DATE / /
PAGE NO.

Tutorial - 5

Name: SONAM
Sec: B
Rollno: 2014892

Q1 Diff. Between BFS and DFS

BFS	DFS
• Uses queue data Structure	• Uses stack data Structure
• Stands for Breadth First Search	• stands for Depth first search.
• Can be used to find single source shortest path in an unweighted graph and we reach a vector with min no. of edge from a source vertex.	• have to traverse through more edges to reach a destination vertex from a source.
• Siblings are visited before the children.	• Children are visited before the siblings.
• Application:	• Applicati
1. Peer to Peer Network.	1. Detecting cycle in a graph.
2. Social Networking Website	2. Path finding
3. GPS Navigation System	3. Topological Sorting
4. Shortest path & Minimum Spanning tree for unweighted graph.	4. Solving puzzle with only one soln.

Ans 2 In BFS we use Queue Data Structure as Queue is used when things don't have to be processed immediately, but have to be performed in FIFO manner.

In DFS stack is used as DFS uses Backtracking. for DFS, we retrieve it from root to the farthest node as much as possible that is the same idea as ~~FIFO~~ LIFO.

Ans 3 In Dense graph no. of edge is close to the maximal no of edges.

In Sparse graph no. of edges is close to the minimal no of edges. It can be a disconnected graph.

We prefer Adjacency list for sparse graph whereas adjacency matrix is prefered for dense graph.

Ans 5 The disjoint set data structure is also known as union-find data structure and merge find set. It is a data structure that contains a collection of disjoint or non overlapping set

The disjoint set means that when the set of is partitioned into the disjoint subsets, various optim can be performed on it.

In this case, we can add new sets, we can merge the sets and we can also find the representation member of set. It also allows to find out whether the two elements are in the same set or not efficiently

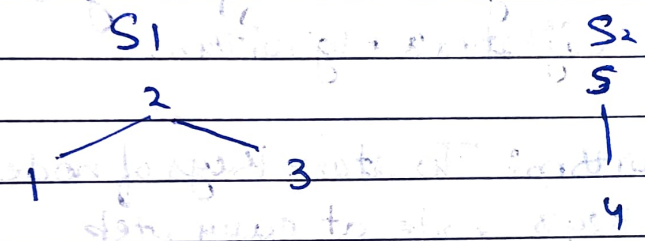
operation on disjoint set.

1. Union : If S_1 and S_2 are two disjoint sets, their union $S_1 \cup S_2$ is a set of all elements x such that x is in either S_1 or S_2.

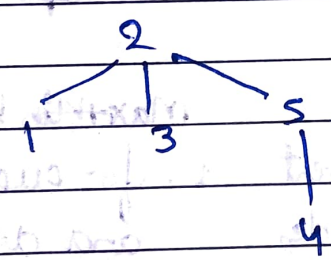
2. As the set should be of disjoint $S_1 \cup S_2$ replace S_1 and S_2 which no longer exist.

3. Union is achieved by simply making one of the trees ~~an~~ as a ~~subset~~ subtree of other i.e. to set parent field of one of the roots of the ~~trees~~ trees to other root.

eg

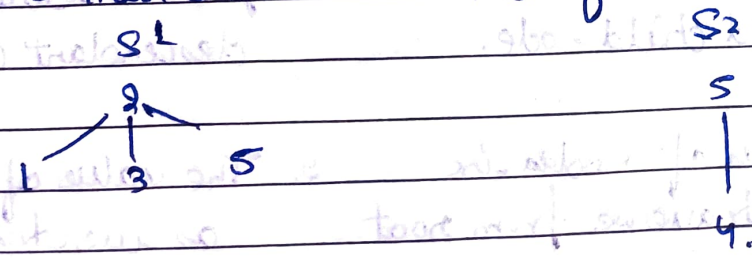


$S_1 \cup S_2$



4. find : Given an element x, to find the set containing it.

eg



find(3) → (S_1)

find (5) → (S_2)

5. Make Set Creates Set Containing X:

Ans9 Heap data Structure is preferred for priority Queue implementation because heap provide better performance compared to arrays or linked list.

priority Queue is used for.

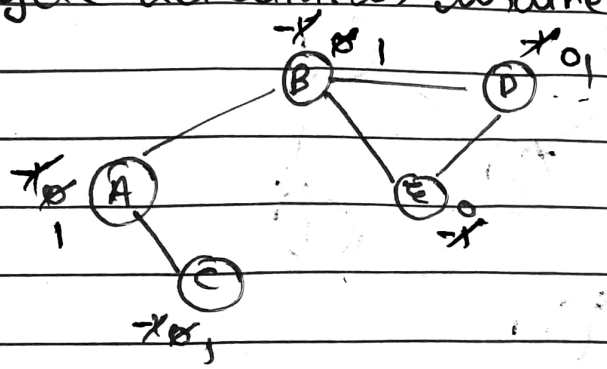
Dijkstra's Shortest path Algorithm: when the graph is stored in the form of adjacency list or matrix, priority Queue can be used to extract minimum ~~priority Queue~~ efficiently while performing dijkstra's Algorithm.

Prim's Algorithm: To store Keys of node and extract minimum Key's node at every step.

Q10 Difference between Max and Min heap.

Min ~~Max~~ heap	Max ~~Min~~ heap
1. for every pair of the parent and descendant child node, the parent node always has lower value than descended child node.	1. for every pair of the parent and descendant childnode and the parent node has greater value than descendant child node.
2. The values of nodes inc as we traverse from root to leaf node	2. The value of node decreases as we traverse from root to leaf node
3. Root node has lowest value	3. Root node has greatest value

Q4 Cycle detection in Undirected graph.



-1 = unvisited
0 = into the queue
1 = traversed.

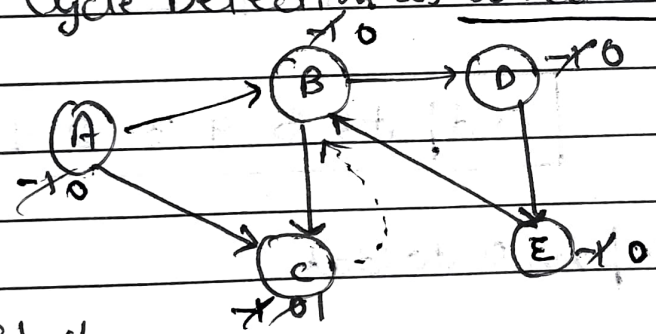
Queue: | A | B | C | D | E |

Visited node: | A | B | C | D |

when D checks it's adjacent vertices it finds E with 0.

→ if any vertex finds the adjacent vertex with flag 0, it contains cycle.

Cycle Detection in Directed Graph : DFS



-1 = unvisited.
0 = in stack.
1 = visited & popped

Stack

E
D
B
A

Visited Set
A B C D E.

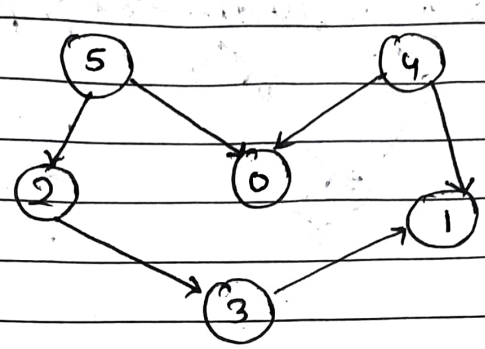
B → D → E → B

parent map.

Vertex	Parent
A	–
B	A
C	B
D	B
E	D.

Here E finds B with 0.
⇒ it contains a cycle.

Q8 Topological sorting & DFS.



Adj. list (G)

0 → NULL
1 → NULL
2 → 3
3 → 1
4 → 0, 1
5 → 2, 0

	0	1	2	3	4	5
visited	∅	∅	∅	∅	0	0
	1					
		1				
			1	1		
					1	
						1

1. ~~visited [0] = 1~~
~~Stak is empty~~
~~push 0~~

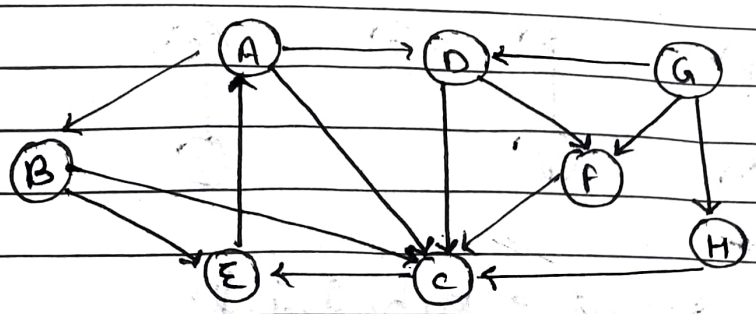
0	1	3	2	4	5

2. ~~visited [1] = 1~~

Output: 5 4 3 2 1 0

1. visited [0] = 1
no outgoing edge ; push 0 to stak
2. visited [1] = 1
no outgoing edge ; push 1 to stak
3. visited [2] = 1
visited [3] = 1
visited [1] is already true ; push 3 to stack ; push 2 to stack
4. visited [4] = 1
visited [0] is already true
visited [1] is already true, push 4 to stak.
5. visited [5] = 1
no outgoing edge
put 5 to stack.
6. Print element of stak top to bottom.

Q6



BSF: Queue.

~~B~~	~~E~~	~~C~~	~~A~~	~~D~~	~~F~~	

visited:

A	B	C	D	E	F	G	H
~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	0	0
1	1	1	1	1	1		

Starting node B. visit child E and C, visited = 1 for B
push E and C to Queue, pop B

E visit A, visited = 1 for E
push A to Queue, pop E.

C visit none visited = 1 for C, pop C

A visit D, visited = 1 for A pop A
push D to Queue

D visit F, visited = 1 for D pop D.
push F to Queue

F visit none, visited = 1 for F
pop F.

order of traversal B → E → C → A → D → F.
G & H cannot be visited.

DSF. Stack

~~B~~	~~E~~	~~C~~	~~A~~	~~D~~	~~F~~

visited:

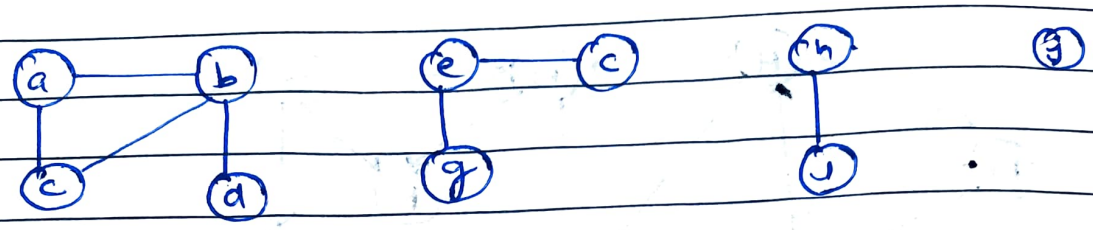
A	B	C	D	E	F	G	H
~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	0	0
1	1	1	1	1	1		

Starting node B.

output: B C E D F
G and H cannot be visited.

Date / /
Page No.

Q1.



$V = \{a, b, c, d, e, g, h, i, j, d\}$

$E = \{(a,b), (a,c), (b,c), (b,d), (e,i), (e,g), (h,d), (j)\}$

	{a}	{b}	{c}	{d}	{e}	{f}	{g}	{h}	{i}	{j} {d}
(a,b)	{a,b}	{c}	{d}	{e}	{g}	{h}	{i}	{j}	{d}	
(a,c)	{a,b,c}	{d}	{e}	{g}	{h}	{i}	{j}	{d}		
(b,c)	{a,b,c}	{d}	{e}	{g}	{h}	{i}	{j}	{d}		
(b,d)	{a,b,c,d}	{e}	{g}	{h}	{i}	{j}	{d}			
(e,i)	{a,b,c,d}	{e,i}	{g}	{h}	{j}	{d}				
(e,g)	{a,b,c,d}	{e,i,g}	{h}	{j}	{d}					
(h,d)	{a,b,c,d}	{e,i,g}	{h,d}	{j}						
(j)	{a,b,c,d}	{e,i,g}	{h,d}	{j}						

we have

{a, b, c, d}

{e, i, g}

{h, d}

{j}