LAB ASSIGNMENT

Flood Fill Algorithm
1. 8-connected

```
from collections import deque as dq
def floodFill8(matrix, row, column, target_x, target_y, target_color):
        color = matrix[target_x][target_y]
        que = dq([(target_x,target_y)])
        X = [0,0,-1,1,1,-1,1,-1]
        Y = [1,-1,0,0,1,1,-1,-1]
        while que:
                current_x, current_y= que.pop()
                matrix[current_x][curr_y] = target_color
                for i in range(8):
                        next_x = current_x + X[i]
                        next_y = current_y+ Y[i]
                        if(next_y>=0 and next_y<column and next_x>=0 and next_x<row
                        and matrix[next_x][next_y] == color):
                                matrix[next_x][next_y] = target_color
                                que.append((next_x,next_y))
                for i in range(row):
                        for j in range(column):
                                print(matrix[i][j],end=" ")
                        print()
                print()
        matrix=[]
        rows = int(input())
        columns = int(input())
        for row in range(rows):
                l=list(map(int,input().split()))
                matrix.append(l)
        t=int(input())
        x,y = map(int,input().split())
        floodFill8(matrix, rows, columns, x, y, t)
```

2. 4-connected:

```
from collections import deque as dq
def flood_Fill_4(matrix, row, column, target_x, target_y, target_color):
        color = matrix[target_x][target_y]
        que = dq([(target_x,target_y)])
        X = [0,0,-1,1]
        Y = [1,-1,0,0]
        while que:
                current_x, current_y= que.pop()
                matrix[current_x][curr_y] = target_color
                for i in range(4):
                        next_x = current_x + X[i]
                        next_y = current_y+ Y[i]
                        if(next_y>=0 and next_y<column and next_x>=0 and next_x<row
                        and matrix[next_x][next_y] == color):
                                matrix[next_x][next_y] = target_color
                                que.append((next_x,next_y))
                for i in range(row):
                        print(*matrix[i])
                print("----------------------")

        matrix=[]
        print("Enter matrix dimension:")
        print("Enter no of rows")
        rows = int(input())
        print("Enter no of col:")
        columns = int(input())
        print("Enter matrix:")

        for row in range(rows):
                l=list(map(int,input().split()))
                matrix.append(l)
        print("Enter color")
        t=int(input())
        print("Enter target cell")
```

```
    x,y = map(int,input().split())
    floodFill4(matrix, rows, columns, x, y, t)
```

Boundary Fill Algorithm
1.8-Connected:

```
def bound_fill_8(matrix,color,bound_color,n):
    i,j=0,0
    q=[]
    q.append([i,j])
    f[n*i+j]=1
    while(q):
        x,y=q.pop()
        if x+1<n and matrix[x+1][y]!=bound_color and f[n*(x+1)+y]==0:
            matrix[x+1][y]=color
            q.append([x+1,y])
            f[n*(x+1)+y]=1
        if y+1<n and matrix[x][y+1]!=bound_color and f[n*x+y+1]==0:
            matrix[x][y+1]=color
            q.append([x,y+1])
            f[n*(x)+y+1]=1
        if x-1>=0 and matrix[x-1][y]!=bound_color and f[n*(x-1)+y]==0:
            matrix[x-1][y]=color
            q.append([x-1,y])
            f[n*(x-1)+y]=1
        if y-1>=0 and matrix[x][y-1]!=bound_color and f[n*x+y-1]==0:
            matrix[x][y-1]=color
            q.append([x,y-1])
            f[n*(x)+y-1]=1
        if x+1<n and y+1<n and matrix[x+1][y+1]!=bound_color and
        f[n*(x+1)+y+1]==0:
            matrix[x+1][y+1]=color
            q.append([x+1,y+1])
            f[n*(x+1)+y+1]=1
```

```
            if x-1>=0 and y+1<n and matrix[x-1][y+1]!=bound_color and
            f[n*(x-1)+y+1]==0:
                    matrix[x-1][y+1]=color
                    q.append([x-1,y+1])
                    f[n*(x-1)+y+1]=1
            if x-1>=0 and y-1>=0 and matrix[x-1][y-1]!=bound_color and
            f[n*(x-1)+y-1]==0:
                    matrix[x-1][y-1]=color
                    q.append([x-1,y-1])
                    f[n*(x-1)+y-1]=1
            if x+1<n and y-1>=0 and matrix[x+1][y-1]!=bound_color and
            f[n*(x+1)+y-1]==0:
                    matrix[x+1][y-1]=color
                    q.append([x+1,y-1])
                    f[n*(x+1)+y-1]=1
            for i in range(n):
                    print(*matrix[i])
            print("----------------------------------")
    return
    print("enter matrix dimension:")
    no_of_rows=int(input())
    matrix=[]
    print("enter matrix:")

    for _ in range(no_of_rows):
            matrix.append(list(map(int,input().split())))
    print("color to be filled:")
    color=int(input())
    bound_color=matrix[0][0]
    f=[0]*(no_of_rows*no_of_rows)
    i,j=1,1
    print("Boundary fill for 4 connected")
    bound_fill_8(matrix,color,bound_color,no_of_rows)
    print("Final colored matrix")

    for i in range(no_of_rows):
            print(*matrix[i])
```

1. 4-connected:

```
def bound_fill_4(matrix,color,bound_color,n):
    i,j=0,0
    q=[]
    q.append([i,j])
    f[n*i+j]=1
    while(q):
        x,y=q.pop()
        if x+1<n and matrix[x+1][y]!=bound_color and f[n*(x+1)+y]==0:
            matrix[x+1][y]=color
            q.append([x+1,y])
            f[n*(x+1)+y]=1
        if y+1<n and matrix[1][y+1]!=bound_color and f[n*x+y+1]==0:
            matrix[1][y+1]=color
            q.append([x,y+1])
            f[n*(x)+y+1]=1
        if x-1>=0 and matrix[x-1][y]!=bound_color and f[n*(x-1)+y]==0:
            matrix[x-1][y]=color
            q.append([x-1,y])
            f[n*(x-1)+y]=1
        if y-1>=0 and matrix[x][y-1]!=bound_color and f[n*x+y-1]==0:
            matrix[x][y-1]=color
            q.append([x,y-1])
            f[n*(x)+y-1]=1
        for i in range(n):
            print(*matrix[i])
        print("---------------------------------")
    return

    print("enter matrix dimension:")
    no_of_rows=int(input())
    matrix=[]
    print("enter matrix:")

    for _ in range(no_of_rows):
        matrix.append(list(map(int,input().split())))
```

```
print("color to be filled:")
color=int(input())
bound_color=matrix[0][0]
f=[0]*(no_of_rows*no_of_rows)
i,j=1,1
print("Boundary fill for 4 connected")
bound_fill_4(matrix,color,bound_color,no_of_rows)
print("Final colored matrix")
for i in range(no_of_rows):
        print(*matrix[i])
```