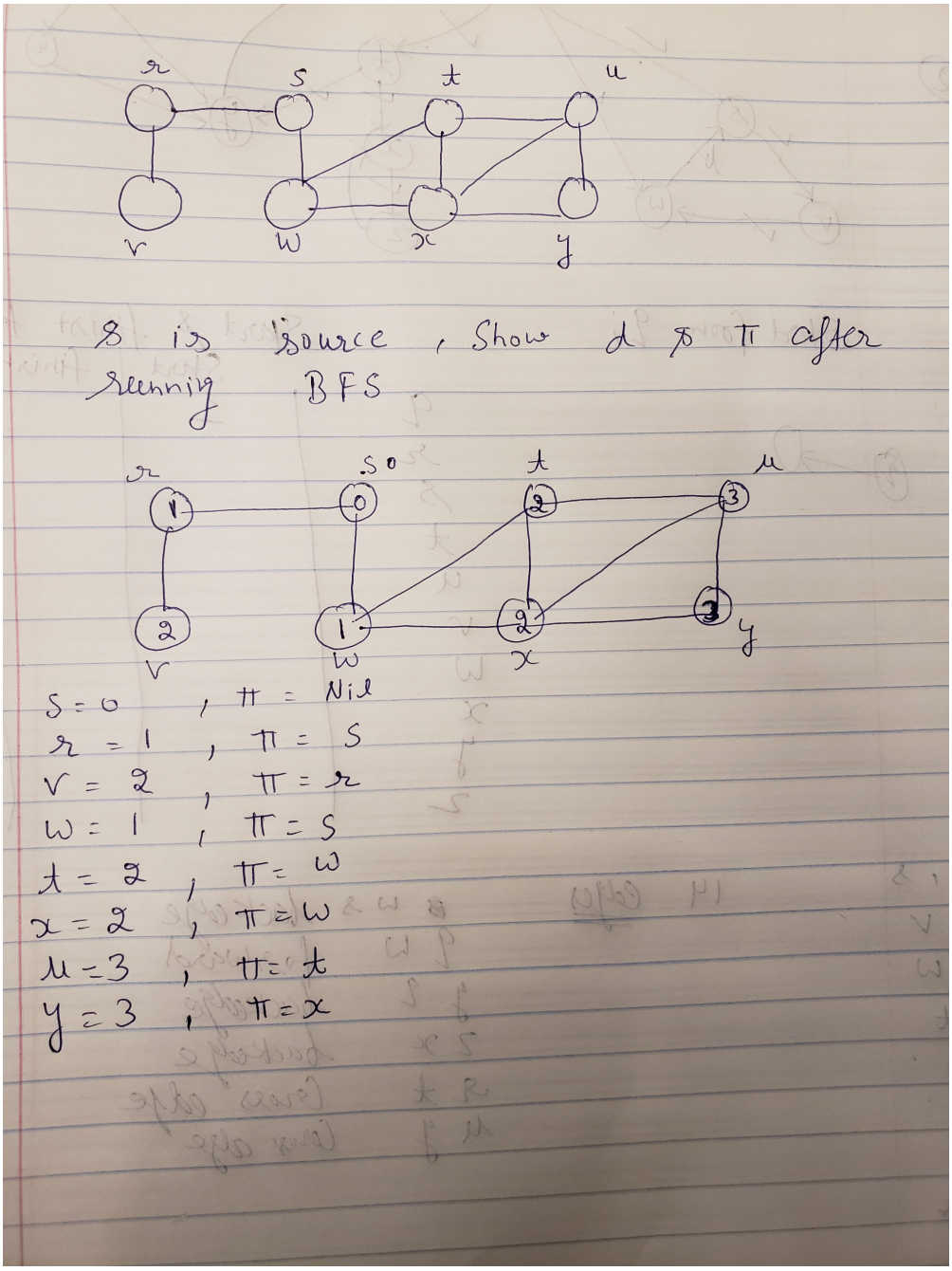
Fundamental Algorithms, Graphs , HW - 10

Tulsi Jain

1. BFS


r
s
t
u
v
w
x
y
s is source, Show d & π after running BFS
r
s 0
t
u
1
0
2
3
2
1
2
3
v
w
x
y
s = 0 , π = Nil
r = 1 , π = s
v = 2 , π = r
w = 1 , π = s
t = 2 , π = w
x = 2 , π = w
u = 3 , π = t
y = 3 , π = x

2. DFS

Start and Finish time

Vertex	Start	End
q	1	16
r	17	20
s	2	7
t	8	15
u	18	19
v	3	6
w	4	5
x	9	12
y	13	14
z	10	11

Edge classification

Tree edges	Back edges	Forward edges	Cross edges
(q, s)	(w, s)	(q, w)	(r, y)
(s, v)	(y, q)		(u, y)
(v, w)	(z, x)		
(q, t)			
(t, x)			
(x, z)			
(t, y)			
(r, u)			

3. Cycle in Undirected Graph

Algorithm 1 IsCycle

```
function IsCycle(G)                                   ▷ where G graph
    for u in G.V do
        u.color = WHITE;
        u.parent = NIL;
    end for
    for u in G.V do
        if u.color == WHITE then
            isCyclecFound = Node-VISIT(G,u)
            if isCyclecFound then
                return true;
            end if
        end if
    end for
    return false;
end function
```

Algorithm 2 Node-VISIT

```
function Node-VISIT(G, u)                             ▷ where G graph, u Vertex
    u.color = GRAY;
    for v in adjacent to u do
        if v.color == GRAY and u.parent != v  then
            return true
        end if
        if v.color == WHITE then
            v.parent = u;
            isCyclecFound = Node-VISIT(G,u)
            if isCyclecFound then
                return true;
            end if
        end if
    end for
    return false;
end function
```

This algorithms looks like depth first search but its running time is like it. DFS running time is O($|V| + |E|$) but above O($|V|$) independent of edges. There are two cases if there is no cycle then $|E|$ would be less then $|V|$ hence running time is $|V|$. If there is a cycle we would be able to find it at or before last vertex. And every time we moves along a edge we traverse to new vertex. Hence running time is O($|V|$).

4. Breadth-First Search properties

4.1 Un-directed

4.1.1 Back and Forward

In breadth first search every node at level, k get discovered before any nodes at level k + 1. Level increased from 0. As given graph is un-directed, from any vertex, u we can move to adjacent vertex, v. Hence, in BFS no forward and backward occurs because node are discovered immediate connection fashion.

4.1.2 Tree Edge v.d = u.d +1

If there is tree edge between and u and v. Then v distance would be u.d + 1 because v is discovered only when moving one step from u to v.

4.1.3 Cross Edge

For cross edge between u and v, only either v.d = u.d or v.d = u.d + 1 is possible. Their is not other relationship possible. Because if one is discovered other can be discovered just moving one step.
Following are the cases then this would occur. For cross link between u and v, v.d = u.d occurs when both both are connected to node whose distance is less than v.d.
For cross link between u and v, v.d = u.d + 1 occurs when v is already discovered by a node who is at same level at u. If v is yet to be discovered than u,v would have become a tree edge.

4.2 Directed

4.2.1 Forward

In breadth first search every node at level, k get discovered before any nodes at level k + 1. Given graph is directed. Let's say a vertex u. Every node connected to u would be discovered if u is connected by forward arrow. And then this arrow would be called tree arrow. Hence no forward arrow in directed graph in BFS but there may exist backward arrow. In this case that node can not be discovered now.

4.2.2 Tree Edge v.d = u.d +1

If there is tree edge between and u and v. Then v distance would be u.d + 1 because v is discovered only when moving one step from u to v.

4.2.3 Cross Edge

There is edge between u to v. So if v is not discovered yet we can discovered in u.d + 1. Hence for v distance u.d + 1 is upper bound. Hence v.d <= u.d + 1.

4.2.4 Back Edge

Path distance can be less then zero. Hence 0 <= v.d. For a back-edge between u to v. It means v is discovered before u. Hence 0 <= v.d <= u.d

5. Topological-Sort

Topological Sort Order P, N, O, S, M, R, Y, V, X, W, Z, U, Q, T
Start and Finish time

Vertex	Start	End
M	1	20
N	21	26
O	22	25
P	27	28
Q	2	5
R	6	19
S	23	24
T	3	4
U	7	8
V	10	17
W	11	14
X	15	16
Y	9	18
Z	12	13

6. Strongly-Connected-Components

Start and Finish time

Vertex	Start	End
q	1	16
r	17	20
s	2	7
t	8	15
u	18	19
v	3	6
w	4	5
x	9	12
y	13	14
z	10	11

Forest found is
r, u, q $\rightarrow$ y $\rightarrow$ t, x $\rightarrow$ z, and s $\rightarrow$ w $\rightarrow$ v