TUTORIAL → 5

NAME:→ Srijan

SECTION:→ F

ROLLNO.:→ 30

UNIVERSITY ROLLNO.:→ 2017067

Ques:-

Solution:→

BFS	DFS
•) It stands for Breadth First Search	•) It stands for Depth First Search
•) It uses Queue data structure	•) It uses stack data structure
•) It is more suitable for searching.	•) It is more suitable when there are solution away from source
•) BFS considers all neighbours first and therefore not suitable for decision making things used in games & puzzles	•) DFS is more suitable for game or puzzle problem we make a decisions. And if decision leads to win situation, we stop.
• It requires more memory	• It requires less memory.

APPLICATIONS:→

•) BFS:→ Bipartite graph and shortest path, peer to peer networking, crawlers in search engine and GPS navigation System.

•) DFS:→ Acyclic graph, topological order, scheduling problems, sudoku puzzle.

2Qns:-

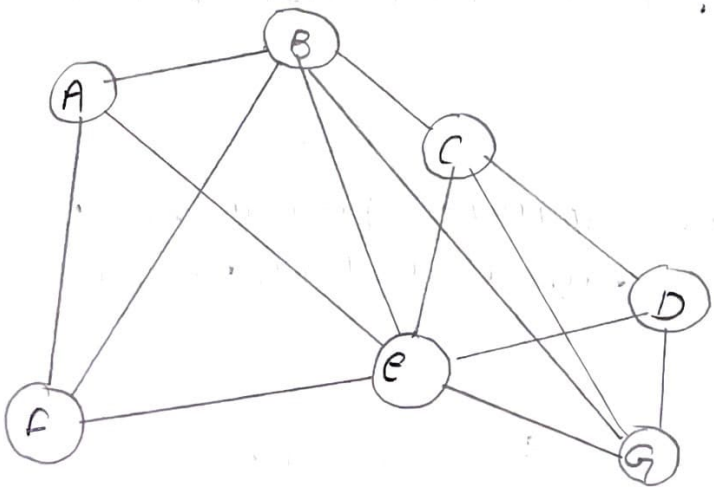
Solution :→ For implementing BFS we need a queue data structure for finding shortest path between any node. We use queue because things don't have to be processed immediately, but have to be processed in FIFO order like BFS. BFS searches for node one wise, i.e. it searches node w.r.t their distance from root (source) for this queue is better to use in BFS.

For implementing DFS we need a stack data structure as it traverses a graph in depthward motion and uses stack to remember to get the next vertex to start search, when a dead end occurs in any iteration

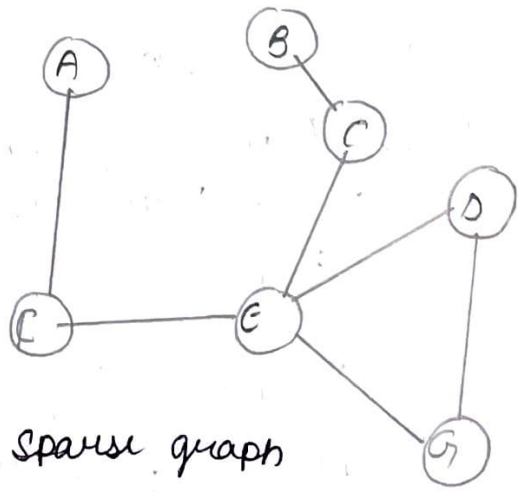
3Qns :→

Solution :→ Dense graph is a graph in which no. of edges is close to maximal no. of edges.

Sparse graph is graph in which no of edges is very less.



Dense Graph
(many edges between nodes)



Sparse graph
(few edges b/w nodes)

Qus

Solution:→ For detecting cycle in a graph using BFS we need to use Kahn's algorithm for topological sorting :-

The steps involved are :→

1) Compute in-degree for each of vertex present in graph and initialize count of visited nodes as 0.

2) Pick all vertices with in-degree as 0 and add them in queue.

3) Remove a vertex from queue and then
   - Increment count of visited nodes by 1.
   - Decrease in-degree by 1 for all its neighbouring nodes
   - If in-degree of neighbouring node is reduced to zero then add to queue.

4) Repeat 3) until queue is empty.

5) If count of visited nodes is not equal to no. of nodes in graph has cycle, otherwise not.

5Qus

Solution:→ A disjoint set is a data structure that keeps track of set of elements. partitioned into several disjoint subsets. In otherwords, a disjoint set is a group of sets where no item can be in more than one set.

3 operations

•) Find → can be implemented by recursively traversing the parent array until we hit a node who is parent to itself

Eg:-
```
int find(int i){
    if (parent[i] == i){
        return i;
    }
    else{
        return find(parent[i]);
    }
}
```

- **Union:→** It takes 2 elements as input and find representations of these sets using the find operation & finally put either one of the tree under root node of other tree, effectively merging the tree and sets.

Eg:-
```
void union(int i, int j)
{
    int irep = this.Find(i);
    int jrep = this.find(j);
    this.parent[irep] = jrep;
}
```

- **Union by Rank:→** we need a new array rank[] size of array is same as parent array. If i is representative of set, rank[i] is height of tree.

Eg:-
```
void unit(int i, int j){
    int irep = this.Find(i);
    int jrep = this.find(j);
    if (irep == jrep) return;
    irank = rank[irep];
    jrank = rank[jrep]
    if (irank < jrank)
        this.parent[irep] = jrep;
    else if (jrank < irank)
        this.parent[jrep] = irep;
    else{ this.parent[jrep] = jrep; irank[jrep]++;
    }
}
```

Quiz 6

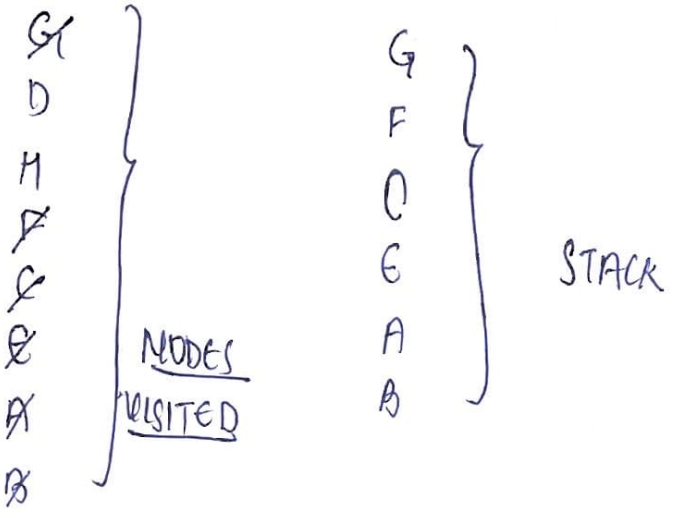
solution

BFS

child	G	H	D	F	C	E	A	B
parent		G	G	G	H	C	E	A

Path → G → H → C → E → A → B

DFS



Path → G → F → C → E → A → B.

Quiz 7

solution :- V = {a, b, c, d, e, f, g, h, i, j}

E = {a,b}, {a,c}, {b,c}, {b,d}, {e,f}, {e,g}, {h,i}, {j}

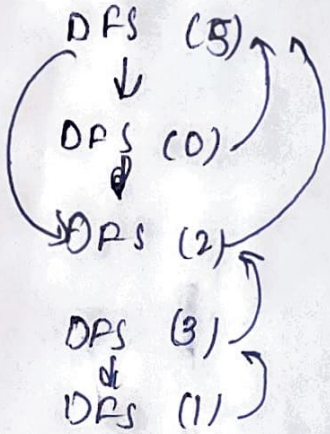
(a,b)	{a,b} {c} {d} {e} {f} {g} {h} {i} {j}
(a,c)	{a,b,c} {d} {e} {f} {g} {h} {i} {j}
(b,c)	{a,b,c} {d} {e} {f} {g} {h} {i} {j}
(b,d)	{a,b,c,d} {e} {f} {g} {h} {i} {j}
(e,f)	{a,b,c,d} {e,f} {g} {h} {i} {j}
(e,g)	{a,b,c,d} {e,f,g} {h} {i} {j}
(h,i)	{a,b,c,d} {e,f,g} {h,i} {j}

No. of connected components = 3 Ans.

Ques 8

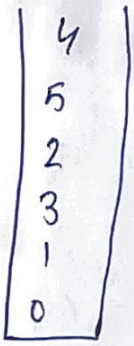
Solution we take source node as 5.

Applying topological sort



DFS(4)
⇓
Not possible.

DFS



Stack.

4 → 5 → 2 → 3 → 1 → 0 Ans

Ques 9

Solution :→ Yes heap data structure can be used to implement priority queue. It will take O(log N) time to insert and delete each element in priority queue. Based on heap structure priority queue has two type max-priority queue based on max heap and min heap. Heaps provide better performance comparison to array and LL.

The graph like dijkstra's shortest path algorithm, Prim's minimum spanning tree use priority queue

Q.No 10

Solution:

Min Heap	Max Heap
• In this, key present at root node must be less than or equal to among key present at all of its children	• In this, the key present at root node must be greater than or equal to among key present at all of its children
• The minimum key element is present at the root.	• The Max. key element is present at the root
• It uses ascending priority	• It uses descending priority
• The smallest element has priority while construction of min-heap	• The largest element has priority while cons. of max. heap.
• The smallest element is the first to be popped from the heap	• The largest element is the first to be popped from the heap.