Chapter (3) - 3 or 4 Q

2 sections

1) Divide and Conquer - 1 or 2 Q
2) Greedy method - 2 Q

1) Divide and Conquer Method

→ This technique can be divided into the following three parts

1) Divide:
→ This involves dividing the ~~sub~~ problem into smaller sub-problems

2) Conquer → solving the subproblems recursively.

3) Combine → Combine the subproblems to get the final solution of the whole problem

Defn

→ In divide and conquer method the problem is divided into smaller subproblems. These subproblems are solved independently. finally all the solutions of sub problems are collected together to get the solution of the given problem.

Scanned with CamScanner

Divide and Conquer (1 or 2 Q)

1) Explain binary search technique using divide and conquer method of problem solving and analyze its time complexity

→ It is an efficient algorithm for searching in a sorted array.

→ This search algorithm works on the principle of divide and conquer.

→ Searching is finding some particular item in the list.

Working

→ ~~Binary search looks for a particular item by comparing the~~

→ The array list is divided into two halves, and then item is compared with the middle element of the list. If the match is found then the location of the middle element is returned. Otherwise, if the middle item is greater than the item, then the item is searched in the sub array to the left of the middle item. Otherwise, the item is searched for in the sub-array to the right of the middle item. The process continues until the item is found.

Scanned with CamScanner

Algorithm

Let x is the item to be searched in a sorted array list array of size N

Step 1: Compare x with the middle element of an array

Step 2: If x matches with the middle element, we return the mid index.

Step 3: Else if (x is greater) than the mid element recur for the right half

Step 4: Else (x is smaller) recur for the left half.

→ ~~Example:~~ The time complexity for best case is $O(1)$, average & worst case is $O(\log n)$

→ Space complexity is $O(1)$

Example:

0	1	2	3	4	5	6
3	4	5	6	7	8	9

find 4

Step 1: find the middle element

$$mid = \frac{(start + end)}{2} = \frac{(0+6)}{2} = 3$$

~~Step 2:~~ 4 < A[mid]

4 < A[3]

4 < 6 (true)

So we skip the right side of the array

Scanned with CamScanner

Step 2:

0	1	2
3	4	5

$$mid = \frac{start + 2}{2} = \frac{0+2}{2} = 1$$

$$x = A[mid]$$
$$4 = 4$$

It is found at index 1

Successful search.

Scanned with CamScanner

Sorting - Divide & Conquer

2) Write an algorithm for merge-sort and find it's time complexity. Trace out the algorithm for the following array A[] = {16, 7, 15, 17, 18, 25, 55, 32}

→ It performs divide, conquer & combine.

→ In this algorithm, first the array is divided into $\frac{n}{2}$ sub-arrays and the process is followed until the single array will be left in sub-array. Then the sorted arrays are merged again into a new array.

Algorithm

Step 1: if it is only one element in the list it is already sorted, return.

Step 2: divide the list recursively into two halves until it can no more be divided.

Step 3: merge the smaller lists into new list in sorted order.

→ It's best, average and worst time complexity is $O(n \log n)$.

Disadvantages → it uses extra space.

Scanned with CamScanner

Given array

16	7	15	14	18	25	55	32

Step 1: There is ~~the~~ a total of 8 elements in the array. Thus, mid=4. So we will divide this array into two arrays of size 4

16	7	15	14	18	25	55	32

Step 2: we recursively divide these arrays into further halves. The half of 4 is 2

16	7	15	14	18	25	55	32

Step 3: Further dividing:

16	7	15	14	18	25	55	32

Step 4: Now we will compare each element with its consecutive elements and arrange them in a sorted manner and combine them.

7	16	15	14	18	25	55	32

Step 5: compare two arrays and sort them

7	14	15	16	18	25	32	55

Scanned with CamScanner

Step 6: we will compare the elements of the two arrays each size of 4 and we will get our resultant sorted array

| 7 | 14 | 15 | 16 | 18 | 25 | 32 | 55 |

Ans

3) Write an algorithm for bubble sort and trace out the algorithm for bubble sort with suitable example.

-> This sorting algorithm is comparision based algorithm in which each pair of adjacent elements is compared and the elements are swapped if they are not in order. The next element is compared with its adjacent element and the same process is repeated for all the elements in the array until ~~the~~ we get a sorted array.

Algorithm

Step 1: Starting with the first element (index=0) compare the current element with the next element of the array.

Step 2: If the current element is greater than the next element of the array, swap them.

Step 3: If the current element is less than the next element, move to the next element.

Step 4: Repeat Step 1, 2 & 3 until the array is sorted.

-> The average & worst case complexity is $O(n^2)$ and best case complexity is $O(n)$.

-> Space complexity is $O(1)$.

Scanned with CamScanner

Let the array be A[] = {13, 32, 26, 35, 10}

- First Pass

Step1: Sorting will start from the first two element. Compare to check which is greater.

| 13 | 32 | 26 | 35 | 10

∴ 32 > 13 . Already sorted.

Step2: Compare 32 & 26. ∴ 26 < 32 Swap the array

| 13 | 26 | 32 | 35 | 10 |

Step3: Compare 32 & 35 ∴ 35 > 32 . Already sorted

Step4: Compare 35 & 10 ∴ 10 < 35 swap them

The sorted array will look like

| 13 | 26 | 32 | 10 | 35 |

The position of 35 is fixed.

- Second Pass

Step1 Repeating above process from first element

∴ 13 < 26 already sorted

Step2: ∴ 26 < 32 already sorted

Step3 ∴ 32 > 10 ∴ Swap

| 13 | 26 | 10 | 32 | 35 |

The position of 32 & 35 is fixed

- Third pass

Step1: 13 < 26 ∴ already sorted

Step2 26 > 10 ∴ Swap

| 13 | 10 | 26 | 32 | 35 |

Scanned with CamScanner

Fourth Pass

13 > 10 (swap)

10	13	26	32	35

sorted array using bubble sort.

III a) Write the algorithm for quick sort and compute the time complexity. Point out the scenario where quick sort gives the worst case performance. Trace each steps of the algorithm on following array

A[ ] = ~~{16, 17, 15, 14, 18, 25, 55, 32}~~

= {15, 10, 13, 9, 12, 17}

Ans: It follows divide and conquer approach.

-> Divide: In divide, first pick an pivot element. After that, partition the array into two sub-arrays such that each element in the left subarray is less than or equal to the pivot element and each element in the right sub-array is larger than the pivot element.

Conquer: recursively, sort two arrays with quick sort.

Combine: Combine the already sorted array.

Choosing pivot

-> There are various method to choose pivots like

1) Select median as pivot

2) Pivot can either be leftmost element or the right most element of the given array.

Scanned with CamScanner

Note: P < R → move right pointer to left
∴ P > L → move left pointer to right
∴ P > R (swap pivot & element position)
∴ L > P (swap " " ")

left < Pivot < Right

Algorithm

Step1: Choose the first element as a pivot
Step2: Take two pointers to point leftmost element and rightmost element.
Step3: Left points to the low index
Step4: Right points to the ~~right~~ high index
Step5: While value at (left < Pivot). Move left pointer to right
Step6: While value at (right > Pivot). Move right pointer to left.
Step7: If both step 5 & 6 doesnot match. Swap the element of ~~rig~~ left pointer with pivot ~~and~~ the element of right pointer with pivot respectively.
Step8: If left pointer, pivot & right pointer are pointing at the same position then the position is fixed.
Step9: Perform Step 1 to Step 8 until all the position are fixed and array is sorted.
Step10: EXIT

→ The worst case time complexity is $O(n^2)$.
→ The average & ~~base~~ best case time complexity is $O(n\log n)$.
→ Space complexity is $O(n\log n)$.

Scanned with CamScanner

Given array [15 | 10 | 13 | 9 | 12 | 17]

P - Pivot element
L - Left pointer
R - right pointer

↓P

15	10	13	9	12	17

↑L ↑R

Step1: Check (P < R) : true move ~~left~~ right pointer to ~~right~~ left.

↓P

13	10	13	9	12	17

↑L ↑R

Step2: Check (P < R) : false (swap P & R)

↓P

12	10	13	9	15	17

↑L ↑R

Step3: Check (P > L) : true move L to right
Here, L is pointing to 10
Check (P > L) : true move L to right
Similarly
15 > 13
15 > 9

↓L ↓P

Step4:

12	10	13	9	15	17

↑R

Since L, P & R met at same point it is a fixed position. Now divide array.

Step 5 Let's perform quick sort for left & right

Step 6:

12	10	13	9

↓P ↑L ↑R

Check (P < R)

∴ (12 < 9) ∴ false (swap P & R)

Step 7:

9	10	13	12

↓P ↑L ↑R

Similarly

∴ (9 < 12) true ∴ Move L to right

∴ (10 < 12) true ∴ Move L to right

∴ (13 < 12) false ∴ swap

Step 8:

9	10	12	13

↓P ↑L ↑R

∴ (12 < 13) true ∴ Move R to left

Step 9:

9	10	12	13

↓P ↑L↑R

Since all met in same place again divide the array.

Step 10:

9	10

↓P ↑L ↑R

Check: 9 < 10 (Move R to left

9	10

↓P ↑L↑R

Finally, combine all arrays.

9	10	12	13	15	17

Ans

5) Explain a MaxHeap Data Structure. Give an example where such a data structure may be used. Write the algorithm to dele the max element from a MaxHeap and analyse it's time complexity.

Heap Data Structure

-> Heap is a special case of balanced binary tree data structure where the root-node is compared with its children and arranged accordingly.

Max-Heap

-> Here, the value of the root node is greater than or equal to either of its children.

-> It is used for ascending sorting.

Max-Heap Construction Algorithm (-> not asked in exam. Just for understanding)

Step 1: Create a new node at the end of heap.

Step 2: Assign new value to the node.

Step 3: Compare the value of this child node with its parent.

Step 4: If value of parent is less than child, then swap them.

Step 5: Repeat step 3 & 4 until Heap property holds.

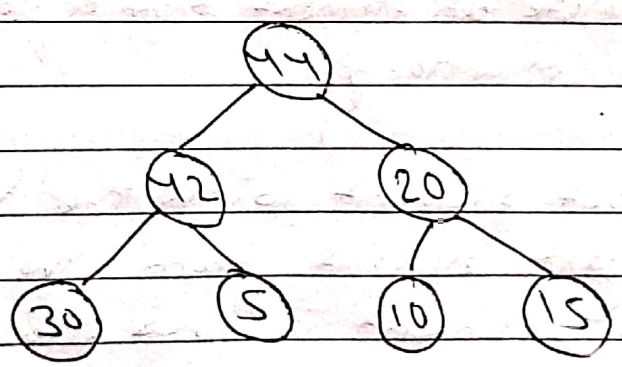
Scanned with CamScanner

Max Heap Deletion Algorithm (Asked in Exam)

Step1: Replace the root element with the last element
Step2: Delete the last element from the heap.
Step3: Since the last element is now placed at the position of the root node. So it may not follow the heap property. Therefore, heapify.

The worst time complexity is O(logn)

Example of MaxHeap

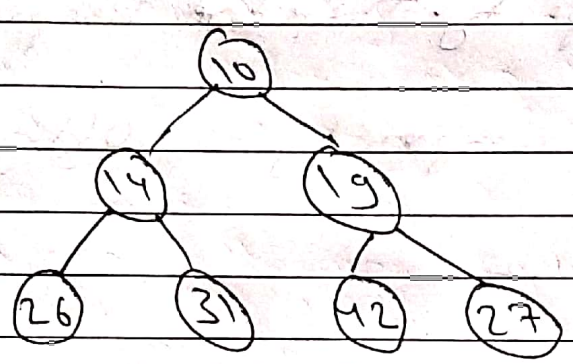


Scanned with CamScanner

6) Explain a min Heap. Write an algorithm to delete an element from a minHeap and find its time complexity.

Ans: In minHeap, the root node is less than or equal to either of its children.

Eg:



Algorithm to delete an element from a minHeap.
(Same like MaxHeap)

Step1: Replace the root element with the last element.
Step2: Delete the last element from the heap.
Step3: Since, the last element is now placed at the position of the root node.
Step4: If it ~~no~~ doesnot follow the minHeap property then Perform Heapify

~~Step5:~~

→ The worst case time complexity is O(logn)

17) Finding MIN and MAX algorithm using iterative method:

```
minMax (A,n)
{
    max = min = A[0]
    for (i=1; i<n; i++)
    {
        if (A[i] > max)
            max = A[i];
        else if (A[i] < min)
            min = A[i];
    }
}
```

→ The worst case complexity to find min max using iterative method is O(n).

Scanned with CamScanner

Q) Compute x^{29} by using divide and conquer technique.

Divide and Conquer approach for solving exponential problem

→ In divide and conquer approach, exponential of x^n is achieved by creating sub problems of size $x^{n/2}$. They are sub problems are divided recursively until they hit ~~the~~ to the power 1. The recursive solution to the problem is depicted in following equation

$$x^n = \begin{cases} x^{n/2} \cdot x^{n/2} & \text{if } n \text{ is even} \\ x^{(n-1)/2} \cdot x^{(n-2)/2} \cdot x & \text{if } n \text{ is odd} \end{cases}$$

If power of x is even, then simply we can find x^n by computing $x^{(n/2)} \times x^{(n/2)}$
If power of x is odd, we can compute x^n by solving $x^{(n-1)/2} \times x^{(n-1)/2} \times x$.

Problem size reduces by factor of 2 in every recursive call and very soon it hits a base case. This division of problem creates tree of height $\log_2 n$.

Scanned with CamScanner

1111.8) Strassen's Matrix Multiplication

or

Provide the algo. Explain each step with example.

or,

Divide & conquer approach to solve Strassen's Matrix Multiplication

Ans: Let us consider two matrices A and B. We want to calculate the resultant C by multiplying A and B.

In Normal method

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \qquad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$$A \times B = \begin{bmatrix} A_{11} \times B_{11} + A_{12} \times B_{21} & A_{11} \times B_{12} + A_{12} \times B_{22} \\ A_{21} \times B_{11} + A_{22} \times B_{21} & A_{21} \times B_{12} + A_{22} \times B_{22} \end{bmatrix}$$

$$\therefore C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = A \times B$$

There will be three for loops in this method. So the time complexity of this algorithm is $O(n^3)$.

Scanned with CamScanner

In this context, using Strasser's Matrix Multiplication ~~can~~ algorithm, ~~the~~ time consumption can be improved a little bit.

Strasser's matrix multiplication can be performed only on square matrices where n is a power of 2. Order of both matrices are $n \times n$.

$A \times B = C$ (we should compute this)

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}, \quad C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

Using Strasser's Algorithm compute the following:-

$P = (A_{11} + A_{22})(B_{11} + B_{22})$

$Q = B_{11}(A_{21} + A_{22})$

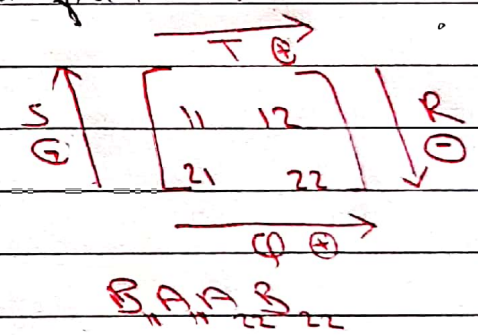
$R = A_{11}(B_{12} - B_{22})$

$S = A_{22}(B_{21} - B_{11})$

$T = B_{22}(A_{11} + A_{12})$

$U = (A_{21} - A_{11})(B_{11} + B_{12})$

$V = (B_{21} + B_{22})(A_{12} - A_{22})$



C_{ij}

$C_{11} = P + S - T + V$

$C_{12} = R + T$

$C_{21} = Q + S$

$C_{22} = P + R - Q + U$

RAT

→ Time complexity of Strasser's Matrix Multiplication is $O(n^{\log 7})$

Scanned with CamScanner

Greedy Approach (20)

1) How does greedy method help in problem solving.

Ans: A greedy algorithm is an approach for solving a problem by selecting the best option available at the moment. It doesn't worry weather the current best solution will bring the overall optimal solution.

→ The algorithm never reverses the earlier decision even if the choice is wrong.

→ Greedy algorithm look for locally optimal solutions in the hopes of finding a global optimal solution.

Applications of Greedy Algorithm

1) It is used in finding the shortest path.
2) It is used to find minimum spanning tree using prims or kruskals algorithm.
3) It is used in Job sequencing with deadline.
4) It is used to solve fractional knapsack problem.

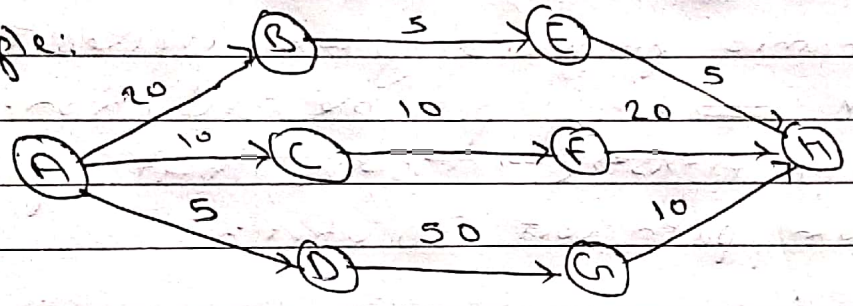
Scanned with CamScanner

2) Can greed approach be optimal always? Provide an example to support your answer (in terms of time/space usage).

Ans: Greedy algorithms look for locally optimal solution in the hopes of finding a global optimal solution.

→ Hence, greedy algorithm can make a guess that looks optimal at the time but becomes costly down the line and doesn't guarantee a global optimal solution.

Example:



We have to travel from source to destination ~~with~~ at minimum cost.

Using greedy approach

A → D = 5 unit

A → C = 10 unit

A → B = 20 unit

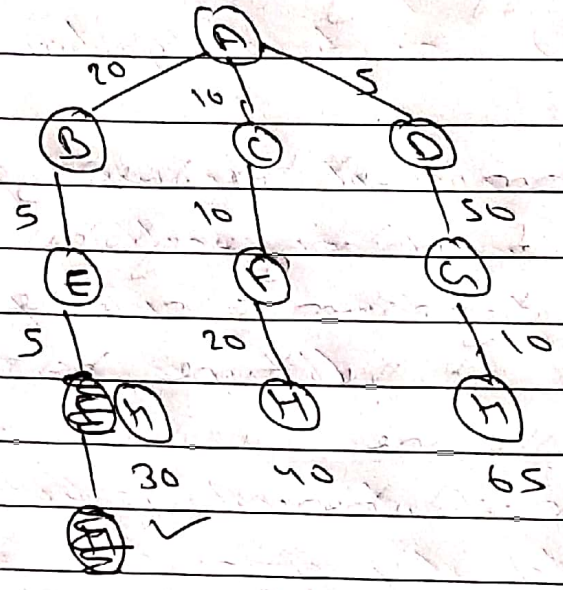
We choose A → D since it has the lowest cost.

Similarly,

We choose path A → D → G → H with path cost 65 units by using greedy method.

Scanned with CamScanner

Using dynamic programming



→ In dynamic programming, we checked the path cost for ~~one~~ every possible solution.
Here, A→B→E→H is the best path with minimum path cost 30 unit.

Path cost using greedy was 65 unit with path A→D→G→H.

Here, greedy approach doesnot always give optimal solution.

Scanned with CamScanner

Q3) Job sequencing with deadline

or

Explain Job sequencing algorithm

or

Greedy approach to solve Job sequencing problem. Analyze its complexity

Ans: In Job sequencing problem, the objective is to find a sequence of jobs, which is completed within their deadlines and gives maximum profit.

→ Here, several jobs are given with its ~~profit~~ associated profit if the job is finished before the deadline.

→ One job should be done in one unit of time.

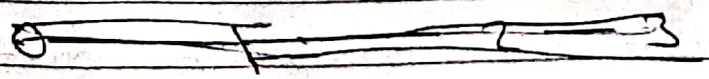
→ We should select jobs in such a way that we will receive a maximum profit.

Example:

Job	J_1	J_2	J_3	J_4	J_5
Profit	20	10	15	25	5
Deadline	1	2	3	1	3

From above table

→ First, we will select a job with maximum profit and place it in a deadline.



Scanned with CamScanner

J_4

0 —— 1 —— 2 —— 3

→ J_4 has the maximum profit so ~~placing~~ we select it and place it in (0 – 1)

→ Moving forward, J_1 has maximum profit and it's deadline is 1. But the space is already taken by J_4. So we donot take this job i.e (the job is rejected)

→ Again, moving forward, J_3 has maximum profit from the remaining job and its deadline is 3. So place it on (2 – 3)

J_4 J_3

0 —— 1 —— 2 —— 3

→ Similarly, placing J_2 on the deadline

J_4 J_2 J_3

0 —— 1 —— 2 —— 3

→ Since all the position is filled we do not have to check further.

→ All the Jobs are placed and the maximum profit is : $J_4 + J_3 + J_2$

$= 25 + 15 + 10 = 50$

→ Using greedy method, we chose a job in a sequence which will give us a maximum profit.

Scanned with CamScanner

minimum cost.

4) Optimal Merge Pattern

Ans: Optimal Merge Pattern is a process of merging a given set of files of different len

4) Optimal Merge Pattern

→ In this method, we have to merge a given set of files into a single file. We need to find an optimal solution, where the resultant file will be generated in minimum computation (i.e minimum time)

→ This process is known as Optimal Merge Pattern

→ The files should before starting the operation.

→ We use the greedy method by merging the two smallest size files among all the files present and repeat the process until all the ~~merges~~ files is merged into one single file.

→ Time complexity is O(nlogn).

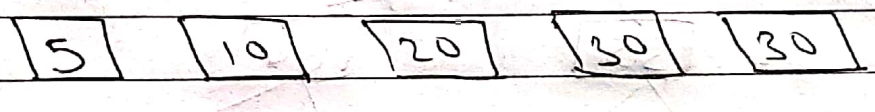
Scanned with CamScanner

Example:

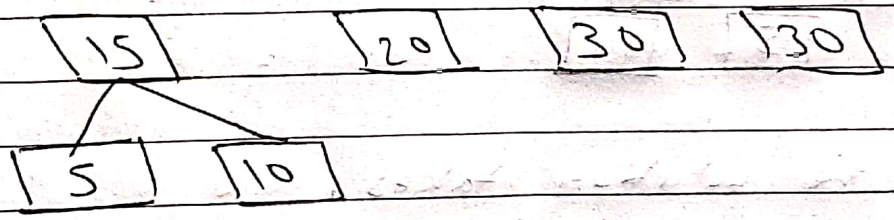
Let us consider the given files f_1, f_2, f_3, f_4 & f_5 with 20, 30, 10, 5 and 30 number of elements respectively. (no. of records)

Sorting → 5, 10, 20, 30, 30

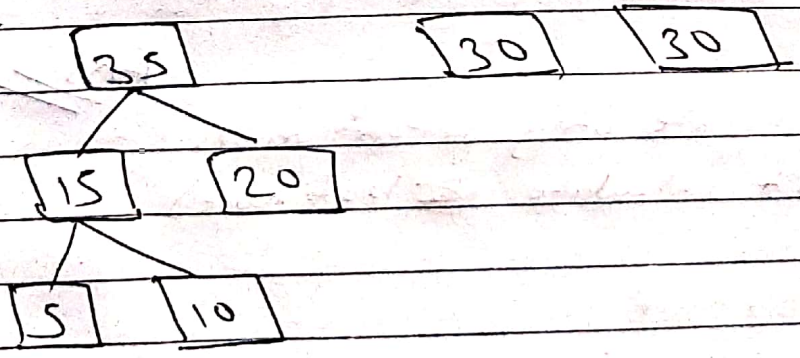
Initial step



Step-1

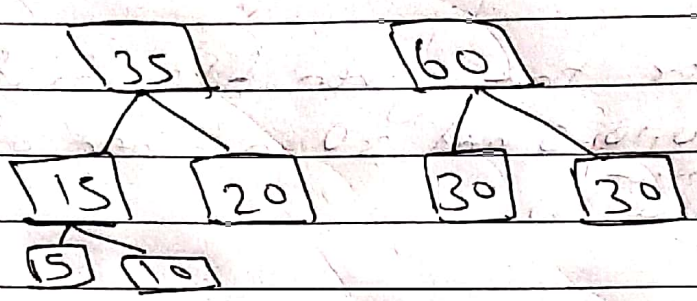


Step-2

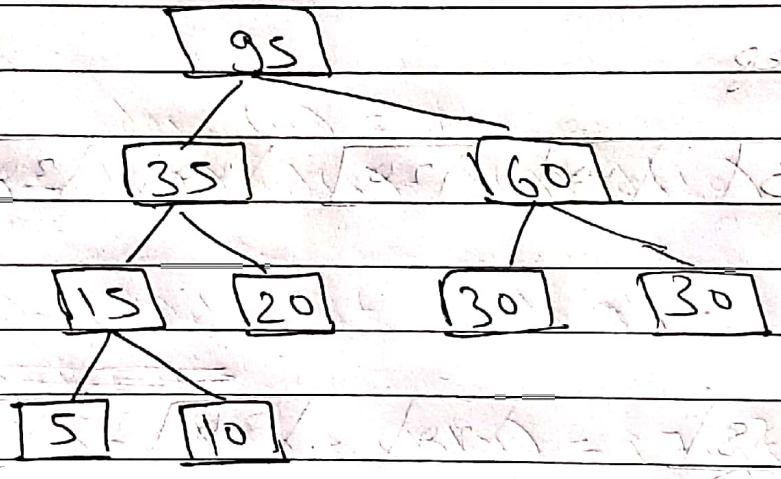


Scanned with CamScanner

Step-3



Step-4



Here the solution takes,

$15+35+60+95 = 205$ computation.

Ans

Note: Calculate only parent nodes.
(Don't calculate leaf node)

Scanned with CamScanner

~~5) Tree~~

5) Tree Vertex Splitting Problem

or

Provide an analysis of tree vertex splitting

Ans: It uses greedy method of problem solving.

→ It is based on a problem that if the delay of a vertex is greater than the condition delta (i.e δ) then its child should be splitted.

delay $d(u) = \max_{v \in ch(u)} \{ d(v) + c_e(u,v) \}$

here,

c_e → cost of edge — u - parent node

$ch(u)$ → set of child of u — v - child node

$d(u)$ - delay of u node

for all the leaf node → u

$d(u) = 0$

u is a parent node of v such that

$\{ d(u) + c_e(u,v) \} > \delta$

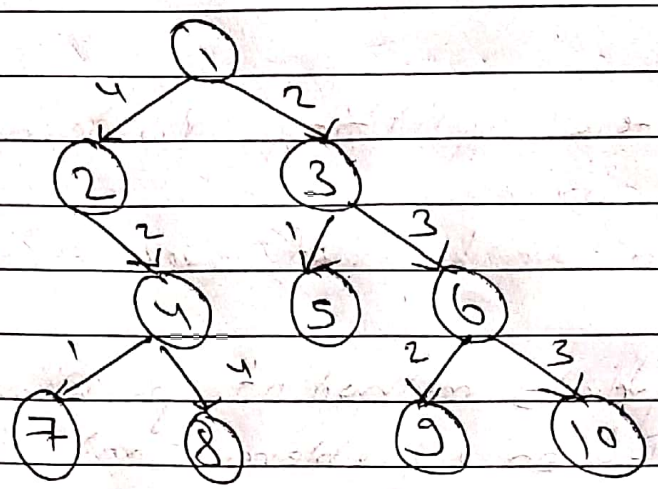
then node v gets splitted into v_i, v_o

i - incoming

o - outgoing

$d_u = \max_{v \in ch(u)} \{ d(v) + e(u,v) \}$ directed graph

Example:



$\delta = 5$

Step 1:

Leaf nodes, $d(u) = 0$

$d(5) = d(7) = d(8) = d(9) = d(10) = 0$

Calculating delay of node 4:

~~ch(4)~~

child of 4 i.e $ch(4) = \{7, 8\}$

$$d(4) = \max_{v \in 7,8} \{ d(7) + e(4,7), d(8) + e(4,8) \}$$

$$= \max\{0+1, 0+4\}$$

$$= 4 < \delta \quad \text{(no need to split)}$$

$\therefore d(4) = 4$

Similarly, calculating for node 2

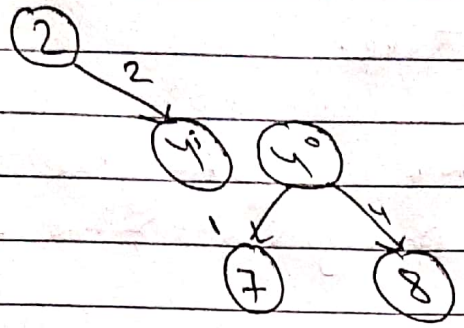
$ch(2) = \{4\}$

$$d(2) = \max\{d(4) + e(2,4)\}$$

$$= 4 + 2 = 6 > \delta$$

$\therefore$ Node (4) has to be splitted.

Scanned with CamScanner



Similarly, for node 6

$ch(6) = \{9, 10\}$

$d(6) = \max_{v \in 9,10} \{ d(9) + ce(6,9), d(10) + ce(6,10) \}$

$= \max(0+2, 0+3) = 3 < \delta$

$\therefore$ no need to split $\quad \therefore d(6) = 3$

Similarly for node 3

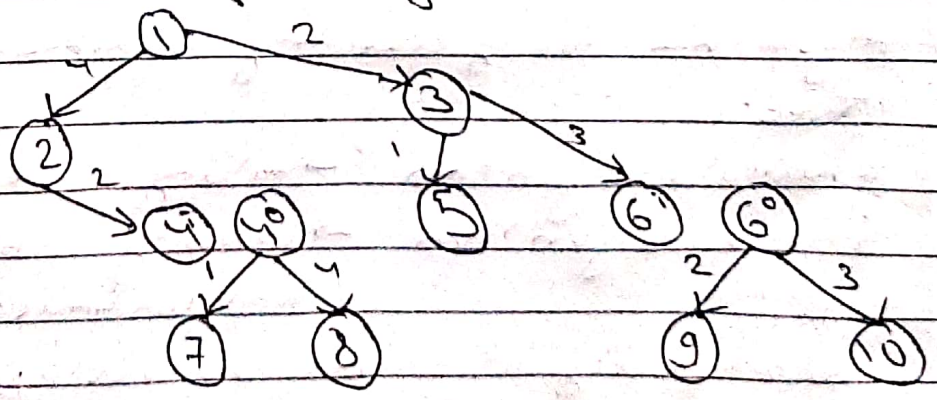
$ch(3) = \{5, 6\}$

$d(3) = \max_{v \in 5,6} \{ d(5) + ce(3,5), d(6) + ce(3,6) \}$

$= \max\{0+1, 3+3\} = 6 > \delta$

$\therefore$ Split vertex 6.

Final Tree after splitting vertex 4 & 6 is



classmate
Date
Page

11111 6) Minimum Cost Spanning Tree.

or

Describe an algorithm for computing the Minimum Spanning Tree (MST). Analysis its time complexity

i.e.

Prim's Algo
Kruskal's Algo

-> Spanning Tree
↳ a spanning tree of a graph G is a subgraph of G that contains all the vertices of G and is a tree i.e no cycle exists.

-> Minimum Spanning Tree
↳ There exists multiple spanning tree of a graph Among ~~these~~ these several spanning trees a spanning tree whose cost is minimum is called minimum spanning tree.
{Cost is defined as the sum of the weight of all the edges.}

-> Algorithm for generating minimum Spanning Tree.

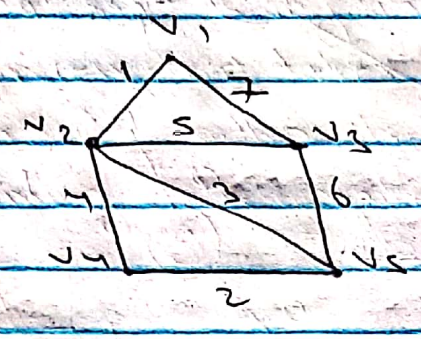
1. Kruskal's algorithm -> taking vertex with min pair
2. Prim's algorithm -> choosing arbitary vertex and continue.

-> Time complexity of Kruskal is $O(E \log V)$
-> Time complexity of Prim's is $O(V^2)$
V-vertex E-edges

Scanned with CamScanner

Abhiyan Dev Rimal
181649

2. Find MST for following graph using prism's & Kraskal Algorithm



(a) Using prism's algorithm

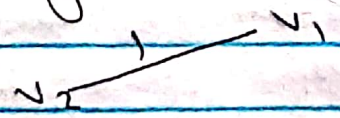
Let, v_1 be the arbitary vertex. Adjacence vertices of v_1 are v_2 and v_3.

Now, the set containing adjacence pair of vertex v_1 is:-

$(v_1, v_2) = 1$

$(v_1, v_3) = 7$

Here, $(v_1, v_2) = 1$ is the vertex point with least weight. So, add it to the tree being formed.



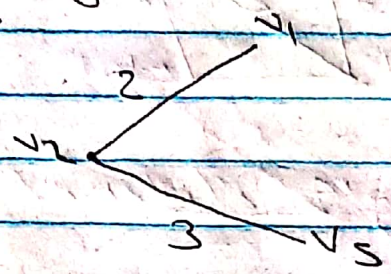
Now, the adjacence vertices of v_2 are v_1, v_3, v_4, v_5. The new set of vertex pair is

$v_2, v_3 = 5$
$v_2, v_4 = 4$
$v_2, v_5 = 3$
$v_2, v_3 = 7$

Scanned with CamScanner

Abhiyan Dev Pimal
181649

Here, $(V_2, V_5) = 3$ is the vertex pair with least weight. So, add it to the tree being formed.

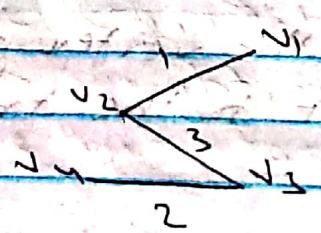


Now, the new arbitrary vertex is V_5. The adjacence vertices of V_5 are V_3, V_2, V_4

The new set of vertex pair is,

$V_5, V_3 = 6$
$V_5, V_4 = 2$
$V_2, V_3 = 5$
$V_2, V_4 = 4$
$V_1, V_3 = 7$

Here the vertex pair $(V_5, V_4) = 2$ has the least weight. So, add it to the tree being formed.



Now, V_4 is the new arbitrary vertex with its adjacence vertices V_1 and V_3

The new set of vertex pair is

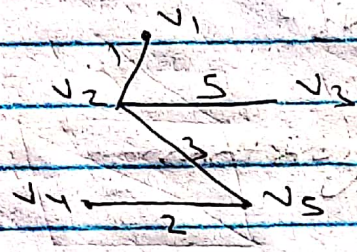
$(V_4, V_2) = 4$
$(V_5, V_3) = 6$
$(V_2, V_3) = 5$
$(V_1, V_3) = 7$

Here, the vertex pair $(V_4, V_2) = 4$ has the least weight. But adding it to the tree forms a loop. So, we discard (V_4, V_2).

Scanned with CamScanner

Abhiyan Dev Rimal
181649

Now, the next minimum weight forming vertex pair is $(v_2, v_3) = 5$.
So, add it to the tree being formed.

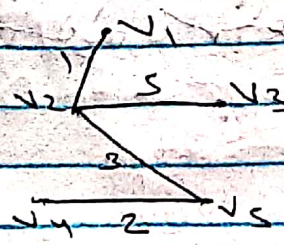


Now, the adjacence vertices of v_3 are v_1, v_2 & v_3. The new set of vertex pair is:

$v_3, v_1 = 7$
$v_3, v_5 = 6$

Here, the vertex pair $(v_3, v_5) = 6$ has the least weight. But adding it to the tree forms a loop. So, we discard the vertex pair (v_3, v_5).
Similarly, the remaining vertex pair (v_3, v_1).

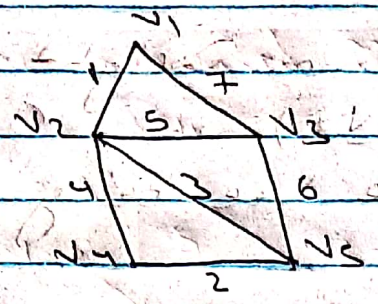
$\therefore$ The final minimum spanning tree is:



total cost $= 1 + 5 + 3 + 2 = 11$

Scanned with CamScanner

Abhiyan Dev Rimal
181649

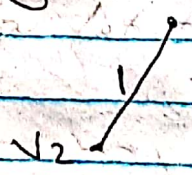


(b) Using Kruskal's Algorithm

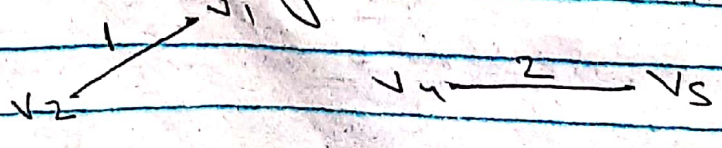
Let us take the vertex pair in ascending order of their weight.

$(v_1, v_2) = 1$
$(v_4, v_5) = 2$
$(v_2, v_5) = 3$
$(v_2, v_4) = 4$
$(v_2, v_3) = 5$
$(v_5, v_3) = 6$
$(v_2, v_1) = 7$

Step 1: Since $(v_1, v_2) = 1$ is the pair with least minimum weight in the set, so add it to the tree being formed.



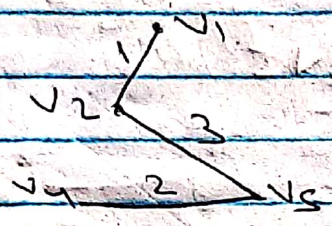
Step 2: Now, $(v_4, v_5) = 2$ is the next pair forming minimum weight. So we add it to the tree.



Scanned with CamScanner

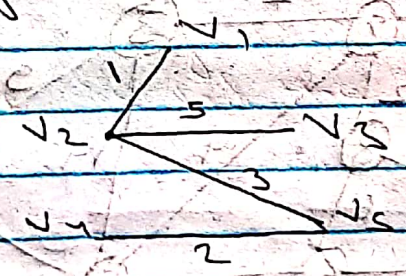
Abhiyan Dev Rimal
181649

Step 3: Now $(v_3, v_5) = 5$ is the pair with next minimum weight so add it to the tree



Step 4: Now $(v_2, v_4) = 4$ is the pair with next minimum weight. But adding it to the tree form a loop so discard pair (v_2, v_4).

Step 5: $(v_2, v_3) = 5$ is the new pair with next minimum weight, so add it to the tree being formed



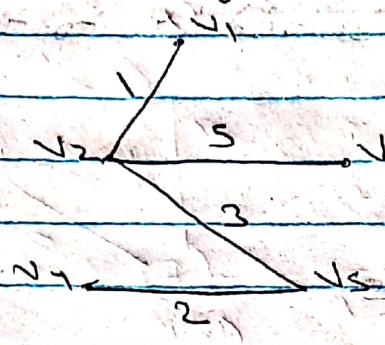
Step 6: The vertex pair $(v_3, v_5) = 6$ is the next minimum weight. But adding it to the tree being formed forms a loop. So, we discard the pair $(v_3, v_5) = 6$

Step 7: The vertex pair $(v_1, v_3) = 7$ is the remaining vertex pair in the set. But adding it to the tree also forms a loop. Hence, we discard the pair (v_1, v_3).

Scanned with CamScanner

Abhiyan Dev Rimal
181649

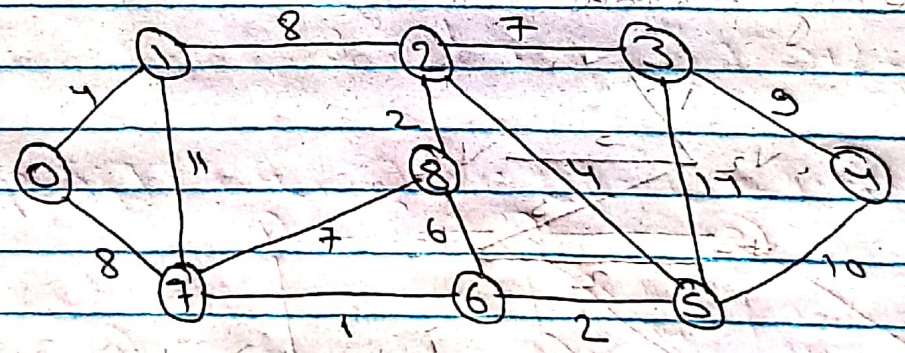
∴ The required MST is:



It's total cost = 1+5+3+2 = 11

only upto here

2s. Find the shortest path in the following graph using Dijakasta Algorithm.



Let us level the ~~the~~ starting node with 0 & all other nodes with infinity.

Let u be the starting node 0 & v be the nearest adjacent of u such that its length from u is d(v).

classmate
Date
Page

11 7) Explain Dijkstra's algorithm for computing the single source shortest path in a graph with suitable example.

Shortest Path

→ The sequence of vertices and edges crossed while moving from source vertex to destination vertex is called path

→ Among the several existing paths, the path with the least cost is called shortest path.

→ Shortest path Algorithm

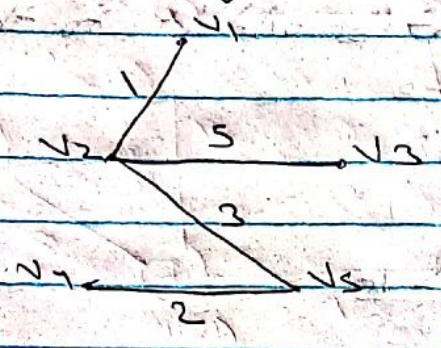
→ Dijkstra's Algorithm

→ Time complexity of Dijkstra's algorithm is $O(V^2)$.

Scanned with CamScanner

Abhiyan Dev Rimal
181649

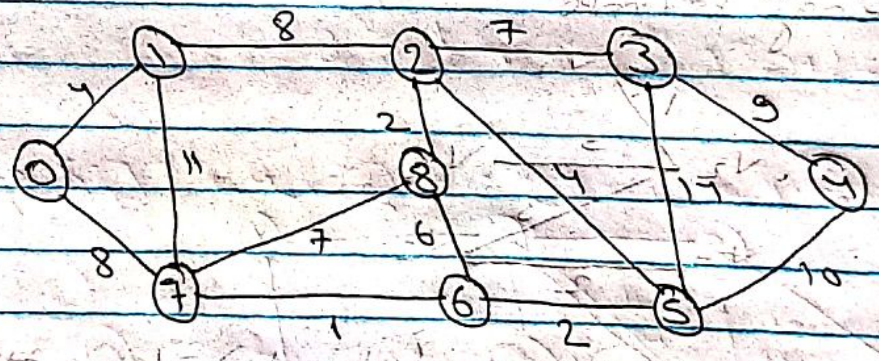
∴ The required MST is:



It's total cost = 1+5+3+2 = 11

continue from here

2s. find the shortest path in the following graph using Dijkastra Algorithm.



Let us level the ~~the~~ starting node with 0 & all other nodes with infinity.

Let u be the starting node 0 & v be the nearest adjacent of u such that its length from u is d(v).

Scanned with CamScanner

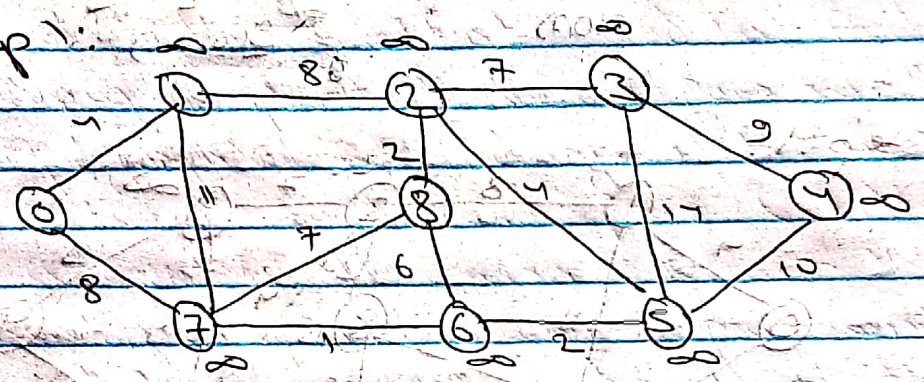
Abhiyan Dev Rimal
181649

For dijkstra Algorithm, we know:

if $(d(u) + c(u,v) < d(v))$
$d[v] = d(u) + c(u,v)$

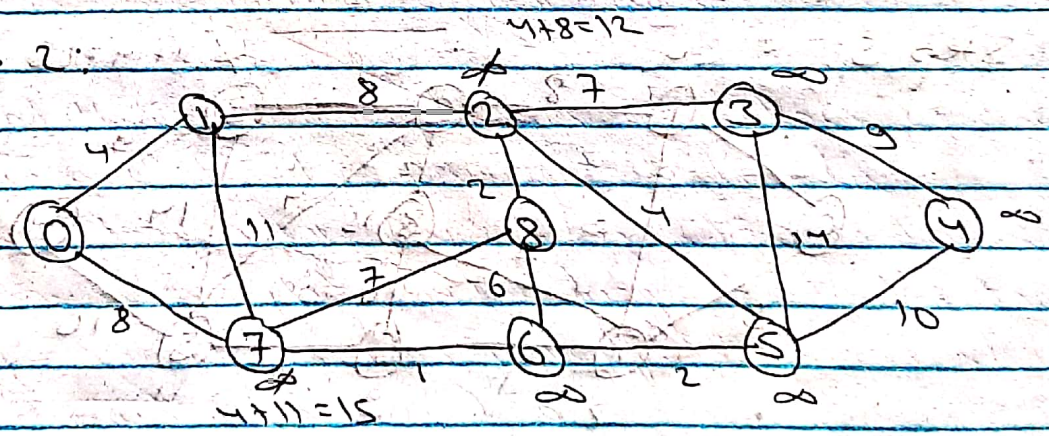
Now,

Step 1:



Here, $(0,1) = 4 < (0,7)$ vertex pair.
So, let us select the vertex pair $(0,1) = 4$

Step 2:



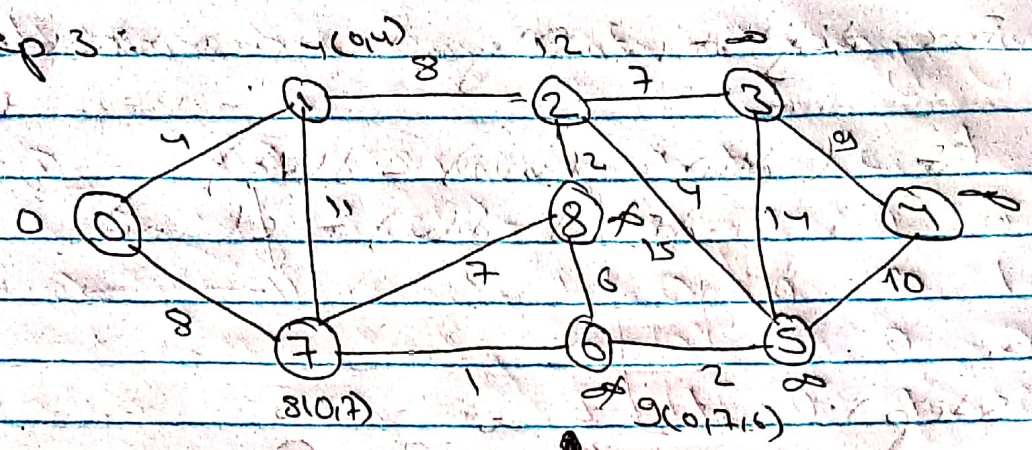
Here, $(0,1,2) = 12 < \infty$. So its new distance $d[v] = 12$. Also, $(0,1,7) = 4+11 > 8$. So the distance $d[v]$ remains same in this case.

Here, $(0,7) = 8$ has the least $d[v]$. So we select the new vertex pair $(0,7)$.

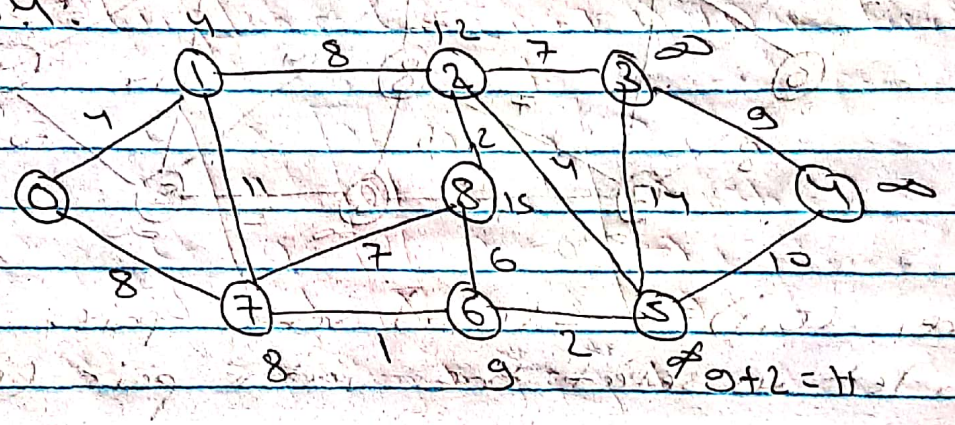
Scanned with CamScanner

Abhiyan Dev Rimal
181649

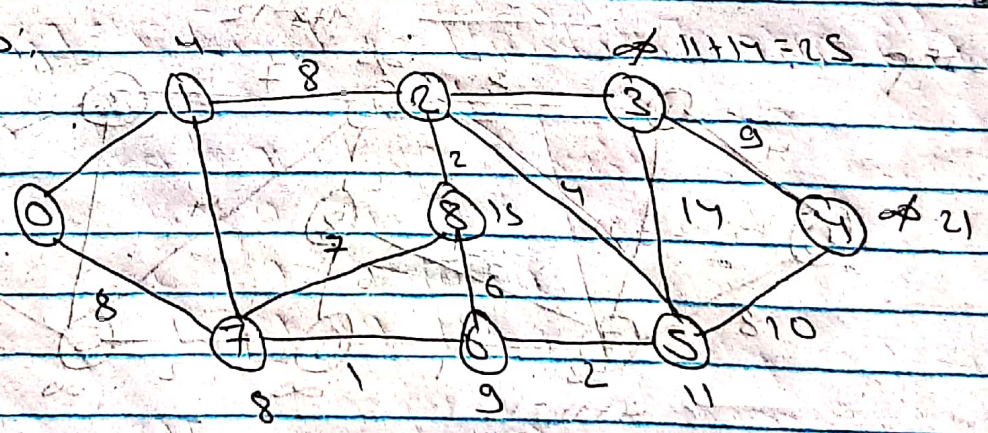
Step 3:



Step 4:



Step 5:

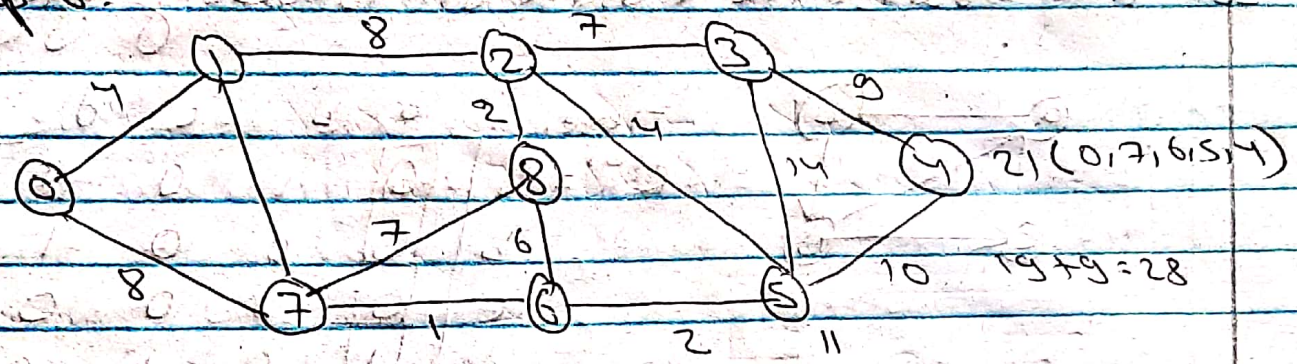


The process is continued until all the path for destination node are calculated. The final distance obtained after tracing the algorithm is the shortest path.

Scanned with CamScanner

Abhiyan Dev Rimal
181679

Step 6:



① 8 has min distance, so select 8 vertex.
① 3 has min distance so select 3 vertex now.

Now, moving from vertex 3 to 4 produces minimum distance of 28 which is greater than the distance formed from vertex 5 to 4.

So, we have the min distance length from 0 to 4 is 21.

21(0,7,6,5,4)

Scanned with CamScanner