CS & IT ENGINEERING

Algorithms

Miscellaneous Topics

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

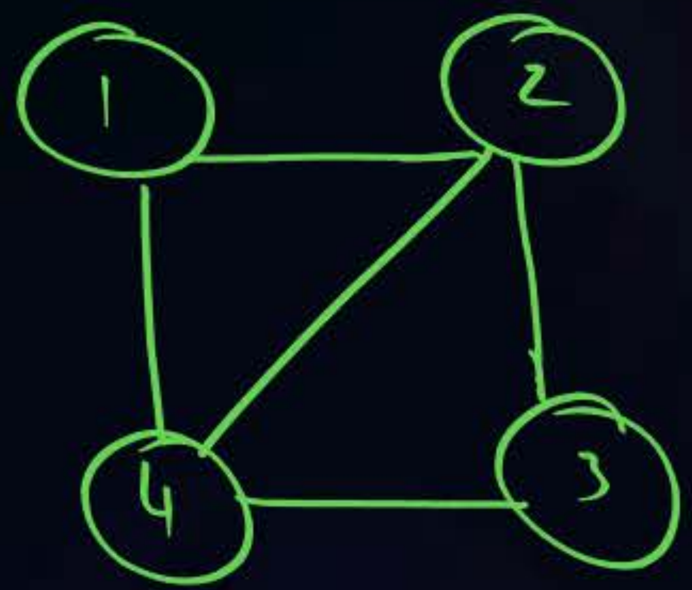
Min. Cost Spanning Tree (MCST)

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

Representation of Graphs

- Adjacency Matrix
- Adjacency list

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



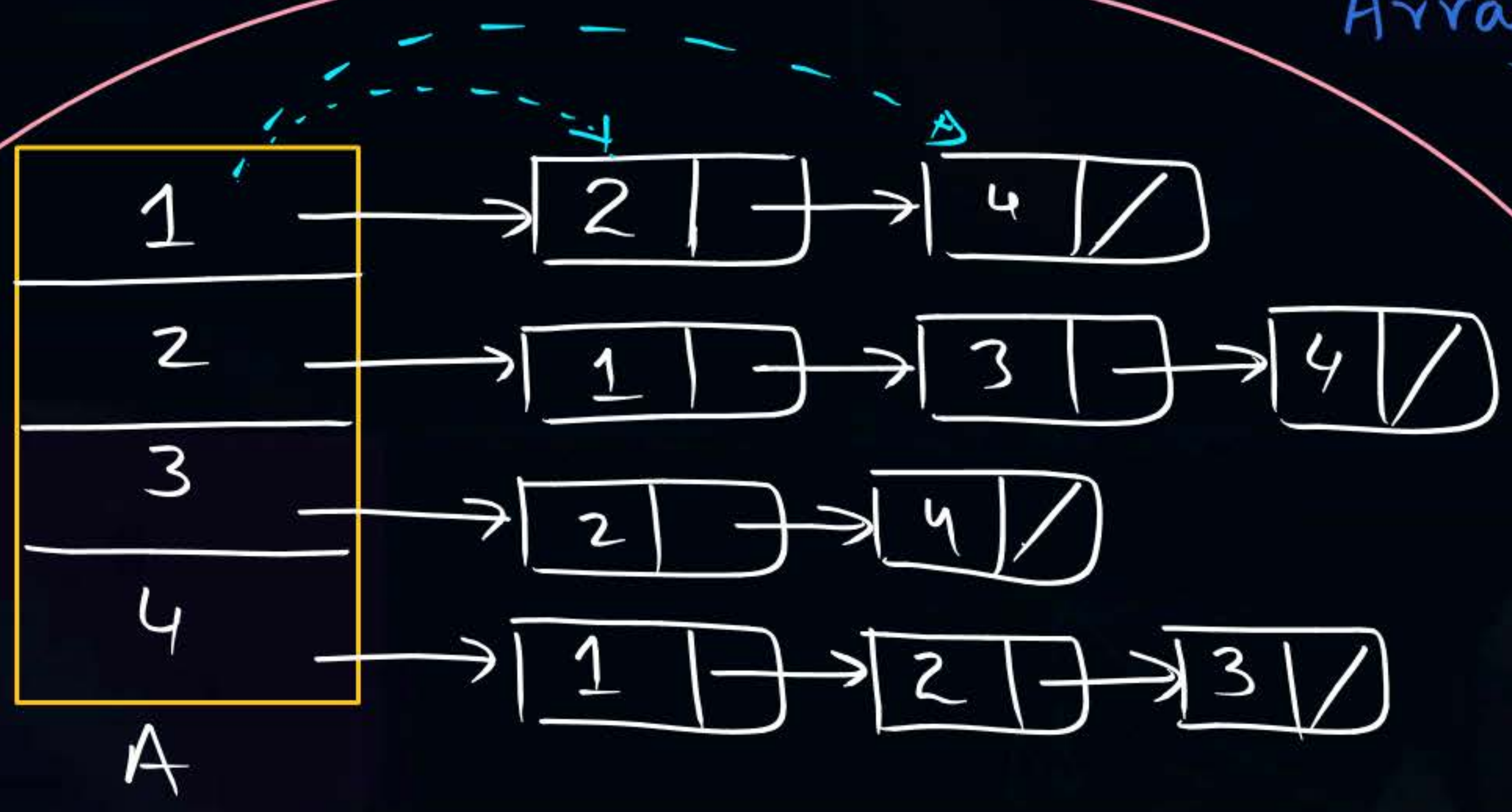
A	1	2	3	4
1	0	1	0	1
2	1	0	1	1
3	0	1	0	1
4	1	1	1	0

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

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

(ii) Adjacency list

Array of linked list



a) undirect Graph

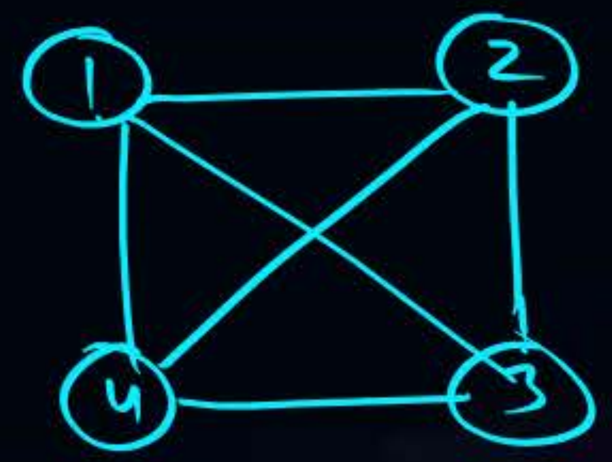
$(n + 2 \cdot e)$

b) Directed Graph

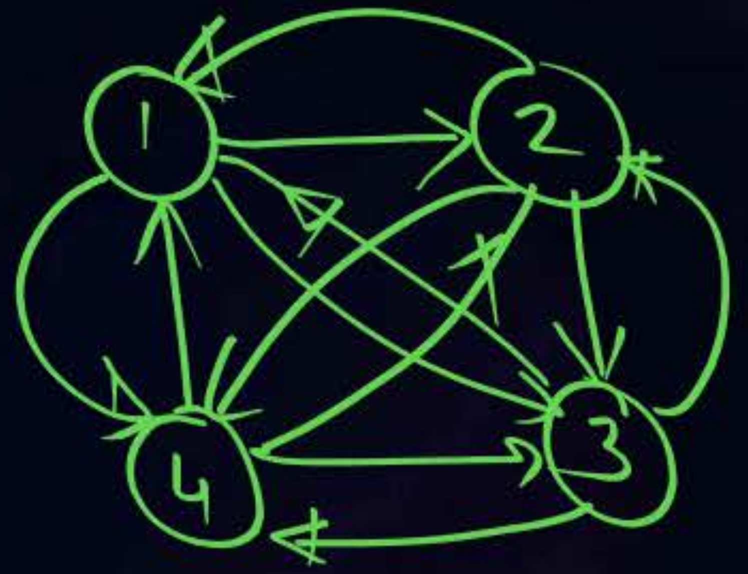
$(n+e)$

$O(n+e)$

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



Complete Graph



undirected

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

$$\boxed{e = n(n-1)}$$

directed

$$O(n^2)$$

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

$$\boxed{\log e \sim O(\log n)}$$

Graphs

Dense Graphs

<almost Complete>

$\boxed{e \sim O(n^2)}$

(Matrix Repr. is better)

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

Sparse Graphs

<Not Complete>

→ Adj. list

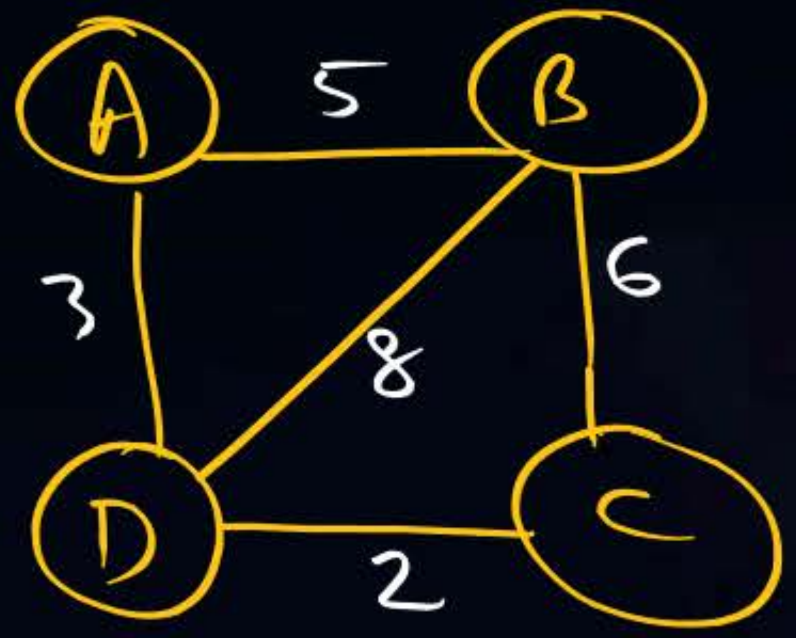
$O(n+e)$

Def'n of Spanning Tree:

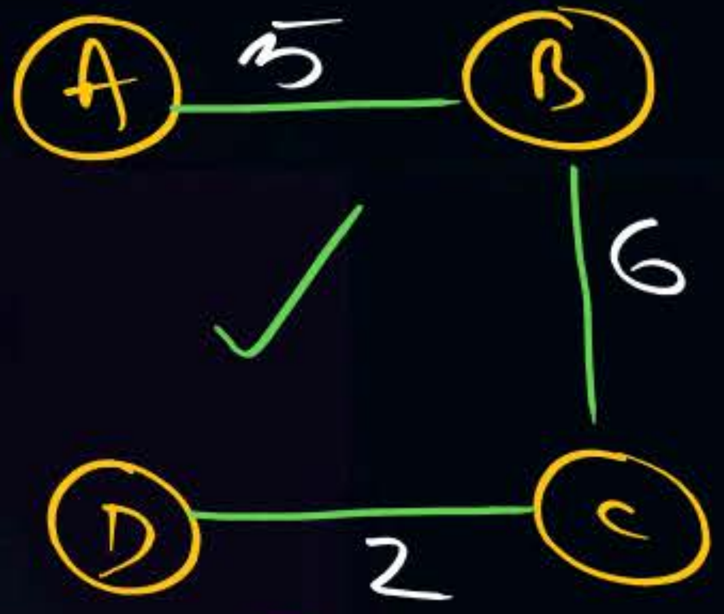
(Cost of Sp. Tree)

A Subgraph $T(V, E')$ of given Graph $G(V, E)$, where $E' \subset E$ is a Spanning Tree, iff T is a TREE

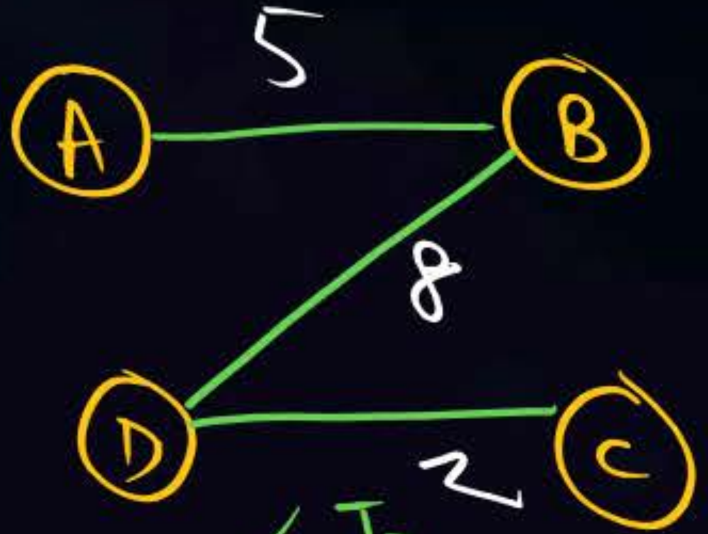
(Acyclic)



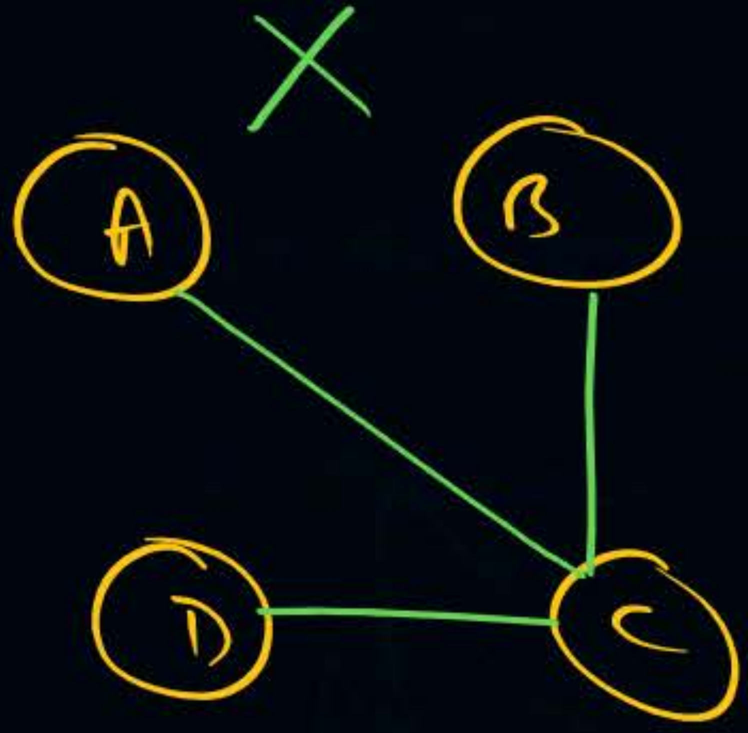
$G = (V, E)$



T_1



T_2



T_3

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 ______ ;

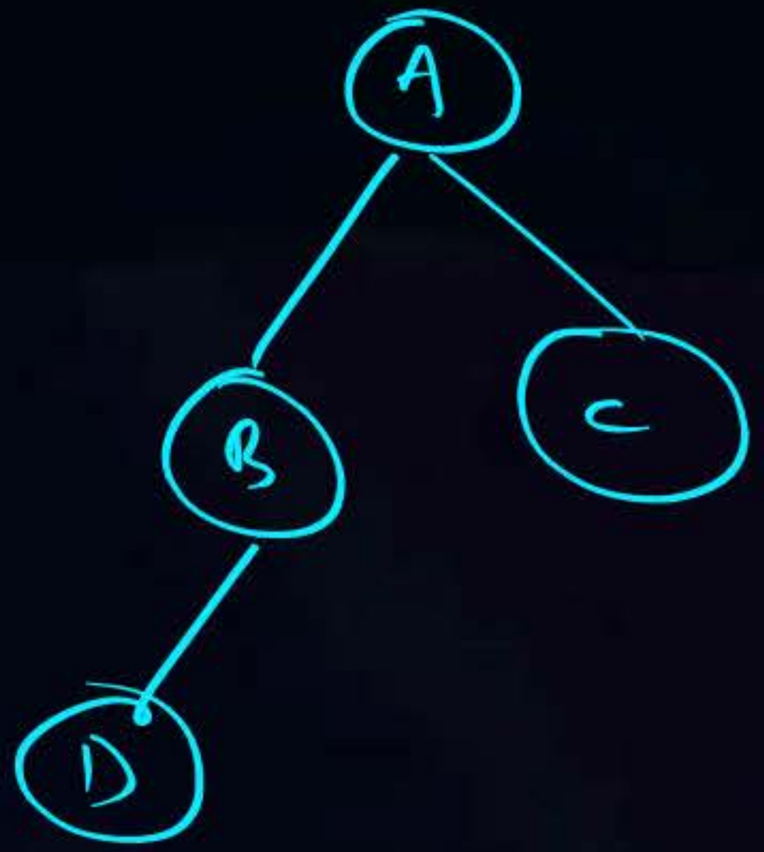
→ Total edges in Graph : e

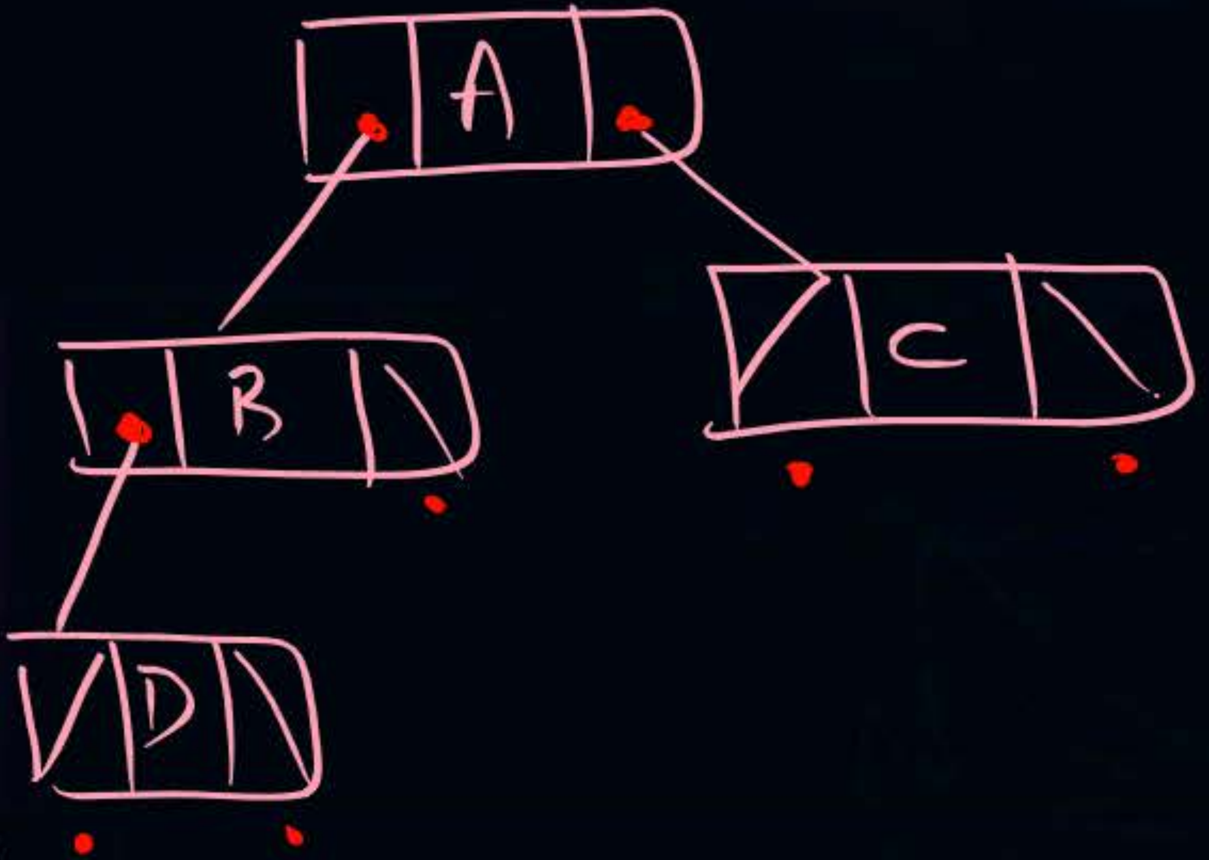
→ Spanning Tree with 'n' vertices will have : $\boxed{(n-1) \text{ edges}}$ <used>

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

: $\boxed{e-n+1}$

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





Total links : $2n$

Edges : $(n-1)$

Unused (NILL fields) : $2n-(n-1)$: $(n+1)$

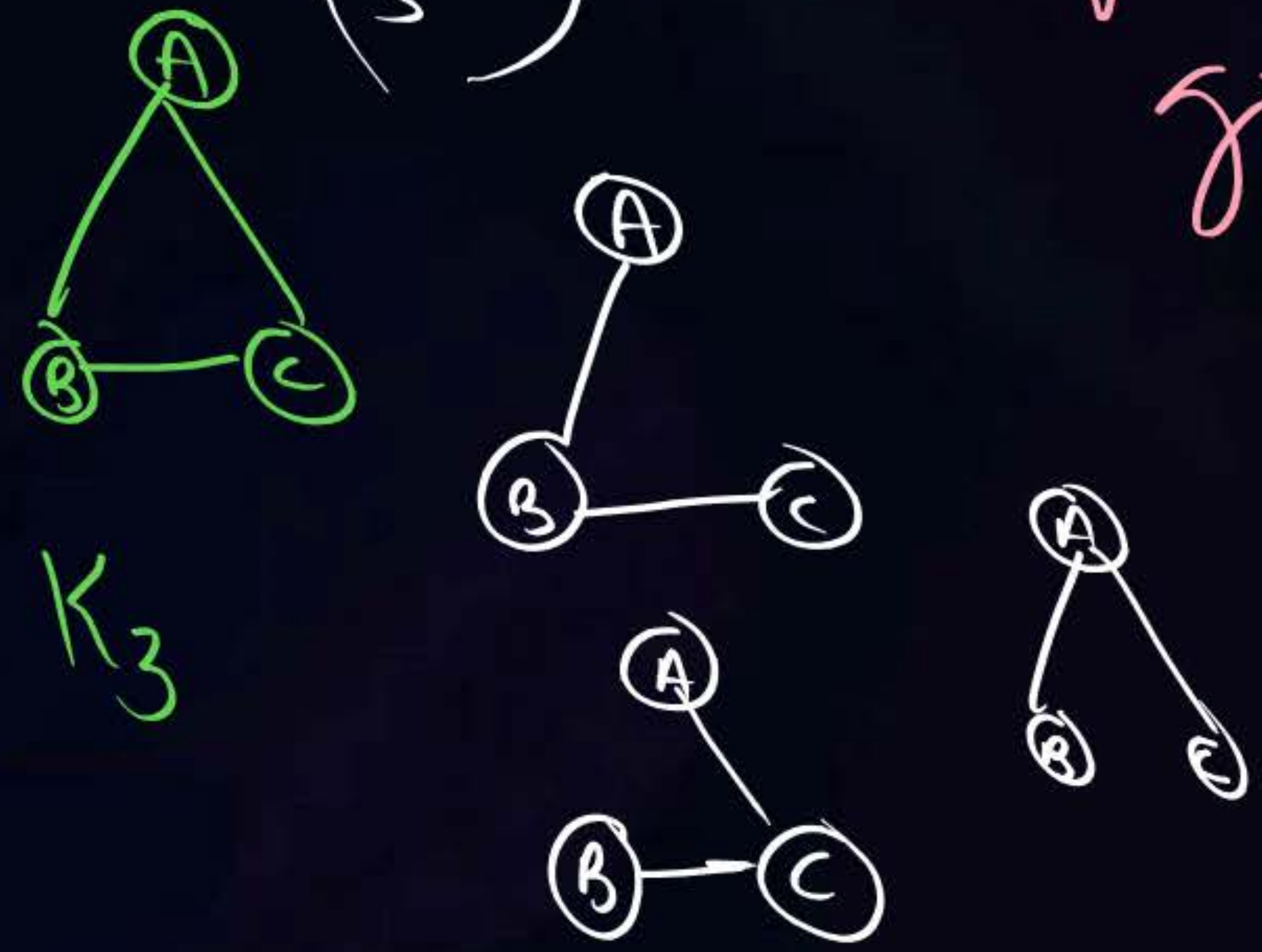
Soln Space: For a Given Graph with n-vertices & 'e' edges the max. # of Sp. Trees (Soln Space)

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

→ For Complete Graph K_n

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

$\left(3^{3-2}\right)$



Applications of Spanning Tree :

(i) Multicasting & Broadcasting in both wired & Wireless Networks;
S.T

(ii) Construct | Implement Circuits (Electronic | Electrical) in communication networks;
MCST

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

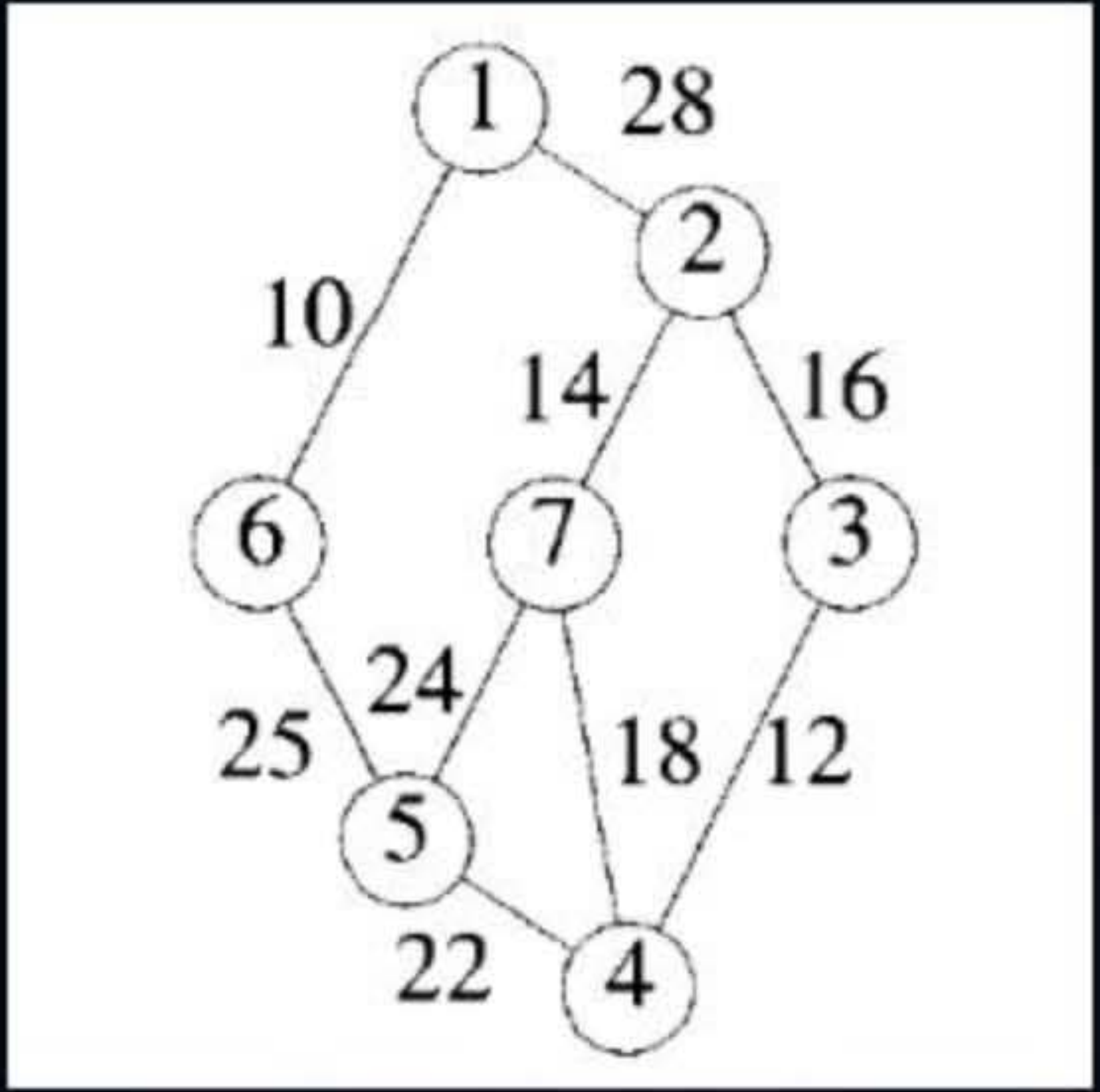
(i) Prim's - Jarnik Algo

(ii) Kruskal's Algo.

(iii) Dijkstra's Algo.

Topic : Greedy Method

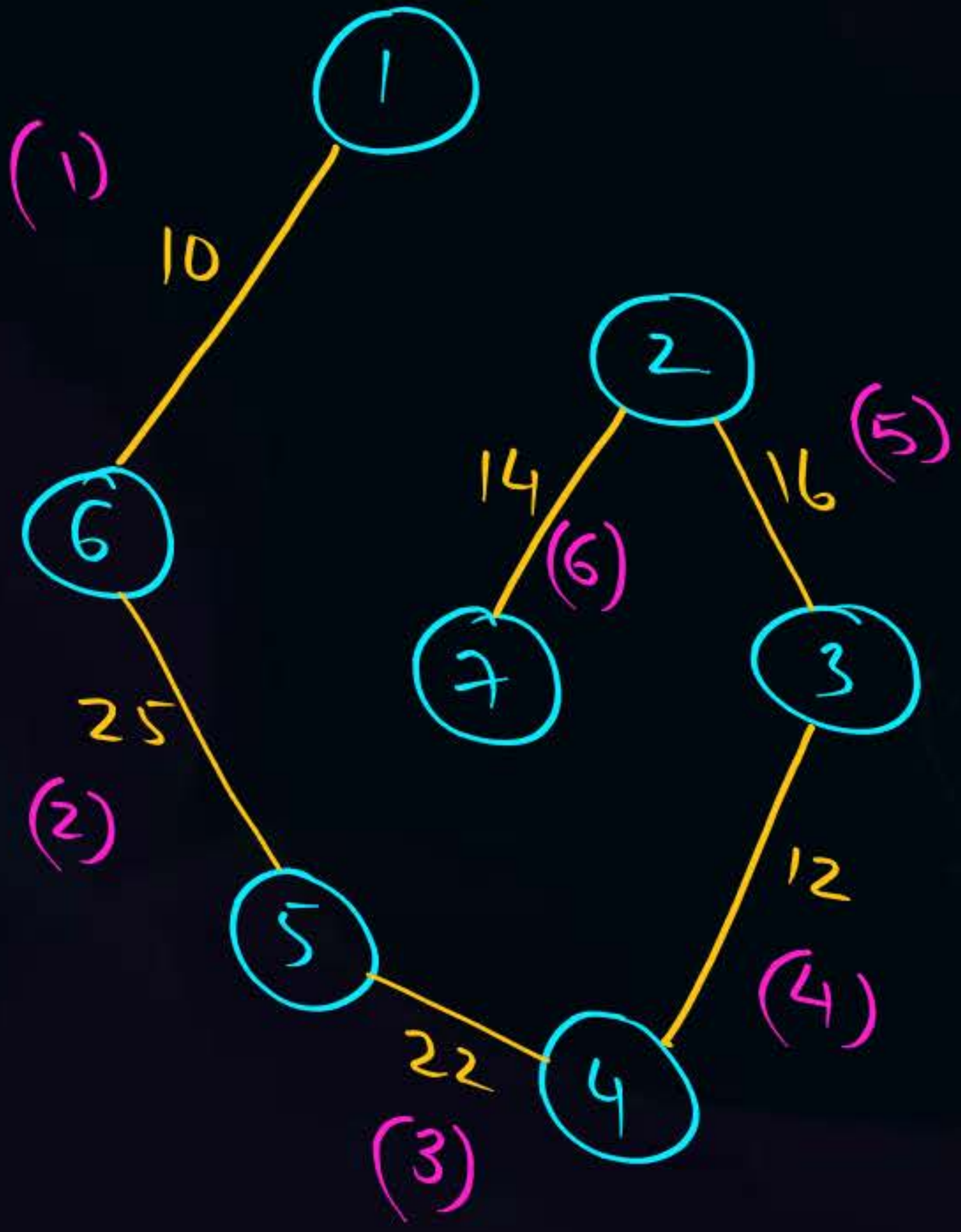
$O(e \cdot \log e)$ Min-Heap



$G = (V, E)$

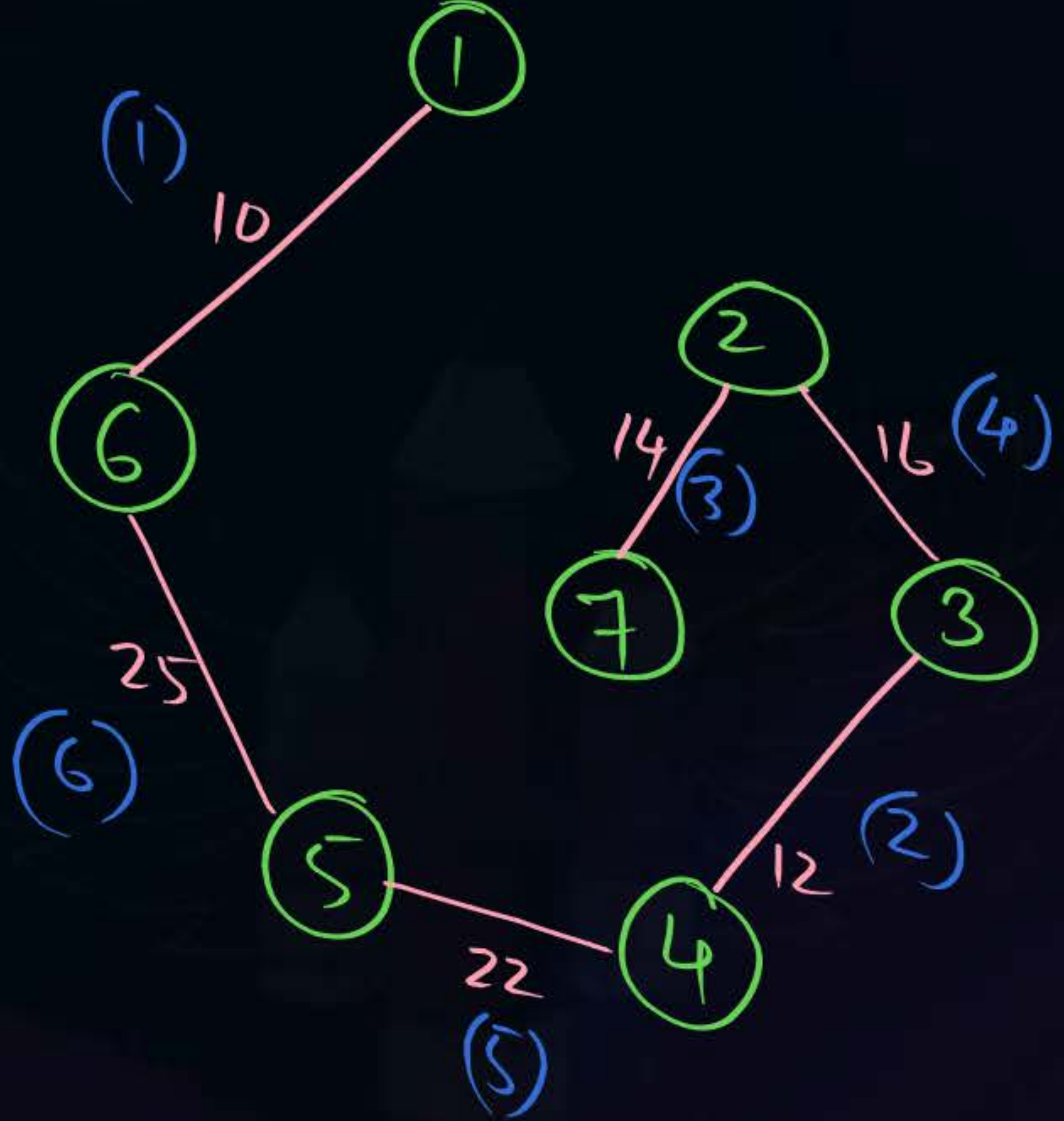
Prim's Algo

Time : $O(n^2)$



Kruskal's Algo

$\langle : \cancel{10}; \cancel{12}; \cancel{14}; \cancel{16}; \cancel{18}; \cancel{22}; \cancel{24}; \cancel{25}; 28 \rangle$



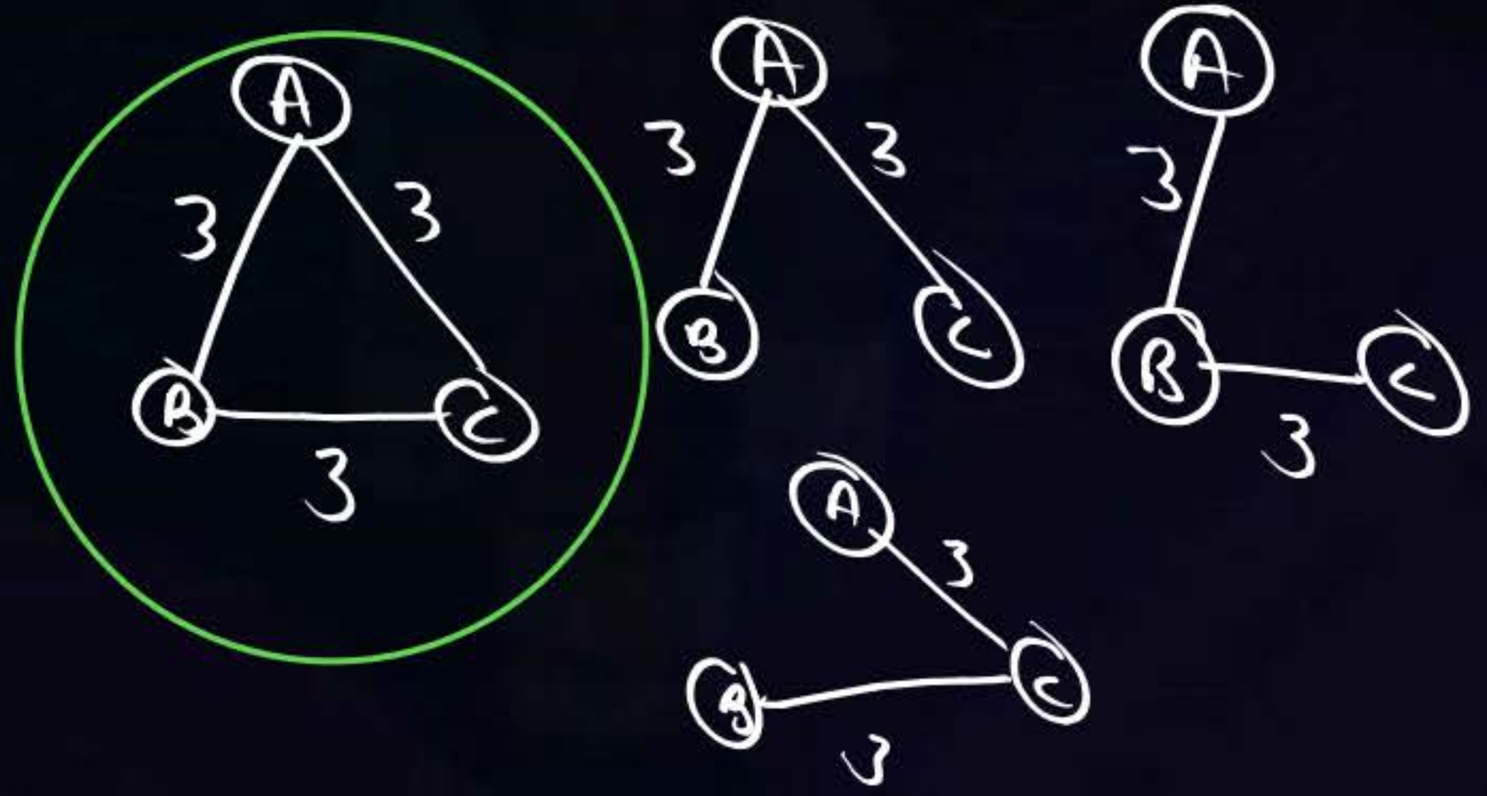
→ Prim's method always maintains Tree Structure Property at each Step;

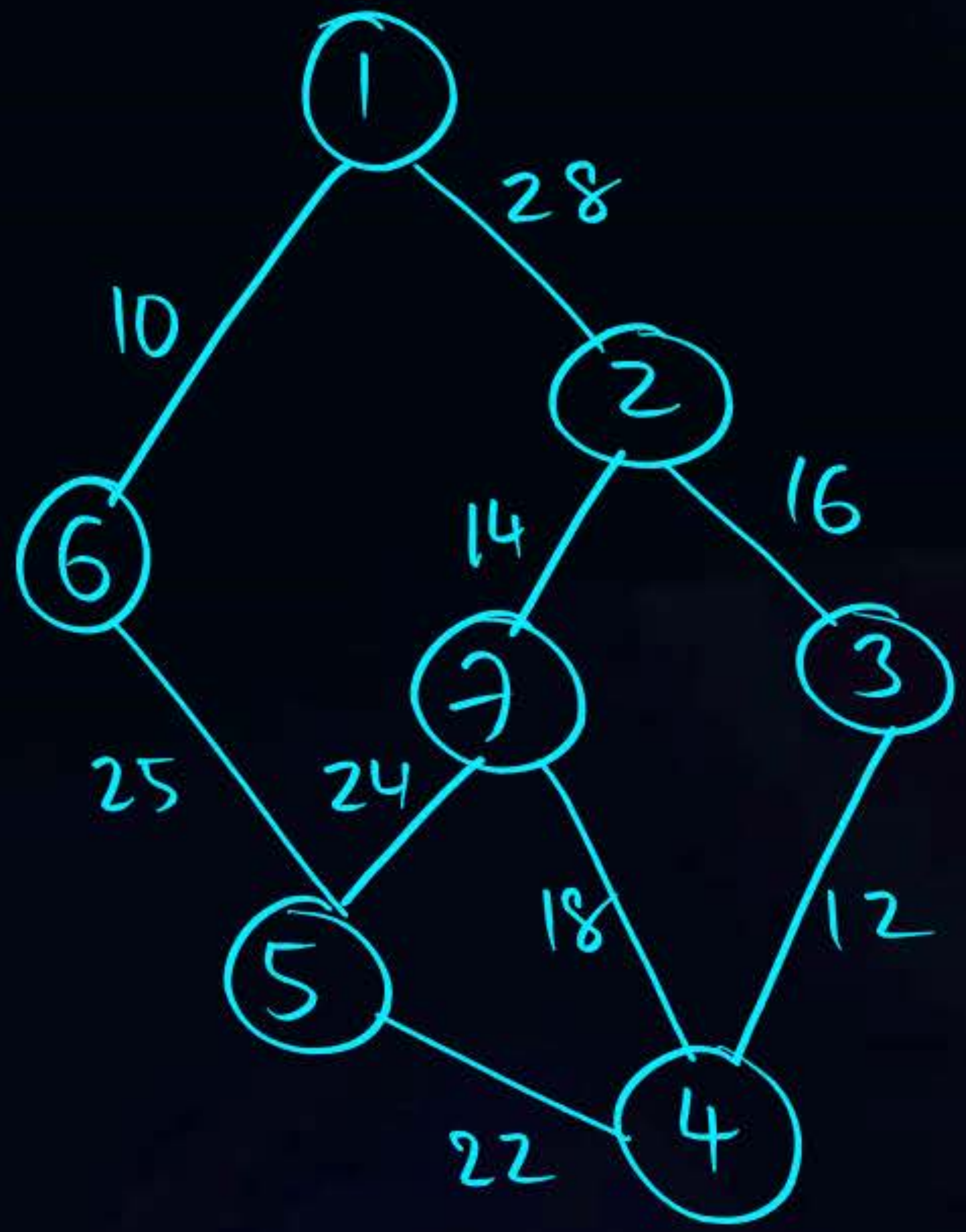
→ Cost of Spanning Tree with both approaches will always be Same;

→ The Tree Structure may/may NOT be Same;

→ The Tree will also be Same iff all edges have unique cost (distinct)

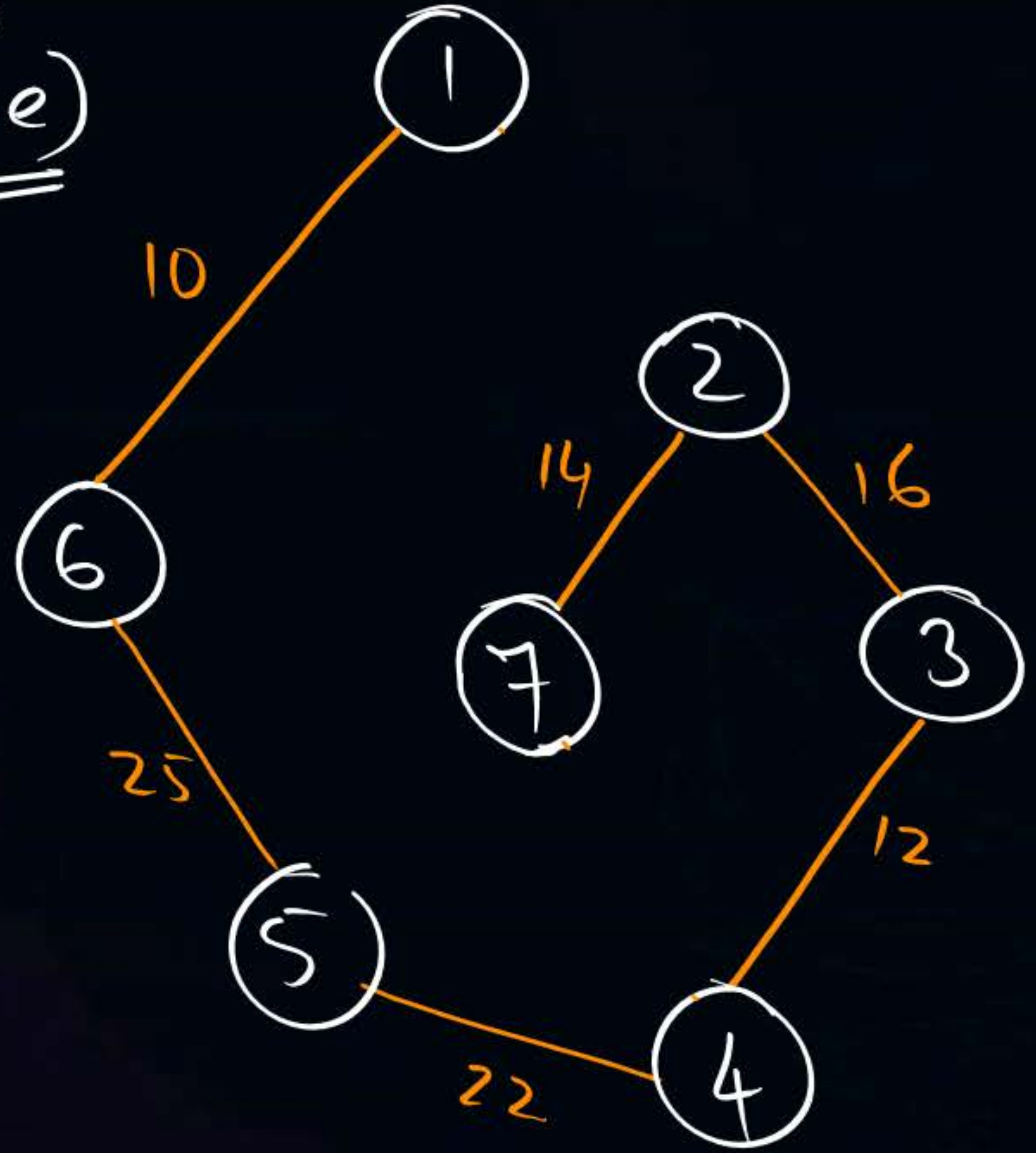
$^3C_2$



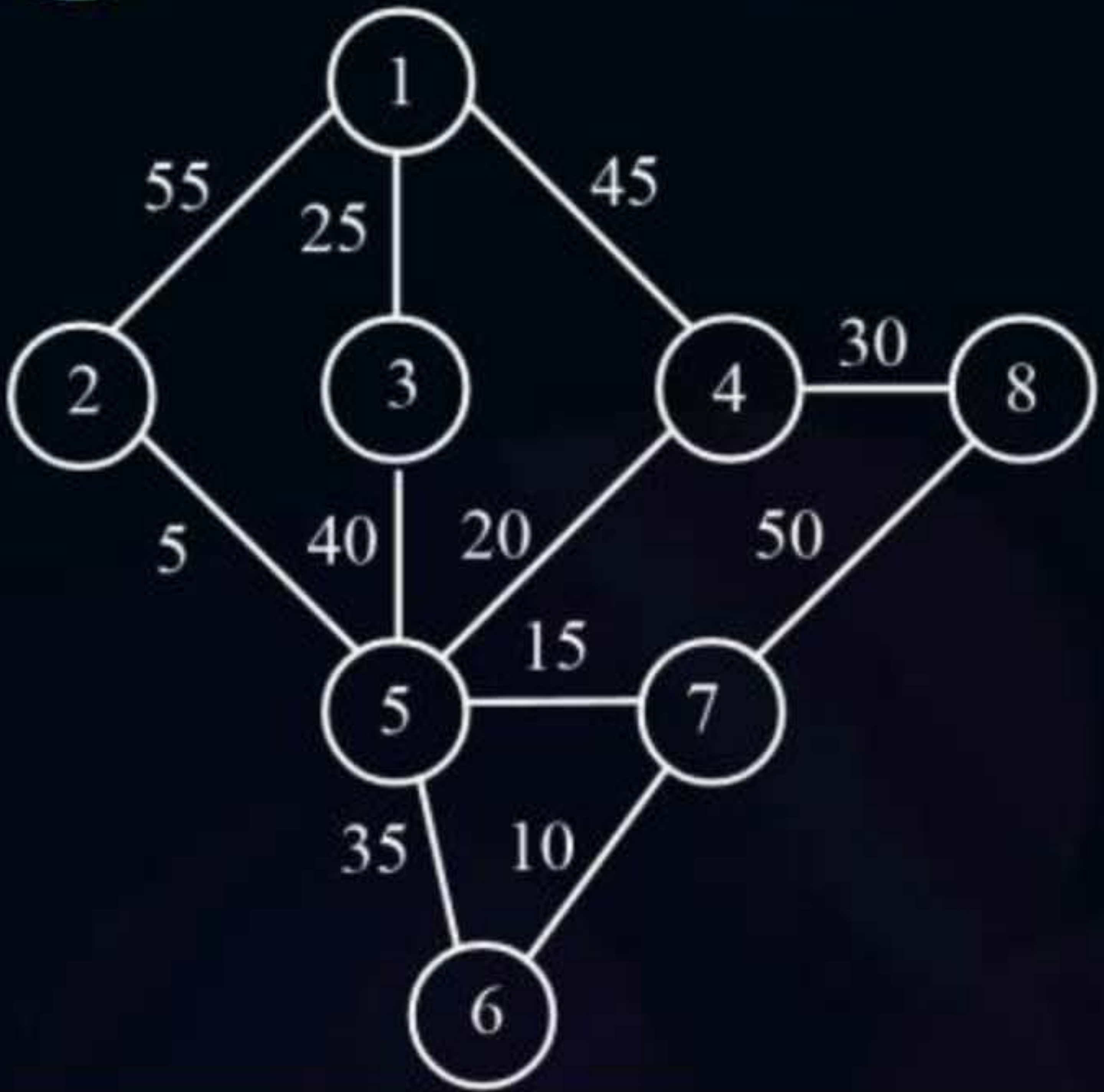


Dijkstra's Algo :

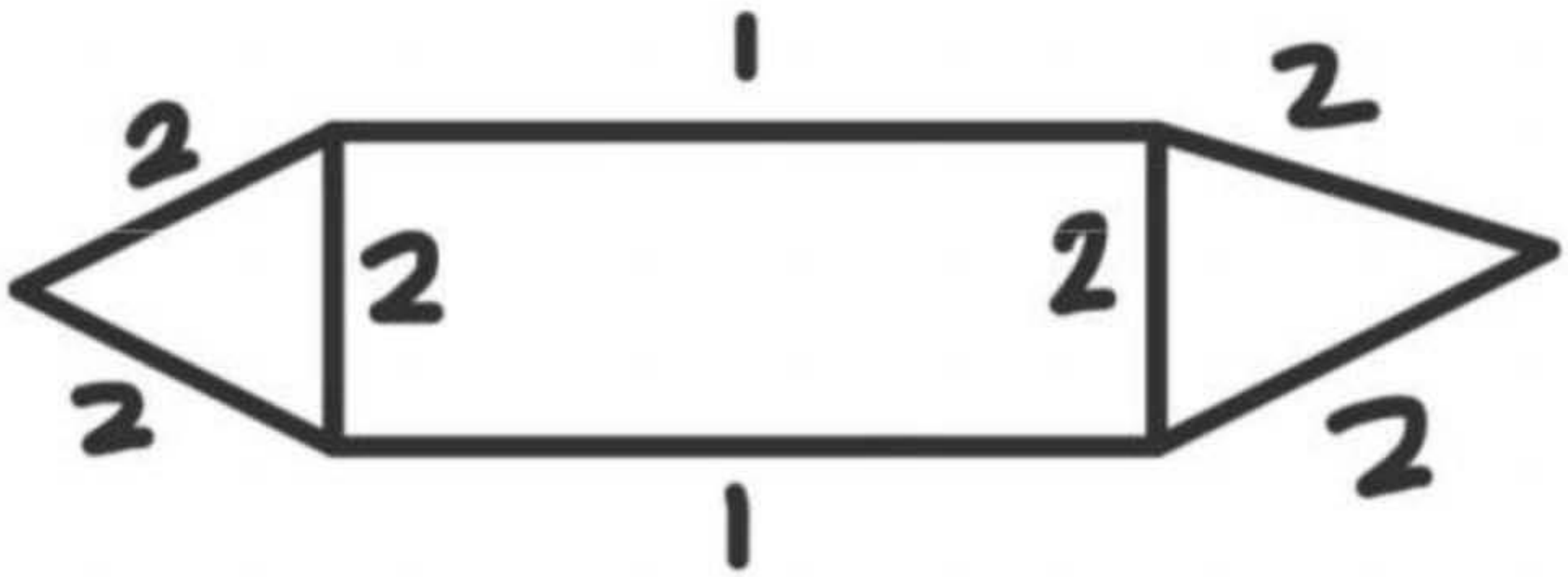
Time :
$O(n \cdot e)$



Topic : Greedy Method



No of minimal spanning trees



Topic : Greedy Method

n; e

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

vertex

```
Algorithm Prim( E , cost ,n , t)
{                                            t[1..n-1, 1..2]
                                                          e
1) Let (k,l) be an edge of minimum cost in E;
2) mincost := cost [k, l];  - 1
3) t[1,1] := k; t[1,2] :=l;  - 1
4) for i := 1 to n do // Initialize near .                      } n
5) if (cost[i, l] < cost [i, k]) then near[i] := l;
   else near[i] := k;
6) near[k] := near[l] := 0;   1
7) for i := 2 to n – 1 do   (n-2) edges
   { // Find n – 2 additional edges for t.
(n)
a) Let j be an index such that near[j] ≠ 0 and
        cost[j, near[j]] is minimum;   } log n
b) t[i,1] := j ; t [i, 2] : = near [j];
c) mincost := mincost + cost [j, near [j]];
d) near[j] := 0;
            n
e) for k:= 1 to n do // Update near[ ].
      if ((near[k] ≠ 0) and (cost[k, near[k]] > cost[k,
                                                        j]))
            then near[k]:= j;
   }
   return mincost;
}
```

Modify key : log n

Slide 8

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) $n^2 * \log n$
9) $e * \log n$

Heap creation $: O(n)$

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

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

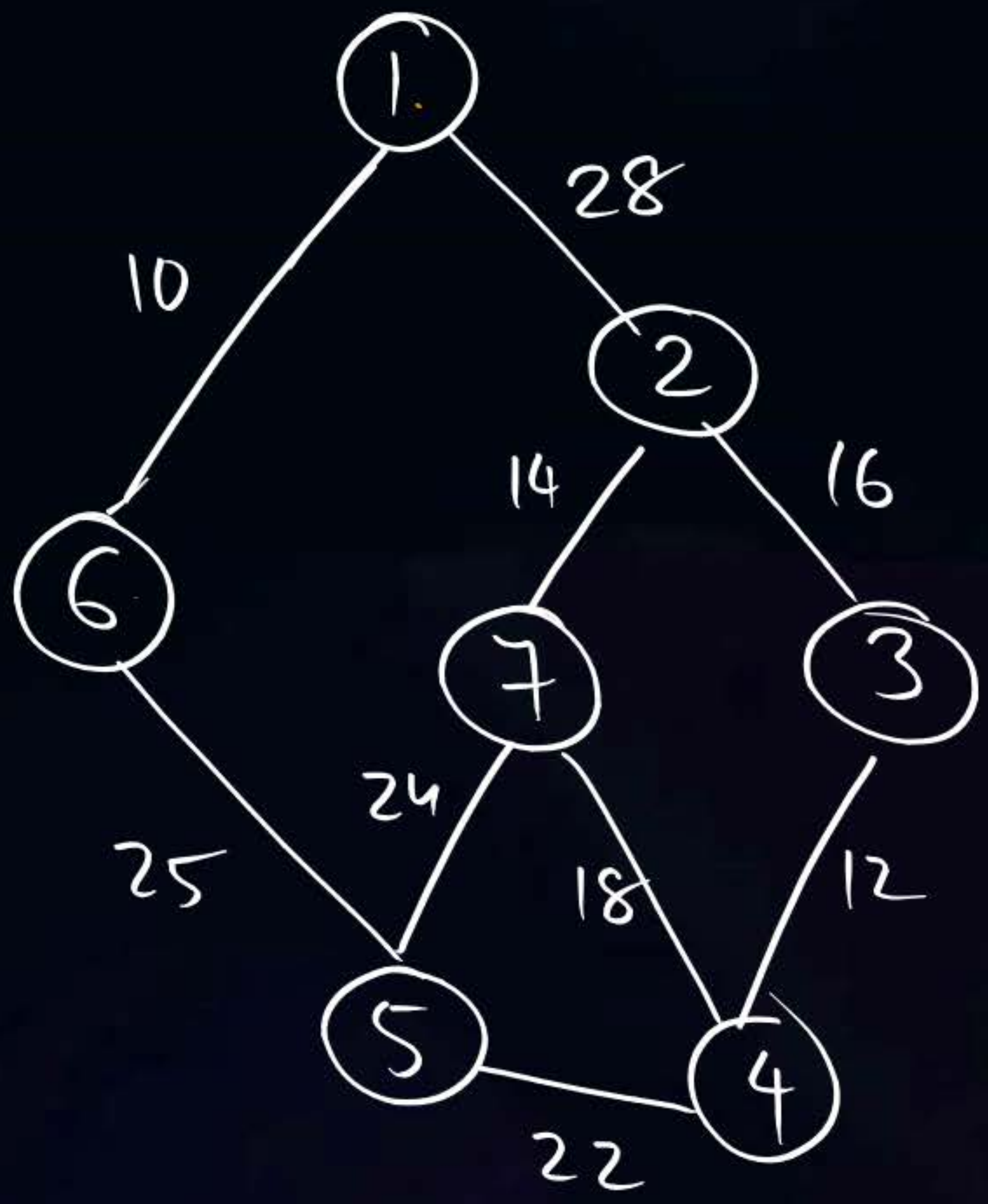
$: \boxed{O((n+e) * \log n)}$

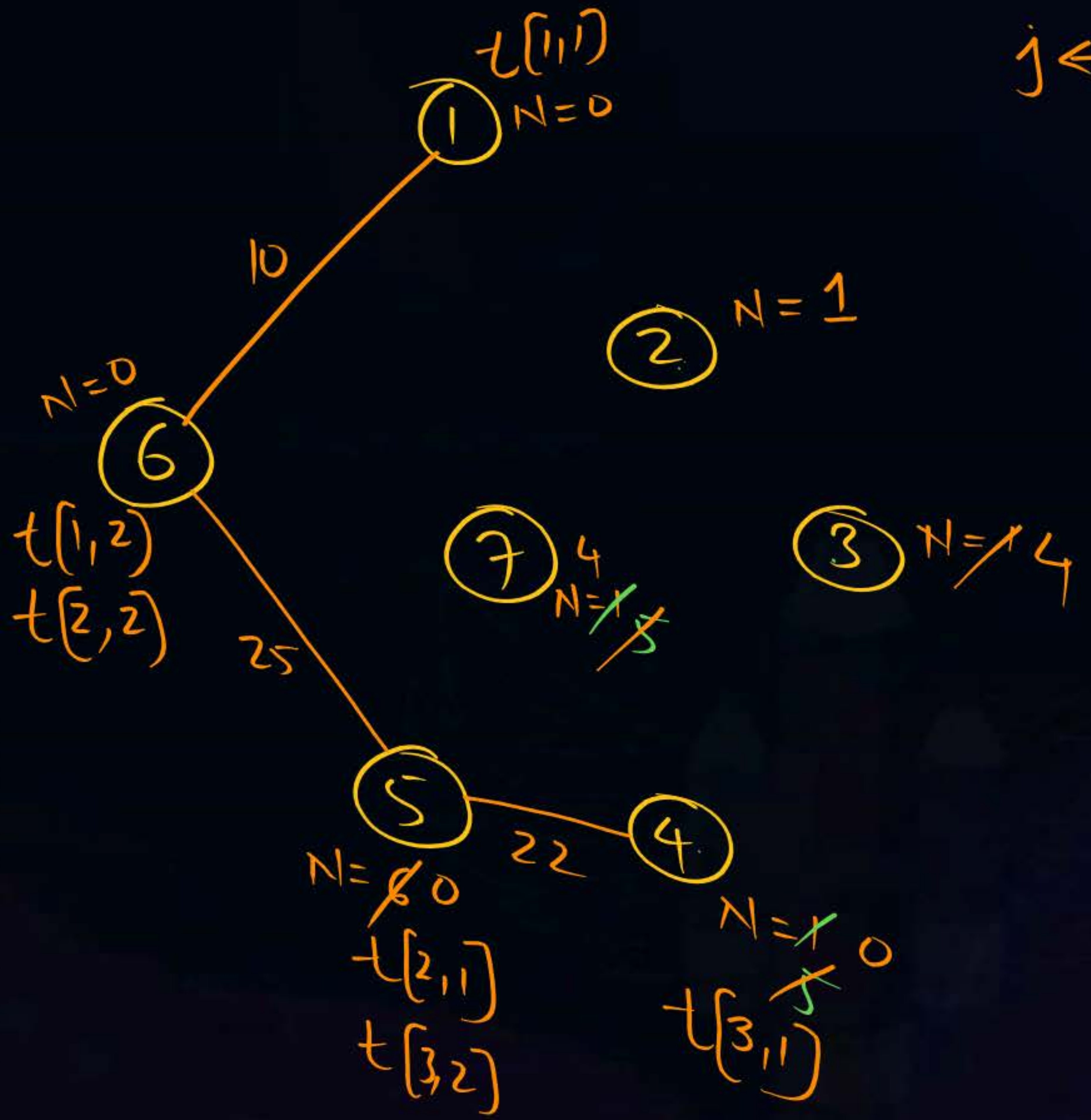
$(k, l) \leftarrow (1, 6)$

$mincost \leftarrow 10 + 25 + 22$

$j \leftarrow 5$

$j \leftarrow 4$





```
t := ∅;
while ((t has less than n − 1 edges) and (E ≠ ∅)) do      e
{
  a) Choose an edge (v, w) from E of lowest cost;   } loge
  b) Delete (v, w) from E;                          }
  c) if (v, w) does not create a cycle in t then add (v, w) to t;
     else discard (v, w);                    } loge
}
```

$\boxed{O(e * \log e)}$

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

(High-level Implementation)

Topic : Greedy Method

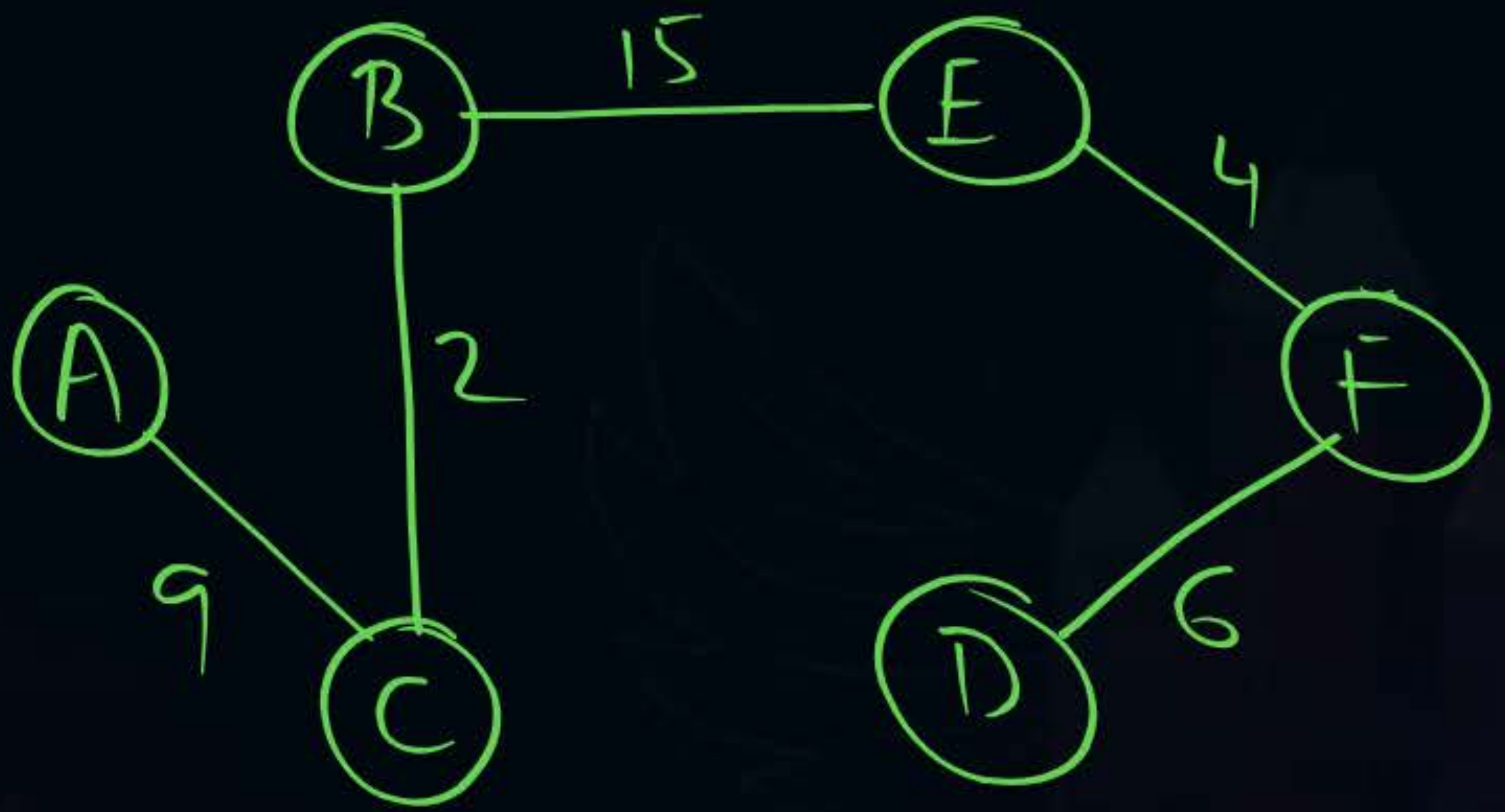
```
Algorithm Kruskal (E, cost, n, t)
{
Construct a heap out of the edge costs using Heapify;
for i : = 1 to n do parent[i] := — 1;
i := 0; mincost := 0.0;
while ((i < n — 1) and (heap not empty)) do
{
Delete a minimum cost edge (u, v) from the heap
and reheapify using Adjust;
j:= Find(u); k; = Find(v);.
if (j ≠ k) then
{
    i := i + 1;
    t[i, 1] := u; t[i, 2] := u;
    mincost := mincost + cost[u, v];
    Union (j, k);
    }
}
if (i ≠ n –1) then write ("No spanning tree");
else return mincost;
}
```

PYQ (NAT)

Topic : III. Greedy Method

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~~).

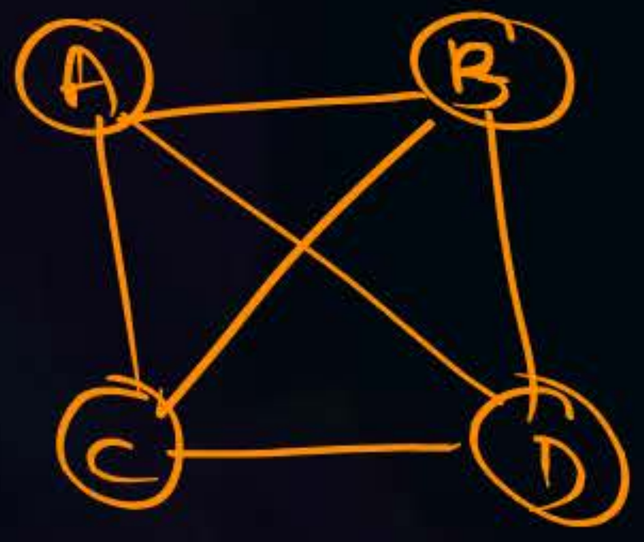
66

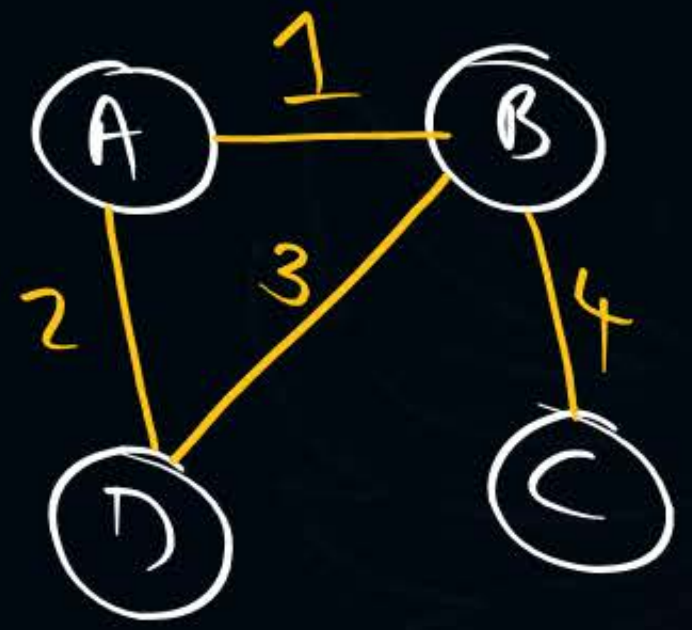


Topic : III. Greedy Method

PYQ/NAT

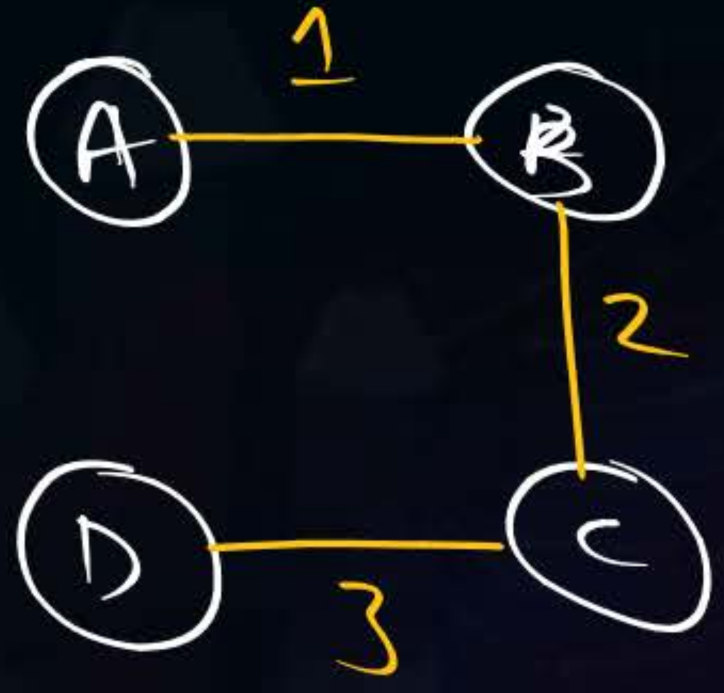
Q. Let G be a complete undirected graph with 4 vertices and edge weights are {1, 2, 3, 4, 5, 6}. The maximum possible weight that a minimum weight Spanning Tree can have is 7.





$(1+2+4)=7$

Normal Situation



: 6

THANK - YOU