

CS & IT ENGINEERING

Algorithms

Greedy Method

Lecture No. - 04

By- Dr. Khaleel Khan
sir



Recap of Previous Lecture



Topic

Optimal Merge Pattern

Huffman Coding



Topics to be Covered



Topic

Minimum Cost Spanning Trees





Topic : Greedy Method

Min. Cost Spanning Tree (MCST)



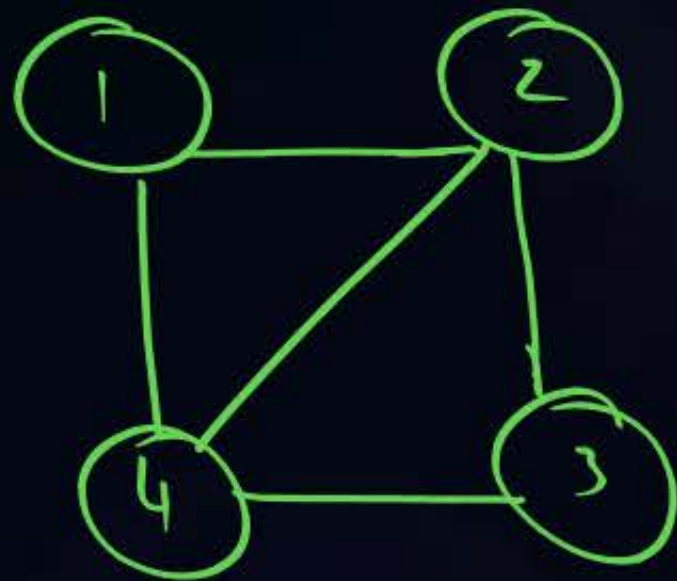
$$G = (V, E) \quad |V| = n; |E| = e$$

Representation of Graphs

Adjacency Matrix

Adjacency List

(i) Matrix $[n \times n]$



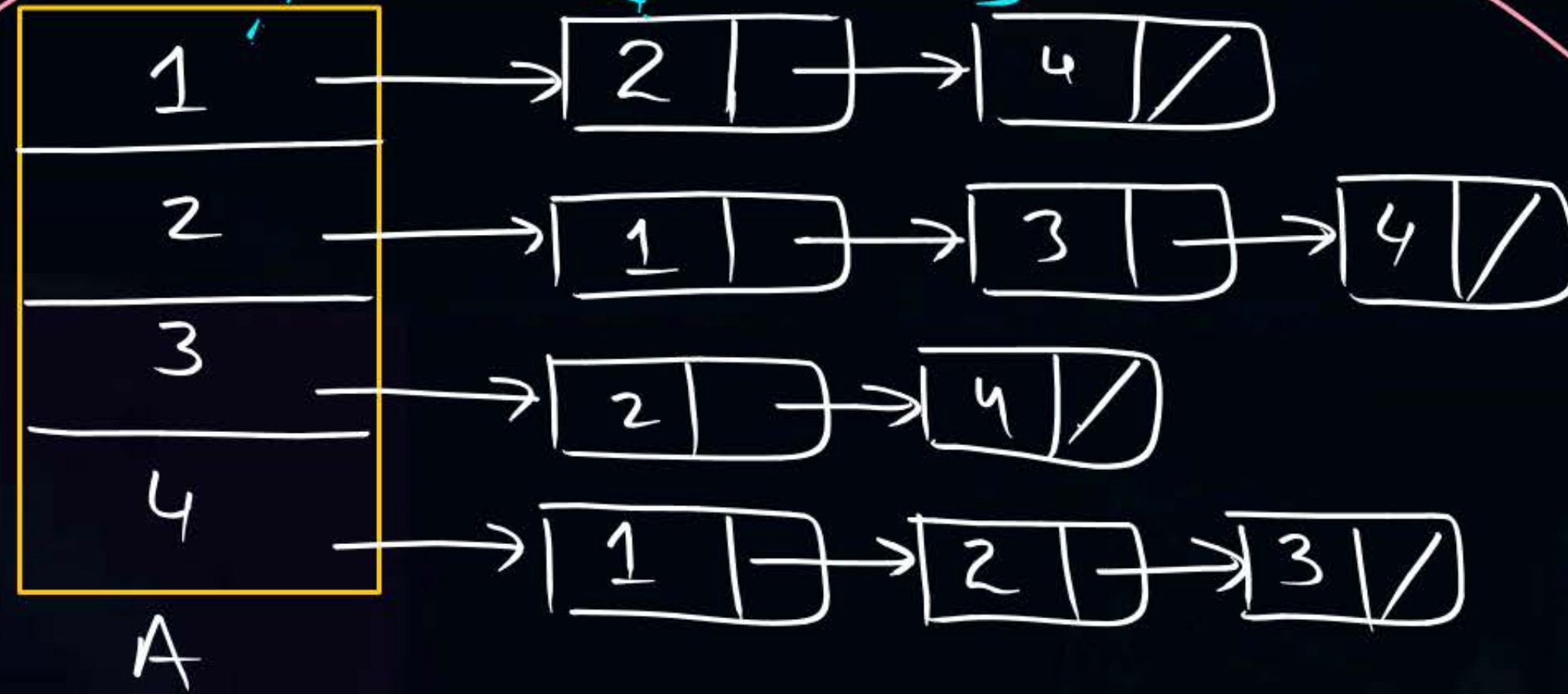
A	1	2	3	4
1	0	1	0	1
2	1	0	1	1
3	0	1	0	1
4	1	1	1	0

$$A[i, j] = 1, \quad \langle i, j \rangle \in E$$
$$= 0, \quad \langle i, j \rangle \notin E$$

$(n \times n)$ $O(n^2)$

(ii) Adjacency list

Array of linked list

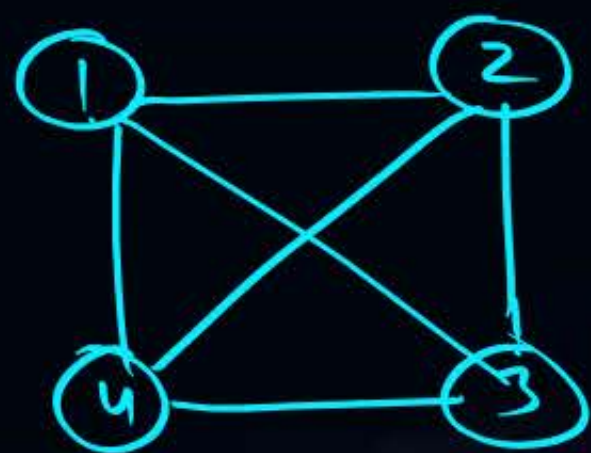


a) undirect Graph
($n + 2 \cdot e$)

b) Directed Graph
($n + e$)

$O(n + e)$

" $e \propto n$ " (in worst case)



Complete graph

undirected

$$e = \frac{(n-1) \cdot n}{2}$$

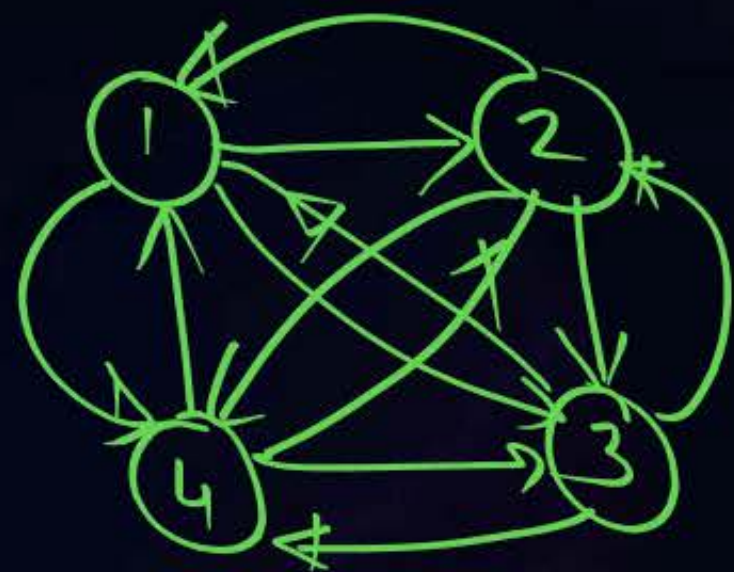
$$O(n^2)$$

$$e = O(n^2)$$

$$\log e \sim O(\log n)$$

$$e = n(n-1)$$

directed



Graphs

Dense Graphs

<almost Complete>

$$e \sim O(n^2)$$

(Matrix Repr. is better)

$$O(n+e) = O(\underline{\underline{n+n^2}})$$

Sparse Graphs

<Not Complete>

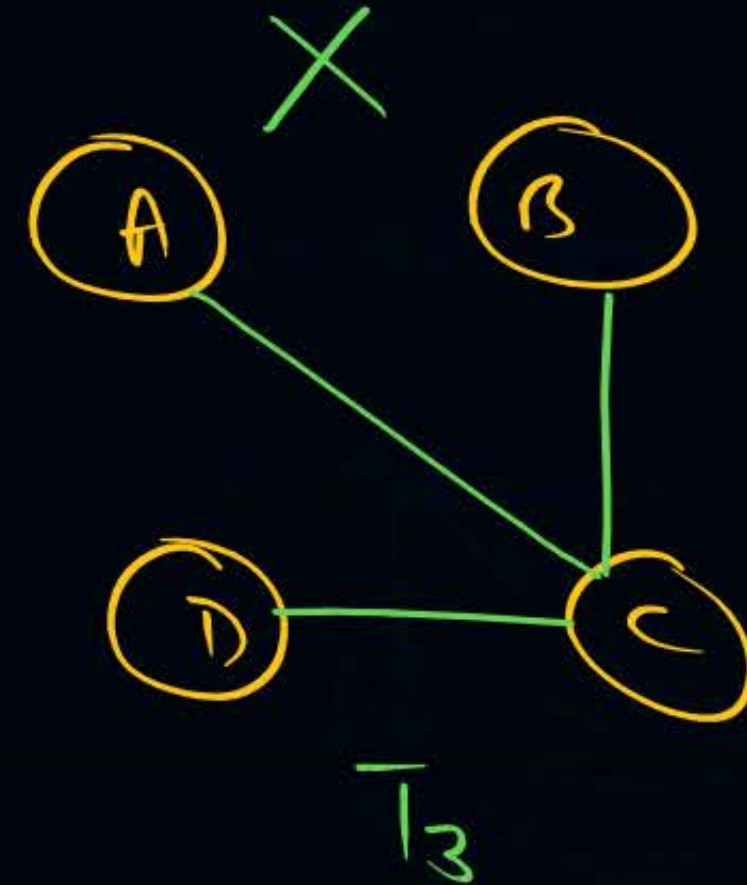
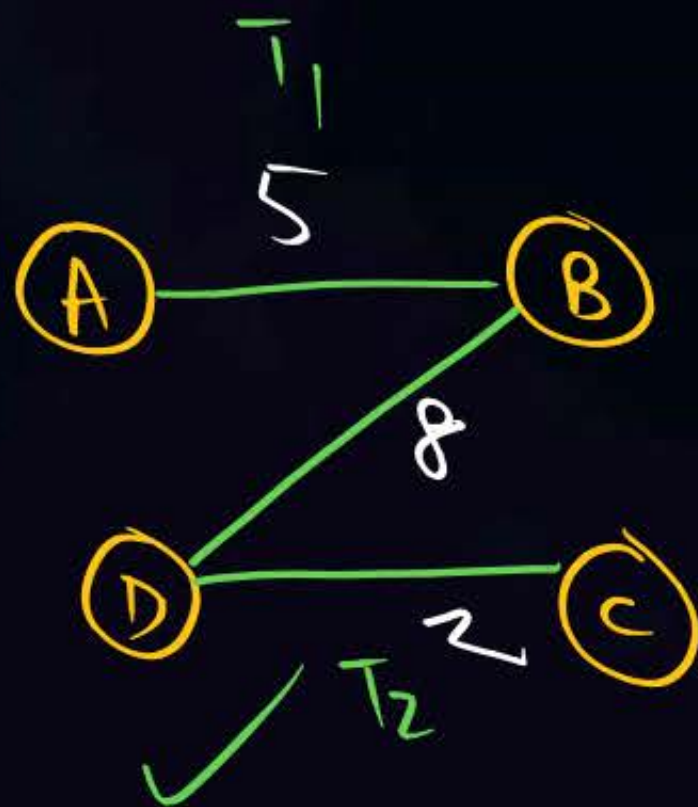
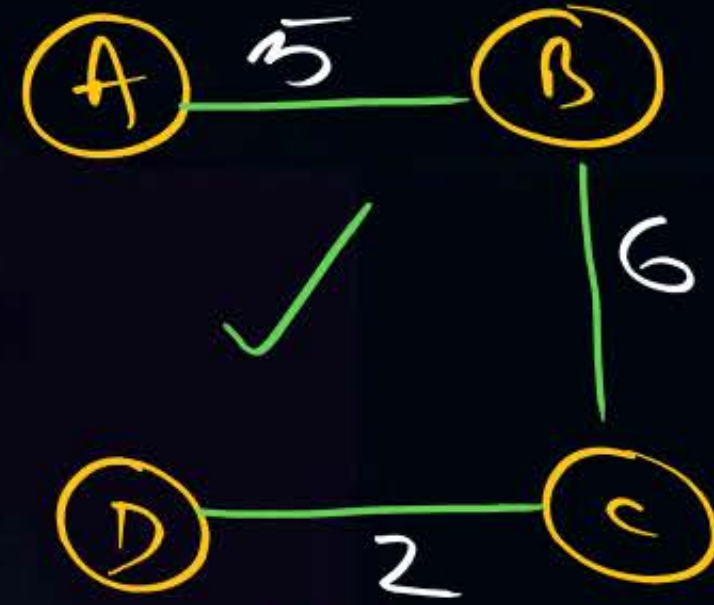
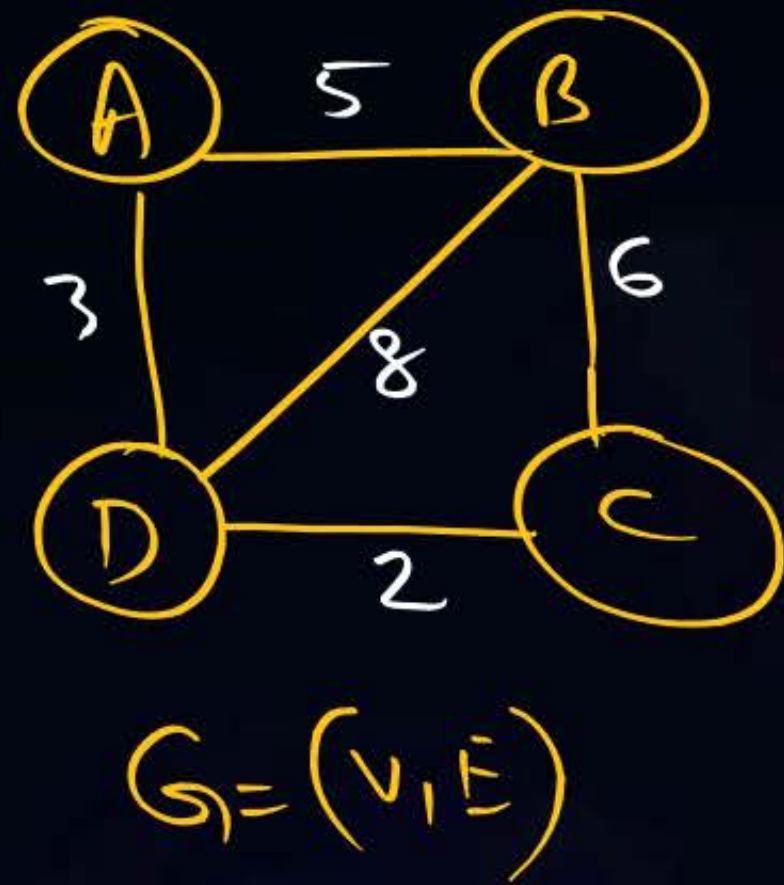
→ Adj. List
 $O(n+e)$

Def'n of Spanning Tree:

(Cost of Sp. Tree)

A Subgraph $\underline{T}(V, E')$ of given graph $G(V, E)$, where $E' \subseteq E$ is a Spanning Tree, iff T is a TREE

(Acyclic)



A Tree with n -vertices will have $(n-1)$ edges

Q1) Given a graph having n -vertices & e -edges, then the no. of edges that must be removed from the graph to get a Spanning Tree is _____,



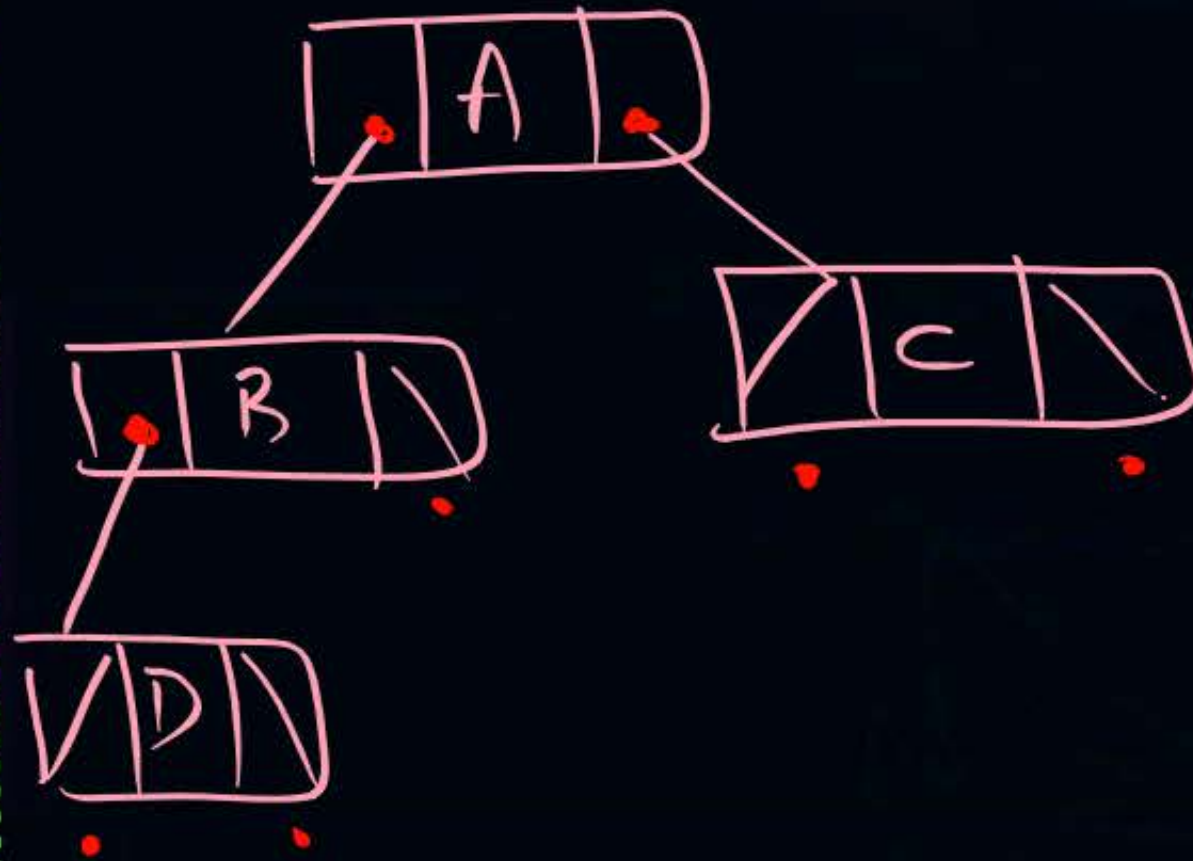
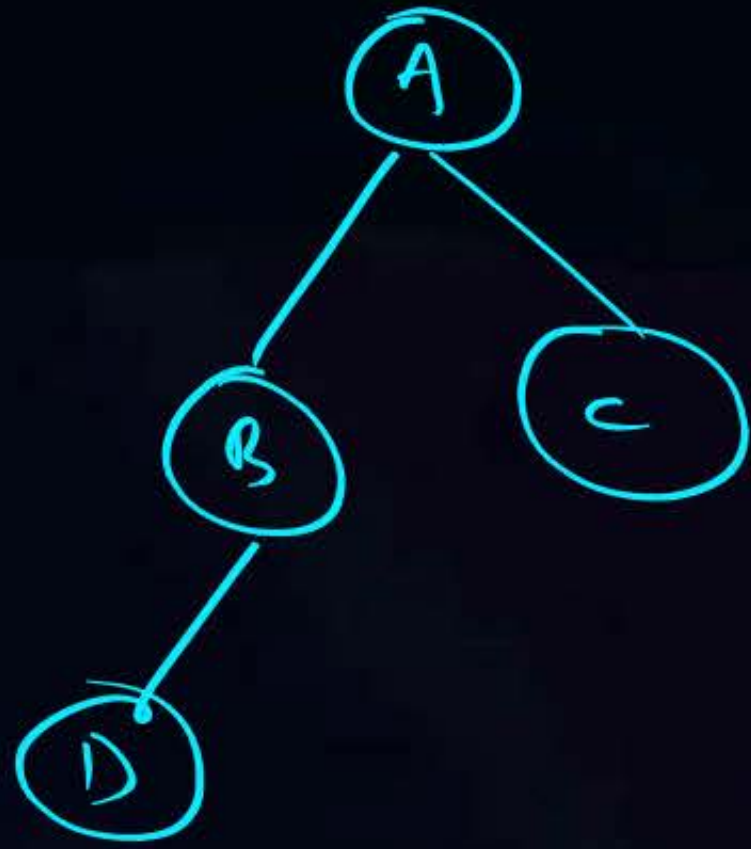
→ Total edges in graph: e

→ Spanning Tree with ' n ' vertices will have: $(n-1)$ edges <used>

Removed edges: $e - (n-1)$

: $e - n + 1$

→ A Tree with n -vertices will have $\underline{(n+1)}$ nil fields (Null-links) 



Total links : $2n$

Edges : $(n-1)$

Unused : $2n - (n-1)$

(NIL) fields : $(n+1)$



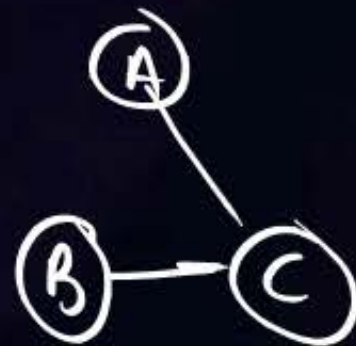
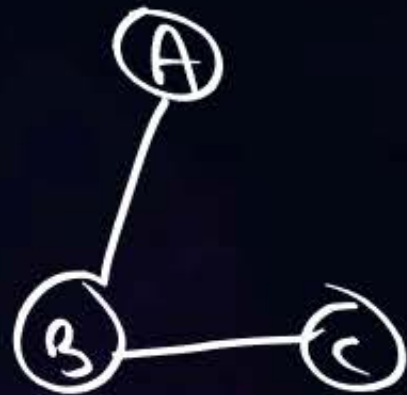
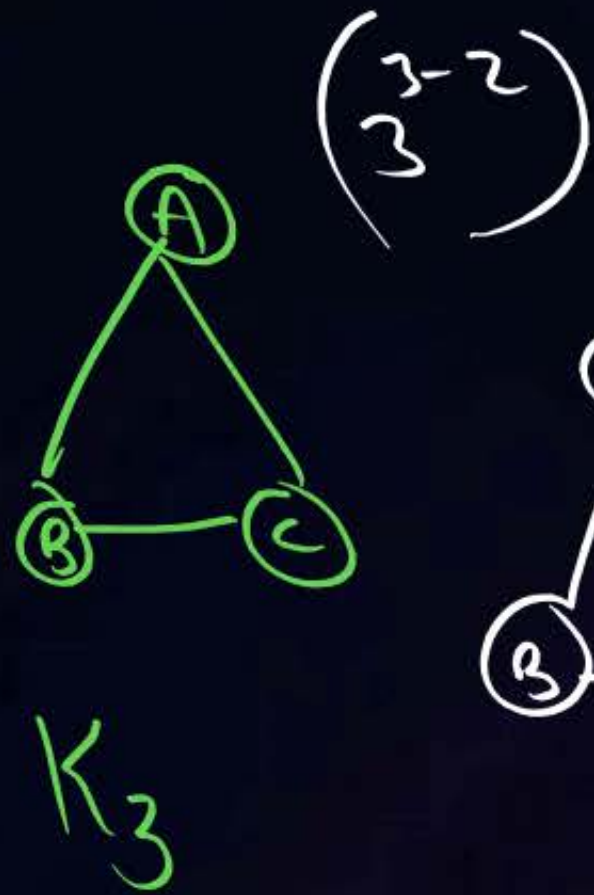
Sohu Space: For a given graph with n -vertices & e -edges

the max. # of Sp-Trees (Sohu Space)

$$= \frac{\binom{n-2}{n}}{\binom{n}{n^2}} \checkmark \left(\frac{n^n}{n^2} \right) \underline{\underline{O(n^n)}}$$

→ determining co-factor
of an element of the
given matrix (representing the
graph)

→ For Complete
graph
 K_n



Applications of Spanning Tree :



- (i) ^{S.T} Multicasting & Broadcasting in both wired & wireless Networks;
- (ii) ^{MCST} Construct / Implement Circuits (Electronic / Electrical) in communication networks;

Algo's for Construction of Min. Cost Sp. Trees

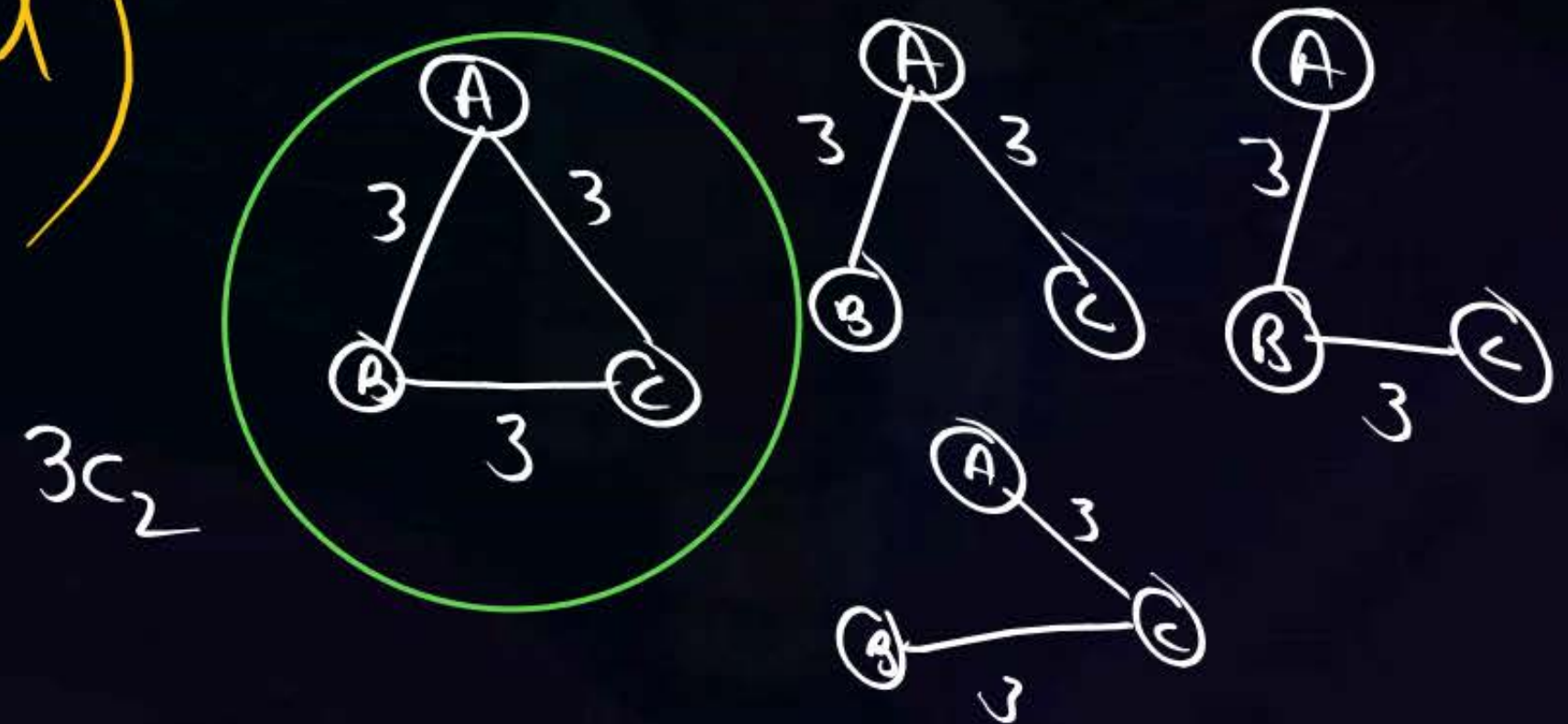


(i) Prim's - Jarník Algo

(ii) Kruskal's Algo

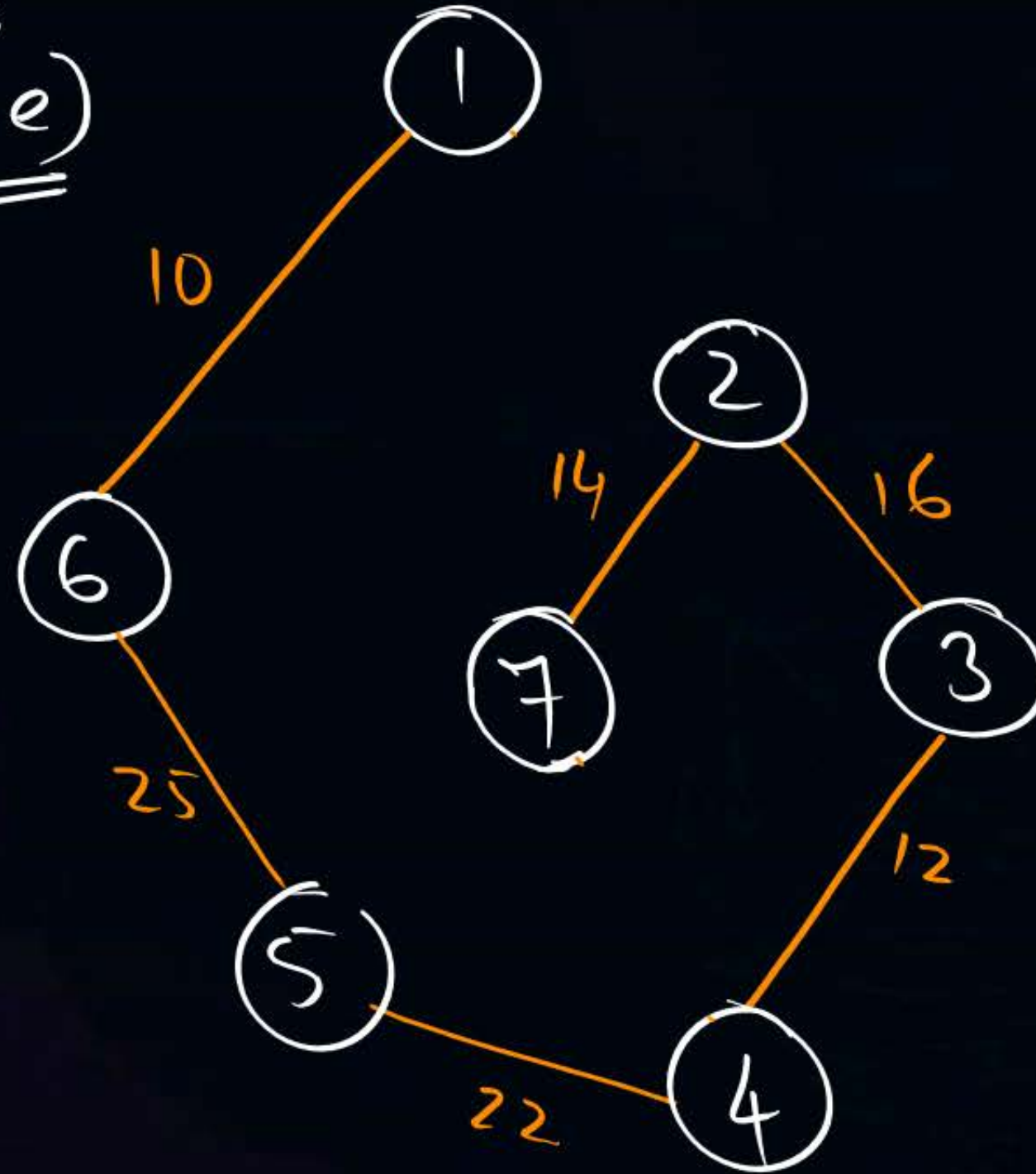
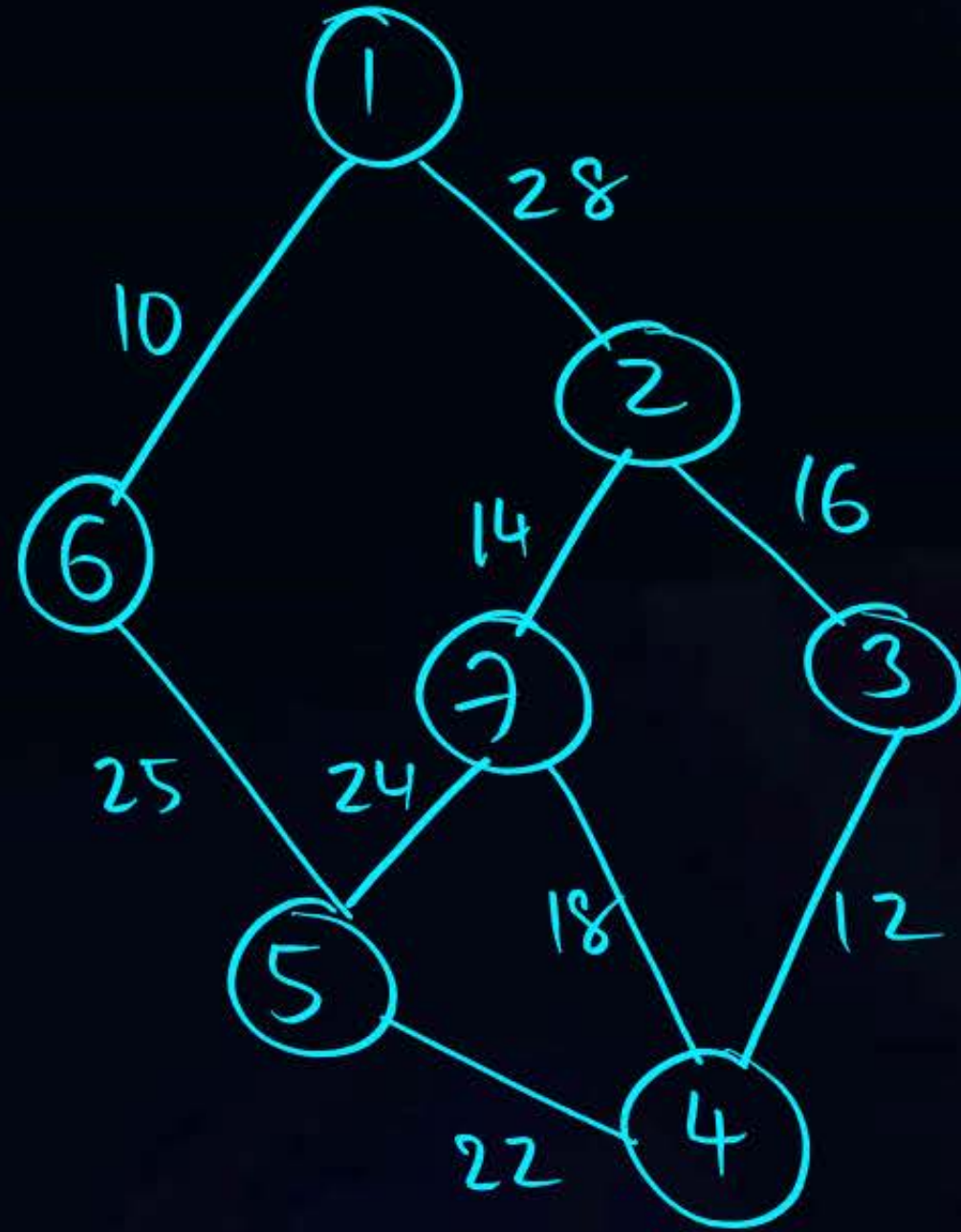
(iii) Dijkstra's Algo

- Prim's method always maintains Tree Structure Property at each Step;
- Cost of Spanning Tree with both approaches will always be Same;
- The Tree Structure may/may NOT be Same;
- The Tree will also be Same iff all edges have unique cost (distinct)



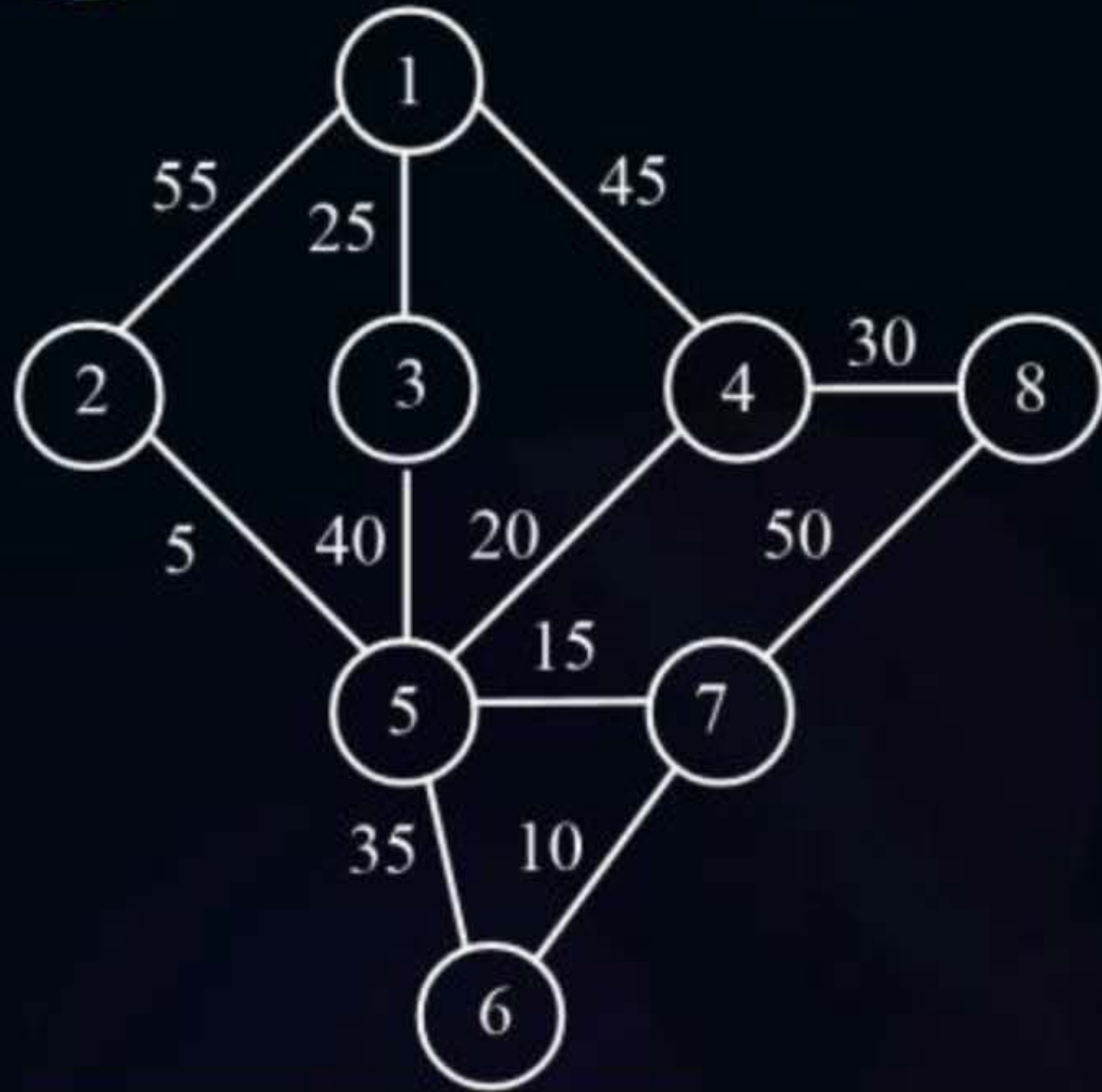
Dijkstra's Algo :

Time :
 $O(n \cdot e)$

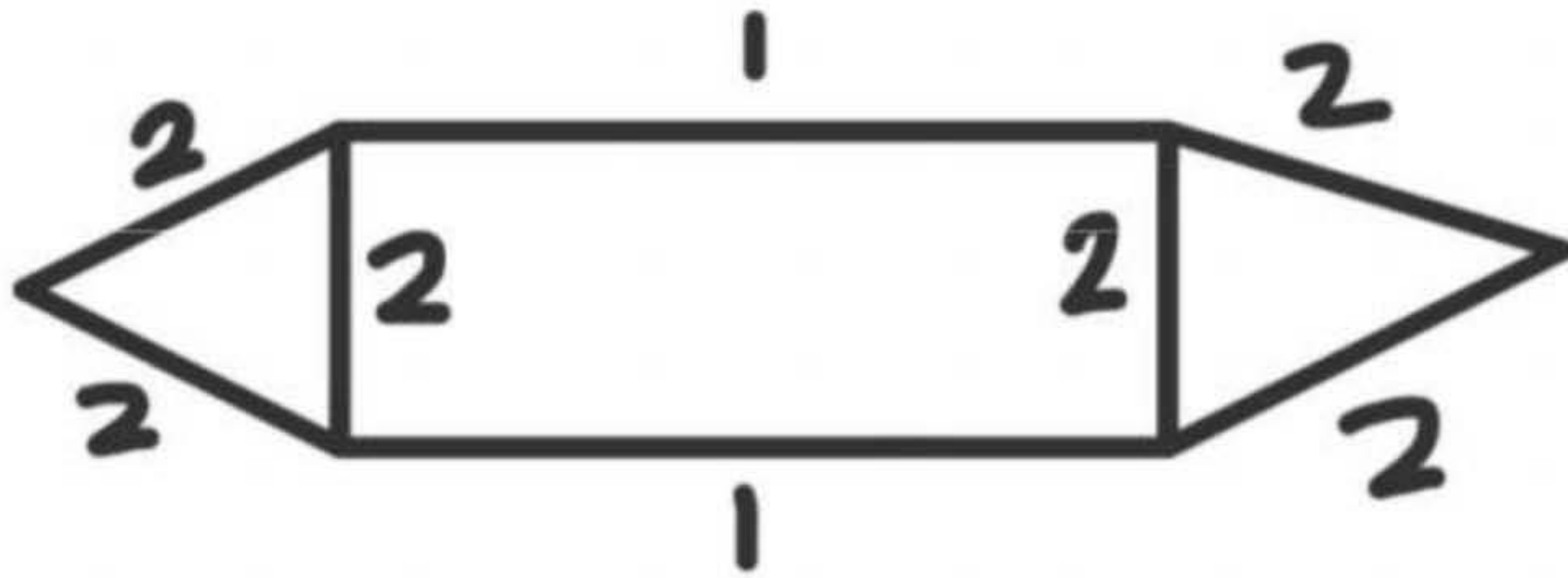




Topic : Greedy Method



No of minimal spanning trees:





Topic : Greedy Method



1. Algorithm **Prim**(E , cost , n , t)
2. { $t[1..n-1, 1..2]$ } $\rightarrow e$
3. 1) Let (k,l) be an edge of minimum cost in E;
3. 2) mincost := cost [k, l]; $- 1$
4. 3) (t[1,1] := k; t[1,2] := l;) $- 1$
5. 4) for i := 1 to n do // Initialize near .
6. 5) if (cost[i, l] < cost [i, k]) then near[i] := l; } n
7. else near[i] := k;
8. 6) near[k] := near[l] := 0; 1
9. 7) for i := 2 to n - 1 do $(n-2)$ edges
10. 8) { // Find $(n-2)$ additional edges for t.

n, e

$$T(n) = e + c + n + n^2 = O(n^2)$$

\uparrow
vertices

11. a) Let j be an index such that near[j] \neq 0 and
12. $(\text{cost}[j, \text{near}[j]])$ is minimum; } $\log n$
13. b) t[i,1] := j ; t [i, 2] := near [j];
14. c) mincost := mincost + cost [j, near [j]];
15. d) near[j] := 0;
16. e) for k:= 1 to n do // Update near[].
17. if ((near[k] \neq 0) and (cost[k, near[k]] > cost[k,
j]))
18. then near[k]:= j;
19. } *
20. return mincost;
21. }

Modify key
: $\log n$

If Heap is used in the Implementation of



Prim's Algo :

1) $O(e)$

2) $O(\log e)$

3) c

4) n

5) $n * \log n$

6) n

7) n

8) n

9) $n^2 * \log n$

10) $e * \log n$

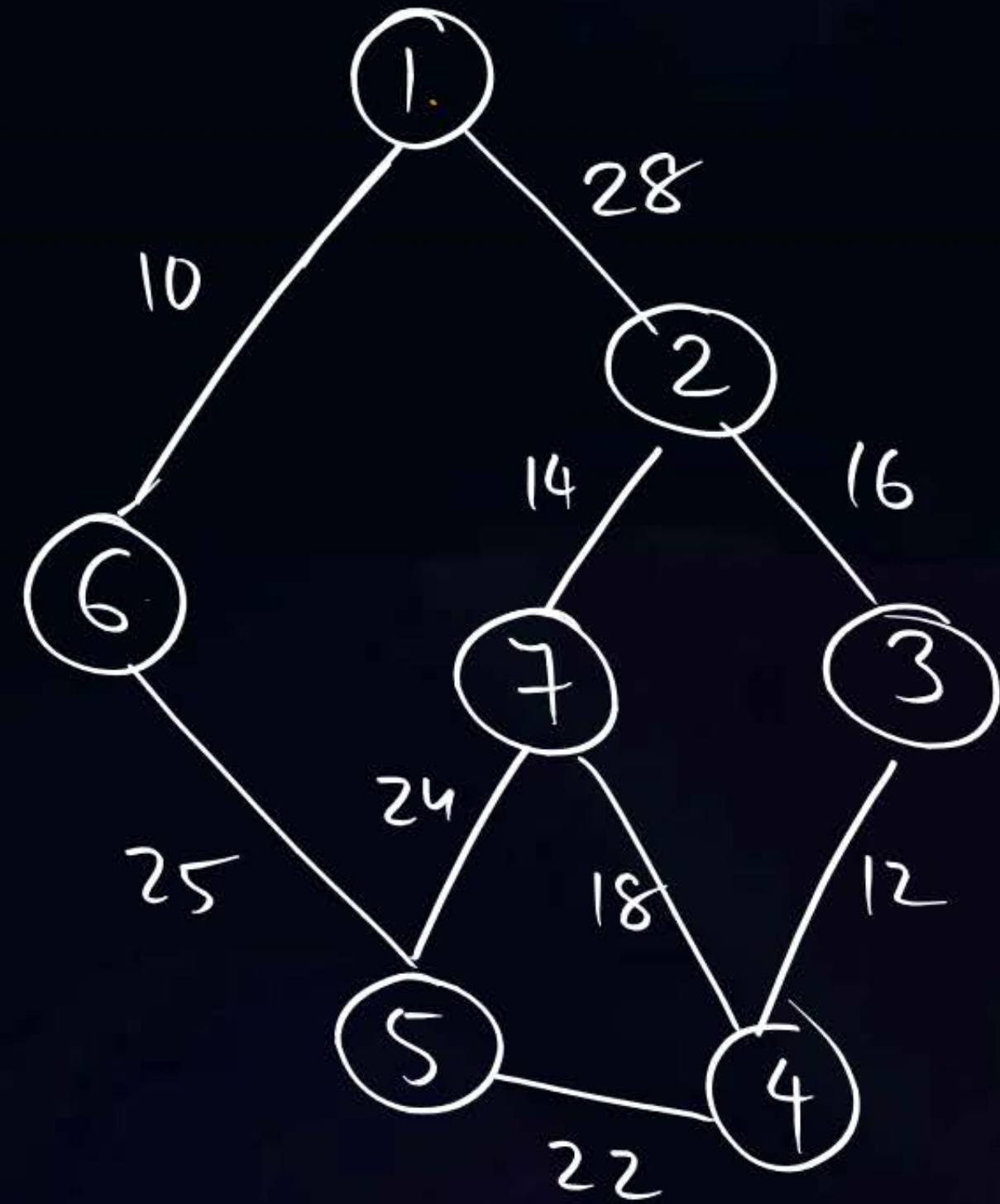
Heap creation
 $O(n)$

$$e = O(n^2)$$

Time : $e + \log e + n + n * \log n + e \log n$

$$: O((n+e) * \log n)$$

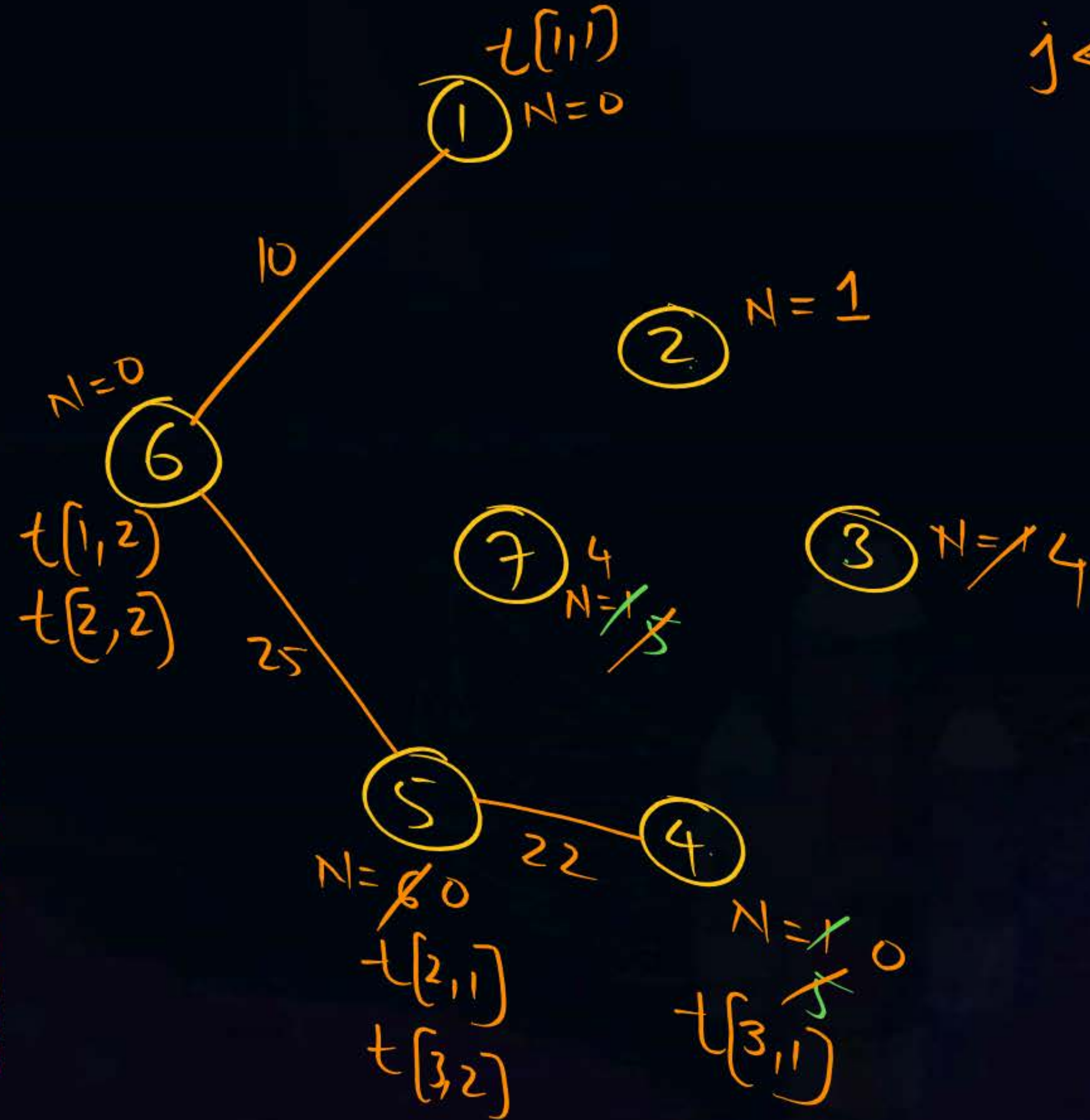




$$(k, e) \leftarrow (1, 6)$$

$$\text{minCost} \leftarrow 10 + 25 + 22$$

$$j \leftarrow 5$$
$$j \leftarrow 4$$




```

1   $t := \emptyset;$ 
2  while (( $t$  has less than  $n - 1$  edges) and ( $E \neq \emptyset$ )) do  $e$ 
3  {
4      a) Choose an edge  $(v, w)$  from  $E$  of lowest cost;
5      b) Delete  $(v, w)$  from  $E$ ;
6      c) if  $(v, w)$  does not create a cycle in  $t$  then add  $(v, w)$  to  $t$ ;
7          else discard  $(v, w)$ ;
8  }
```

$O(e * \log e)$

Algorithm 4.9 Early form of minimum-cost spanning tree algorithm due to Kruskal

(High-level Implementation)



Topic : Greedy Method



```
1.  Algorithm Kruskal (E, cost, n, t)
2.  {
3.  Construct a heap out of the edge costs using
    Heapify;
4.  for i := 1 to n do parent[i] := - 1;
5.  i := 0; mincost := 0.0;
6.  while ((i < n - 1) and (heap not empty)) do
7.  {
8.  Delete a minimum cost edge (u, v) from the heap
9.  and reheapify using Adjust;
10. j:= Find(u); k; = Find(v);.
11. if (j ≠ k) then
12. {
13.     i := i + 1;
14.     t[i, 1] := u; t[i, 2] := v;
15.     mincost := mincost + cost[u, v];
16.     Union (j, k);
17. }
18. }
19. if (i ≠ n - 1) then write ("No spanning
    tree");
20. else return mincost;
21. }
```

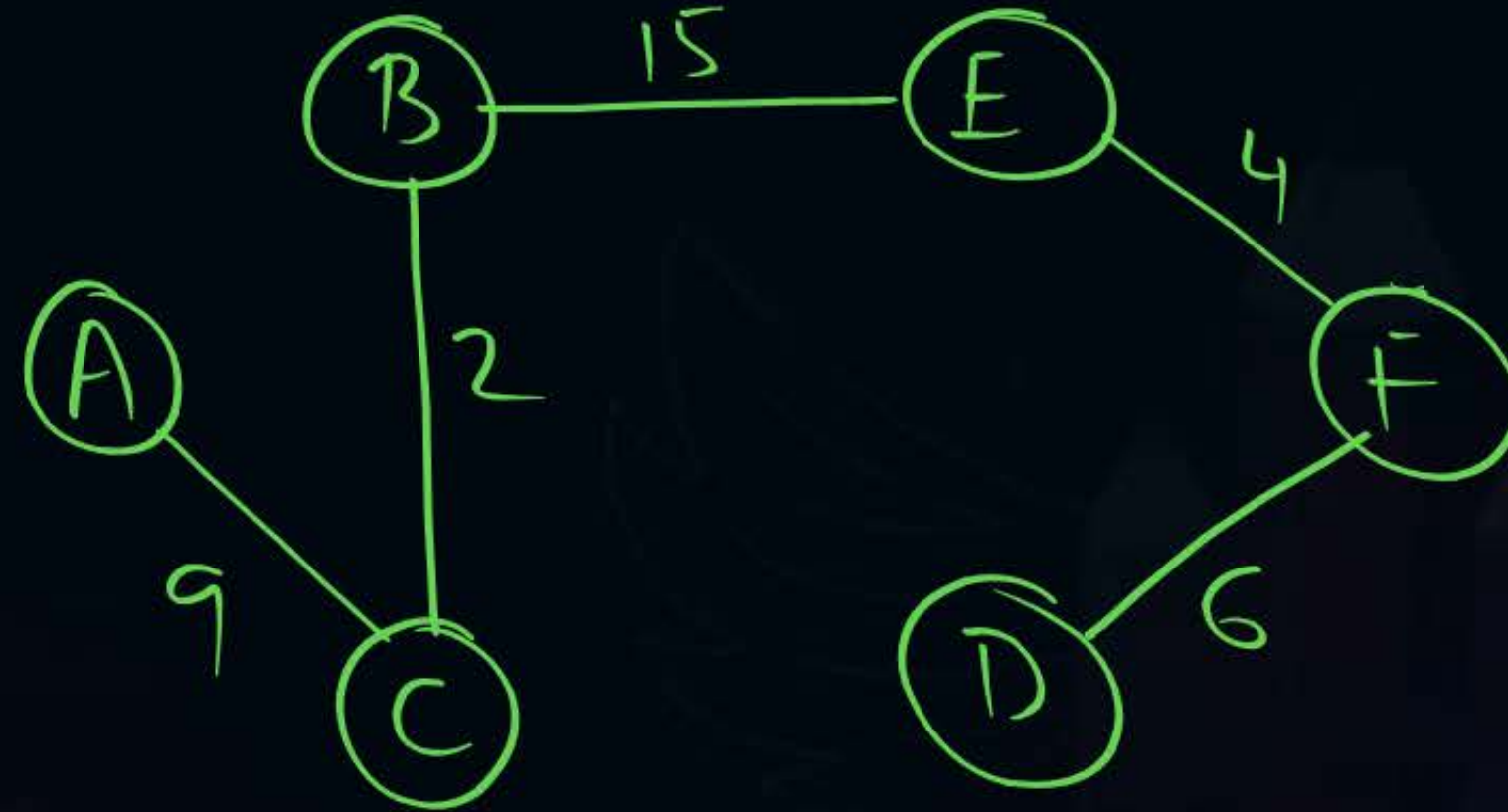



Topic : III. Greedy Method

PYQ
(NAT)



- Q. Consider the following Graph whose Minimum Cost Spanning Tree marked with edge values has a weight of 36. (Minimum) possible sum of all edges of the graph G is 69. (~~Assume that all edges have distinct cost~~). (66)



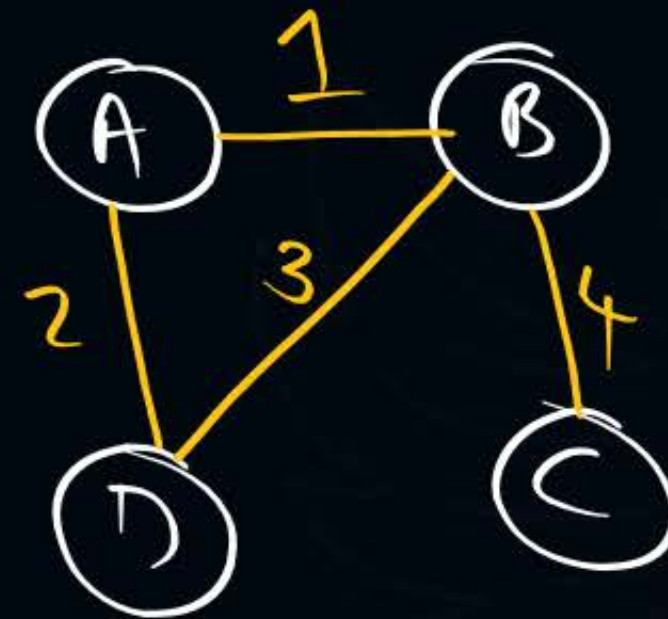
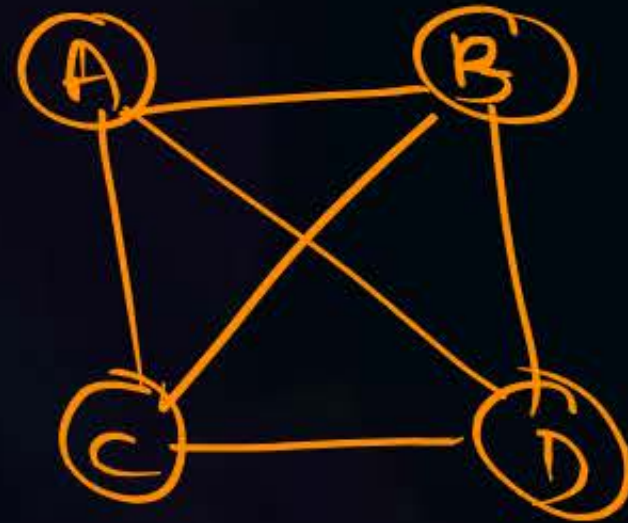


Topic : III. Greedy Method

PYQ/NAT

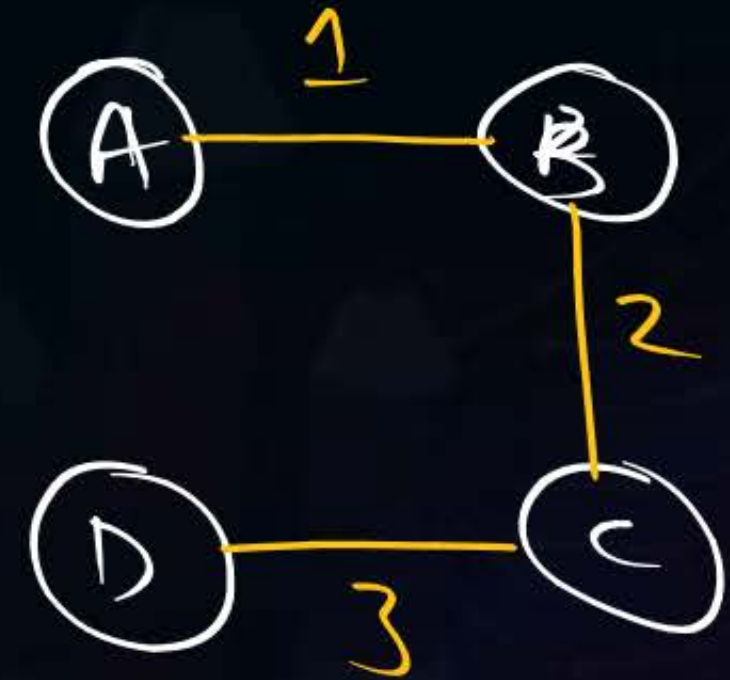


- Q. Let G be a complete undirected graph with 4 vertices and edge weights are $\{1, 2, 3, 4, 5, 6\}$. The maximum possible weight that a minimum weight Spanning Tree can have is 7.



$$(1 + 2 + 4) = 7$$

Normal Situation



$$= 6$$

THANK - YOU