AKSHAY MITTUR
IBM18CS010

07-10-20

Number of Islands ⇒ Disjoint Sets

Algo:

i) Start
ii) int result=0
iii) traverse through matrix,
  i) if arr[i][j]= 1
    ⇒ check its neighbours
    ⇒ if neighbour=1, take union of it and its neighbour
iv) Create array to store frequency of sets
v) Traverse matrix again, if arr[i][j] ⇒ find its set
vi) If it's frequency is 0, result += 1.

function : Pseudo Code

```
int countIsland(vector<vector<int>> a) {
    int n= size(a)
    int m= size(a[0])
    dus => disjointunionset(n*m)
    for j in (0→n):
        for k in(0→m):
            if a[j][k]= 0 continue;
            if j+1<n & a[j+1][k]==1
            dus.union(j*m+k,(j+1)*m+k)
            if(j-1>=0 & a[j-1][k]==1
            dus.union(j*m+k, j*m+k+1)
            dif(k+1<m & a[j][k+1] == 1
              dus.Union(j*m+k, j*m+(k+1)
```

(PTO)

AKSHAY MITTUR
1BM18CS010

7-6-20

```
if(j+1<n & k+1<m & a[j+1][k+1]=1)
    dus.union(j*m+k, (j+1)*m+k+1)
if(j+1<n && k-1>=0 & a[j+1][k-1]==1)
    dus.Union(j*m+k, (j-1)*m+k+1)
if(j-1>=0 && k-1>=0 & a[j-1][k-1]=1)
    dus.Union(j*m+k, (j-1)*m+k-1)
```

Disjoint Union Sets:

```
vector<int> rank, parent;
int n;
DisjointUnionSets(int n){
    rank.resize(n);
    parent.resize(n);
    this->n = n;
    iota(begin(parent), end(parent), 0);
}
int find(int x){
    if(parent[x] != x)
        return find(parent[x]);
    return x;
}
```