Page No.

Tutorial 5

Name - Prakshep Gusain
Sec - F : 19
Roll No - 2016907

1) What is difference b/w DFS & BFS. Write applications of both algo.

Ans: **BFS:**
- Breadth First Search
- Uses queue
- Suitable for searching vertices closer to source.
- Considers all neighbours.
- No backtracking.
- Requires more memory.

DFS:
- Depth First Search
- It uses stack
- Suitable for far away solutions
- More suitable for game or puzzle. We make a decision, then explore all paths through this decision. And if decision leads to win situation,

Teacher's Signature

we stop.
- It is a recursion algorithms that uses backtracking
- It require less memory.

Applications

- BFS → Bipartite graph & shortest path, peer to peer networking etc..
- DFS → Acyclic graph, topological order, scheduling problems etc

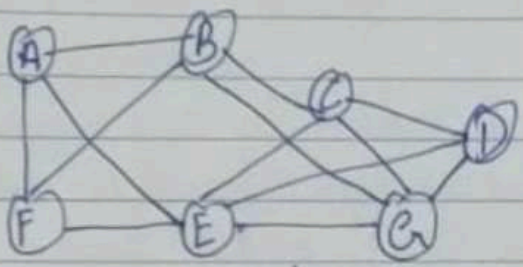
2. Which data structure are used to implement BFS & DFS & why?

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, it searches nodes wrt 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 depthward motion & uses stack to remember to get the next vertex to start a search, when a dead end occurs in any iteration.

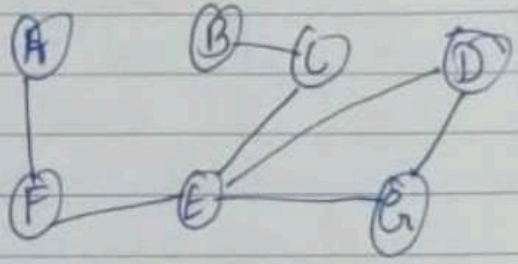
Date

3. What do you mean by spare & demo graph? Which representation of graph is better for a spare & demo graph?

Ans Demo graph is a graph in which no. of edges is close to maximal no. of edges.
Spare graph is a graph in which no. of edges is very less.



Demo graph
(many edges blw nodes)



Spare graph
(few edges blw nodes)

- For spare graph it is preferred to use adjacency list.
- For demo graph it is preferred to use adjacency matrix.

4. How can you detect a cycle in a graph using BFS and DFS?

Ans. For detecting cycle in a graph using BFS we need to use Kahn's algorithm for Topological Sorting. The steps involved are:

Date

1) Compute in degree for each of vertex present in graph & initialize count of visited nodes as 0.
2) Pick all vertices with in-degree as 0 & add them in queue.
3) Remove a vertex from queue & then
   - Increment count of visited nodes by 1.
   - Decrease in-degree by 1 for all its neighbouring.
   - If in-degree of neighbouring nodes is reduced to zero then add to queue.
4) Repeat.
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 graphs 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 can check for a cycle in individual trees by checking back edges. To detect a back edge, keep track of vertices currently in recursion track for DFS traversal. If a vertex is reached that is already in recursion attack, then there is a cycle.

Date

5. What do you mean by disjoint set data structure? Explain 3 operations along with examples which can be performed on ~~Dis~~ disjoint sets?

Ans A disjoint set in 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.

3 operations

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

e.g.
```
int find (int i) {
    if (parent[i] == i) {
        return i;
    }
    else {
    return find (parent[i]);
    }
}
```

b) 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 & sets.

e.g.
```
void union (int i, int j) {
    int irep = this.Find(i);
```

Date

```
        int jrep = this.Find(j);
        this.parent[irep] = jrep;
    }
```

2) 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 tree. If we are uniting 2 trees, we call them left & right, then it all ~~depends~~ depends on rank of left & right.

- If rank of left is less than right then it's best to move left under right & vice versa.
- If ranks are equal, rank of result will always be one greater than rank of trees.

e.g.

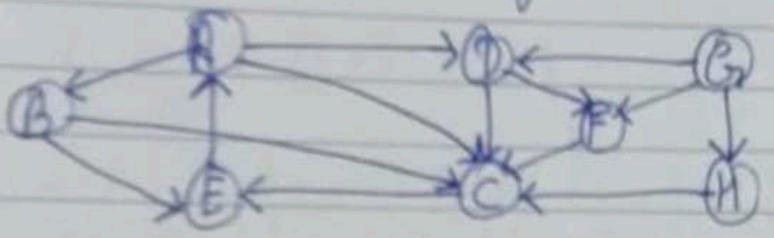
```
void union (int i, int j) {
    int irep = this.Find(i);
    int jrep = this.Find(j);

    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.pared[irep] = jrep;
```

Mohan

Date

```
        Rank[yrep]++;
    }
}
```

6) Run BFS & DFS on graph shown below



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

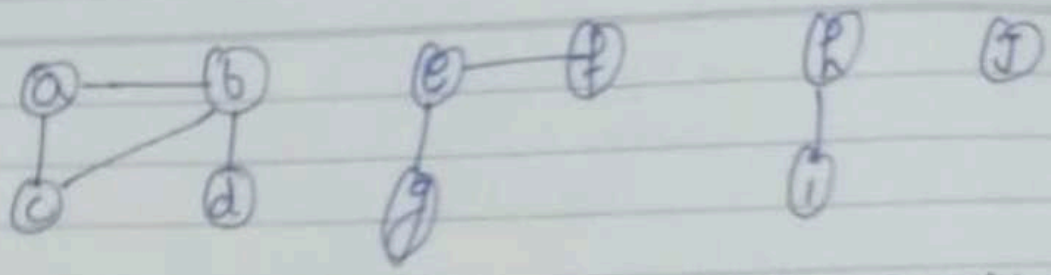
DFS

Nodes Visited: G, D, H, F, C, E, A, B

Stack: G, F, C, E, A, B

Path → G → F → C → E → A → B

7) Find out no. of connected components & vertices in each component using disjoint set data structure.



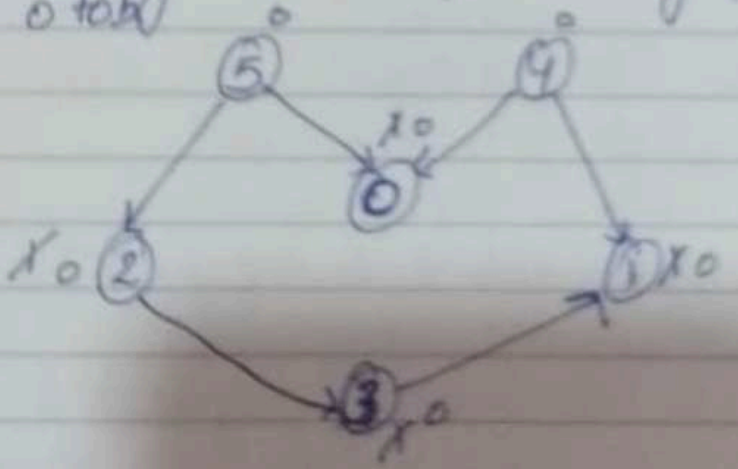
Ans $V = \{a\}\{b\}\{c\}\{d\}\{e\}\{f\}\{g\}\{h\}\{i\}\{j\}$

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

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

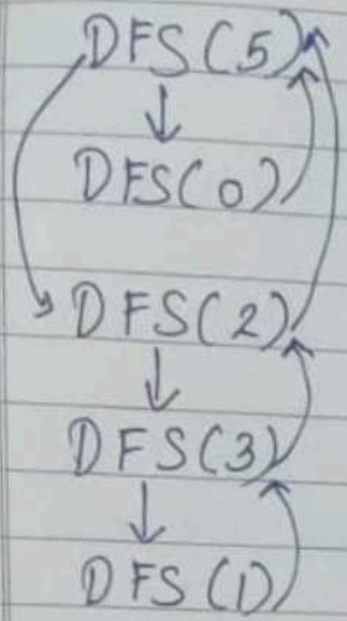
No. of connected components = 3 Ans.

Q) Apply topological sort of DFS on graph having vertices from 0 to 5



Ans We take source node as 5 q: 5/4, Pop 5 & decrement indegree of 2 & 0 by 1

Date



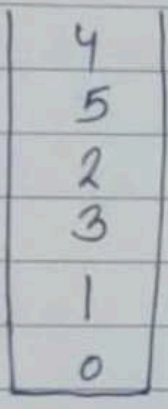
q: 4/2, Pop 4 & decrement indegree & push 0

q: 2/0 Pop 2 & decrement indegree & push 3

q: 0/3 Pop 0, Pop 3, Push 1

q: 1; Pop 1

DFS



Stack

4 → 5 → 2 → 3 → 1 → 0

Answer: 5 4 2 0 3 1

Topological sort

10) Differentiate between Min heap & Max heap.

Ans

Min Heap	Max Heap
• Key present at root node must be less than or equal to among keys present at all.	• Key present at root node must be greater than or equal to among keys present at all of its children.
• 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.

Tutorial 6

Name - Prakshep Gusain
Sec - F : 19
Roll No - 2016907

1) What is min. Spanning Tree? What are its applications?

Ans: Min. Spanning Tree is a subset of edges of a connected edge-weighted undirected graph that connects all vertices together without any cycles & min possible edge weighted.

Applications:

(i) Consider n stations are to be linked using network & laying of communication link between any 2 stations. The ideal solution would be extract a subgraph termed as min cost spanning tree.

(ii) Google Maps

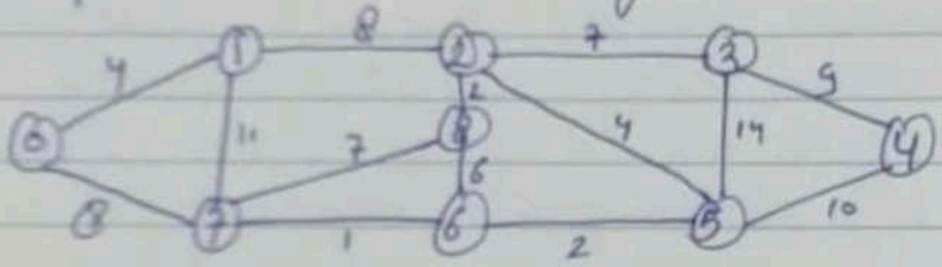
2) Analyze time & space complexity of Prim, Kruskal, Dijkstra & Bellman Ford Algo.

Ans: T.C. of Prim's Algo : $O(|E| \log |V|)$
S.C " " " : $O(|V|)$

Teacher's Signature

⇒ Time complexity of Kruskal's Algorithm : $O|E|\log|E|$
⇒ Space " " " " : $O|V|$
⇒ Time " " Dijkstra's " : $O[V^2)$
⇒ Space " " " " : $O(V^2)$
⇒ Time " " Bellman Ford's Algorithm: $O(VE)$
⇒ Space " " " " " : $O(E)$

3) Apply Kruskal & Prim's Algorithm on given graph to compute MST & its weight.



Ans

Kruskal's Algorithm

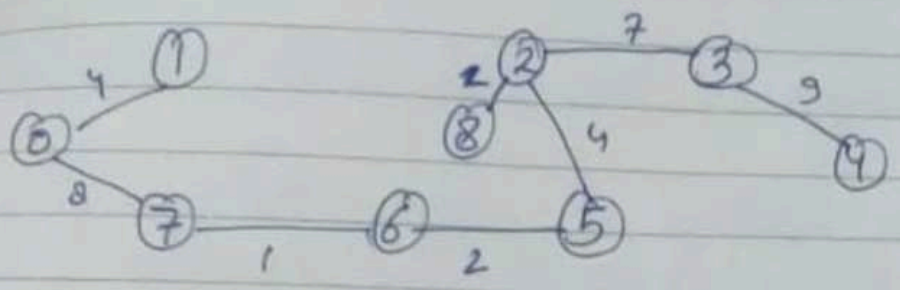
u	v	w	
6	7	1	✓
5	6	2	✓
2	8	2	✓
0	1	4	✓
2	5	4	✓
6	8	6	X
2	3	7	✓
7	8	7	X
0	7	8	✓
1	2	8	X
4	3	9	✓
4	5	10	X
1	7	11	X
3	5	14	X

Prim's Algorithm

Weight = 4+8+2+4+2+7+9+3
= 37

Mohan

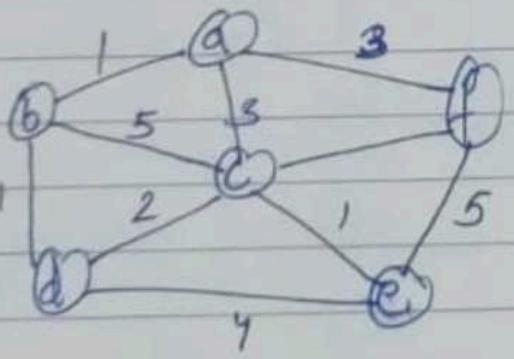
Date



Weight: 37

4) Given a directed weighted graph. You are also given the abstract path from a source vertex 's' to a destination vertex 't'. Does the shortest path remain same in following cases:

i) If weight of every edge is increased by 10 units.
ii) If " " " " " multiplied " " ".



Ans i) The shortest path may change. The reason is that there may be different no. of edges in different paths from 's' to 't'.

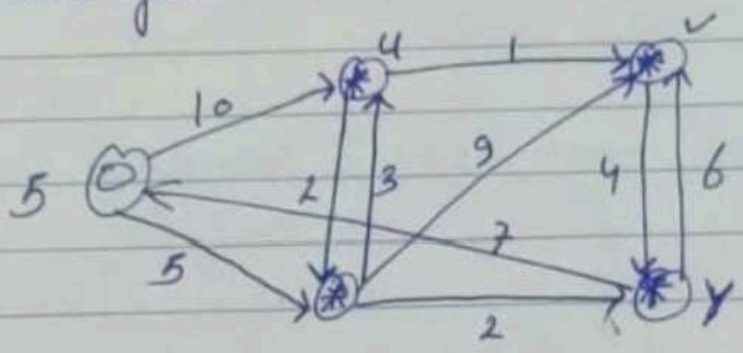
For e.g. - Let the shortest path of weight 15 & has edges 5. Let there use another path with 2 edges & total weight 25. The weight of shortest path is increased by 5*10 & becomes 15+50. Weight of other path is increased by 2*10 & becomes 25+20. So, the shortest

Date

path changes to other path with weight as 45.

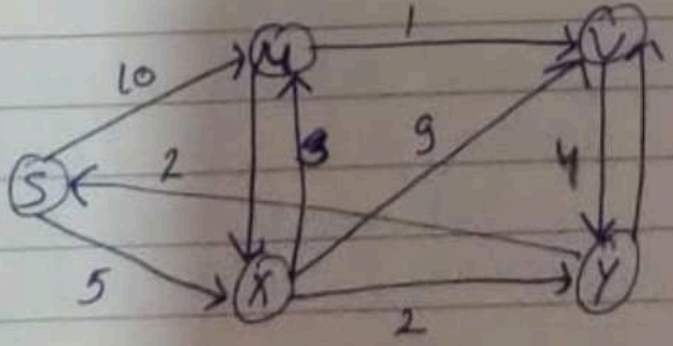
ii) If we multiply all edges weight by 10, the shortest path does not change. The reason is that weights of all path from 's' to 't' gets multiplied by same unit. The number of edges on path doesn't matter.

5. Apply Dijkstra & Bellman Ford Algorithm on graph given right side to compute shortest path to all nodes from node s.



Ans Dijkstra's Algorithm

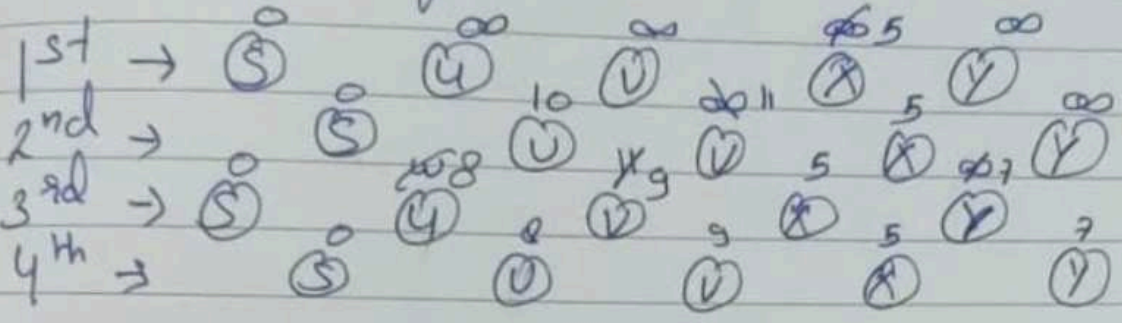
Node	Shortest Distance from source node
u	8
x	5
v	9
y	7

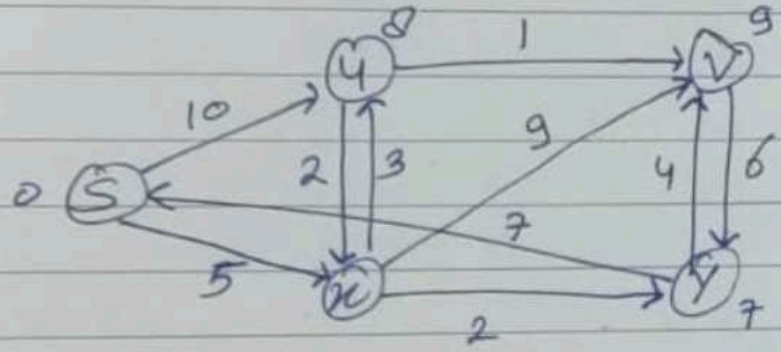


Mohan

Date

Bellman Ford Algorithm →

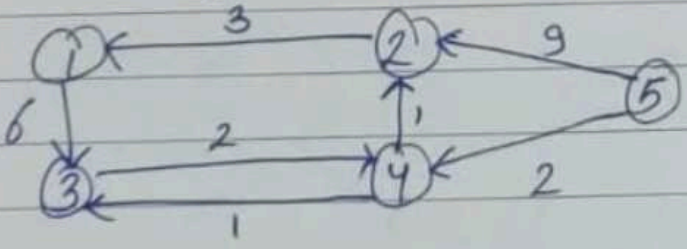




Final graph

6. Apply all pair shortest path algorithm. Floyd Warshall on below mentioned graph. Also analyse space & time complexity of it.

Ans



	1	2	3	4	5
1	0	∞	6	3	∞
2	2	0	∞	∞	∞
3	∞	∞	0	2	∞
4	∞	1	1	0	∞
5	∞	4	∞	2	0

Date

	1	2	3	4	5
1	0	∞	6	3	∞
2	2	0	8	5	∞
3	∞	∞	0	2	∞
4	∞	1	1	0	∞
5	∞	4	∞	2	0

	1	2	3	4	5
1	0	∞	6	3	∞
2	2	0	8	5	∞
3	∞	∞	0	2	∞
4	3	1	1	0	∞
5	6	4	12	2	0

	1	2	3	4	5
1	0	∞	6	3	∞
2	2	0	8	5	∞
3	∞	∞	0	2	∞
4	3	1	1	0	∞
5	6	4	12	2	0

Time complexity → $O(|V|^3)$

Space complexity → $O(|V|^3)$

} Ans

Mohan