Page No.
Date: / /

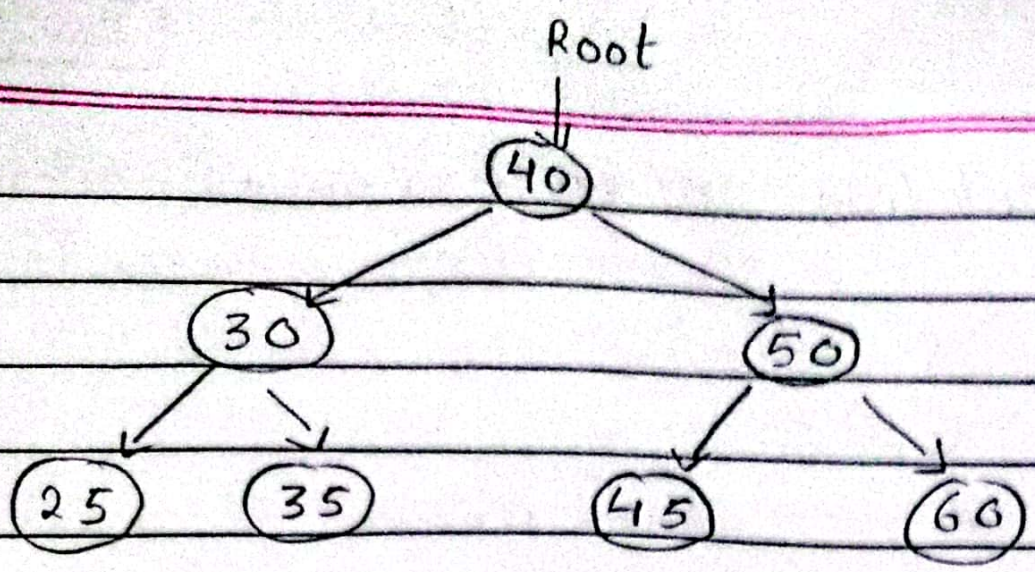
10.1 Write the steps required to delete a node from a Binary Search Tree (BST). Show your BST after deletion of 40.

=> Binary search is the search technique that works efficiently on sorted lists. It follows the divide and conquer approach.

A Binary Search Tree (BST) is a tree in which all the nodes follow the below-mentioned properties -
- The left sub-tree of a node has a key less than or equal to its parent node's key.
- The right sub-tree of a node has a key greater than or equal to it's parent node's key

In BST, we must delete a node from the tree by keeping in mind that the property of BST is not violated.

To delete a node from BST, there are three possible situation occur:-
- The node to be deleted is the leaf node or
- The node to be deleted has only one child and
- The node to be deleted has two children.



According to question we must dele 40 from tree i.e. third case is applicable

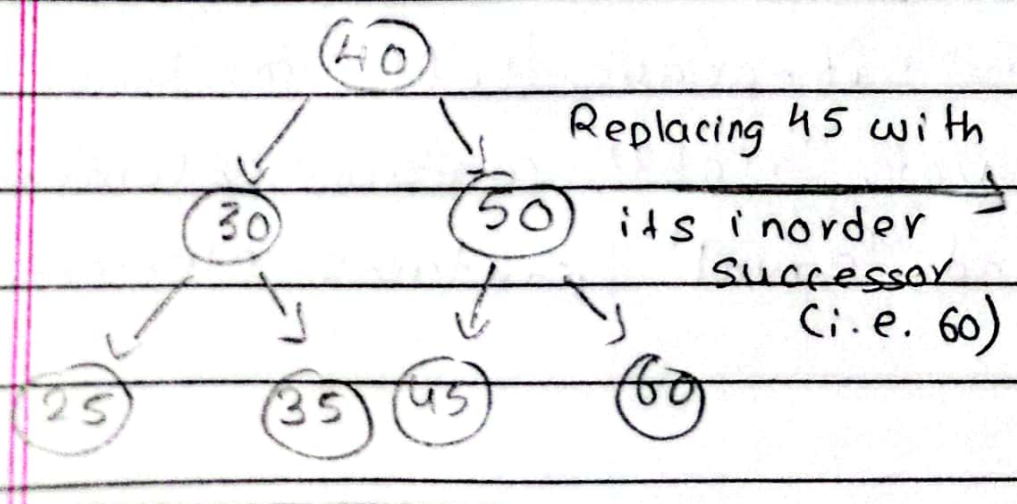
To dele 40, following steps should be implemented:

When the node to be deleted has two children

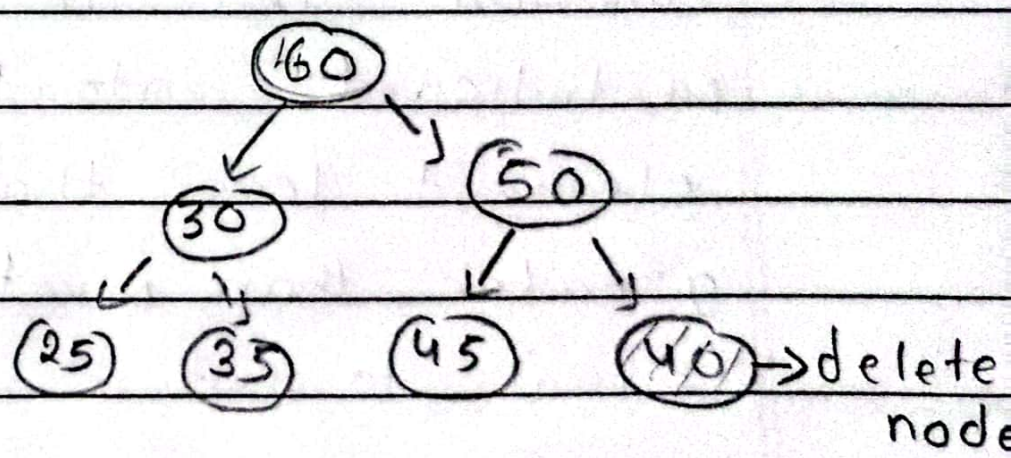
STeps:-

1) First, find the inorder successor of the node to be deleted.
2) After that, replace that node with the inorder successor until the target node is placed at the leaf of tree.
3) And, At last replace node with NULL and free up allocated space.

i.e. :-

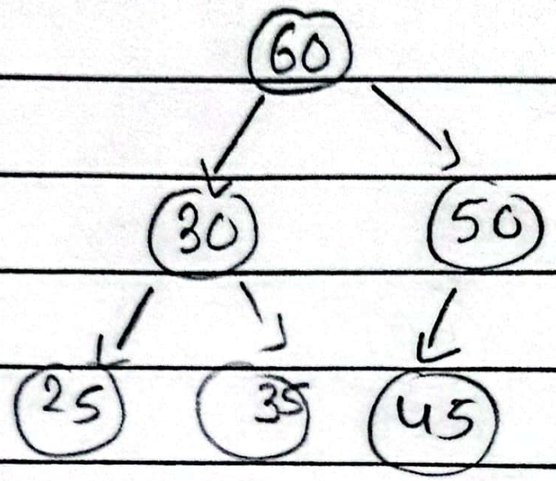


Replacing 45 with its inorder successor (i.e. 60)



Delete node 40

BST after deletion of 40:-



Q no.2 What is sorting? Write an algorithm for quick sort.

=> The arrangement of data in a preferred order is called sorting in data structure.

A Quick sort is the widely used sorting algorithm that makes nlogn comparisions in average case for sorting an array of n elements

eturn arr

- x - x - x - x - x - x - x

Algorithm

In the following algorithm, arr is the given array start is the starting index, and end is the ending index, p is partitioning index.

Quick_SORT(arr, start, end)

```
if ( start < end)
Set   p = partition (a,start , end)
Quick_SORT(arr, start, p - 1)
Quick_SORT ( arr, p+1, end);
 end of if
END Quick_SORT
```

The important part of the merge quick sort is Quick function. This function perform three sorted that are, partition, swap left swap right.

3. Explain various collision resolution techniques in hashing with example.

=> Finding an alternate location for the hashed key is called collision resolution.
Several commonly used collision resolution techniques are:-

Collision Resolution Techiques

- Separate Chaining (Open Hashing)
- Open Addressing (Closed Hashing)
  - → Linear probing
  - → Quadratic Probing
  - → Double Hashing

1. Separate chaining

In separate chaining each location in the data structures (eg: hash table) contains a linked list or another data structure to store multiple elements that hash to same location.

Eg:

Index	Elements		
0			
1			
2	7	→ 17	→ 27
3	12		
4	22		

Page No.
Date : / /

Consider a hash table with array of size 5. Using sample hash function, ("key·1.5") we get hash values for keys: 7, 12, 17, 22, &27. All these keys hash to same location. instead of ~~appending~~ replacing element separate chaining appends them to a linked list at index 2. as shown in table.

2. closed hashing

a) linear probing

insert k_i at first free location from $(u+i)\cdot 1.m$ where $i = 0$ to $(m-1)$.

It is a method where, when a collision occurs, the algorithm searches for the nex available empty slot in the data structure to place the element.
let's consider, has function key ·1·7 and seaurence : 50, 700, 76, 85, 92, 73, 101.

0		0		0	700	0	700	0	700	0	700	
1		1	50	1	50	1	50	1	50	1	50	
2		2		2		2	85	2	85	2	85	
3		3		3		3		3	92	3	92	
4		4		4		4		4		4	73	
5		5		5		5		5		5	101	
6		6		6	76	6	76	6	76	6	76	
with	Empty table		Insert 50		Insert 700 & 76		Insert 85: collision occur Insert at next slot		Insert 92: collision occur Insert next slot		Insert 73 & 701	

The problem with linear probing is clustering.

Page No.
Date: / /

2. Quadrating probing

in

Quadrating probing is similar to linear probing but instead of a linear increment to find next empty slot. It uses a quadratic function.

insert k_i at free space from $(u+i^2)\%m$ where

$i = 0$ to $(m-1)$.

Eg:

Table size = 10, hash(x) = x mod 10, $f(i) = i^2$,

$h_i(x) = (hash(x) + f(i)) \% \text{ table size}$

Insert keys : 89, 18, 49, 58, 69.

$h_0(58) = has(58) = 58 \% 10 = 8$

$h_1(58) = has(58) + f(1) \% 10$

$= (8+1^2) \% 10 = 9$

$h_2(58) = hash(58) + f(2) \% 10$

$= (8+2^2) \bmod 10 = 2$

0	49
1	
2	58
3	
4	
5	
6	
7	
8	18
9	89

3. Double Hashing.

Double hashing uses two different hash functions to calculate the next probing index when a collision occurs. The second hash function helps to resolve collisions more effectively than linear probing

insert k_i at first free place from $(u+v*i) \% m$

Page No.
Date : / /

where i = 0 to (m-1) & u = h, k %. m and V = h2 k %. m.

Eg:

Table size = 10 elements

Hash1 (key) = key %. 10

Hash2 (key) = 7 - (k %. 7)

Insert keys : 89, 18, 49, 58, 69

Hash (89) = 89 %. 10 = 9

Hash (18) = 18 %. 10 = 8

Hash (49) = 49 %. 10

= 9 (collision)

= 7 - (49 %. 7)

= 7 position

Hash (58) = 58 %. 10 = 8

= 7 - (58 %. 7)

= 5 positions from [8]

Hash (69) = 69 %. 10 = 9

= 7 - (69 %. 7)

= 1

0	49
1	
2	
3	69
4	
5	
6	
7	58
8	18
9	89

Q11.4 Explain Adjacency matrix representation of graph with examples in undirected and directed graphs.

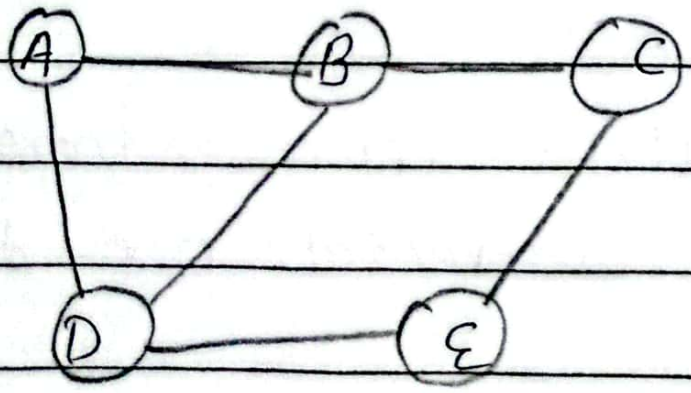
Page No.
Date : / /

An adjacency list is used in the linked representation to store the graph in the computer's memory whereas,
An adjacency matrix is used in the matrix representation to store the graph in computer's memory.

If an Undirected Graph G consists of n vertices then the adjacency matrix of a graph is an nxn matrix $A = [a_{ij}]$ and defined by'

$$a_{ij} = \begin{cases} 1, & \text{if } \{v_i, v_j\} \text{ is an edge i.e. } v_i \text{ is adjacent to } v_j \\ 0, & \text{if there is no edge bet}^n\ v_i \& v_j \end{cases}$$

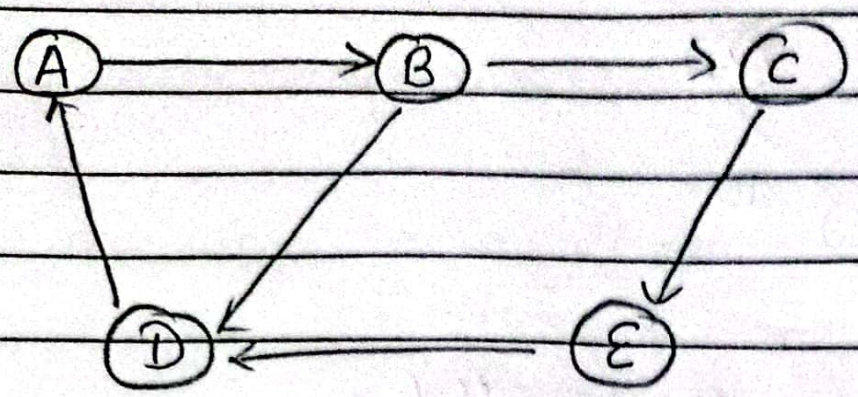
If there exist an edge between vertex v_i & v_j , where i is row & j is a column then the value of $a_{ij} = 1$.
If there is no edge between vertex v_i & v_j then $a_{ij} = 0$,
let 'us, see example :-
Undirected graph.



	A	B	C	D	E
A	0	1	0	1	0
B	1	0	1	1	0
C	0	1	0	0	1
D	1	1	0	0	1
E	0	0	1	1	0

Adjacency matrix representation.

(b) Directed Graph



	A	B	C	D	E
A	0	1	0	0	0
B	0	0	1	1	0
C	0	0	0	0	1
D	1	0	0	0	0
E	0	0	0	1	0

Adjacency matrix representation.

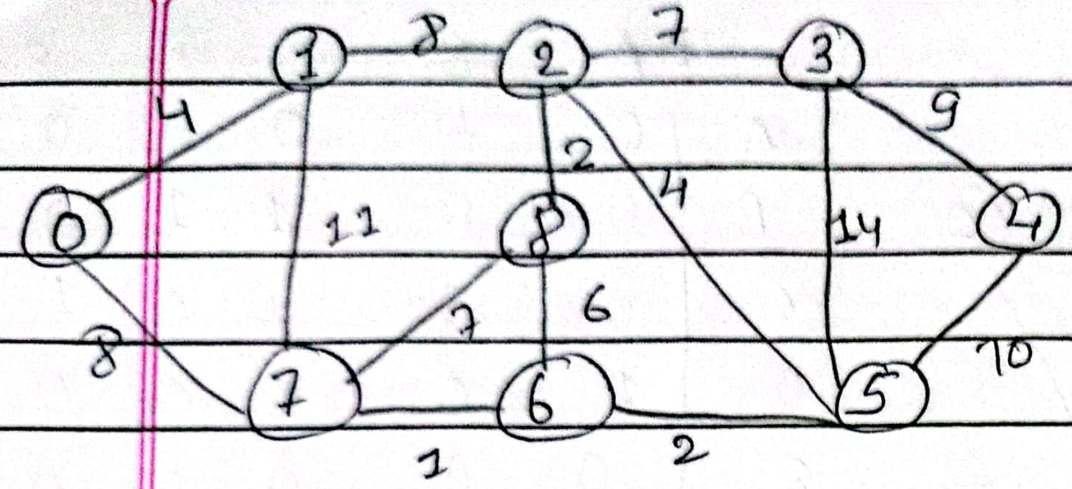
(5) Define graph traversing. Find the shortest path from 0 using Dijkstra's Algorithm.

⇒ Graph traversing, also known as graph traversal, is the process of visiting and exploring all the vertices or nodes of a graph in a specific order.

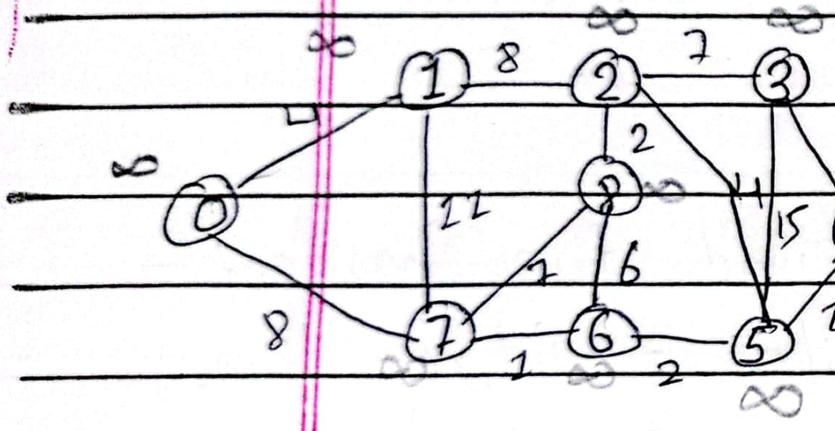
Dijkstra's Algorithm to find the shortest path

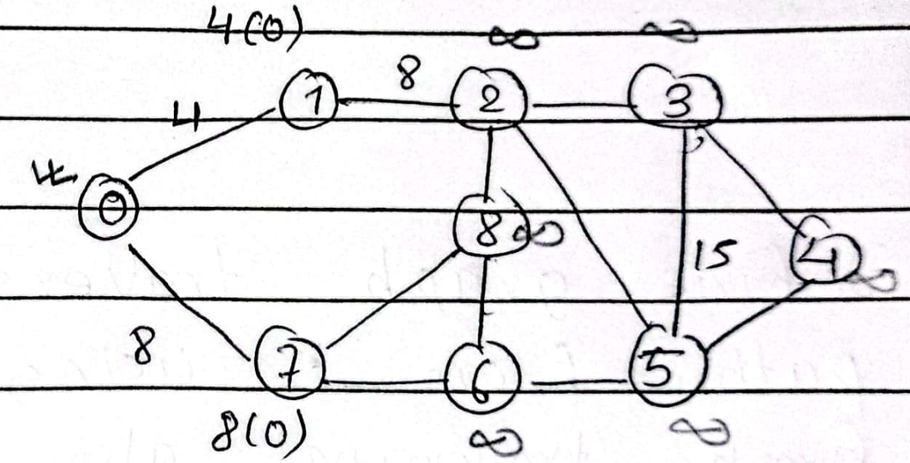
1. Mark all the vertices as unknown.
2. For each vertex u keep a distace du from source vertex s to u initially expect for s which is set to $d_s = 0$.
3. repeat these steps until all vertices are known
   i) Select a vertex u, which has the smallest d_u among all the unknown vertices
   ii) mark u as visited.
   iii) For each vertex v adjacent to u.
      if v is unvisited and $d_u + cost(u,v) < d_v$
      update dv to $d_u + cost(u,v)$.

To find shortest path from 0:-



Given graph with weights.





(a)

(b)