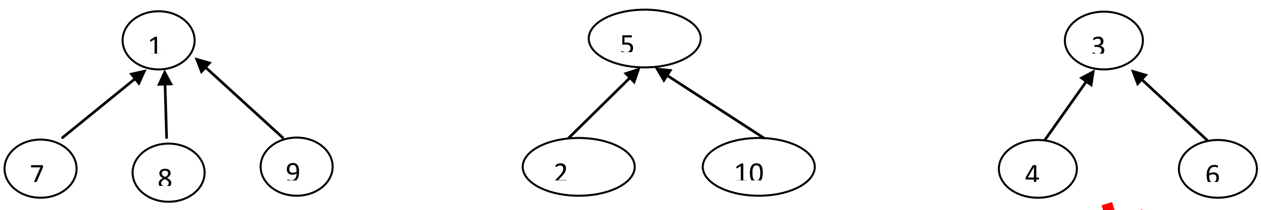
UNIT-II

Sets and Disjoint Set Union : set is a group of elements. Set consists of { 1, 2, 3, … n} numbers as their elements. If S_i and S_j , I # j are two sets , these two sets are called pairwise disjoint, when there is no element that is in both S_i and S_j

For example n=10, elements are partitioned into 3 disjoint sets s1 = { 1, 7, 8, 9 } , s2 = { 2, 5, 10 } S3 = { 3, 4, 6}

Representation of sets as trees :

S1 S2 S3



Operations that can be performed on these set are

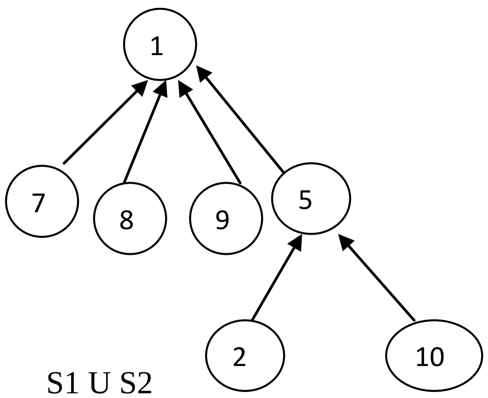
1. Disjoint Set Union : if S_i and S_j are two disjoint sets, then their union S_i U S_j = all elements x such that x is in S_i or S_j. since all the sets are disjoint the sets S_i and S_j are replaced by S_i U S_j in collection of sets.

   Example : S1 U S2 = { 1, 7, 8, 9, 2, 5, 10 }

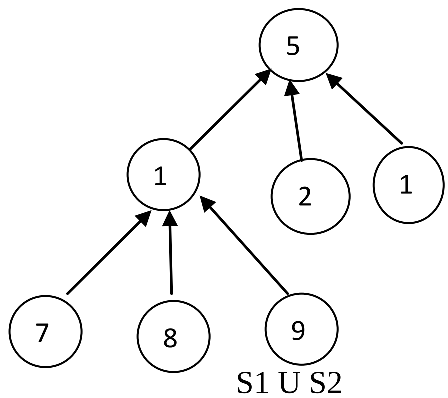
2. Find(i) : Given an element i, find the set containing i. thus , 4 is in set S3, and 9 is in Set S1 etc.

Union and Find Operations :

If we want to obtain the union of S1 and S2. Make one of the trees a sub tree of the other. S1US2 could then have one of the representations.



S1 U S2



S1 U S2

If adjacency lists are used, a BFT will obtain the connected components in Θ(n + e) time.

In the similar way DFT can also be used to find the connected components by modifying DFS algorithm.

Spanning Trees :

The graph G has a spanning tree iff G is connected. BFS easily determines the existence of a spanning tree. Modify the algorithm BFS by adding statements t := 0 ; initially and t = tU{(u, w)} when a new vertex is visited. Call the resulting algorithm BFS*. If BFS* is called with 'v' any vertex on connected undirected graph g, then on termination, the edges in t form a spanning tree of G. the spanning tree obtained using BFS is called Breadth first spanning tree.

```
Algorithm BFSTree(v)
//  Breadth first search Spanning tree using BFS
{
u := v ;
visited[v] := 1;
t := 0;
repeat
  {
   For all vertices w adjacent from u do
   {
      If ( visited [w] = 0 ) then
      {
         Add w to q;   // w is unexplored
          Visited[w] := 1;
t := t U { ( u,w) };
      }
  }
If q is empty then return ;    // no explored vertex
Delete the next element u from q ;   // get next  unexplored vertex.
} until ( false );
```

}

Similarly If DFS Is Modified by adding t := 0 and t = t U {(u,w)} when it terminates the edges in t define a spanning tree for the undirected graph G, if G is connected. A spanning tree obtained in this manner is called a depth first spanning tree.

```
Algorithm DFSTree (v)
// Depth first spanning Tree using DFS
{
  Visited [ v ] := 1;
t := 0;
for each vertex w adjacent from v do
   {
       If ( visited [w] = 0 ) then
       {
       t := t U { ( v , w ) }
       DFS(w);
   }
}
```


Example :