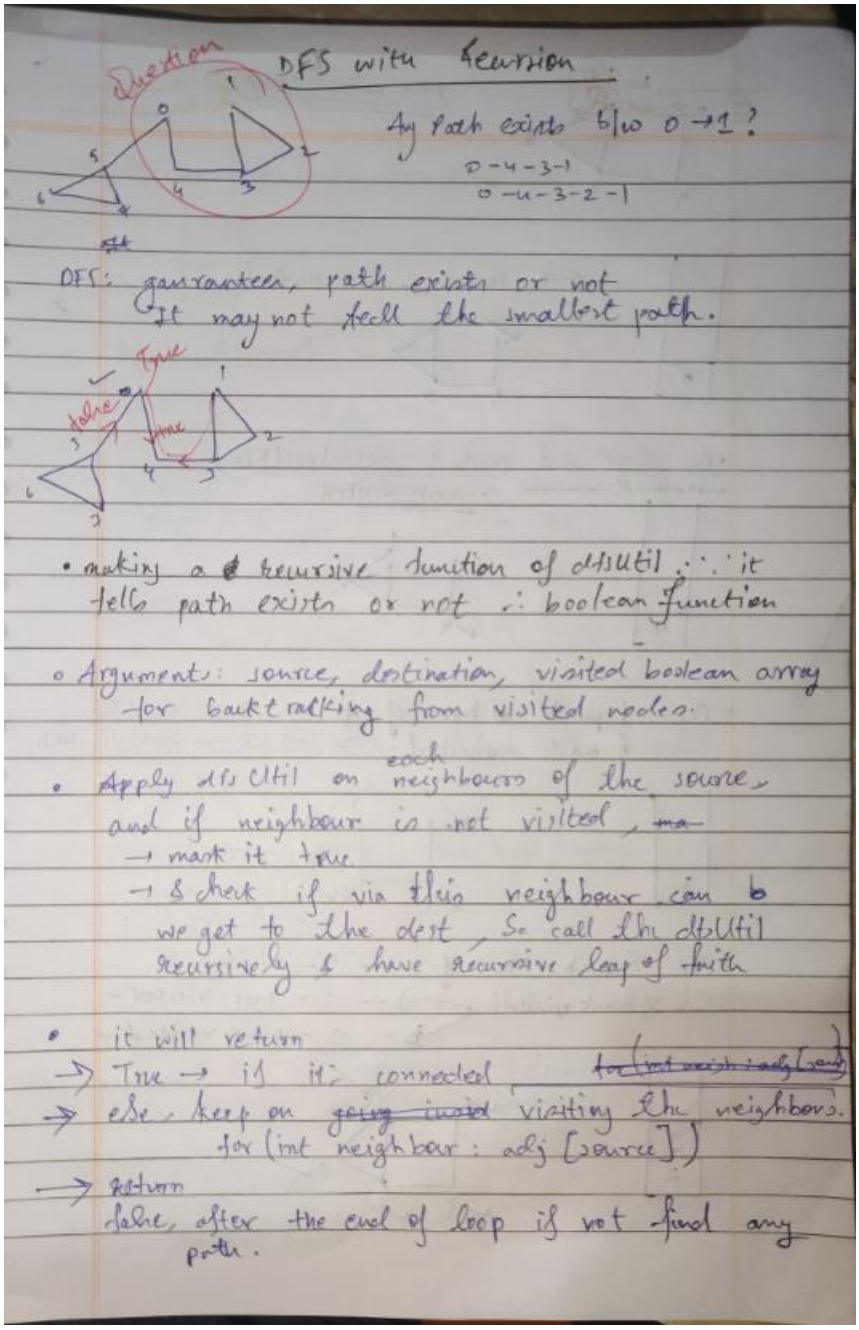
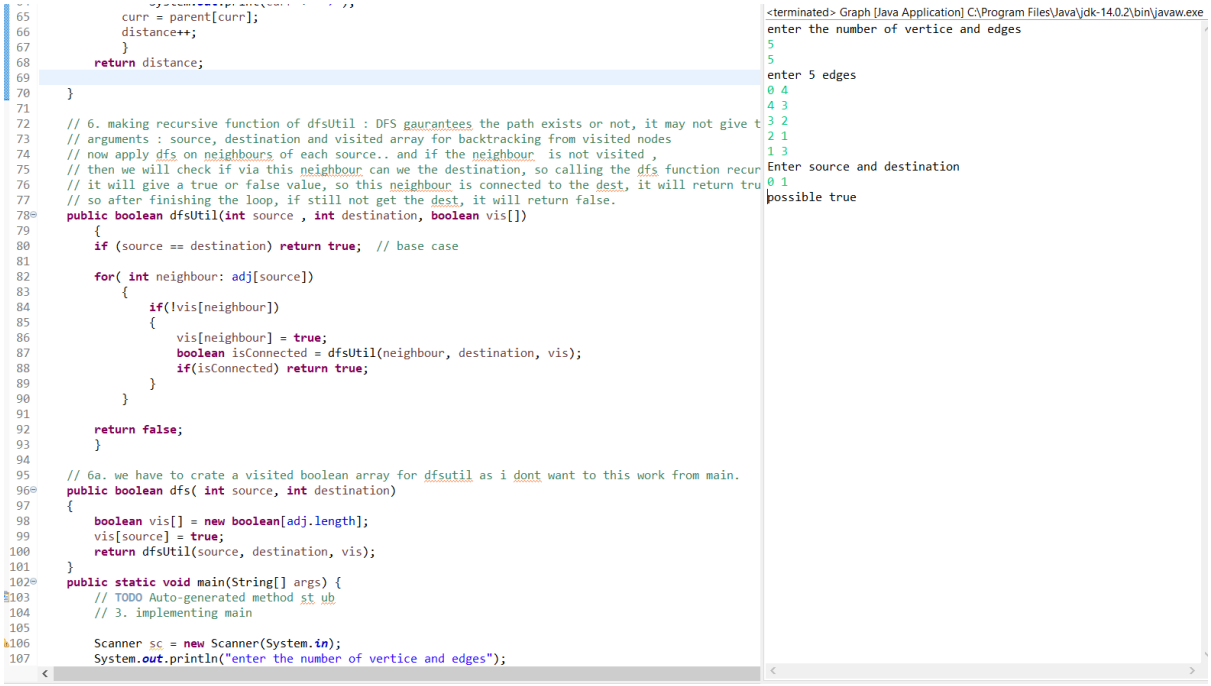
Dfs with stack and recursion



Dfs with stack and recursion

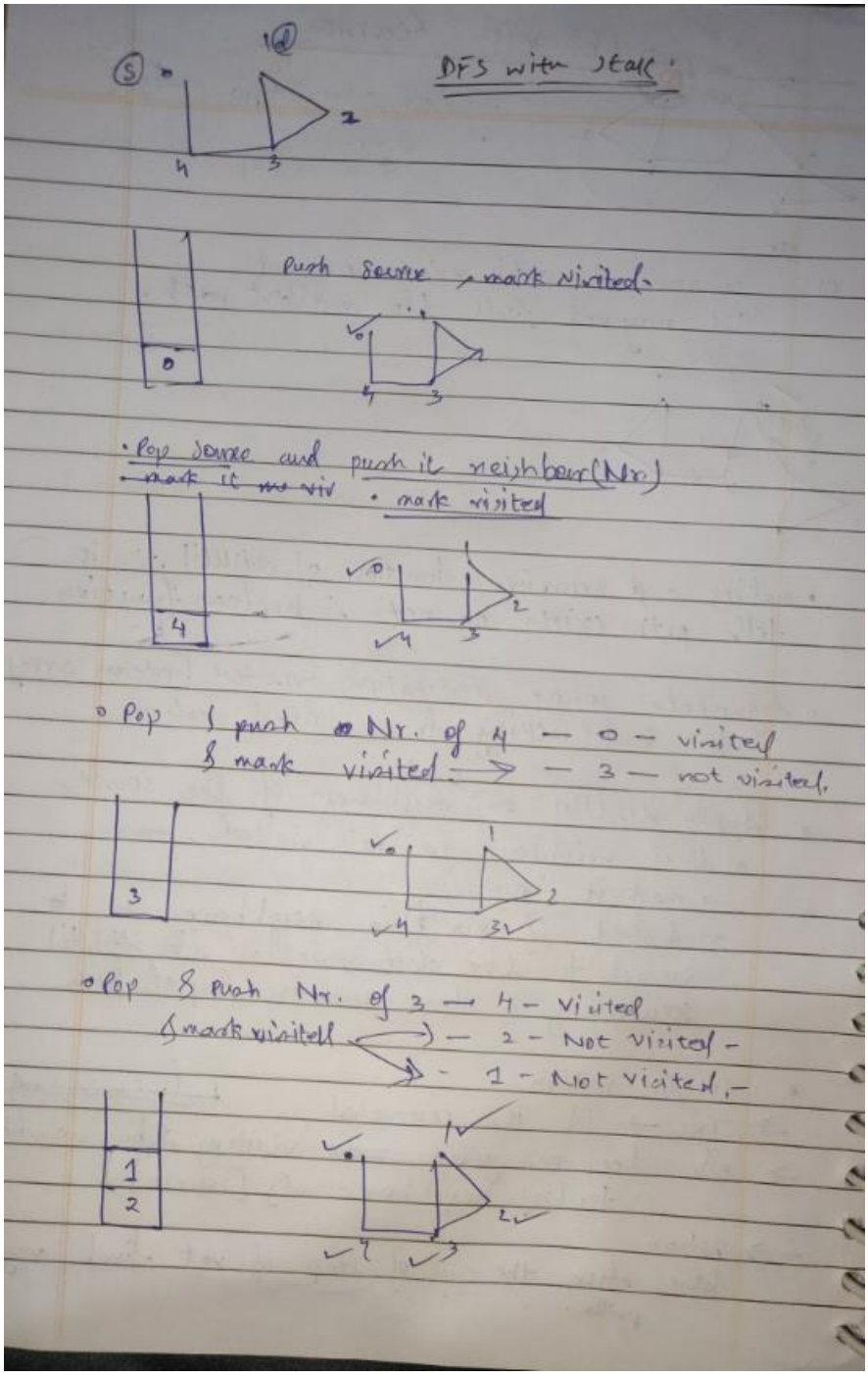


```
            curr = parent[curr];
            distance++;
            }
        return distance;

    }

    // 6. making recursive function of dfsUtil : DFS gaurantees the path exists or not, it may not give t
    // arguments : source, destination and visited array for backtracking from visited nodes
    // now apply dfs on neighbours of each source.. and if the neighbour  is not visited ,
    // then we will check if via this neighbour can we the destination, so calling the dfs function recur
    // it will give a true or false value, so this neighbour is connected to the dest, it will return tru
    // so after finishing the loop, if still not get the dest, it will return false.
    public boolean dfsUtil(int source , int destination, boolean vis[])
        {
        if (source == destination) return true;  // base case

        for( int neighbour: adj[source])
            {
                if(!vis[neighbour])
                {
                    vis[neighbour] = true;
                    boolean isConnected = dfsUtil(neighbour, destination, vis);
                    if(isConnected) return true;
                }
            }

        return false;
        }

    // 6a. we have to crate a visited boolean array for dfsutil as i dont want to this work from main.
    public boolean dfs( int source, int destination)
    {
        boolean vis[] = new boolean[adj.length];
        vis[source] = true;
        return dfsUtil(source, destination, vis);
    }
    public static void main(String[] args) {
        // TODO Auto-generated method st ub
        // 3. implementing main

        Scanner sc = new Scanner(System.in);
        System.out.println("enter the number of vertice and edges");
```

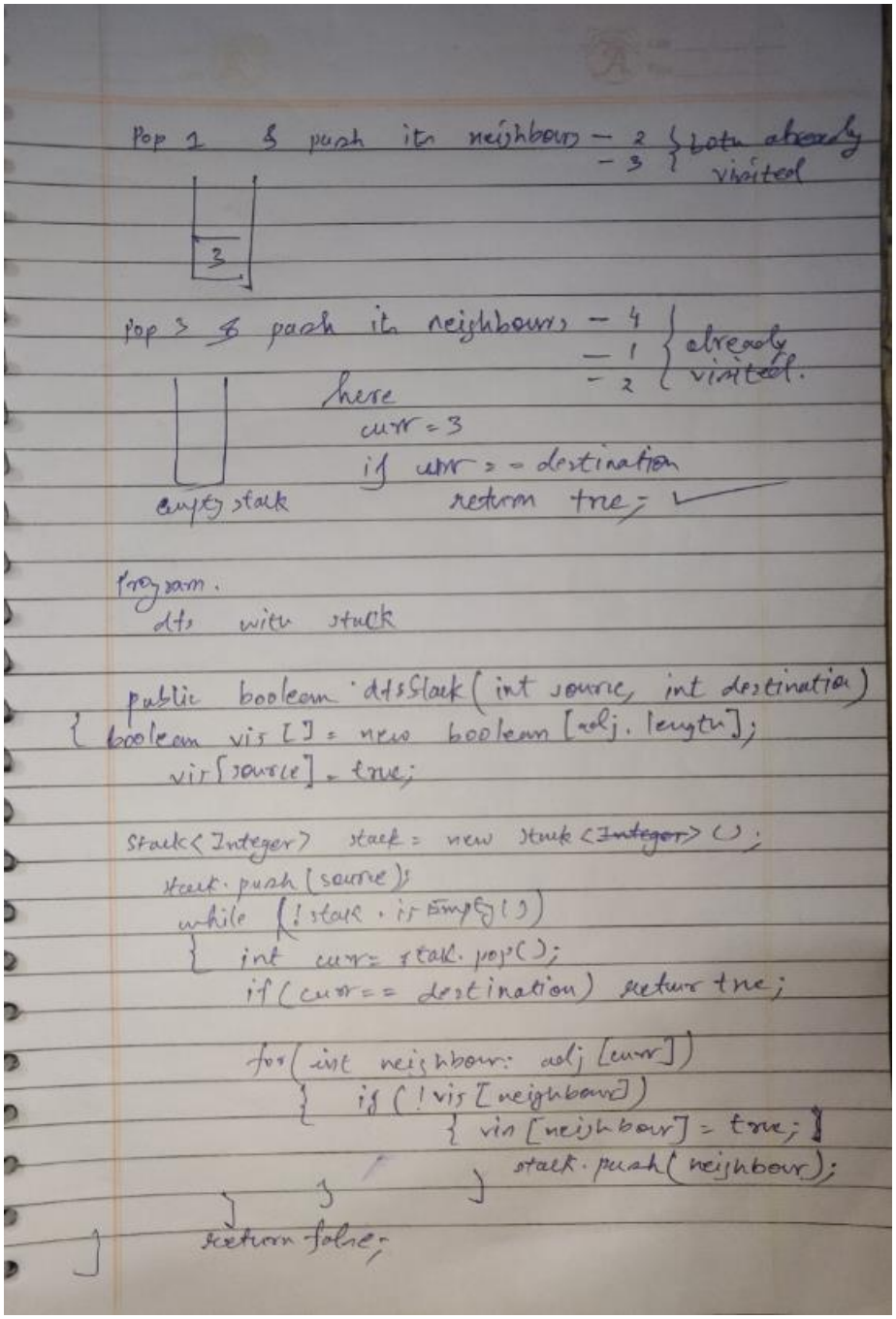
```
<terminated> Graph [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe
enter the number of vertice and edges
5
5
enter 5 edges
0 4
4 3
3 2
2 1
1 3
Enter source and destination
0 1
possible true
```

Dfs with stack and recursion

DFS WITH STACK



Dfs with stack and recursion



Dfs with stack and recursion

```java
//6b lets see how dfs is implemented with stack
// will have source and dest as arg , and a boolean visited array is req.
// creating a stack and push the source.
// and keep on popping the top element=curr and pushing its neighbours, marking them as visited till(
//and add a base condition, if curr == dest  return true line(118) else no path is possible it will r
public boolean dfsStack(int source, int destination)
{
    boolean vis[] = new boolean[adj.length];
    vis[source] = true;
    Stack<Integer> stack = new Stack<>();

    stack.push(source);

    while(!stack.isEmpty())
        {
            int curr = stack.pop();

            if( curr == destination) return true; // base
            for ( int neighbour : adj[curr])
                {
                    if(!vis[neighbour])
                        {
                            vis[neighbour] = true;
                            stack.push(neighbour);
                        }
                }
        }
    return false;
}
```

```
<terminated> Graph [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe  (
enter the number of vertice and edges
5
5
enter 5 edges
0 4
4 3
3 2
2 1
1 3
Enter source and destination
0 1
possible true
```