SHIVAM SUNDRIYAL

D

137

PAGE No.

DATE

ASSIGNMENT - 5

1.

BFS	DFS
• Breadth first Search	• Depth first Search
• Uses Queue data structure for finding the shortest path	• Uses stack data structure.
• Used to find single source shortest path in an unweighted graph	• Used to find one of the possible path from source to destination
• There is no concept of backtracking	• Backtracking is possible as stack is used.
• Requires more memory	• Requires less memory

Applications of BFS

1. To find a spanning tree in unweighted graph
2. GPS navigation system
3. Garbage Collection

Applications of DFS

1. Topological Sorting
2. Scheduling Problems
3. Cycle detection in graphs.

2. Queue is used in BFS

Reason - Since BFS algorithm traverses a graph in a breadthward motion (i.e. in level order) it uses a queue to remember to get the next vertex to start a search, when a dead end occurs in any iteration.

Firstly visit the adjacent unvisited vertex, mark it as visited & insert it in the queue.

Then if no adjacent vertex is found, remove the first vertex from the queue.

Repeat the process until the queue is empty.

When the queue gets empty, the program is over

Stack is used in DFS

Reason - Since DFS algorithm traverses a graph in a depthward motion it uses stack to remember to get the next vertex to start a search, when a dead end occurs in any iteration.

Visit the adjacent unvisited vertex, mark it as visite push it in the stack.

If no adjacent vertex is found, pop up a vertex from stack.

Pop the vertex unil a node with an unvisited adjacent node.

PAGE No.
DATE

3. Dense graph is a graph in which the number of edges is close to the maximal number of edges.
Sparse graph is a graph in which the number of edges is close to the minimal number of edges. Sparse graph can be a disconnected graph.

It is ideal to represent sparse graph by adjacency list & dense graphs by an adjacency matrix.

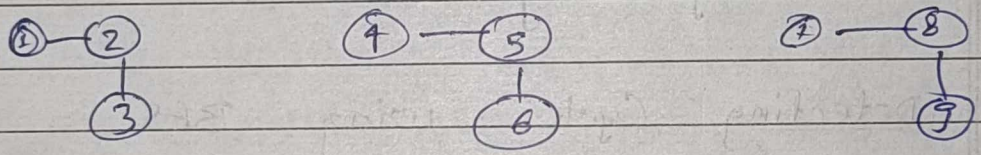
4. Detecting Cycle using BFS
When we do a BFS traversal of a given graph. For every visited vertex 'v', if there is an adjacent 'u' such that u is already visited & u is not a parent of v, then there is a cycle in the graph. If such an adjacent for any vertex is not found, then there is no cycle. A parent array is used to keep track of the parent vertex.

Detecting Cycle using DFS
DFS for a connected graph produces a tree. There is a cycle in a graph only if there is a back edge present in the graph. A back edge is an edge that is from a node to itself (self-loop) or one of its ancestors in the tree produced by DFS

PAGE No.
DATE

5. A disjoint set data structure also called as union - find data structure or merge - find data structure is a data structure that stores a partition of set into disjoint subsets. It provides operations for adding sets, merging sets & finding a representative member of a set.

eg: $S1 = \{1, 2, 3\}$
$S2 = \{4, 5, 6\}$
$S3 = \{7, 8, 9\}$



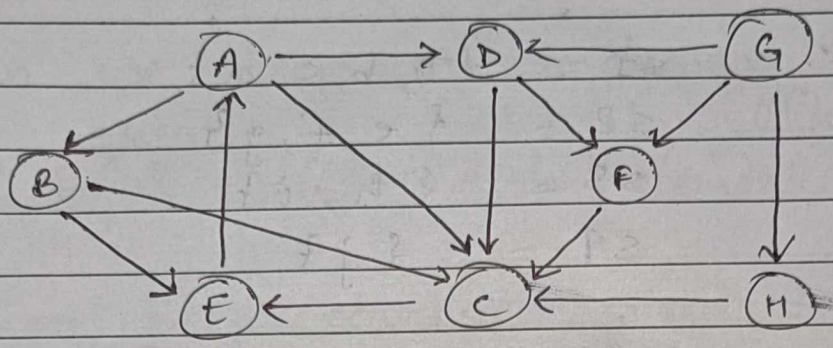
Operations Performed:

① find - It can be implemented recursively by traversing the parent array until we hit a node who is parent to itself. Here, path comparisson can be achieved by keeping track of size of the node

② Union - It takes as input two elements & find the parent of the inputs using find operation & performs merging of the one child to the parent node.

PAGE No.
DATE

6.



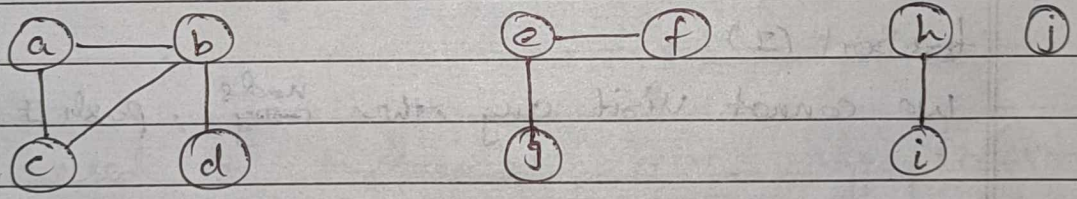
BFS :

Node	B	E	C	A	D	F
Parent		B	B	E	A	D

DFS

B	E	A	D	F	C

7.



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

Here, make, find & Union operation will be performed

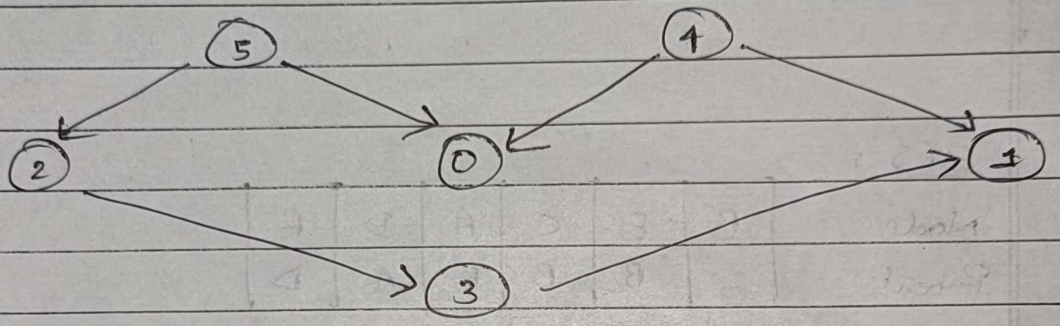
→ First each vertex will be parent of itself.

→ If an edge exists b/w vertices, than we perform find operation & check if their parent is same & perform union operation on them.

PAGE No.
DATE

∴ S1 = {a, b, c, d}
S2 = {e, f, g}
S3 = {h, i}
S4 = {j}

6.



topoSort (0)
As we cannot visit any other node push 0

topoSort (1)
We cannot visit any other node, push 1

topoSort (2) → topoSort (3)
first push 3 then push 2

topoSort (4)
push 4, as other are visited

topoSort (5)
push 5, as other are visited

5
4
2
3
1
0

5	4	2	3	1	0

PAGE No.
DATE

9. We can use heaps to implement the priority queue. It will take O(log N) time to insert & delete each element in priority queue.
Based on heap structure, priority queue has also two types - Max Priority & Min Priority.

Some algos where we need to use Priority Queue:

(i) Dijkstra's Algo -
It is used in calculating shortest path in the algorithm.
When the graph is stored in form of adj list or matrix, priority queue can be used to extract efficiently.

(ii) Prims -
It is used to implement Prim's algo to store keys of nodes & extract minimum key node at every step.

(iii) Data Compression -
It is used in Huffman's coding which is used to compress data.

10.

Min Heap	Max Heap
• In a min heap, the key present at root must be less than or equal to among the keys present at all of its children	• In a max heap, the key present at the root node must be greater than or equal to the keys present at all of its children
• The min. key element is present at the root	• The max. key element is present at root.
• Uses ascending priority	• Uses descending priority.
• The smallest element has priority.	• The largest element has priority.
• The smallest element is to be popped from heap.	• The largest element is to be popped from heap.