Tutorial-5

(1) Difference b/w BFS and DFS

BFS (Breadth First Search)

BFS is a vertex based technique for finding a shortest path in a graph. It uses a queue data structure which follows first in first out.
In BFS one vertex 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.

DFS (Depth First Search)
DFS is an edge based technique. It uses the stack data structure, perform two stages, first visited vertices are pushed into stack and second if there is no vertices then visited are popped.

Applications of BFS and DFS.

DFS: Performing DFS on unweighted graph will create min. spanning tree for all pair shortest path tree.
- Using DFS we can find path b/w two given vertices u and v
- we can detect cycle in a graph using DFS

- BFS: In peer to peer network like bit torrent, BFS is used to find all neighbour nodes.
- Using GPS navigation system BFS is used to find neighbour places.
- Path finding algorithms are based on BFS or DFS.

(2) BFS uses queue data structure. Because in BFS one level is explored completely so queue stores the elements of the level from the front edge and removes the elements from the rear edge. Those elements which had already explored are will removed from the back or rear edge.

DFS uses stack data structure because in DFS we store the previous node till we end at the end of one side of graph so we have to use backtracking for that purpose we use stack data structure for DFS.

(3) Sparse Graph: A graph in which the number of edges is much less than the possible number of edges.

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

Adjacency list is good for dense and sparse graph representation.

(4) Cycle Detection in Graph traversal

Using BFS: when we do a BFS ~~traver~~ from any vertex v in an undirected graph, we may encounter a cross edge that points to a previously discovered vertex that is neither an ancestor nor a descendant of the current vertex. Each "cross edge" defines a cycle in a undirected graph. If the cross edge is $x \to y$, then since y is already discovered, we have a path from v to y where v is the starting vertex of BFS. So we can say that we have a path $v \sim\sim x \sim y \sim\sim v$. that forms a cycle.

Using DFS: when we do ~~g~~ a DFS from any vertex v in an undirected graph, we may encounter a back-edge that points to one of the ancestor of the current vertex v in the DFS. Each "back edge" defines a cycle in an undirected graph. If the back edge is $x \to y$, then since y is the ancestor of node x, we have a path from y to x. So we can say that we have a path $y \sim\sim x \sim y$ that forms a cycle.

5. Disjoint set data structure: Also known as union find data structure or merge find set is a data structure that stores a collection of disjoint sets. disjoint set keeps track of a set of elements partitioned into several disjoint.

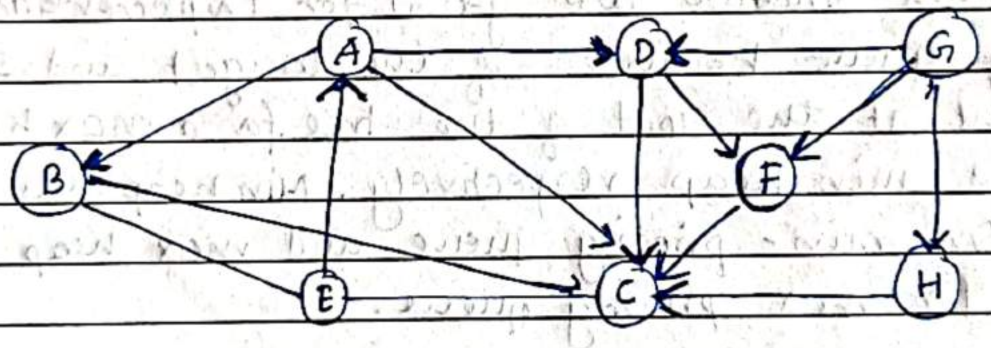
Operations.

(i) Find: It determines in which subset a particular element is in and return the representative of that particular set. An item from this set typically acts as a "representative" of the set.

(2) union: It merge two different subset into one single subset, and the representative of one subset become representative of other.

(3) Making new sets: The make set operation adds a new element. This element is placed into a new set containing only the new element and the new set is added to the data structure.

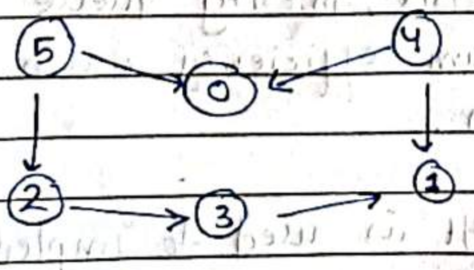
(Ans. 6)



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

BFS : G, D, F, H, C, E, A, B

(8).



An 5, 4, 2, 3, 1, 0
As topological Sort is not unique so there can also

than one.
be ~~are~~ more, sequence

4, 5, 2, 3, 1, 0

or

5, 4, 0, 2, 3, 1

or

4, 5, 0, 2, 3, 1

(9) Priority queue is a type of queue in which every element has a key associated to it and the queue returns the element according to these keys, unlike the queue which works on first come first serve basis.

Heap data structure are great for implementation of a priority queue because of the largest and smallest element at the root of the tree for a max heap and a min heap respectively. Min heap can be used for min-priority queue and max heap can be used for max priority queue.

Graph algorithms where priority queue is used

(1) Dijkstra's shortest Path Algorithm using priority queues: when the graph is stored in the form of adjacency list or matrix, priority queue can be used to extract minimum efficiency when implementing Dijkstra algorithm.

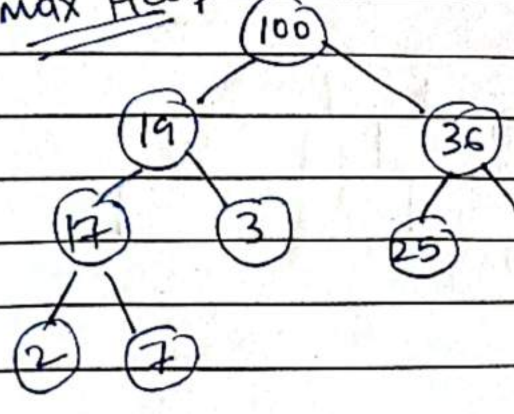
(2) Prism's Algorithm: It is used to implement Prim's Algorithm to store key of nodes and extract minimum key nodes at every step.

- Priority queue are used in Huffman codes for data compression.
- Priority queues are used to sort heaps.

(10) Difference b/w Max and Min heaps.

Min Heap	Max Heap.
1. In a Min-heap the key present at the root-node must be less than or equal to among the keys present at all of the children	In a Max heap the key present at the root node must be greater than or equal to among the keys present at all of its children.
2. In a Min heap the min key element present at the root.	In a max heap the max key element present at the root.
3. A min-heap uses the ascending priority	A Max heap uses the descending priority
4. In the construction of a min-heap, the smallest element has priority.	In the construction of a Max heap, the largest element has priority

max Heap



Min heap