Brief Description of the problem:

Samsung wants to introduce the promotions in mobile sale. They want to setup a booth to sell mobile phones. They will club all areas of same mobile model with 1 single booth. In areas (marked zero), where Samsung mobile is not established, they will consider selling the mobiles that are famous in adjacent areas of higher configuration.

We are given N*N matrix which can have a number between 0 and 5.

Sample input:

5	5	1	4	4
4	0	2	4	2
5	0	0	2	0
5	4	3	0	1
1	3	3	2	1

Sample Output:

5	5	1	4	4
4	5	2	4	2
5	5	5	2	2
5	4	3	3	1
1	3	3	2	1

In this case, 11 independent clusters are formed. This means 11 booths have to be setup.

We need to consider selling the mobile that are famous in adjacent areas. In below case, count is like this:
S3 mobile – 3
S2 mobile – 2
S1 mobile – 2

As S3 is famous in neighboring area, so S3 will be considered for selling at this location.

		2	0
4	3	0	1
3	3	2	1

Problem Analysis:

We need to find clusters of Samsung mobiles around the place where Samsung mobile is not established yet (marked 0 in matrix). After finding clusters, we can choose the famous handset of higher configuration.

Let’s take below example,

5	5		
4	0	2	
5	0	0	2
5	4	3	
	3	3	

While calculating for 0 located at (1,1) position in matrix, we need to find clusters around all the 0s which are connected with each other. For this group of 0, we can find clusters of Samsung mobiles around it and replace these 0s with famous mobile having highest configuration.

We can calculate cluster using BFS or DFS. For above case, we will find below clusters around these 0s:

S5 Mobile – 4 (Marked in Orange)
S4 Mobile - 2 (marked in red)
S3 Mobile – 3
S2 Mobile – 2
S1 Mobile - 0

Highest configuration famous mobile is S5; hence we can replace these 0s with S5 mobile.

5	5		
4	5	2	
5	5	5	2
5	4	3	
	3	3	

Approaches to the problem of traversal:

1) **<u>Recursive Depth-first Search:</u>**
   Find clusters around “Zero” or “Group of Zeros” which are connected using recursive DFS approach. Its simple approach using recursive Calls. However, it would be little slow as compare to other two approaches. DFS solution is given by Sasi. (Source: from sasi folder)

   main_DFS.cpp

2) **<u>DSF using Stack (Non Recursive):</u>**
   We can avoid recursive function calls by implementing our own stack.

```
STACK s
visited[ ]
DFS(v)
    push( s, v )
    WHILE NOT isEmpty( s )
        v ← pop(s)
        IF NOT visited[v]
            visit( v )
            FOR each w in adjacency( v )  //adjacent in UP, RIGHT, DOWN, LEFT  dir
                IF NOT visited[w] and Value is same as previous Node
                    push(s, w)
```

3) <u>Breadth-First Search:</u>

We can find clusters around Zeros using BFS approach as well. BFS would need queue implementation.

```
BFS(G,v) // Graph G, Starting point of search v
    Generate queue
    Insert the starting point v to the queue
    Indicate the point v as visited
    WHILE queue is not empty
        t ← return the first element of queue
        FOR every line connected to t
            u ← neighboring point of t
            If u is not visited point, and value is same as pervious Node
            insert u into queue, and indicate as visited
```

main_BFS.cpp

Sample Input:

input_g.txt

Sample Output:

```
#1: 11
#2: 31
#3: 130
```