DSU

The following capabilities are provided by this data structure. Several elements are supplied to us, each of which is a separate set. A DSU will be able to combine any two sets and will be able to recognise which set a certain element belongs to. The traditional version adds a third operation: it may make a set out of a single element.

- `make_set(x)` - creates a new set consisting of the new element `x`
- `union_sets(x, y)` - merges the two specified sets (the set in which the element `x` is located, and the set in which the element `y` is located)
- ``find_set(v)` - returns the representative (also called leader) of the set that contains the element `v``

Parent Pointer π : Points to the parent of the node

Rank : Height of the subtree hanging from that node.

MakeSet

```
procedure make_set(x)
π(x) = x
rank(x) = 1
```

Findset

```
procedure find_set(int x) {
    if (x == parent[x])
        return x;
    return parent[x] = find_set(parentxv]);
}
```

Merging 2 sets

Merging two sets is easy just make the root of one point to the root of the other. But for the tree to optimal it should be now to keep the tree nearly what we do always make the shorter tree connect to a longer tree.

```
procedure union_set(x, y)
r_x = find(x)
r_y = find(y)
if r_x = r_y: return
if rank(r_x) > rank(r_y): //making the root of the shorter tree connect to longer
//tree for optimizationa
    π(r_y) = r_x
else:
    π(r_x) = r_y
    if rank(r_x) = rank(r_y) : rank(r_y) = rank(r_y) + 1
```

Properties of rank

Property 1 : For any x, rank(x) < rank(π(x)). This is very obvious as we go up the tree height of the tree strictly increases.

Property 2 : Any root node of rank k has at least 2^k nodes in its tree. This also true for any of the node at rank(since they were also root once). A root node rank k can be created by merging 2 ran k(k-1) roots(from union pseudo code).Now using induction $2^{k-1} + 2^{k-1} = 2^k$

Property 3 : If there are n elements overall, there can be at most n/2k nodes of rank k.

Time Analysis

- `Makeset` : $O(1)$ thi
- `Find` : $O(\text{Height of tree }) = O(\log n)$
- `Union` : Find will take $O(\log{(n)})$ and assigning parent $O(1)$. Finally = $O(\log n)$