

Question DFS with recursion

Any path exists b/w 0 → 1?

0-4-3-1
0-4-3-2-1

DFS: guarantee, path exists or not
It may not tell the smallest path.

True
False

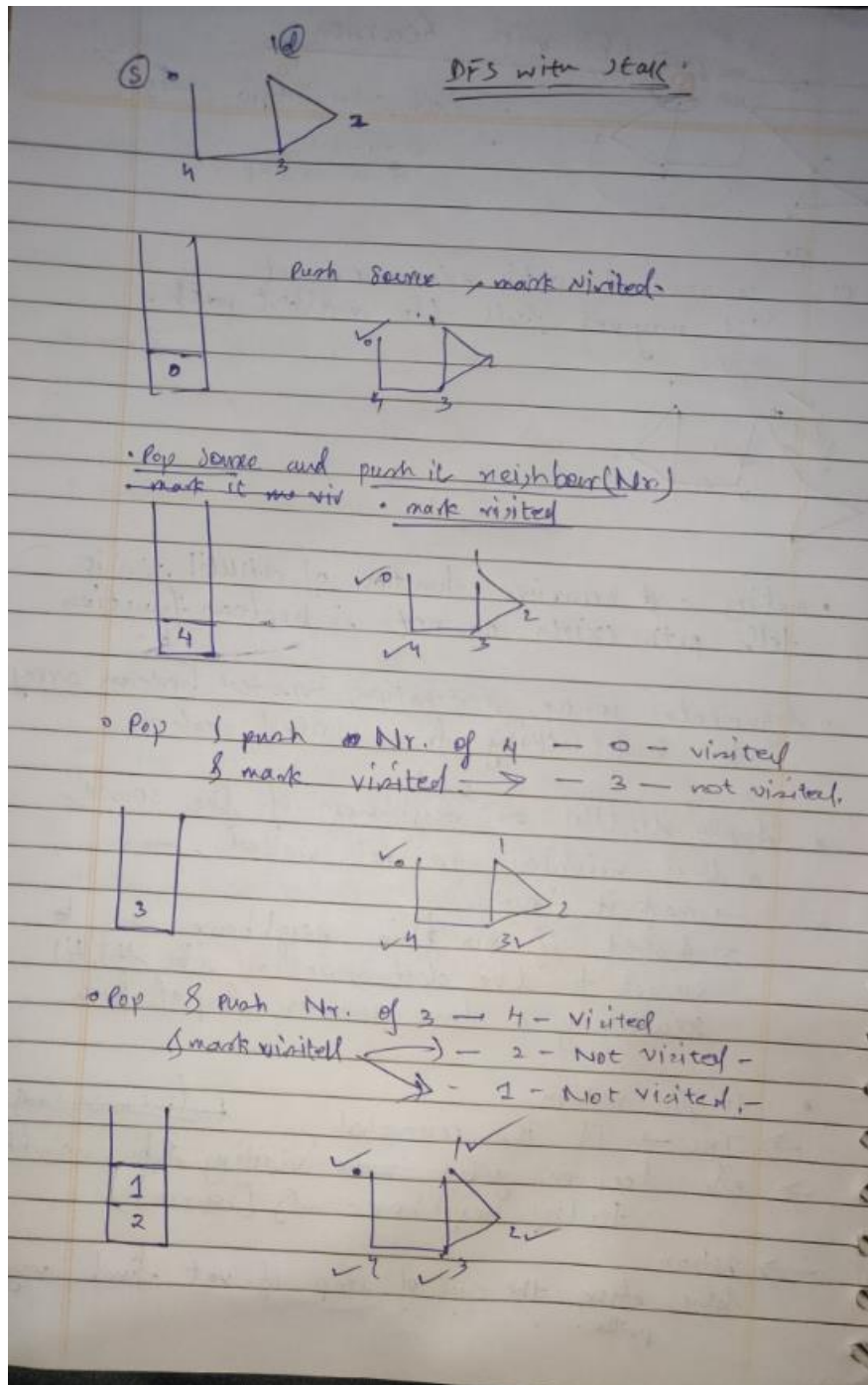
- making a recursive function of dfsUtil ∴ it tells path exists or not ∴ boolean function
- Arguments: source, destination, visited boolean array for backtracking from visited nodes.
- Apply dfsUtil on each neighbour of the source, and if neighbour is not visited, ~~mark~~ → mark it true
→ & check if via this neighbour can we get to the dest, So call the dfsUtil recursively & have recursive loop of truth
- it will return
 - True → if it's connected, ~~for (int neighbour : adj[source])~~
 - else, keep on ~~going inside~~ visiting the neighbors. ~~for (int neighbour : adj[source])~~
 - return false, after the end of loop if not find any path.

Dfs with stack and recursion

```
65         curr = parent[curr];
66         distance++;
67     }
68     return distance;
69 }
70
71 // 6. making recursive function of dfsUtil : DFS gaunantees the path exists or not, it may not give t
72 // arguments : source, destination and visited array for backtracking from visited nodes
73 // now apply dfs on neighbours of each source.. and if the neighbour is not visited ,
74 // then we will check if via this neighbour can we the destination, so calling the dfs function recur
75 // it will give a true or false value, so this neighbour is connected to the dest, it will return tru
76 // so after finishing the loop, if still not get the dest, it will return false.
77 public boolean dfsUtil(int source , int destination, boolean vis[])
78 {
79     if (source == destination) return true; // base case
80     for( int neighbour: adj[source])
81     {
82         if(!vis[neighbour])
83         {
84             vis[neighbour] = true;
85             boolean isConnected = dfsUtil(neighbour, destination, vis);
86             if(isConnected) return true;
87         }
88     }
89     return false;
90 }
91
92 // 6a. we have to crate a visited boolean array for dfsutil as i dont want this work from main.
93 public boolean dfs( int source, int destination)
94 {
95     boolean vis[] = new boolean[adj.length];
96     vis[source] = true;
97     return dfsUtil(source, destination, vis);
98 }
99
100 public static void main(String[] args) {
101     // TODO Auto-generated method st ub
102     // 3. implementing main
103     Scanner sc = new Scanner(System.in);
104     System.out.println("enter the number of vertice and edges");
105
106     <terminated> Graph [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe
107     enter the number of vertice and edges
108     5
109     enter 5 edges
110     0 4
111     4 3
112     3 2
113     2 1
114     1 3
115     Enter source and destination
116     0 1
117     possible true
```

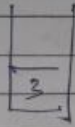
Dfs with stack and recursion

DFS WITH STACK

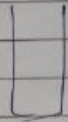


Dfs with stack and recursion

Pop 2 & push its neighbours - 2 } both already
- 3 } visited



Pop 3 & push its neighbours - 4 } already
- 1 } visited.
- 2 }



empty stack

here

curr = 3

if curr == destination

return true; ✓

program.

dfs with stack

```
public boolean dfsStack(int source, int destination)
{
    boolean vis[] = new boolean[adj.length];
    vis[source] = true;
```

```
    Stack<Integer> stack = new Stack<Integer>();
```

```
    stack.push(source);
```

```
    while (!stack.isEmpty())
```

```
    {
        int curr = stack.pop();
```

```
        if (curr == destination) return true;
```

```
        for (int neighbour: adj[curr])
```

```
        {
            if (!vis[neighbour])
```

```
            {
                vis[neighbour] = true;
```

```
                stack.push(neighbour);
```

```
            }
        }
    }
    return false;
```

Dfs with stack and recursion

```
//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
enter 5 edges
0 4
4 3
3 2
2 1
1 3
Enter source and destination
0 1
possible true
```