Preliminary Definitions

A set is a collection of objects.

Set A is a subset of set B if all elements of A are in B.

Subsets are sets

Union of two sets A and B is a set C which consists of all elements in A and B

Two sets are mutually disjoint if they do not have a common element.

A partition of a set is a collection of subsets such that

Union of all these subsets is the set itself

Any two subsets are mutually disjoint

S = {1,2,3,4}, A = {1,2}, B = {3,4}, C = {2,3,4}, D = {4}

Is A, B a partition of S? Yes

Is A, C partition of S? No

Is A, D partition of S? No

Union and Find Operations

Operations on partitions.

Union

Need to form union of two different sets of a partition

Find

Need to find out which set an element belongs to

Every set in the partition has a number.

The numbers can be anything as long as different sets have distinct numbers.

Find(a) returns the number of the set containing a.

Can two different sets contain the same element?

No, the sets in a partition are disjoint

Disjoint Set Data Structure

Every element has a number.

Elements of a set are stored in a tree (not necessarily binary)

The set is represented by the root of the tree.

The number assigned to a set is the number of the root element.

Pseudo Code for Find

```
Find(a) {
    If S[a] = 0, return a;
    else Find(S[a]);
     return;
        }
```

Complexity? O(N)

Pseudo-Code for Union

```
Union(root 1, root 2)
    {
       S[root2] = root1;
     }
```

Complexity? O(1)

Pseudo Code for Find

```
Find(a) {
    If S[a] < 0, return a;
    else Find(S[a]);
     return;
          }
```

Complexity Analysis for Find Operation

If the depth (distance from root) of a node A increases, then the earlier tree consisting the node A becomes a subtree of another.

Since only a smaller tree becomes a subtree of another, total size of the combined tree must be at least twice the previous one consisting A.

Each time depth of a node increases, the size of the tree increases by at least a factor of 2.

At first every node has depth 0

Next time depth is 1, tree size is at least 2,

depth is 2, tree size is at least 4…

depth is k, tree size is at least 2^k

We know that $2^k <= N$

Thus $k <= \log N$

Depth of any tree is at most log N

Complexity of Find operation is O(log N)

Complexity of any M operations is O(MlogN)

Pseudo Code for New Find

```
Find(a) {
    If S[a] < 0, return a;
    else S[a]=Find(S[a]);
     return;
         }
```

Complexity Analysis

Any M operations take $O(M\log^*N)$ if M is $\Omega(N)$

$\log^*N$ is the number of times we take loglog….logN so as to get a number less than or equal to 1 (log base 2, even otherwise asymptotic order remains the same).

$\log^*N$ grows very slowly with N and is less than 4 or 5 for all practical values of N,

$\log^*2^{32}$ is less than 5

Thus the worst case complexity is linear for all practical purposes.

Assignment

Q.1)Write Find_set opertion & Disjoint set operation of disjoint set.

Q.2)Explain data representation & array representation of set S1,S2,S3 where S1={1,7,8,9},S2={2,5,10},S3={3,4,6}.

Q.3)Draw possible representation of S1, S2 where S1={1,7,8,9},S2={2,5,10},