Date.
Page No.

Tutorial - 5

Name - Shagun Gupta
Section - C
Roll No - 11
University Roll No. - 2017012

Q1.
Ans.

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 vertices which are closer to given source	it is more suitable when there are solutio away from source
• BFS considers all neighbours first and therefore not suitable for decision making trees used in games + puzzles	DFS is more suitable for game or puzzle problems. We make a decision, then explore all paths through this decision. And if decision leads to win suitable we stop
• It requires more memory	it requires less memory
• There is no concept of backtracking	It is a recursive algorithm that uses backtracking

Date.
Page No.

⇒ Applications

- BFS:- Bipartite graph and shortest, pees to pees networking crawlers in search engine and GPS ~~no~~ navigation system
- DFS:- acyclic graph, topological order, scheduling problems sudoku puzzle

Q2.

Ans 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 nodes level wise. i.e. it searches nodes w.r.t. their distance from root (source). For this queue is better to use in BFS.

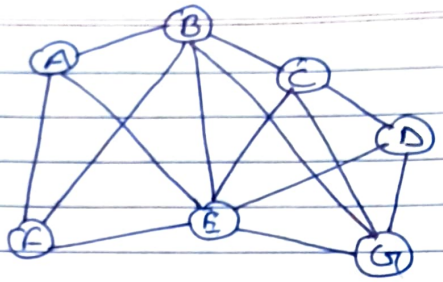
for implementing DFS we need a stack data structure as it traverse a graph in depth wise manner and uses stack to remember to get the next vertex to start a search, when a dead end occurs in any iteration.

Q3

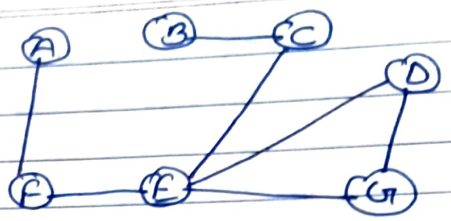
Ans Dense graph is a graph in which no. of edges is close to maximal no. of edges

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

Date.
Page No.



Dense Graph
(many edges b/w nodes)



Sparse graphs
(few edges b/w nodes)

- for sparse graph it is preferred to use Adjacent List.
- for dense graph it is preferred to use Adjacent Matrix

Ans 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 (no. of incoming edges) for each of vertex present in graph & 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 nodes is reduced

Date.
Page No.

to zero then add to queue
4. Repeat 3 until queue's empty
5 if count of visited nodes is not equal to no. of nodes in graph has cycle, otherwise not

for detecting cycle in graph using DFS we need to do following DFS for a connected graph produces a tree. There is cycle in graph 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. For a disconnected graph, get DFS forest as output. To detect cycle, check for a cycle in individual trees by checking back edges. To detect a back edge, keep track of vertices currently in recursive track for DFS traversal. If a vertex is reached that is already in recursion stack, then there is a cycle.

Q5

Ans A disjoint set is a data structure that keeps track of set of elements partioned into several disjoint subsets. In other words, a disjoint set is a group of sets where no item can be in more than one set

Date.
Page No.

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 representatives of this sets using the find operation and finally puts either one of the trees under root node of other tree, effectively merging the trees and sets.

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

Date
Page No.

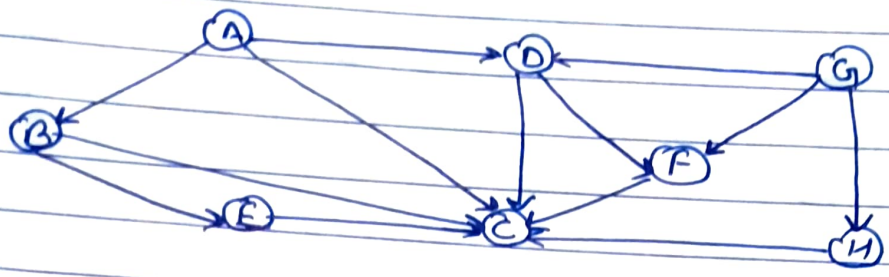
- Union by Rank :- We need a new array rank[]. Size of array same as parent array. If i is representative of set, rank[i] is height of tree. We need to minimize height of trees. If we are uniting 2 trees, we call them left and right, then it all depends on rank of left and right.
- If rank of left is less than right then it's best to move left under right & vice versa
- If rank are equal, rank of result will always be one greater than rank of trees

eg:-

```
void union(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[irep] = jrep;
    Rank[jrep]++;
 }
}
```

Date.
Page No.

6



BFS

Child	G	H	D	E	C	E	A	B
Parent		G	G	G	H	C	E	A

Path → G → H → C → E → A → B

DFS

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

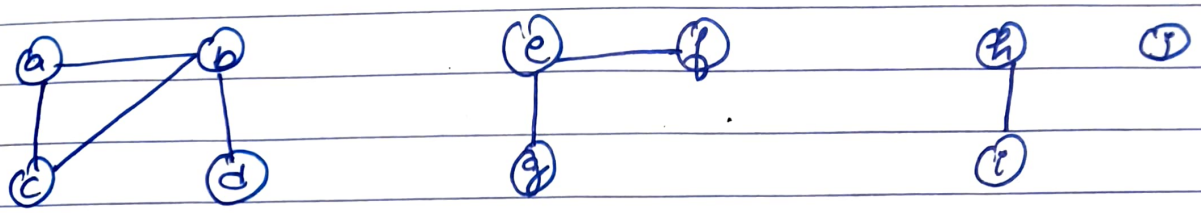
Nodes Visited

G
F
C
E
A
B

Stack

Path → G → F → C → E → A → B

Date.
Page No.



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

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

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

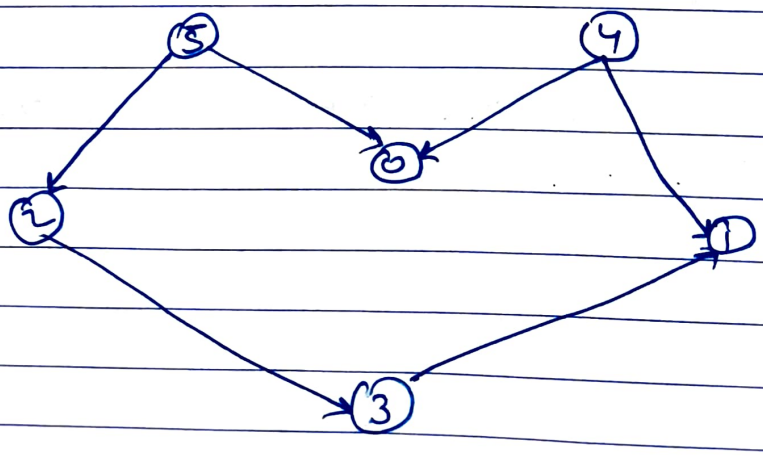
(b,d) $\{a,b,c,d\}\{e\}\{f\}\{g\}\{h\}\{i\}\{j\}$

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

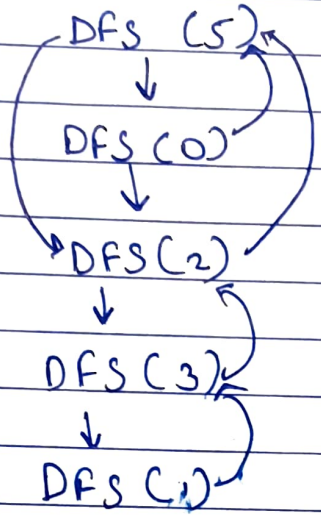
No. of connected components = 3 → Ans



Date.
Page No.

We take source node as 5

Applying Topological Sort



DFS (4)
↓
Not possible

DFS

4
5
2
3
1
0

Stack

4 → 5 → 2 → 3 → 1 → 0

Ans

Date
Page No.

Q9.
Ans Yes, heap data structure can be used to implement priority queue. It will take O(logN) time to insert and delete each element in priority queue. Based on heap structure priority queue has two types max-priority queue based on max heap and min priority queue based on min-heap
Heaps provide better performance comparison to array & LoL.

The graph like Dijkstra's shortest path algorithm. Prim's Minimum Spanning tree use Priority Queue.

- Dijkstra Algorithm → When graph is stored in form of adjacency list or matrix, priority queue is used to extract minimum efficiently when implementing the algorith.
- Prim's Algorithm. It is used to store keys of nodes and extract minimum key nodes at every step.

Q10
Ans

Min Heap	Max Heap
• In min heap key present at root node must be less than or equal to among keys present at all of its children	In max heap the key present at root node must be greater than or equal to among keys present at all of its children.

Date. _____
Page No. _____

◦ The minimum key element is present at the root	The maximum key element is present at the root
◦ It uses ascending priority	it uses descending priority.
◦ The smallest element is the first to be popped from the heap	The largest element is the first to be popped from the heap