Name - Dhruv Saini

Section - ML

Roll no - 2015241, 24

Subject - Design & Analysis of Algo.

TCS 505

Tutorial

1

BFS	DFS
• BFS stands for breadth first search. It uses queue datastructure for finding the shortest path	• DFS stands for Depth First Search It uses stack datastructure.
• BFS can be used to find single source shortest path in an unweighted graph, because in BFS, we reach a vertex with minimum no. of edges from source.	• In DFS, we might traverse through more edges to reach a destination vertex from a source.
• Siblings are visited before the children	• Children are visited before the sibling
- Application	
• Shortest path and min. spanning tree for unweighted graph.	• Path finding
• Peer to Peer networks	• Topological sorting
• Cycle detection in unweighted graph.	• To test if a graph is bipartite.

2

BFS does the search for nodes level by level i.e. it searches the nodes with respect to their distance from root. Here, siblings are visited before children.

We use queue as it is FIFO data structure, we visit

the node which is discovered first from the root.

For DFS, we retrieve it from root to the farthest node as much as possible, same idea as LIFO. Therefore we use stack data structure. Here children are visited before the siblings.

3. A graph with relatively few edges is sparse graph
   It is a graph $G(V, E)$ in which $|E| = O(|V|)$
   - A graph with many edges is dense
     Dense graph is a graph $G(V, E)$ in which $|E| = O(|V^2|)$
   - For sparse graphs, adjacency list can be used for representation.
   - For dense graphs, adjacency matrix can be used for representation

4. Detecting a cycle in directed graph using BFS
   1) Compute in-degree (no. of incoming edges) for each of the vertex present in the graph & initialize the count of visited node as 0.
   2) Pick all the vertices with in-degree as zero & add them into a queue
   3) Remove a vertex from the queue and then:
      - Increment count of visited nodes by 1.
      - Decrease in-degree by 1 for all its neighbouring nodes.
      - If in-degree of a neighbouring node is reduced to zero then add it to the queue
   4) Repeat step 3 until the queue is empty
   5) If count of visited nodes is not equal to the no. of nodes in the graph, the graph has cycle, otherwise not.

Detecting a cycle in directed graph using DFS
1) Create a graph using the given no. of edges & vertices.
2) Create a recursive function that initializes the current index or vertex, visited & recursion stack.

3) Mark the current node as visited and also mark the index in recursion stack

4) Find all the vertices which are not visited & are adjacent to the current node. Recursively call the function for those vertices, if the recursive function returns true, return true.

5) If the adjacent vertices are already marks in the recursion stack then return true.

6) Create a wrapper class, that calls the recursive function for all the vertices and if any function returns true return true. Else if for all vertices the function returns false return false.

5 Disjoint set is basically as group of sets where no item can be in more than one set. It supports union and find operation on subsets.

Assume that you have a set of N elements that are into further subsets and you have to track the connectivity of each element in a specific subset or connectivity of subsets with each other you can use the union find algorithm to achieve this.

operation on disjoint set.

$S1 : \{1, 2, 3\}$
$S2 : \{4, 5, 6\}$

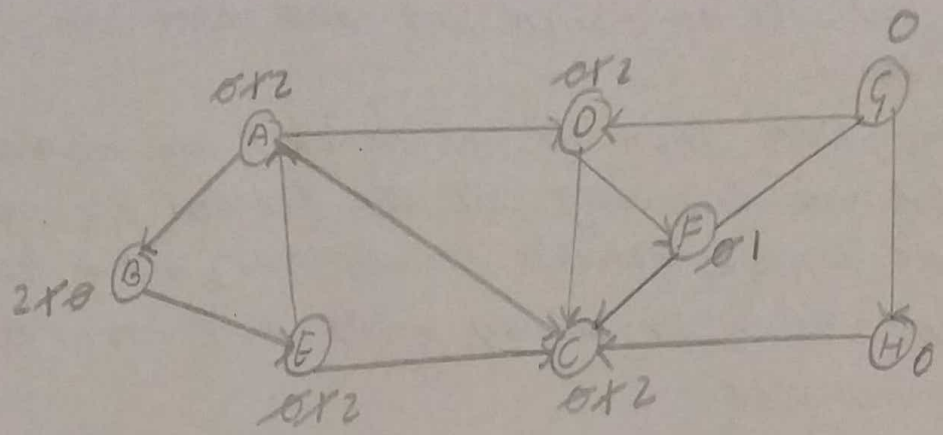
find(): It is used to find in which subset a particular set

find(1) = S1
find(3) = S1

union(): It merges two different subsets into a single subset and representative of other

$S1 \cup S2 = \{1, 2, 3, 4, 5, 6\}$

6

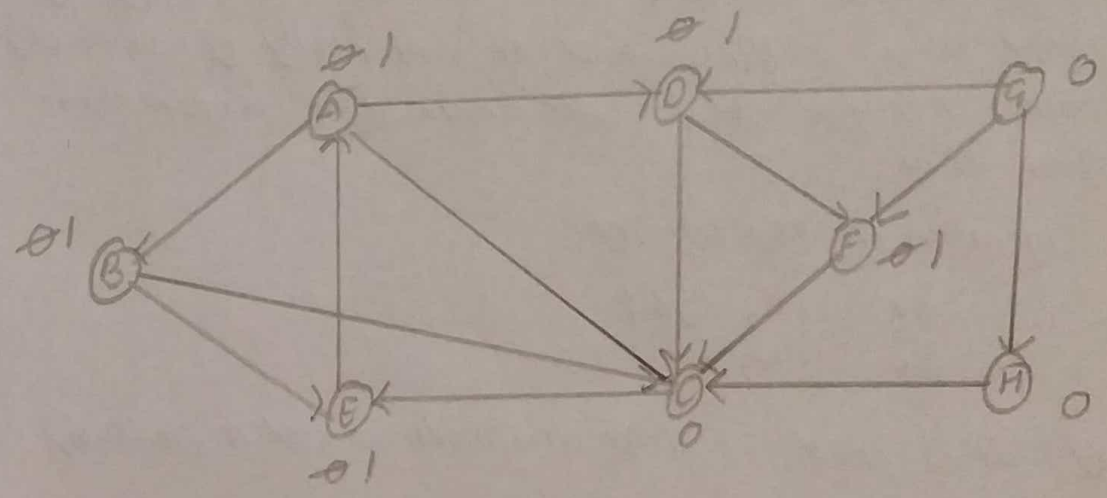


BFS Source = B Destination = F

Queue

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

Path : B → E → A → D → F

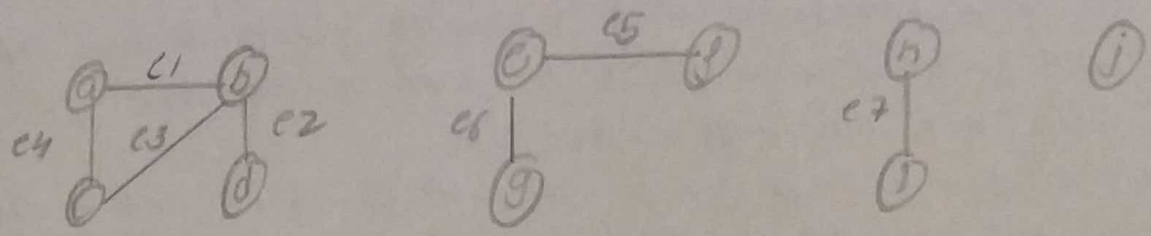


DFS Source B Destination = F

Stack

node processed	stack
-	B
B	GC
G	AC
A	DCC
D	FCCG
F	CCC

7



a b c d e f g h i j

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

-1 a a e -3 c c -2 h not connected

-3 -4 e

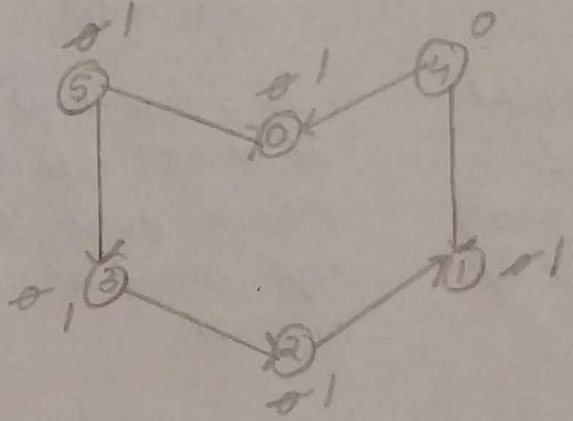
-4

4 connected to a

3 connected to c

2 connected to b

8



Topological sort

5, 2, 4, 0, 3, 1

DFS Source = 5

Node processed	Stack
-	5
5	20
2	30
3	10
1	0
0	-

9

Heaps are great for implementing a priority queue because of largest and smallest element at the root of the tree for a max-heap and min-heap respectively.

We use a max-heap for a max-priority queue and a min heap for min priority queue

Applications

1) Dijkstra's shortest path algorithm using priority queue
   - when the graph is stored in the form of adjacency list or matrix, priority queue can be used to extract minimum efficiency when implementing algo.
2) Prim's Algorithm: It is used to implement Prim's algo to store keys of nodes and extract minimum key node at every step.
3) Data compression: It is used in huffman codes which is used to compress data.

10

Min Heap	Max Heap
• 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 its 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.
• In a min heap the min. key element present at the root	• In a max-heap the maximum key element present at the root
• A min heap uses the ascending priority	• A max-heap uses the descending priority