SNS COLLEGE OF ENGINEERING

Coimbatore-107

COURSE NAME: ANALYSIS OF ALGORITHM

II YEAR/ IV SEMESTER

UNIT – III

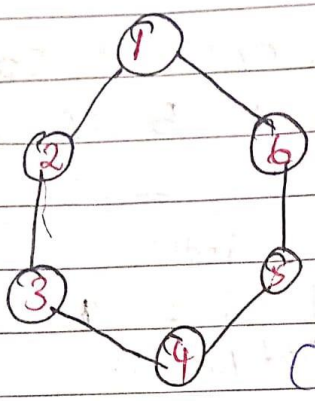
GREEDY TECHNIQUE

Topic

Greedy Technique: Prim's algorithm

Unit -3

Greedy Technique : Prims Algorithm



(1) Graph G = {V, E}

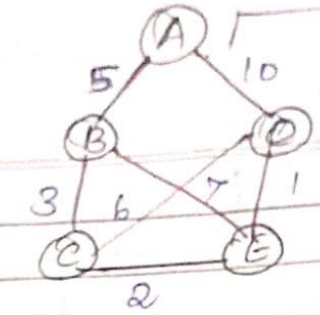
(2) V = set of vertices
= {1, 2, 3, 4, 5, 6}

(3) E = set of edges
{(1,2), (2,3), (3,4), (5,6), (6,1)}

Spanning Tree:

It is a subgraph / subset of graph.

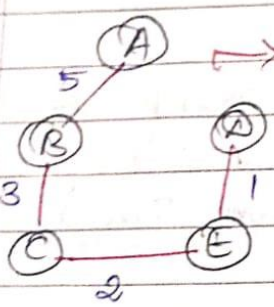
Example:



→ Graph G

2 Spanning Tree: → T_1

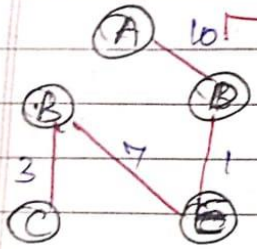
→ T_2



⟼ Spanning Tree T_1.

Weight (T_1) = 5 + 3 + 2 + 1

= 11



⟼ Spanning Tree T_2

Weight (T_2) = 10 + 1 + 7 + 3

= 21

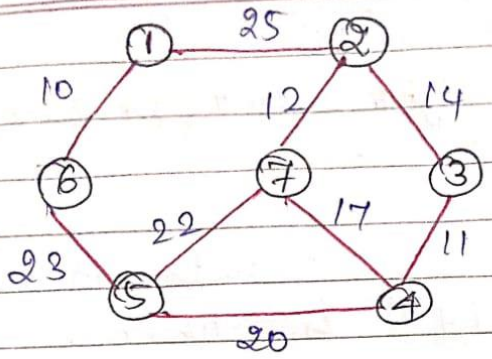
Minimum Spanning Tree = T_1 with Weight

Trying all possible spanning Tree to check MST is lengthy way, So we go for 2 Greedy methods → Prim's Algorithm

→ Kruskal's Algorithm

Example for prims Alg:

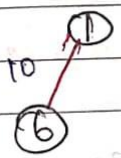
Graph G =



Step 1:

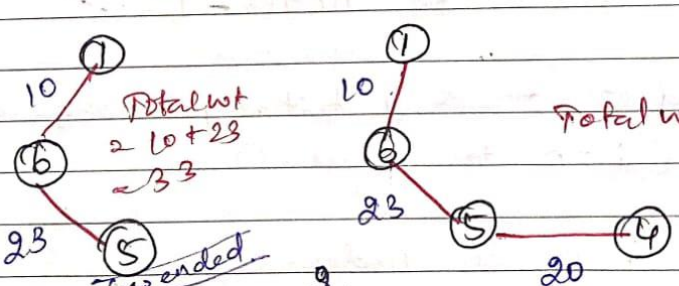
Consider the vertices with minimum weighted edge.

Here 1 —10— 6 is explored



Step 2:

From 1 & 6, 2 paths ~~explored~~ detected to vertices 2 and 5. But chosen 5 since it include minimum weight. (23) → Explore it.



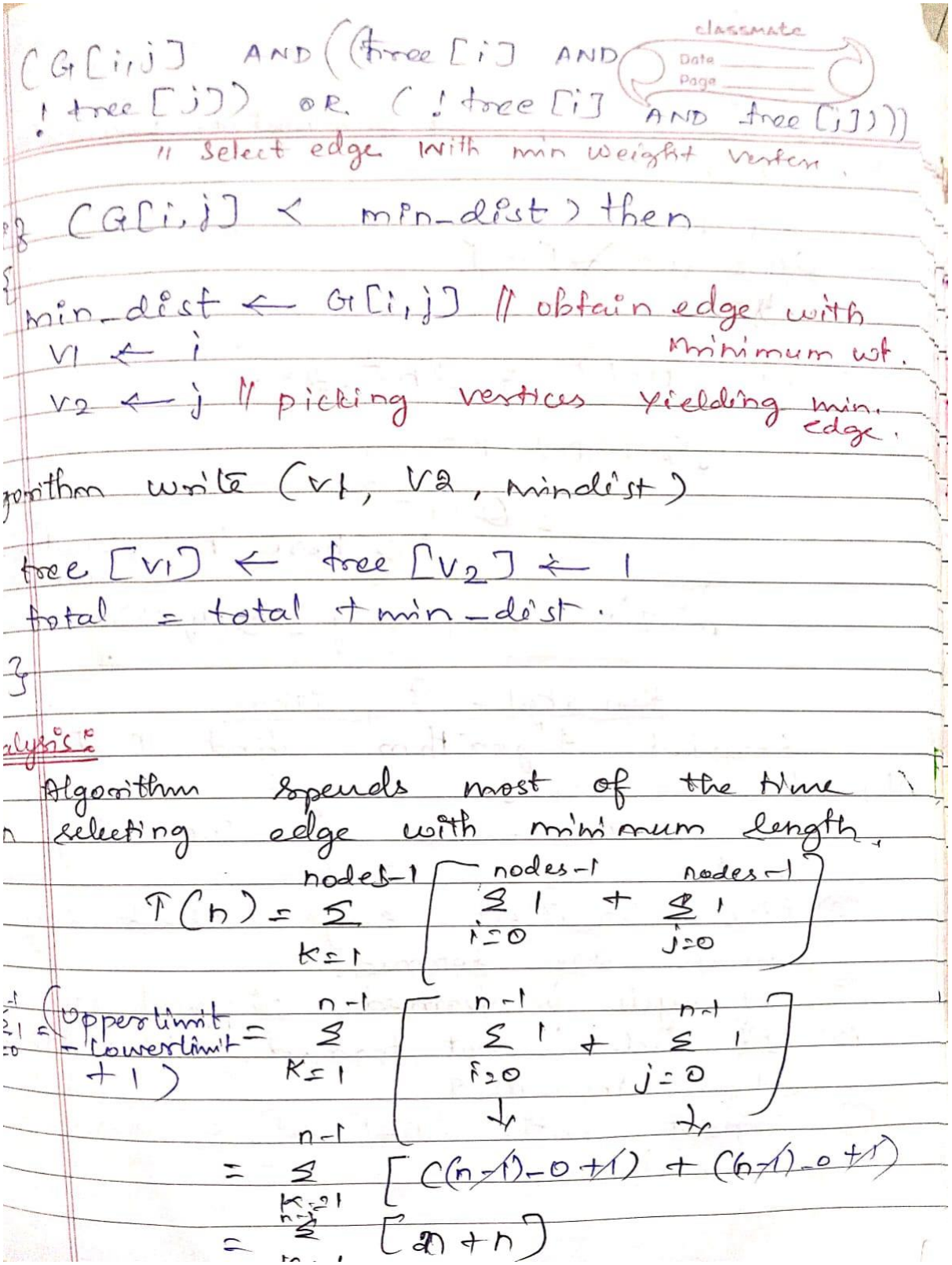
Step 3: From ~~1~~, 5 ~~and~~ 2 paths detected to vertices (4), ~~7~~ and (2). But chosen '4' with min wt = 20

Step 4:
From vertices 4 to 7, 4, 3 paths detected. But chosen '3' with minimum weight = '11'
Total weight = 10 + 23 + 20 + 11
= 64

Step 5:
From vertex '3', 1 path to vertex 2 with weight 14 is chosen.
Total weight = 10+23+20+11+14
= 78

Step 6:
From vertex 2, 2 path detected to 7 & 1. But explored 7 with min wt = 12
Total weight = 10+23+20+11+14+12
= 90

Algorithm:
```
Algorithm prim(G[0..size-1, 0..size-1], nodes)
for (i=0 to nodes-1)
{ tree[i] = 0
  tree[0]=1   // Take initial vertex.
  for (k=1 to nodes-1)
  {
    mindist ← ∝   // Initially assign as '∝'
    for (i=0 to nodes-1)
    {
      for (j=0 to nodes-1)
      {
```



[G[i,j] AND ((tree[i] AND !tree[j]) OR (!tree[i] AND tree[j]))

// Select edge with min weight vertex.

if (G[i,j] < min_dist) then

{

min_dist ← G[i,j] // obtain edge with minimum wt.

v1 ← i

v2 ← j // picking vertices yielding min. edge.

algorithm write (v1, v2, mindist)

tree[v1] ← tree[v2] ← 1

total = total + min_dist.

}

Analysis:

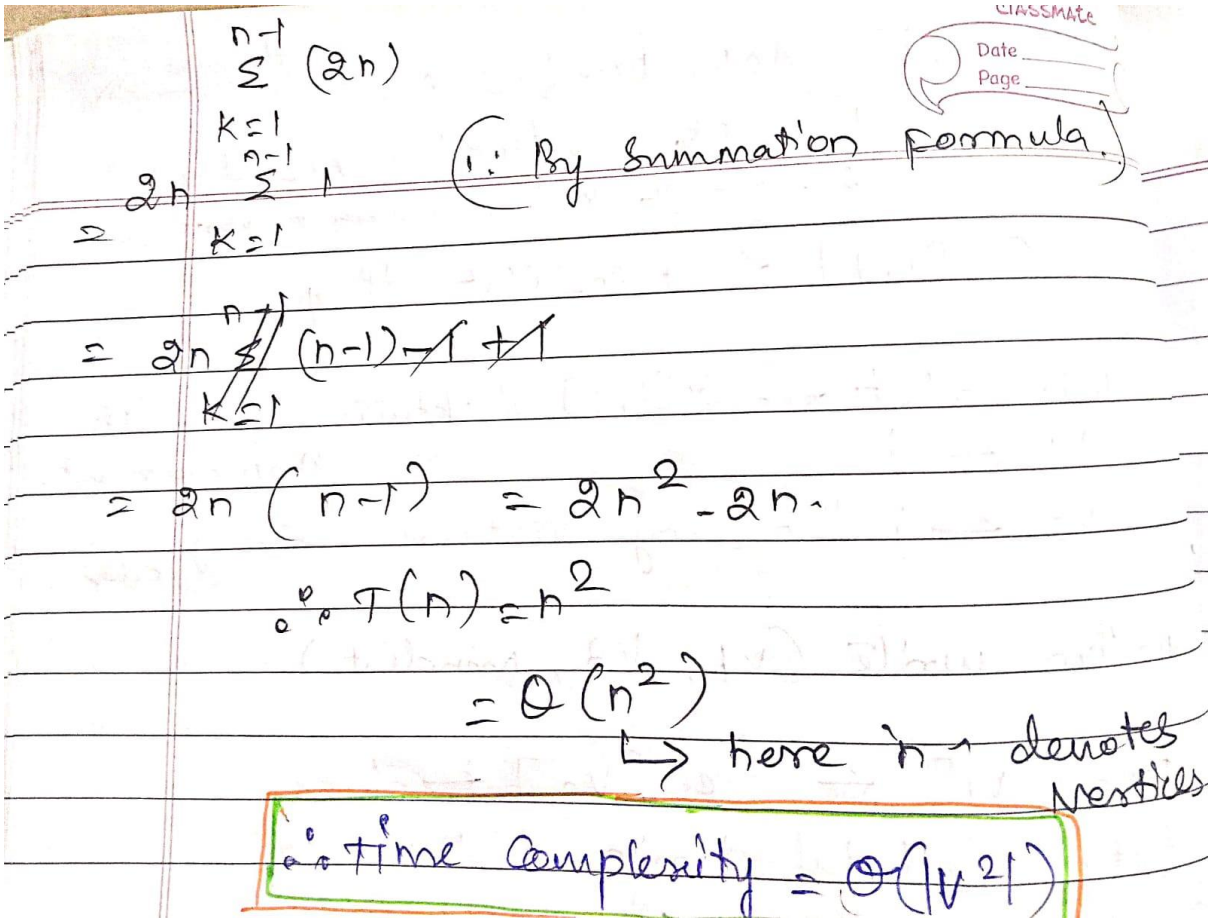
Algorithm spends most of the time selecting edge with minimum length.

$$T(n) = \sum_{k=1}^{nodes-1} \left[\sum_{i=0}^{nodes-1} 1 + \sum_{j=0}^{nodes-1} 1 \right]$$

$$\left(\begin{array}{l} \text{upper limit} \\ - \text{lower limit} \\ + 1 \end{array} \right) = \sum_{k=1}^{n-1} \left[\sum_{i=0}^{n-1} 1 + \sum_{j=0}^{n-1} 1 \right]$$

$$= \sum_{k=1}^{n-1} [(n-1)-0+1) + ((n-1)-0+1)]$$

$$= \sum_{k=1}^{n-1} [2n]$$


$\sum_{k=1}^{n-1} (2n)$
(∵ By Summation formula)
$2n \sum_{k=1}^{n-1} 1$
$= 2n\,(n-1) - 1 + 1$
$= 2n\,(n-1) = 2n^2 - 2n.$
$\therefore T(n) = n^2$
$= O(n^2)$
↳ here 'n' denotes vertices
∴ Time Complexity $= O(|V^2|)$