

```

int *c = new int [m * m];
int numberOfIslands = 0;
for (int j = 0; j < n; j++)
{
    for (int k = 0; k < m; k++)
    {
        if (a[j][k] == 1)
        {
            int x = dfs -> find (j * m + k);
            if (c[x] == 0)
            {
                numberOfIslands ++;
                c[x] ++;
            }
            else
                c[x] ++;
        }
    }
}
return numberOfIslands;
}

```

©Rhoduti

Prog-3 :

Find the number of islands.

Algorithm

```
int countIslands (vector<vector<int>>a)
```

```
{
```

```
    int n = a.size ();
```

```
    int m = a[0].size ();
```

```
    DisjointUnionSets *dus = new newdisjointSet (n*m);
```

```
    for (int j=0; j<n; j++)
```

```
    {
```

```
        for (int k=0; k<m; k++)
```

```
        {
```

```
            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 (k+1<m && a[j][k+1] == 1)
```

```
                dus->union (j*(m)+k, (j)*(m)+k+1);
```

```
            if (j-1>=0 && a[j-1][k] == 1)
```

```
                dus->union (j*(m)+k, (j-1)*(m)+k);
```

```
            if (k-1>=0 && a[j][k-1] == 1)
```

```
                dus->union (j*(m)+k, (j)*(m)+k-1);
```

```
            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<m && 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);
```

```
        }
```

```
    }
```