2018|24

VIVA QUESTIONS

Q-1) Which is the best sorting Algorithm?

ANS-1) An Algorithm will be efficient if it has linear time complexity O(n) & atmost logarithmic space complexity O(logn)

	Best Case		Worst Case	
	Time	Space	Time	Space
Bubble Sort	$O(n^2)$	$O(1)$	$O(n^2)$	$O(1)$
Insertion Sort	$O(n)$	$O(1)$	$O(n^2)$	$O(1)$
Merge Sort	$O(n \log n)$	$O(n)$	$O(n \log n)$	$O(n)$
Quick Sort	$O(n \log n)$	$O(\log n)$	$O(n^2)$	$O(n)$

Q-2) Compare Bubble, merge, Quick & Insertion sort?

ANS-2) ① Bubble Sort

It is a simple sorting algorithm that repeatedly steps through the input list of elements, comparing the current element with one after the another.

Time complexity ⇒ $O(n^2)$ } worst case
Space complexity ⇒ $O(1)$

② Merge Sort

It is a sorting algorithm based on divide & conquer technique. It is a comparison based algorithm. We divide the bigger problem into smaller problems & then accumulate the solⁿ of smaller tasks to obtain solⁿ of bigger problem.

Time complexity ⇒ $O(n \log n)$ } worst case
Space complexity ⇒ $O(n)$

VIVA QUESTIONS

Q-1) Write down 3 applications of Binary Search.

Ans-1) (i) Binary search is effecitive when the data set is sorted.

(ii) Binary search are useful to find nearest value to given target.

(iii) Binary search are used for spell checkers & auto-correct.

Q-2) Write down 3 applications of Linear Search.

Ans-2) (i) Linear search is efficient for small data sets.

(ii) Linear search is efficient when the data set is unsorted.

(iii) Linear search is efficient for simple data processing

Q-3) Which is better: Linear or Binary search?

Ans-3) In majority of cases, Linear search lags behind Binary search as in binary search, we search for element, if it is more than mid, then, we eliminate the left half, else, eliminate right half.

Time complexities

Linear search $\longrightarrow O(n)$ $[T(n) = T(n-1) + c]$

Binary search $\longrightarrow O(\log n)$ $[T(n) = T(n/2) + c]$

Thus, Binary Search is better than Linear Search

③ **Quick Sort**

It is a sorting technique based on the divide & conquer technique. It is implemented using D & C where we divide the bigger problem into smaller problems, cummulate the soln of smaller problem to find soln of bigger problems.

(i) Unsorted Array

Time complexity ⇒ $O(n \log n)$
Space Complexity ⇒ $O(\log n)$ } Best case

(ii) Sorted Array

Time complexity ⇒ $O(n^2)$
Space complexity ⇒ $O(n)$ } worst case

④ **Insertion Sort**

It is a sorting algorithm where we pick element from unsorted part of array & move to the sorted part.

Time complexity ⇒ $O(n^2)$
Space complexity ⇒ $O(1)$

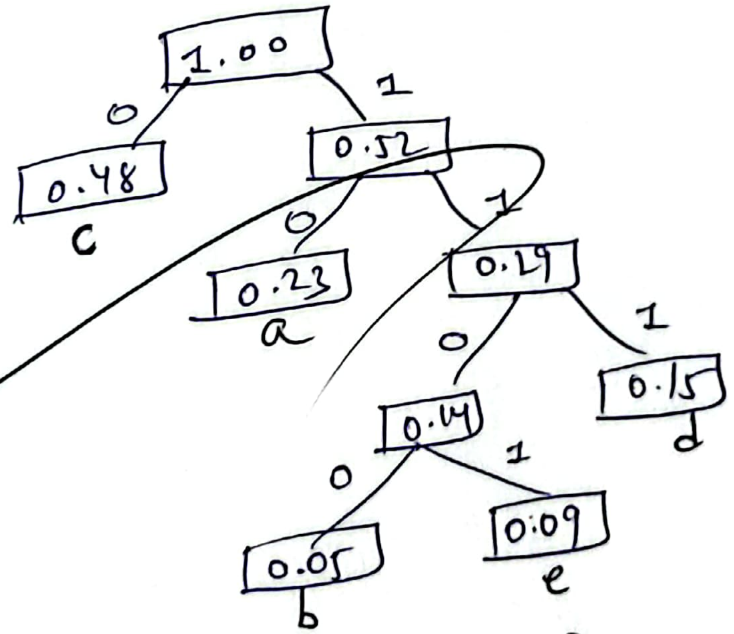
10/9/24

VIVA QUESTIONS

Q.1) What is Huffman Coding? Explain with a example?

Ans.) Huffman coding is been used to assign each character with a variable length case. The Length of each character is been decided based on it's occurances & frequency. The character which occurs the most no. of times would have the smallest case & vice-versa

For ex: $L = \Sigma(a, b, c, d, e)$: <0.23, 0.05, 0.48, 0.15, 0.09>



Therefore,
a : 10
b : 1100
c : 0
d : 111
e : 1101

Let text, = "ccacccadbe"

Uniform encoding ⇒ # Bits = 3×10 = 30 bits

Huffman encoding ⇒ # Bits = 17 bits

$(\# \text{Bits})_{\text{Uniform Encoding}} > (\# \text{Bits})_{\text{Huffman Coding}}$

Topic: Experiment - 04. Date: 03/09/24

⇒ Viva Question on Exp - 04.

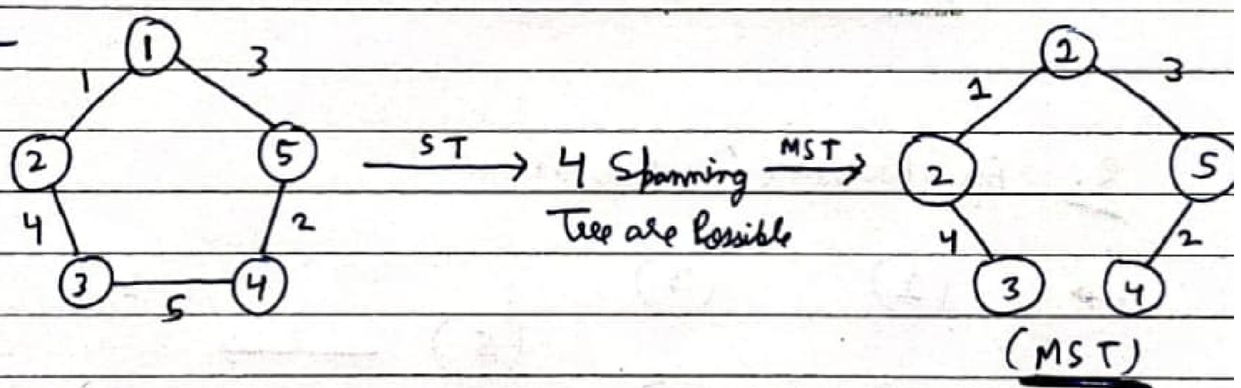
Q → What are Spanning Tree & MST?

→ A Spanning Tree is an Sub-Graph of Undirected Connected Graph. It Includes all The Vertices of The Graph & Min. Possible Edges.

→ The Spanning Tree should have all Connected Components but No Cycle.
→ The Spanning Tree must have V - Vertices & $|V|-1$ Edges.
→ For a Complete Graph of V - Vertices, Then, $\# ST = V^{(V-2)}$ Trees.

→ An MST is a Subset of The Edges of The Graph That Connects all The Vertices Together with No Cycle & Min. Possible Edge Weight Sum.
→ If The Edge Weights are distinct, Then, There Can be only 1 Unique MST.
→ For a Complete Graph, By Removing $(E - V + 1)$ Edges, We Can obtain ST.

Eg :-



Q → In which Way, we Can Represent a Graph?

→ The 2 Ways To Represent Graph are :-

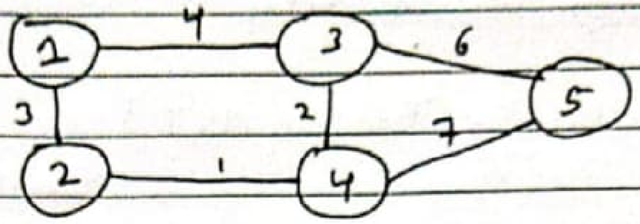
1. Adjacency Matrix : It Is a $(V \times V)$ Matrix, where, $V = \#$ Vertices.

$$a[i][j] = \begin{cases} 1, & \text{If Edge Exist b/w Vertex-i \& Vertex-j} \\ 0, & \text{If Edge Not Exist.} \end{cases}$$

2. Adjacency List : The Adjacency List Is a Way To Represent Graph using Linkedlist. The Space Required Is $O(V + 2E)$.

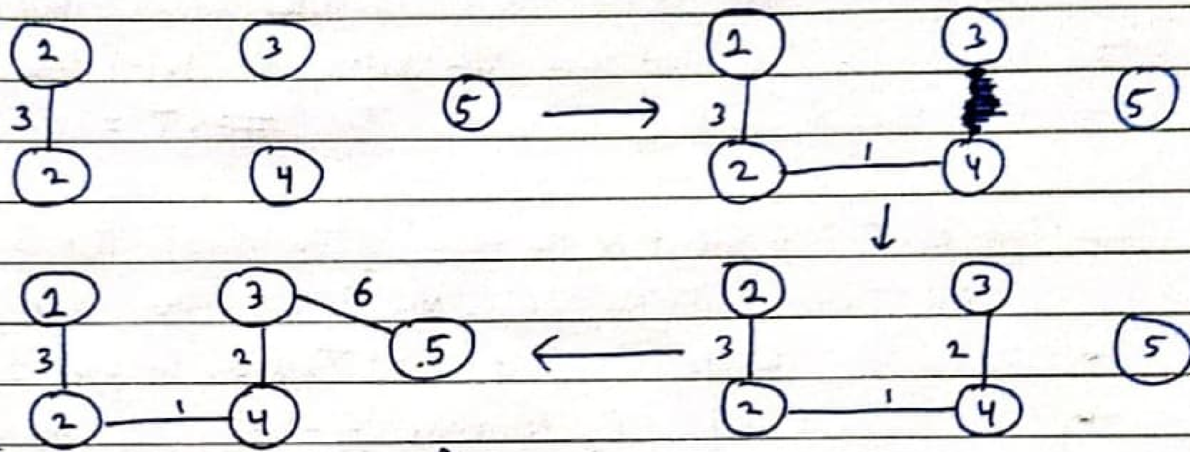
Topic: .... Date: ....

Q → Obtain The MST Of Below Graph using Prim & Kruskul Algo.



1. By Prim's Algorithm

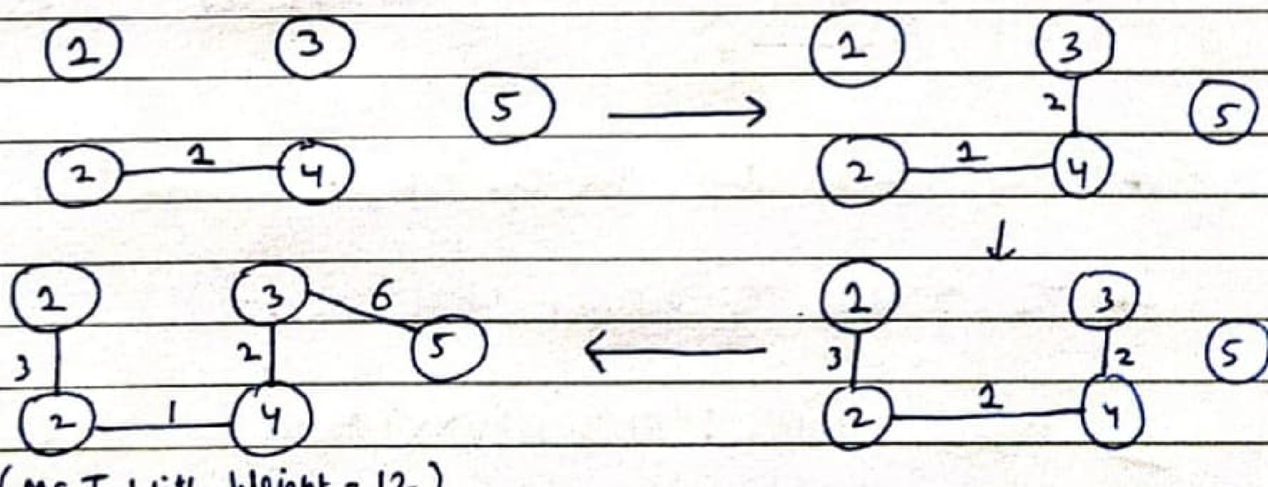
→



(MST with Weight = 12)

2. By Kruskul's Algorithm

→



(MST with Weight = 12)

★ In Prim's Algorithm, The Graph Is Connected In The Intermediate Stages, But, In Kruskul's Algorithm, The Intermediate Graph Can be Disconnected But Final MST's of Both Algo. will be Connected Graphs.

→ Viva Questions of Exp. -05.

Q→ What Is Dijkstra's Algorithm ? What Is It's Time Complexity ?

→ Dijkstra's Algorithm is used To find The shortest Path between The Starting Node and To all other Vertices of The Graph. It Works on The Graph The Edges will have all Non-Negative Weights.

→ In Case of Dijkstra's Algorithm :-

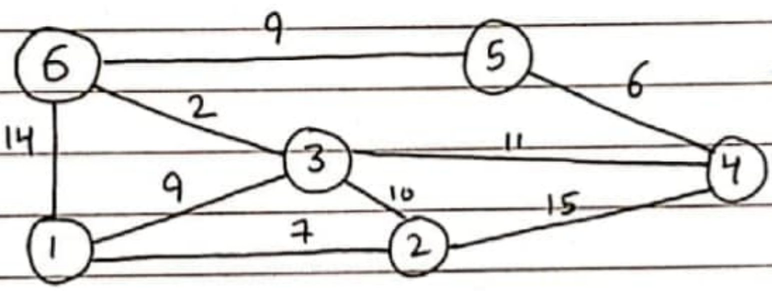
If, dist [A][C] + dist [C][B] < dist [A][B]

Then, dist [A][B] = dist [A][C] + dist [C][B] (Relaxation Step)

→ The Time Complexity of Dijkstra's Algorithm Is O(E log V).

* DRAWBACK :- This Algorithm Fails for (-ve) Weight Cycles.

Q→ Apply Dijkstra's Algorithm for Graph Where Source = 1 ?



Source 1	Destination 2	3	4	5	6
	∞	∞	∞	∞	∞
Itr - I.	7	9	∞	∞	14
Itr - II.	7	9	22	∞	14
Itr - III.	7	9	20	∞	11
Itr - IV.	7	9	20	20	11
Itr - V.	7	9	20	20	11

* The Dijkstra's Algorithm fails for directed Graph having -ve cycles.