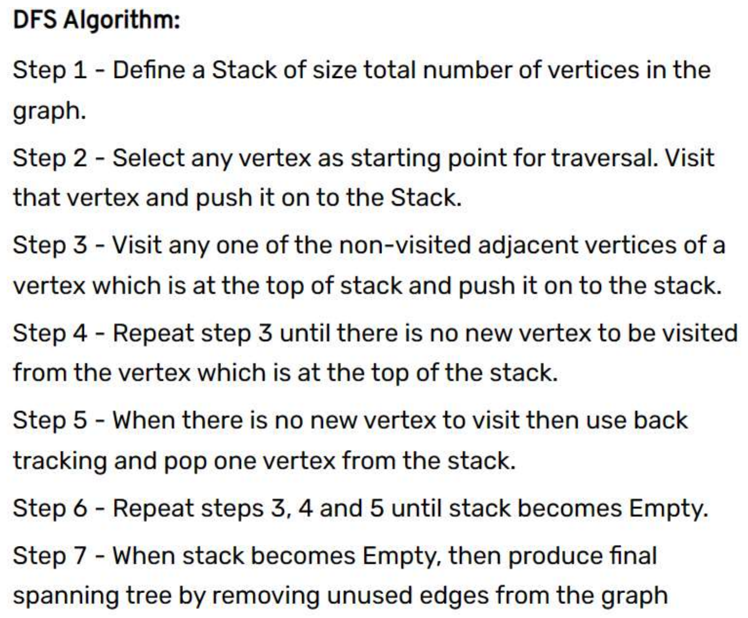
COURSE HANDOUT

Course Code	ACSC13
Course Name	Design and Analysis of Algorithms
Class / Semester	IV SEM
Section	A-SECTION
Name of the Department	CSE-CYBER SECURITY
Employee ID	IARE11023
Employee Name	Dr K RAJENDRA PRASAD
Topic Covered	Depth first search
Course Outcome/s	Use DFS for finding the optimal trees from the graphs
Handout Number	24
Date	

Content about topic covered: Depth First Search

Depth First Search (DFS):

Depth-first Search (DFS) is a method for searching a graph or a tree data structure. The method starts at the root (top) node of a tree and goes as far as it can down a given branch (path). It then goes backwards until it finds an unexplored path, which it then explores.



DFS Algorithm:

Step 1 - Define a Stack of size total number of vertices in the graph.

Step 2 - Select any vertex as starting point for traversal. Visit that vertex and push it on to the Stack.

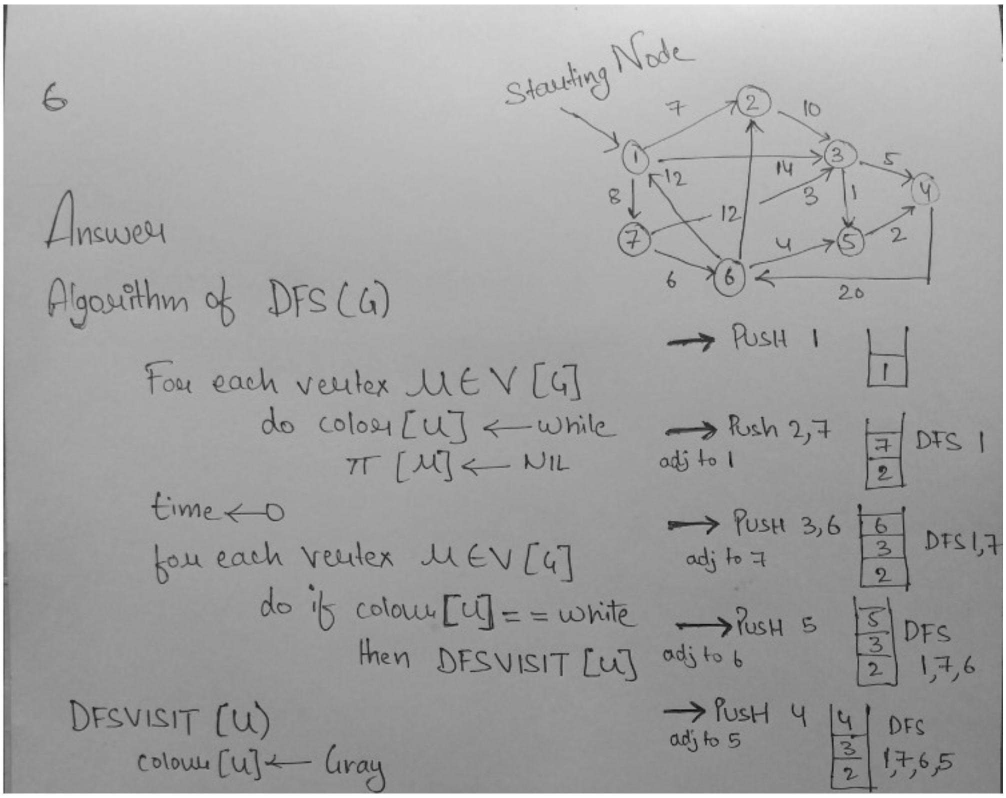
Step 3 - Visit any one of the non-visited adjacent vertices of a vertex which is at the top of stack and push it on to the stack.

Step 4 - Repeat step 3 until there is no new vertex to be visited from the vertex which is at the top of the stack.

Step 5 - When there is no new vertex to visit then use back tracking and pop one vertex from the stack.

Step 6 - Repeat steps 3, 4 and 5 until stack becomes Empty.

Step 7 - When stack becomes Empty, then produce final spanning tree by removing unused edges from the graph



6

Starting Node

1
2
3
4
5
6
7
7
10
14
5
12
8
12
3
1
4
2
6
20

Answer

Algorithm of DFS(G)

```
For each vertex u ∈ V[G]
        do color[u] ← while
           π[u] ← NIL
time ← 0
for each vertex u ∈ V[G]
        do if colour[u] == white
              then DFSVISIT[u]
DFSVISIT(u)
   colour[u] ← Gray
```

→ PUSH 1

1

→ Push 2,7
adj to 1

7
2

DFS 1

→ PUSH 3,6
adj to 7

6
3
2

DFS 1,7

→ PUSH 5
adj to 6

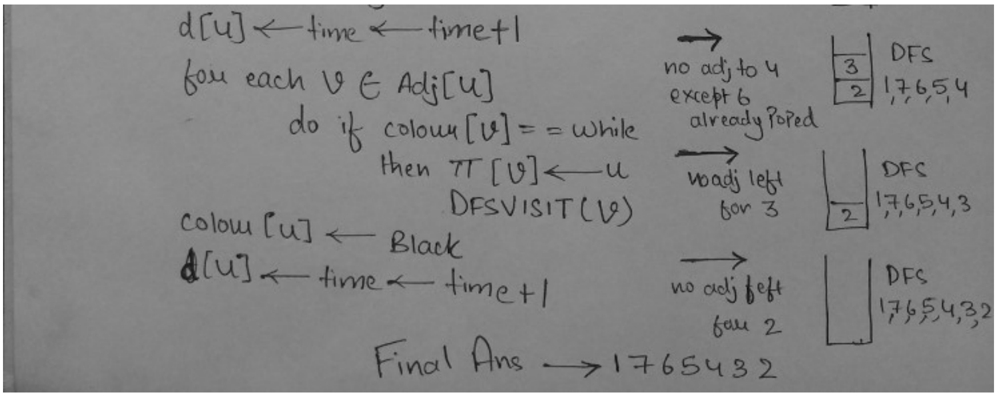
5
3
2

DFS 1,7,6

→ PUSH 4
adj to 5

4
3
2

DFS 1,7,6,5



```
   d[u] ← time ← time+1
   for each v ∈ Adj[u]
        do if colour[v] == while
              then π[v] ← u
                   DFSVISIT(v)
   colour[u] ← Black
   d[u] ← time ← time+1
```

→
no adj to 4
except 6
already Poped

3
2

DFS 1,7,6,5,4

→
no adj left
for 3

2

DFS 1,7,6,5,4,3

→
no adj left
for 2

DFS 1,7,6,5,4,3,2

Final Ans → 1 7 6 5 4 3 2