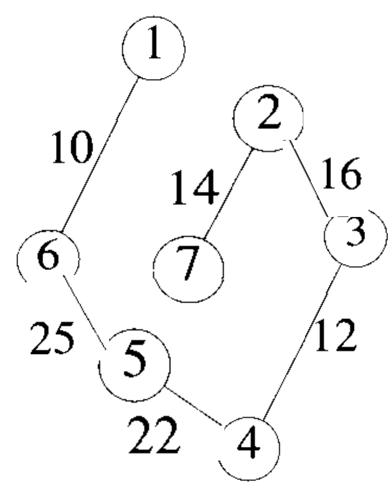
1	6
3	4
2	7
3	4
4	5
5	6

28
(1,2)

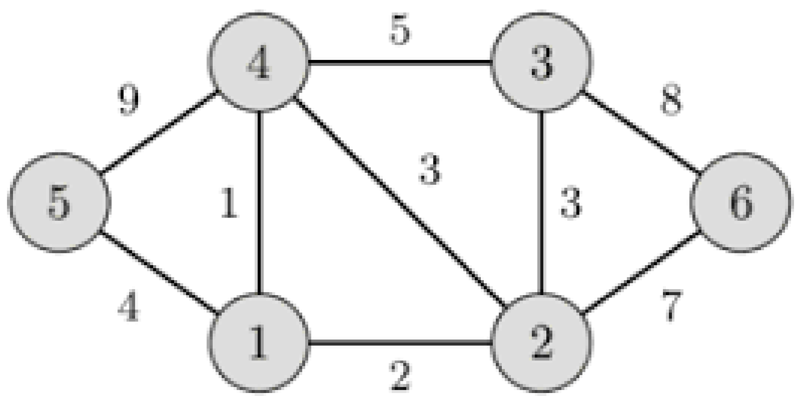


Cost=89

{1,6,2,7,3,4,5}



Kruskal's Algorithm



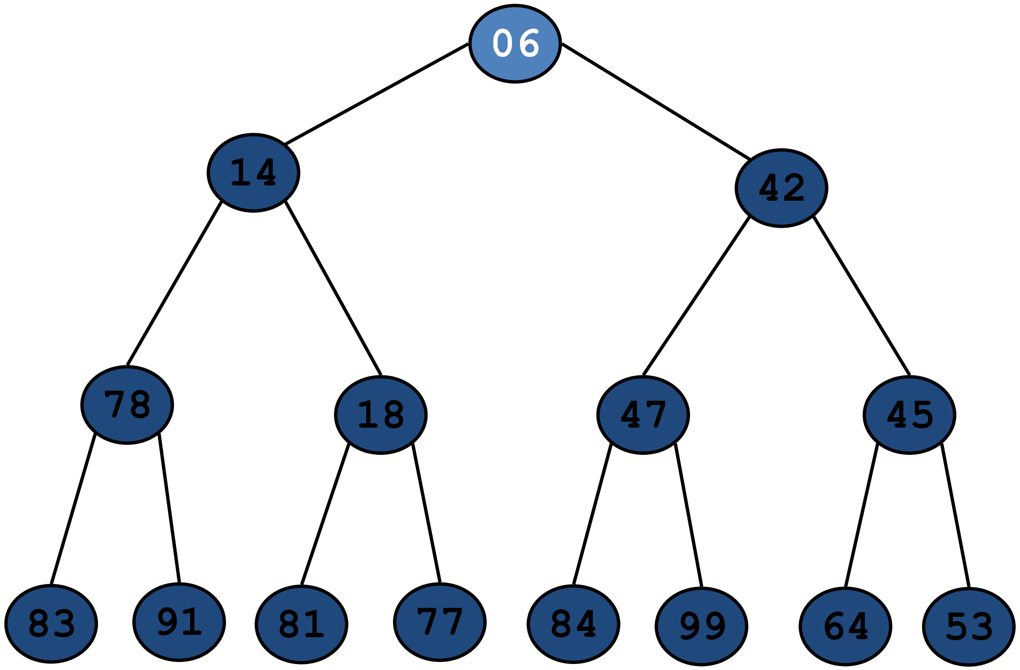
```
Algorithm Kruskal(E,cost,n,t)
//E is the set of edges in G
//n is the number of vertices
//cost is a matrix such that cost[u,v] is the cost of the
//edge(u,v)
//t is the array of  edges included in the MST.
```

```
{
    construct the min heap of edges in E based on cost;
    i:=1;mincost:=0;
    while(i<n and heap not empty) do
    {
        Delete minimum cost edge (u,v) from the heap
        j:=Find(u);k:=Find(v);
        if (j≠k) then
        {
            t[i,1]:=u;t[i,2]:=v;
            mincost:=mincost+cost[u,v];
            Union(j,k);
            i:=i+1;
        }
    }
    if (i≠n) then write ("No Spanning Tree");
    return mincost;
}
```

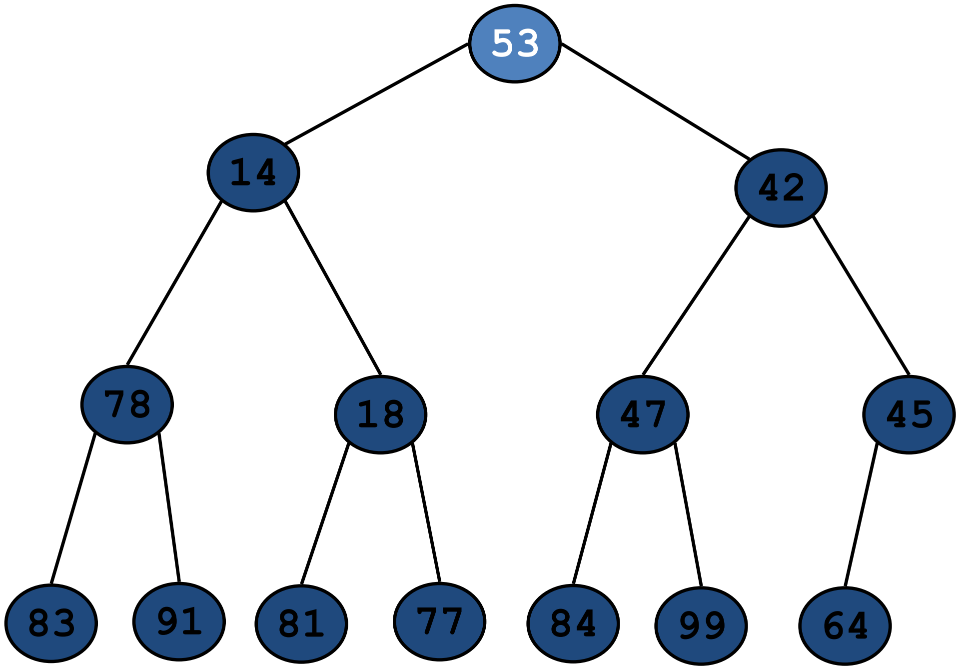
Kruskal's Algorithm

- The complexity of the algorithm is determined by the Heap operations.
- Every time a delete-min is performed, the complexity is $\log|E|$
- We delete and inspect at most $|E|$edges.
- Hence $|E|\log|E|$

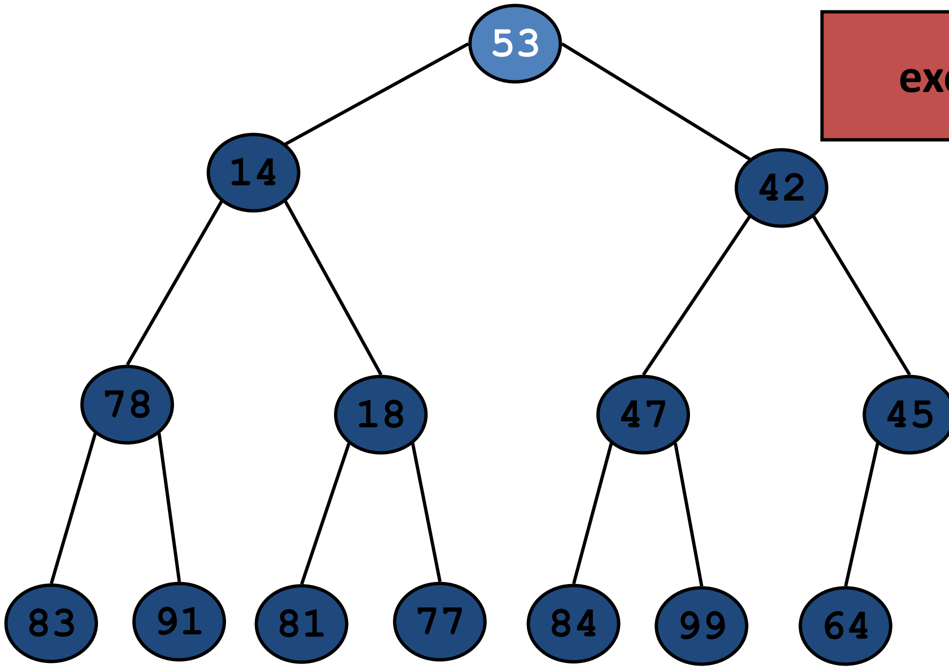
Binary Heap: Delete Min



Binary Heap: Delete Min

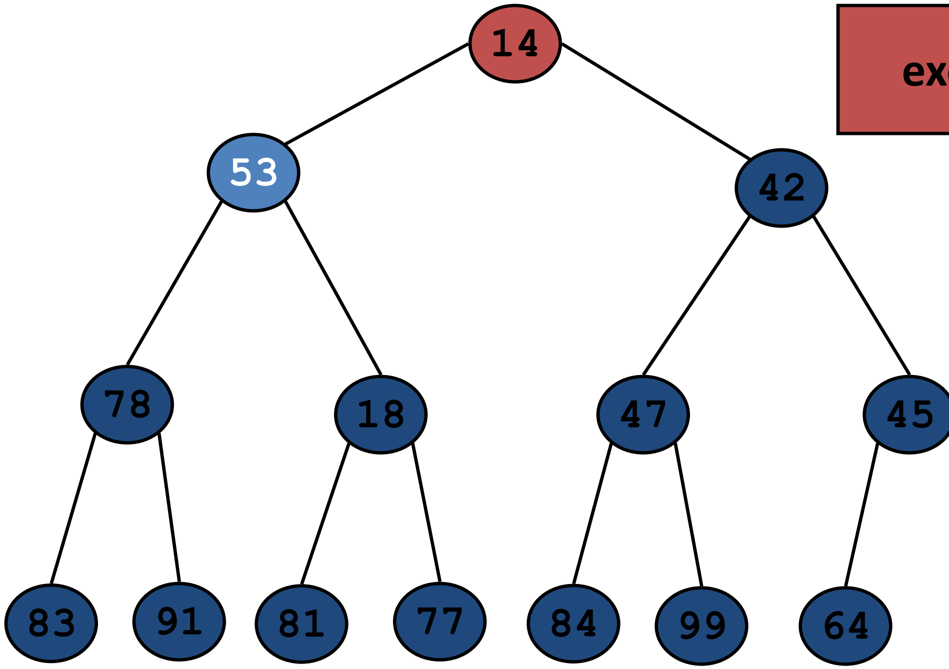


Binary Heap: Delete Min



exchange with left child

Binary Heap: Delete Min



exchange with right child

Binary Heap: Delete Min