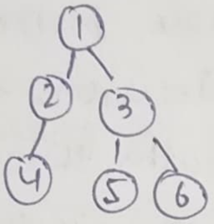
TUTORIAL-5

Q.1 What is difference between DFS and BFS. Please write the applications of both the algorithm.

Ans. BFS (Breadth First Search)- It is used to find shortest path in the graph. It uses a Queue data structure that follows first in first out. In BFS, one vertex/node is selected at a time when it is visited and marked then its adjacent are visited and stored in the queue. It is slower than DFS.

Ex.



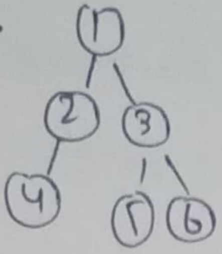
Adj list

1 → 2, 3
2 → 1, 4
3 → 1, 5, 6
4 → 2
5 → 3
6 → 3

BFS traversal → 1, 2, 3, 4, 5, 6

DFS (Depth First search) - It uses stack data structure and performs two stages, first visited vertices are pushed into the stack and second if there are no vertices then visited vertices are popped.

Ex.



DFA travesal

1, 2, 4, 3, 5, 6.

BFS	DFS
1. It uses Queue data structure.	1. It uses stack data structure.
2. It is more suitable for searching vertices which are close to the given source.	2. DFS is more suitable when there are solution away from source.
3. It considers all neighbours first and therefore not suitable for decision making tree used in game or puzzle.	3. DFS is more suitable for game or puzzle problems. We make a decision, then explore all paths through this decision. And if this decision leads to win situation we stop.
4. Time complexity $\rightarrow O(V+E)$	4. Time complexity $\rightarrow O(V+E)$
5 Here siblings are visited before the children	5. Here, children are visited before the siblings.
6 In this there is no concept of back tracking.	6. It is a recursive algo that uses the idea of backtracking

Application of BFS - It is used in various application such as bipartite graph & shortest path etc.

Application of DFS - It is used in various application such as cyclic graph & topological order etc.

Q2) which Data structure are used to implement BFS and DFS & why?

doubt

Ans Bfs uses queue (FIFO - first in first out). Dfs uses stack (Last in first out - LIFO). Bfs uses queue because it gives the shortest path b/w sorcse & des. So it needs something which ~~gi~~ process neighbour nodes first that is why queue is used.

Q3) what do you mean by sparse and dense graph? which representation of graph is better for sparse & dense graph?

Ans A graph in which the number of edges is much less than the possible number of edges is known as sparse graph.

A Dense Graph is a graph in which the number of edges is close to the maximal number of edges.

Adjacency lists can be a good representation of sparse graphs.

Adjacency matrix can be a good representation of dense graph.

Q4] How can you detect a cycle in a graph using DFS and BFS?

Ans In Undirected graph.

In Undirecked graph, As we know that we maintain a visited array which gives us information about which element is visited or not. so if while traversing we get the element which is already visited and that element in not equal to its parent that means graph having cycle.

In Directed graph.

In Directed graph, when we apply dfs for cycle detection we have to make 2 arrays one is visited array and another dfs visited. If the particular element is maked as visited in both array then it means graph having a cycle.

In Directed graph.

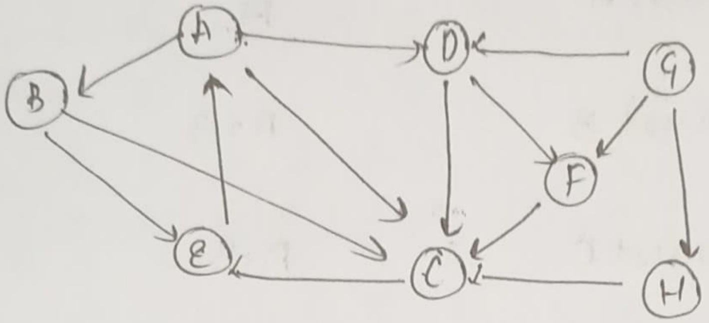
In Directed graph, when we apply bfs for cycle detection In actual we apply topological sort on the given graph because topological sort is only valid for Directed Asyclic graph so for the graph if we are able to find topological sort that means cycle is not present and otherwise cycle is present.

Q5| what do you mean by disjoint set data structure? explain 3 operations along with examples, which can be performed on disjoint sets.

Ans A Disjoint-set data structure also known as union find data structure or merge find set, it stores a collection of disjoint (non-overlapping) sets. Equivalently, it stores a partition of set into disjoint subsets.

Three operations we can perform on disjoint sets are find, union and intersection.

Q6| Run DFS & BFS on graph



Source:-A , Des:- F

Ans BFS.

Queue	Action	Visited
A	Insert A in queue	A
B|C|D	Insert B, C, D in queue	A, B, C, D
C|D|E	Remove A Insert E in queue Remove B	A, B, C, D, E

D | E	Remove c	A, B, C, D, E
E | F	Add F and remove D	A, B, C, D, E, F
F	Remove E	A, B, C, D, E, F
Empty.	Remove F	A, B, C, D, E, F

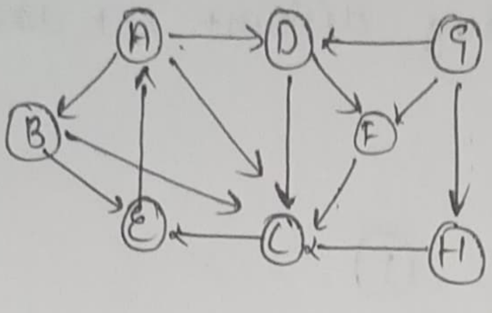
A → B → C → D → E → F

DFA.

Source :- A Des :- F

Stack	Operation	Visited
A	Insert A	A
B A	Insert B	A, B
C B A	Insert C	A, B, C
E C B A	Insert E	A, B, C, E
C B A	Pop E	A, B, C, E
B A	Pop C	A, B, C, E
A	Pop B	A, B, C, E
D A	Insert D	A, B, C, E, D, F
F D A	Insert F	A, B, C, E, D, F &

Q6) Run BFS & DFS on graph shown ↓



Ans BFS.

Adjacency List of the given graph.

A →	B, D, C
B →	C, E
C →	E
D →	C, F
E →	A
F →	C
G →	D, H, F
H →	C

Here we are assuming 1 Based indexing.

	1	1	1	1	1	1	1	1
0	~~∅~~	~~∅~~	~~∅~~	~~∅~~	~~∅~~	~~∅~~	~~∅~~	~~∅~~
0	1	2	3	4	5	6	7	8
	A	B	C	D	E	F	G	H

H
~~G~~
~~F~~
~~E~~
~~D~~
~~C~~
~~B~~
~~A~~

Queue

O/P A, B, C, E, D, F, G, H

DFS. A, B, C, E, D, F, G, H

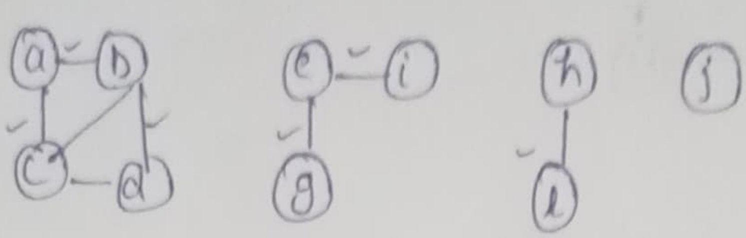
dfs(A)
dfs(B)
dfs(C)
dfs(E)

dfs(D)
dfs(F)

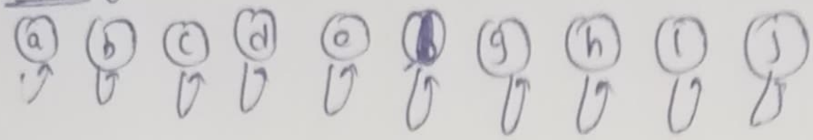
dfs(G)
dfs(H)

(Q7) Find out the no of connected components and vertices in each component using disjoint set data structure.

Ans



Initially



Self loop shows that they are parent of itself initially

~~union(a,b)~~

union(a,b)

edge set

bd

eg

ac

hl

ab

ei

bc

cd

Collection of disjoint sets.

{a} {b} {c} {d} {e} {l} {g} {h} {i} {j}

{a} {b∪d} {c} {d} {e} {l} {g} {h} {i} {j}

{a} {b∪d} {c} {d} {e∪g} {l} {h} {i} {j}

{a,c} {b∪d} {d} {e∪g} {l} {h} {i} {j}

{a∪c} {b∪d} {d} {e∪g} {h∪l} {i} {j}

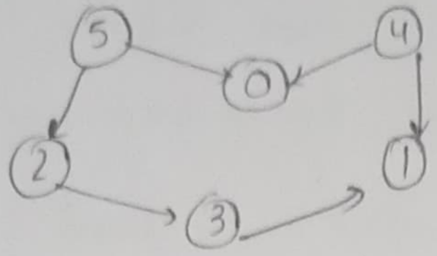
{a∪b∪c∪d} {d} {e∪g} {h∪l} {i} {j}

{a∪b∪c∪d} {e∪g∪i} {h∪l} {j}

~~4 connect~~

4 connected components

Q9) Apply topological sort



Ans Topological sort

0	0	0	0	0	0
0	1	2	3	4	5

5
4
3
2
1
0

stack

Topo(0) → insert 0 into the stack
No neighbours of 0 so topo(0) is over.

Topo(1) → insert 1 into the stack
same it also have no neighbours
so it is also over

Topo(2) → insert 2 into the stack and its neighbour 3
so insert 3 into the stack, neighbour of 3 is 1 but is already present in stack.

Topo(3) → already marked as visited.

Topo(4) → its neighbour is 1 and already visited.

Topo (5) → its neighbour is 0 and already visited.

Ans → ~~[illegible]~~
5 → 4 → 3 → 2 → 1 → 0 &

Q9) Heap data structure can be used to implement priority queue? Name few graph algorithm where you need to use priority queue & why?

Ans) Priority Queue is an extension of the queue but here with each element its priority is associated so according to this priority operation performed on the elements.

Graph Algorithm where we use Priority Queue :-

→ Dijkstra's Algo

→ Bellman ford Algo.

Q10) Diff. b/w Min and Max Heap

Ans)

Min Heap	Max Heap
1. The root element must be less than or equal to its childrens.	1. The root element must be greater or equal to its children.
2. Minimum key value present at root	2. Maximum key value present at root.
3. This uses the Ascending priority.	3. This uses the Decending priority.
4. Smallest element is the first to be popped from heap.	4. Largest element is the first to be popped from heap.