GATE

CRASH COURSE

DS & AI

Algorithms

Graph: Shortest Path

(Part 01) (Lecture 8)

By - Aditya sir

Topics to be Covered

1
2
3
4

DP-based

Graph: Shortest Path Algos
- SSSP
- APSP
- Multi-stage graph
- Travelling Salesman Problem

About Aditya Jain sir

1. Appeared for GATE during BTech and secured AIR 60 in GATE in very first attempt - City topper
2. Represented college as the first Google DSC Ambassador.
3. The only student from the batch to secure an internship at Amazon. (9+ CGPA)
4. Had offer from IIT Bombay and IISc Bangalore to join the Masters program
5. Joined IIT Bombay for my 2 year Masters program, specialization in Data Science
6. Published multiple research papers in well known conferences along with the team
7. Received the prestigious excellence in Research award from IIT Bombay for my Masters thesis
8. Completed my Masters with an overall GPA of 9.36/10
9. Joined Dream11 as a Data Scientist
10. Have mentored working professions in field of Data Science and Analytics
11. Have been mentoring GATE aspirants to secure a great rank in limited time
12. Have got around 27.5K followers on Linkedin where I share my insights and guide students and professionals.

@ADITYASIR_PW

Telegram Link for Aditya Jain sir:

https://t.me/AdityaSir_PW

Topic : (Lecture Schedule)

3. Shortest Path Algos

1. Dijkstra
2. Bellman Ford
3. Floyd Warshall
4. Multi-stage Graph
5. Travelling Salesman Problem

Greedy

SSSP

APSP

DP-based

Topic : Greedy Method

Dijkstra's Algo SSSP

```
Algorithm ShortestPaths(v, cost dist,n)
// dist [j] , 1 ≤ j ≤ n, is set to the length of the shortest
// path from vertex v to vertex j in a digraph G with n
// vertices. dist[v] is set to zero.- G is represented by its
// cost adjacency matrix cost/.[1 : n, 1 : n].
{
        for i := 1 to n do
        { // Initialize S.
        S[i] := false; dist[t] := cost[v , i];
}
S[v] := true; dist[v] := 0.0; // Put v in S.
for num := 2 to n — 1 do
```

$V \rightarrow$ source

C	1	2	3	..	n
1					
2					
3					
:					
n					

c[i,j] :

Topic : Greedy Method


```
{
    // Determine n — 1 paths from v.
    Choose u from among those vertices not
    in S such that dist[u] is minimum;
    S[u] := true; // Put u in S.
    for (each w adjacent to u with S[w] = false) do
    // Update distances.
    if (dist [w] > dist[u] + cost[u, w])) then
    dist[w]:= dist[u] + cost[u, w];
    }
}
```

not yet added

Relaxation Process

	P	Q	R	S
{P}	0	_	_	_
{P, R}	_			_

Time Complexity of Dijkstra's SSSP Algo:

$G(V,E)$

$|V| = n$

$|E| = e$

1) Non-Heap Implementation : $O(n^2)$

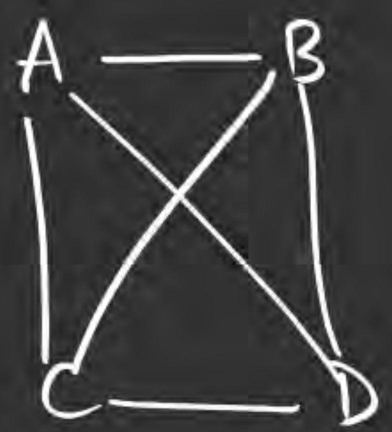
2) Heap-based Implementation : $O((n+e) * \log n)$

In Worst Case, $e = O(n^2)$

Complete graph G

Directed

undirected



$n = 4$
$e = 6$

$n = 4$
$e = 12$

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

$n = 4$

$\frac{4 \times 3}{2}$

$= 6$

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

$n = 4$

$4 \times (4-1)$

$= 4 \times 3 = 12$

Question

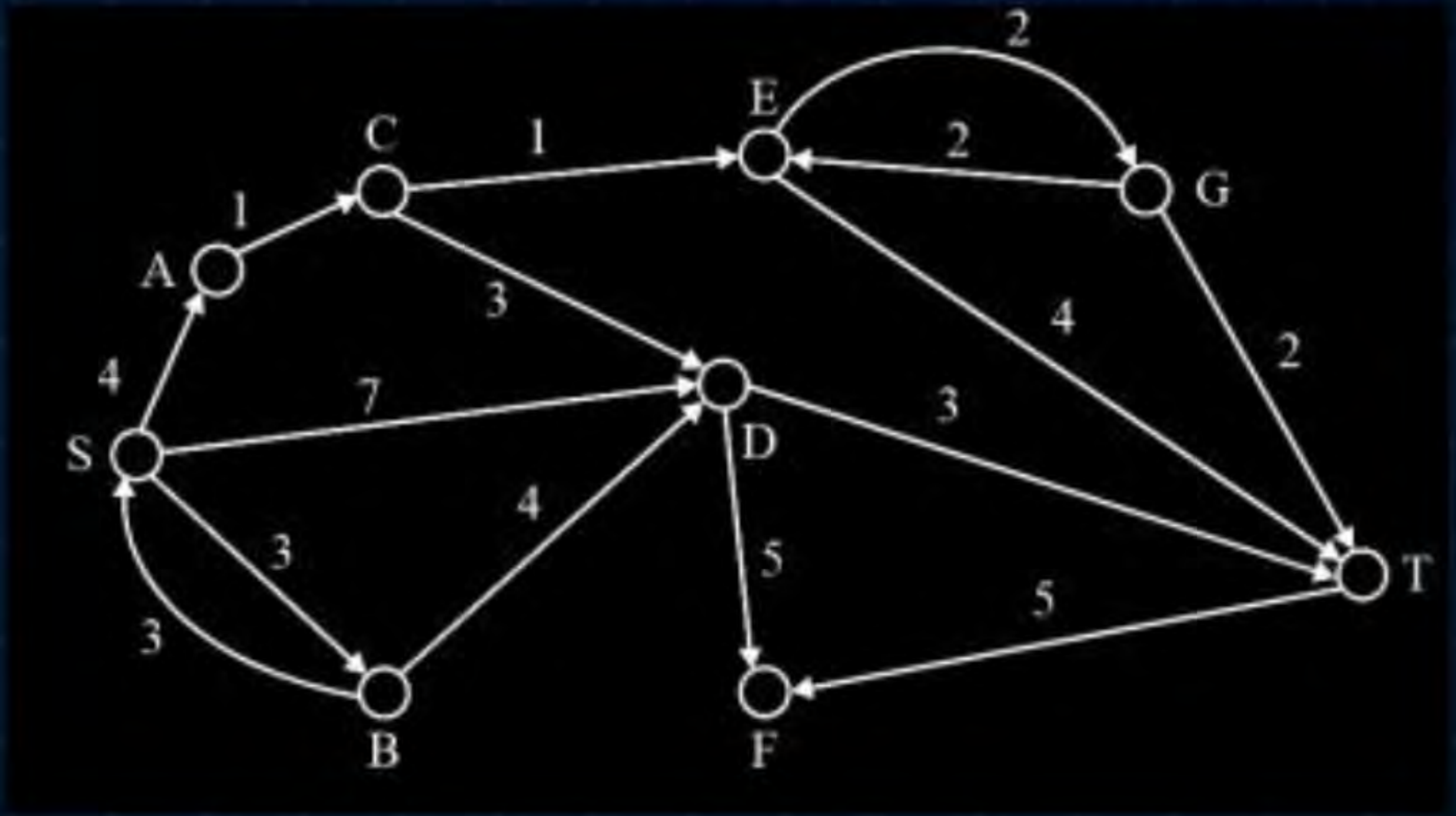
#Q. Applying Dijkstra's Algorithm over the given Graph, Which path is reported from 'S' to 'T';

A) SDT

B) SBDT

C) SACDT

D) SACET



Soln :-

A) SDT : $S \longrightarrow D \longrightarrow T \Rightarrow 7+3=10$ ✓

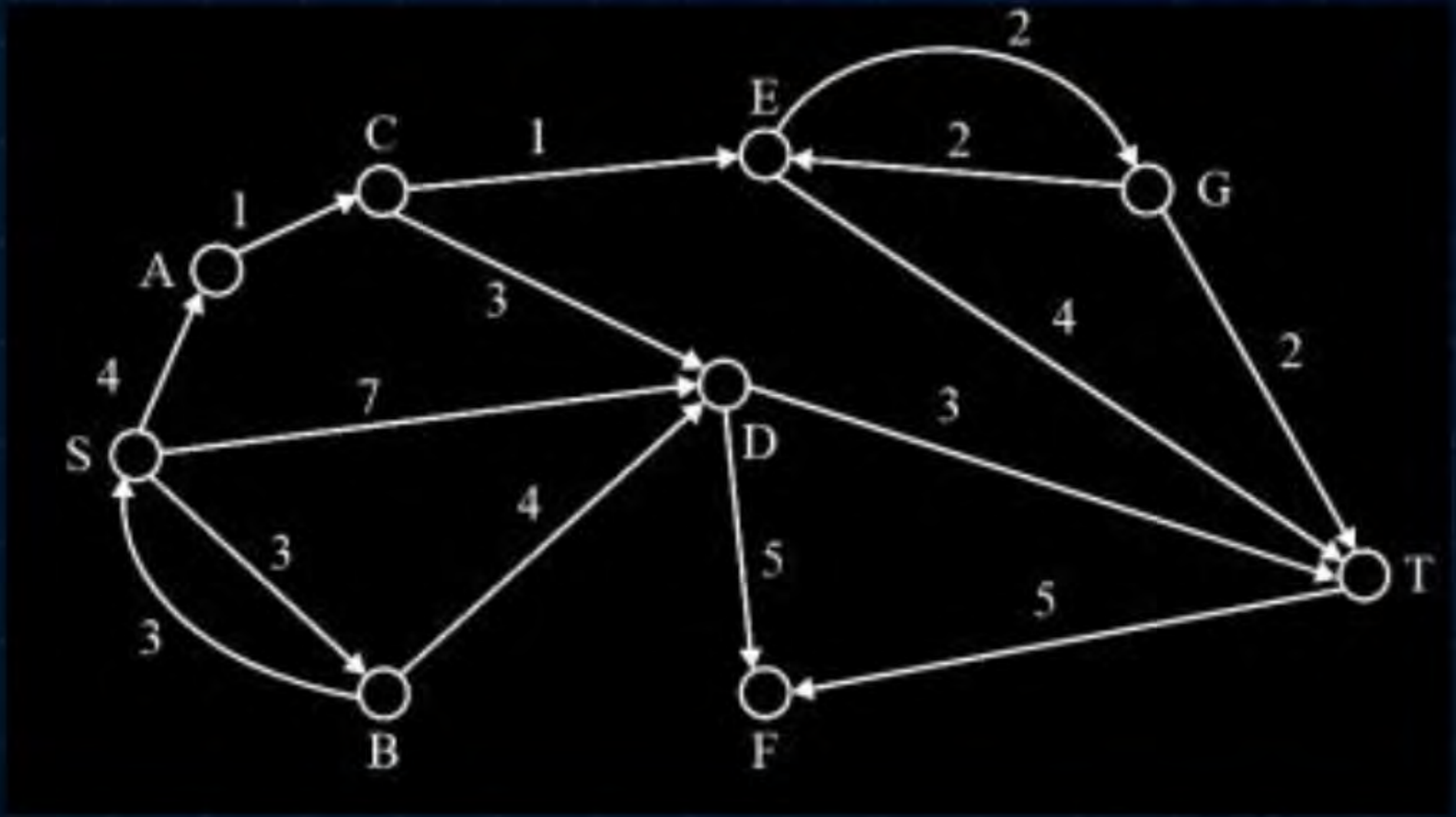
B) SBDT : $S \longrightarrow B \longrightarrow D \longrightarrow T \Rightarrow 3+4+3=10$ ✓

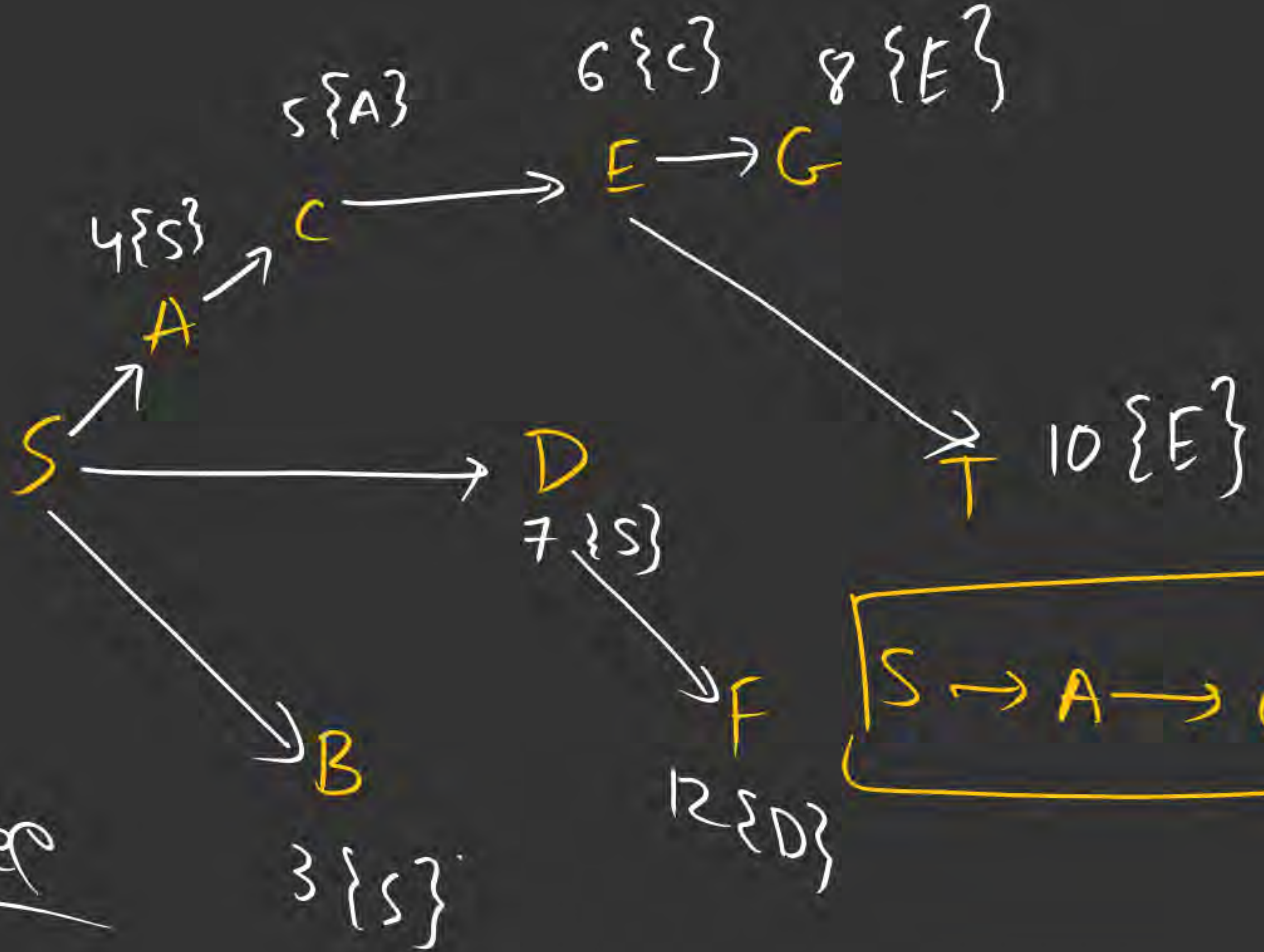
C) SACDT : $S \longrightarrow A \longrightarrow C \longrightarrow D \longrightarrow T \Rightarrow 4+1+3+3=11$ ✗

D) SACET : $S \longrightarrow A \longrightarrow C \longrightarrow E \longrightarrow T \Rightarrow 4+1+1+4=10$ ✓

Question

#Q. Applying Dijkstra's Algorithm over the given Graph, Which path is reported from 'S' to 'T';





Dijkstra

SSSP Tree

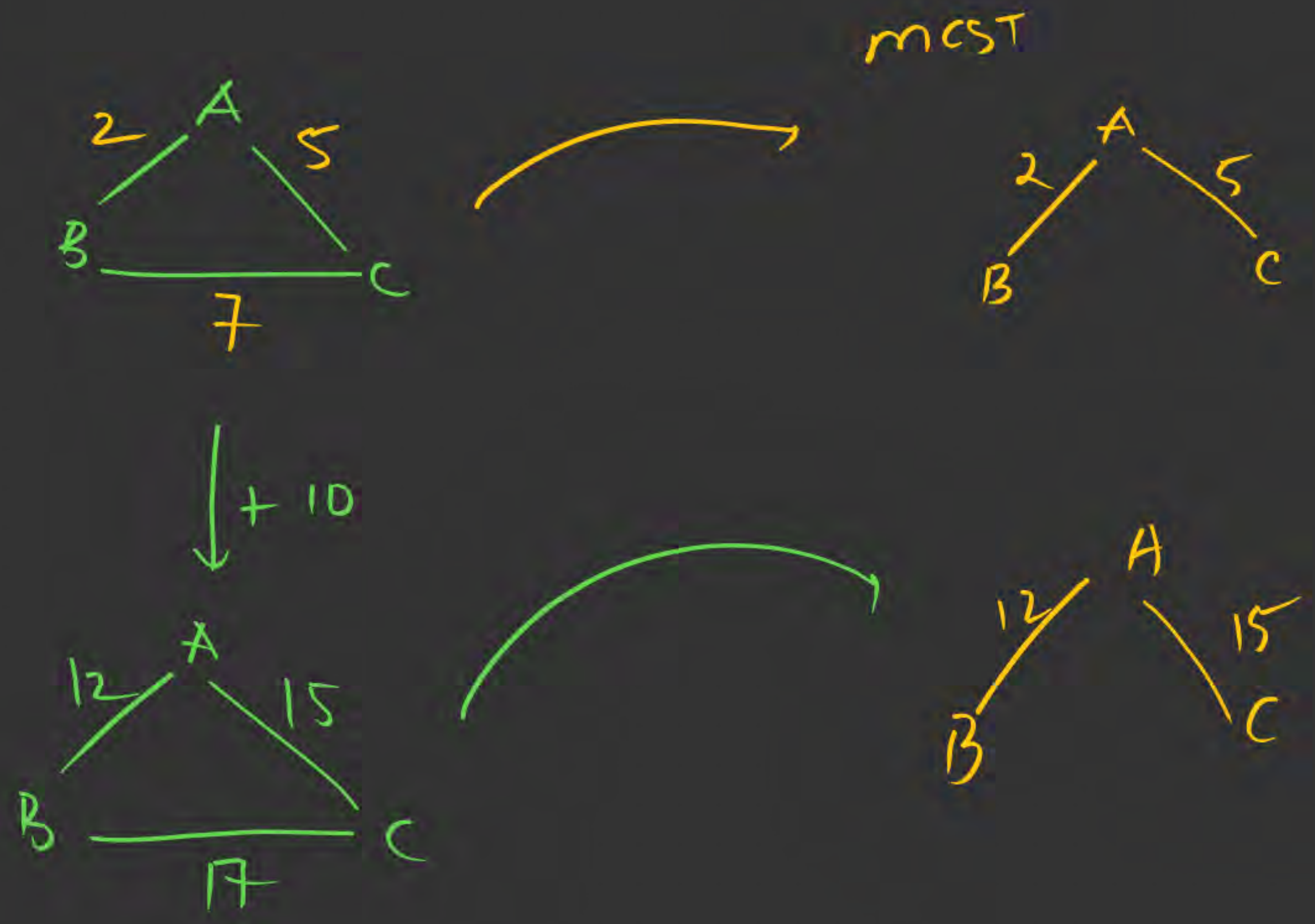
$\boxed{S \to A \to C \to E \to T}$

Question

#Q. Let G be weighted connected undirected graph with distinct positive edge weights. If every edge weight is increased by the same value , then which of the following statements is/are true?

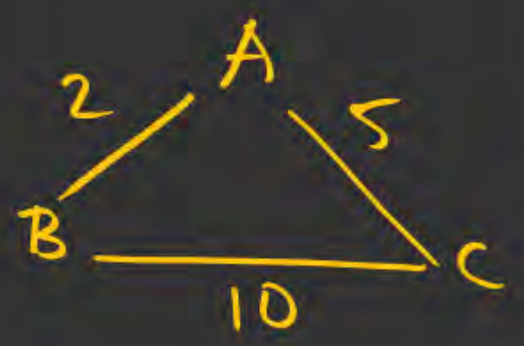
1. Minimum spanning Tree of the graph does not change. True
2. Shortest path between any pair of vertices does not change. False.

eg)



eg)

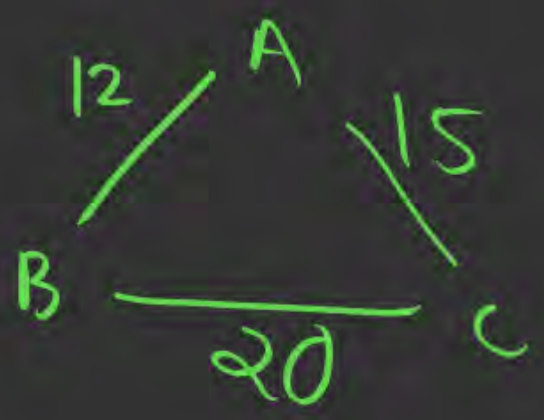
Shortest $B \to C$



$B \to C$: $B \to A \to C$

$2 + 5 = 7$

$+10$



$B \to C$?

$BAC \to 12 + 15 = 27$ ✗

$BC \to 20$ ✓

Question

#Q. Let G = (V, E) be any connected undirected edge- weighted graph. The weights of the edges in E are positive and distinct. Consider the following statements:

(I) Minimum Spanning Tree of G is always unique True

(II) Shortest path between any two vertices of G is always unique. False

Which of the above statements is/are necessarily true ?

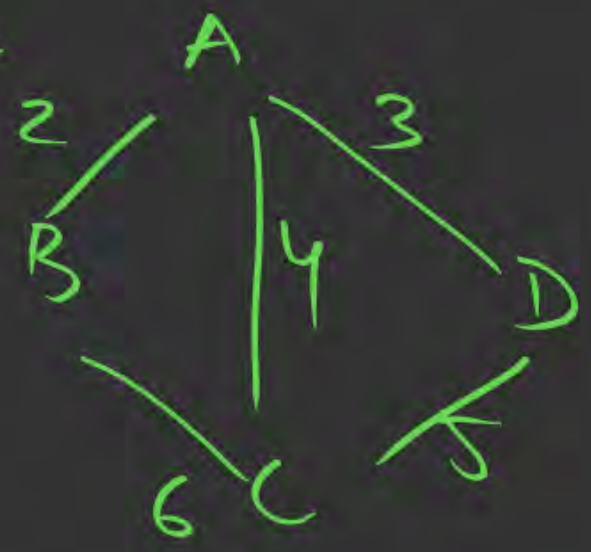
A (I) only

B (II) only

C Both (I) and (II)

D Neither (I) and (II)

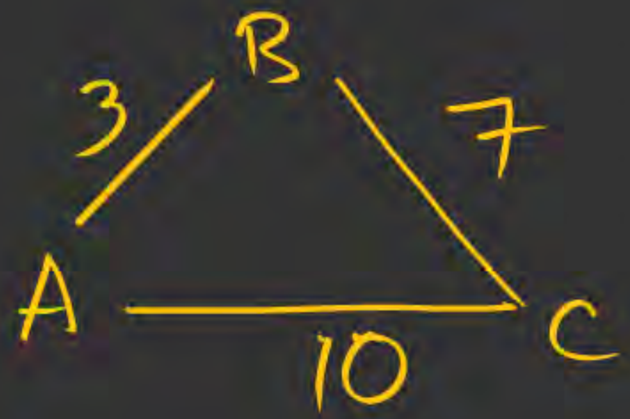
eg:-



MCST

Kruskal MCST

eg2)



1) A – B – C : $3 + 7 = \underline{10}$

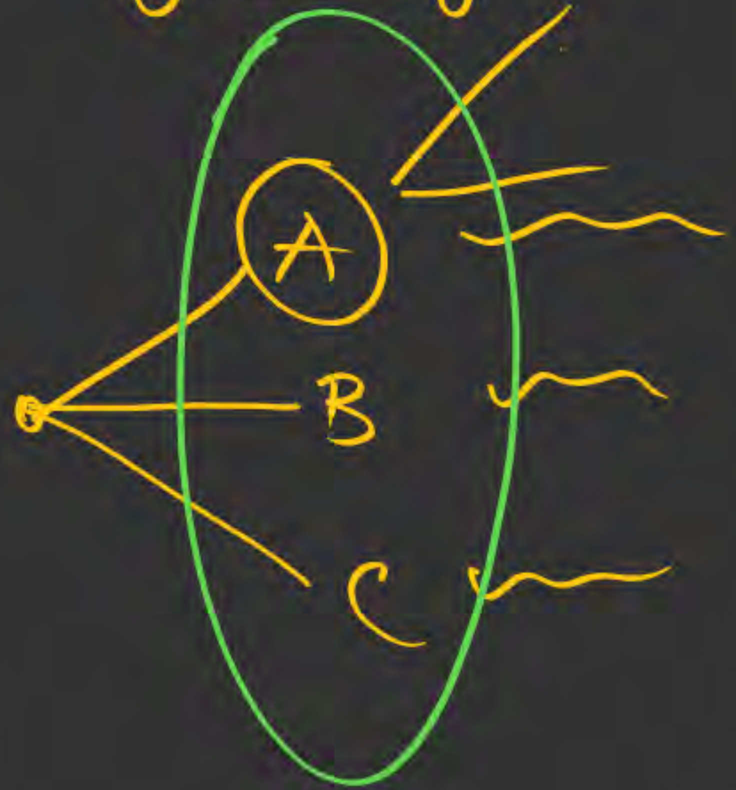
2) A – C : $\underline{10}$

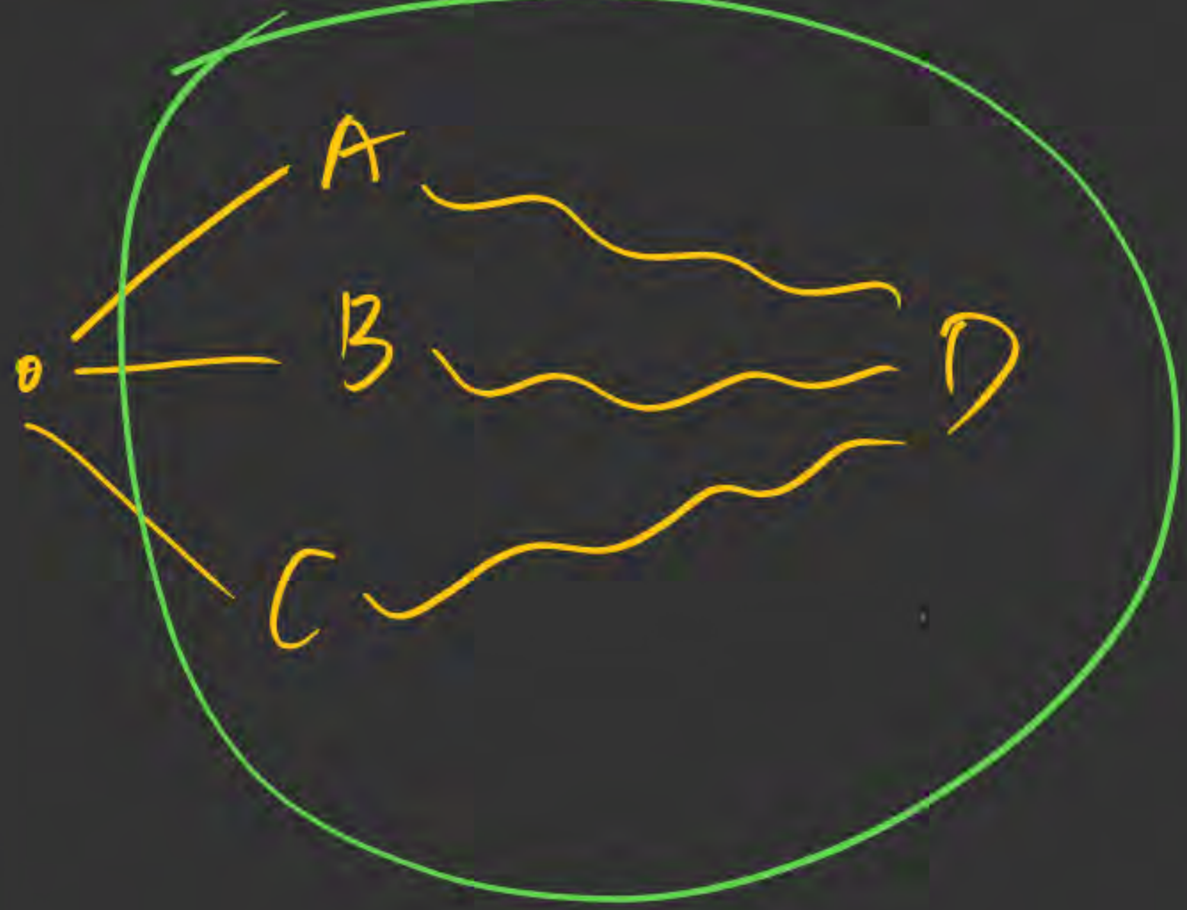
Topic : Dynamic Programming (DP)

Dynamic Programming (DP) is an algorithm design method used for solving problems, whose solutions are viewed as a result of making a set / sequence of decisions.

- One way of making these decisions is to make them one at a time in a step-wise (sequential) step-by-step manner and never make an erroneous decision. This is true of all problems solvable by Greedy method.)
- For many other problems it is not possible to make step-wise decisions based on local information available at every step, In such a manner that the sequence of decision made is optimal.

Greedy vs DP

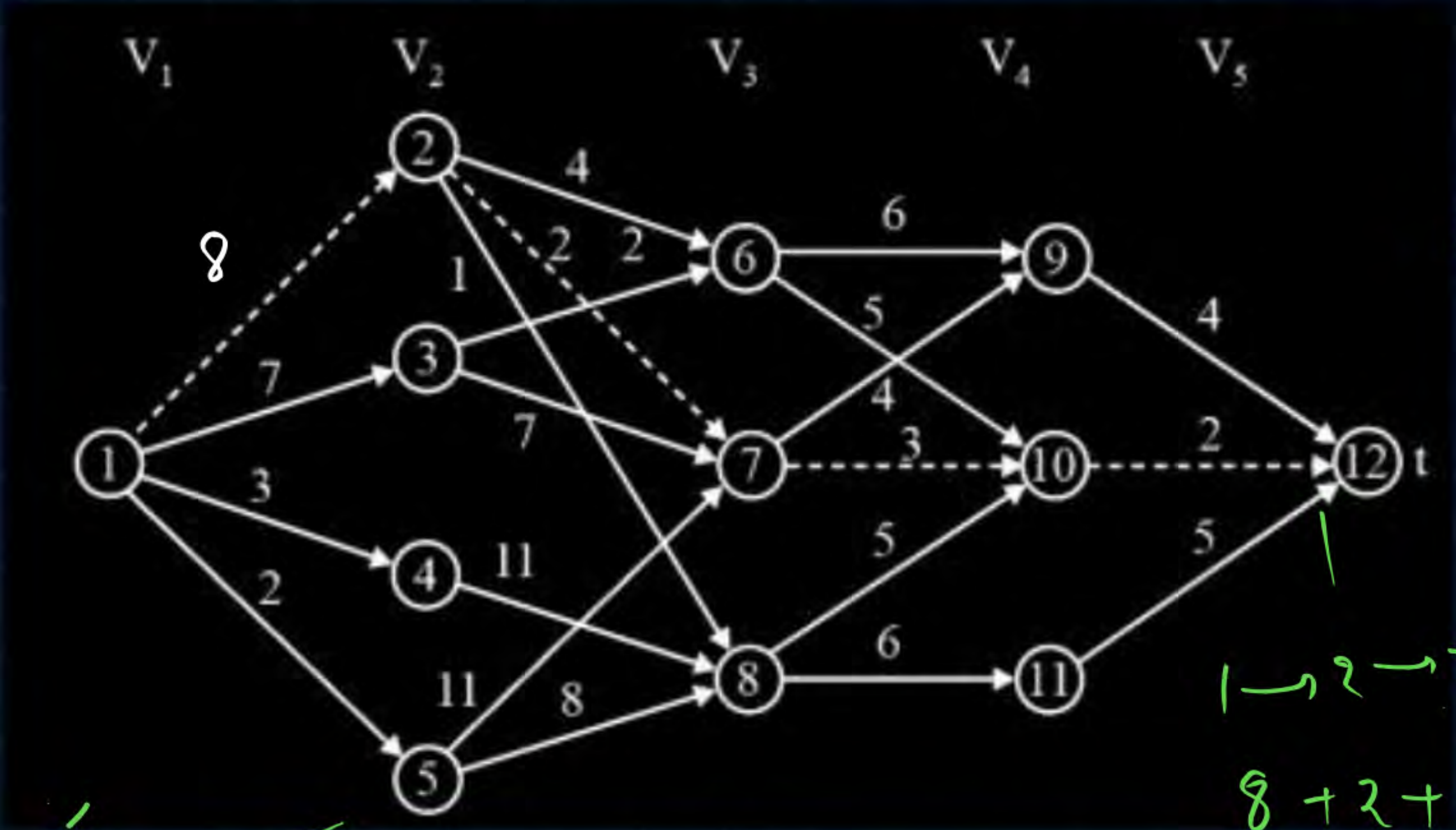




Topic : Dynamic Programming (DP)

$1 \longrightarrow 12$

Greedy

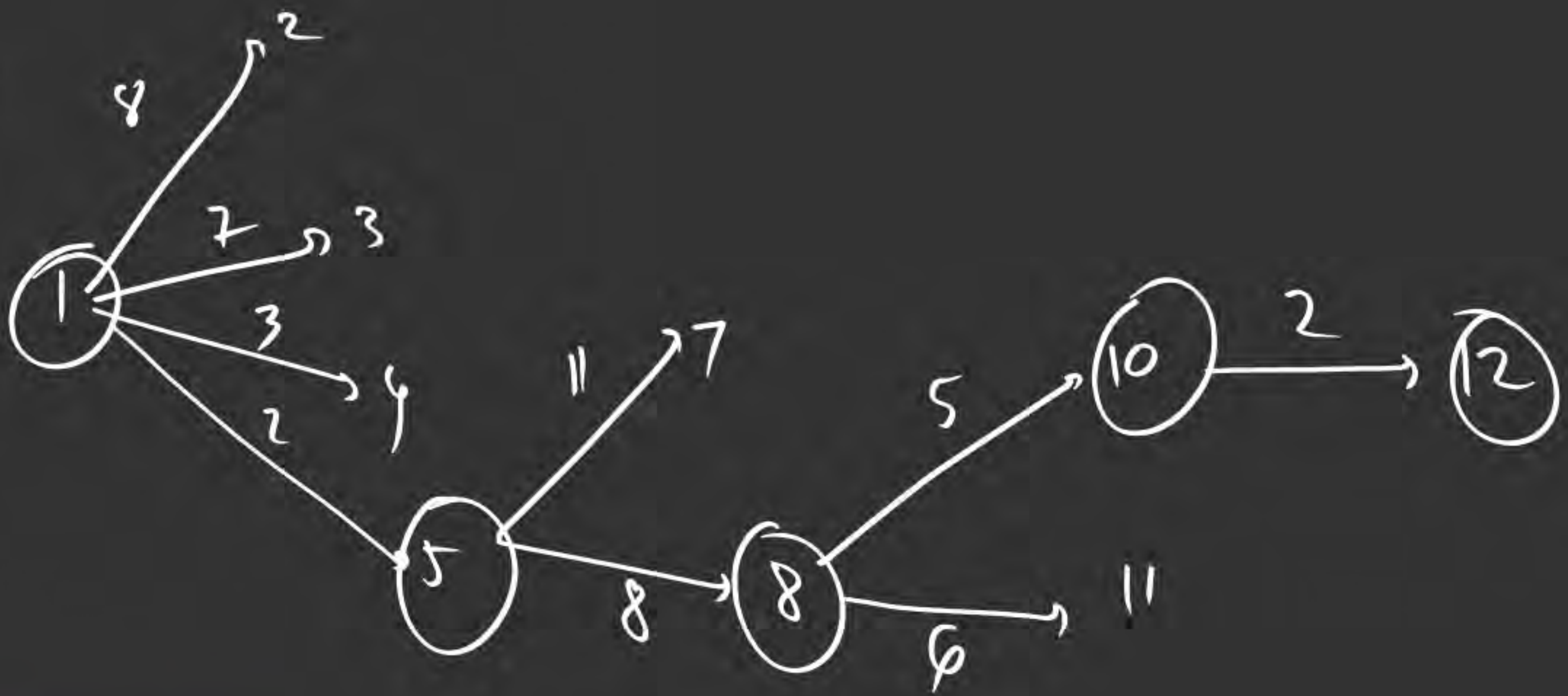


Actual min

$1 \to 2 \to 7 \to 10 \to 12$

$8 + 2 + 3 + 2 = 15$

Greedy:



1 → 12

1 → 5 → 8 → 10 → 12

2 + 8 + 5 + 2 = 17

Is 17 the minimum/optimal cost to $1 \longrightarrow 12$?

Topic : Dynamic Programming (DP)

- Another important feature of D.P is that optimal solutions of the sub problems are retained (cached/ stored in a table) to avoid recomputing their values. (Invariably this feature also leads to saving of time)

V.V. Imp

D.P implémentation → Memoization (Top-Down Approach)

D.P implémentation → Tabulation (Bottom-Up)

Fibonacci

$$F(n) = F(n-1) + F(n-2)$$

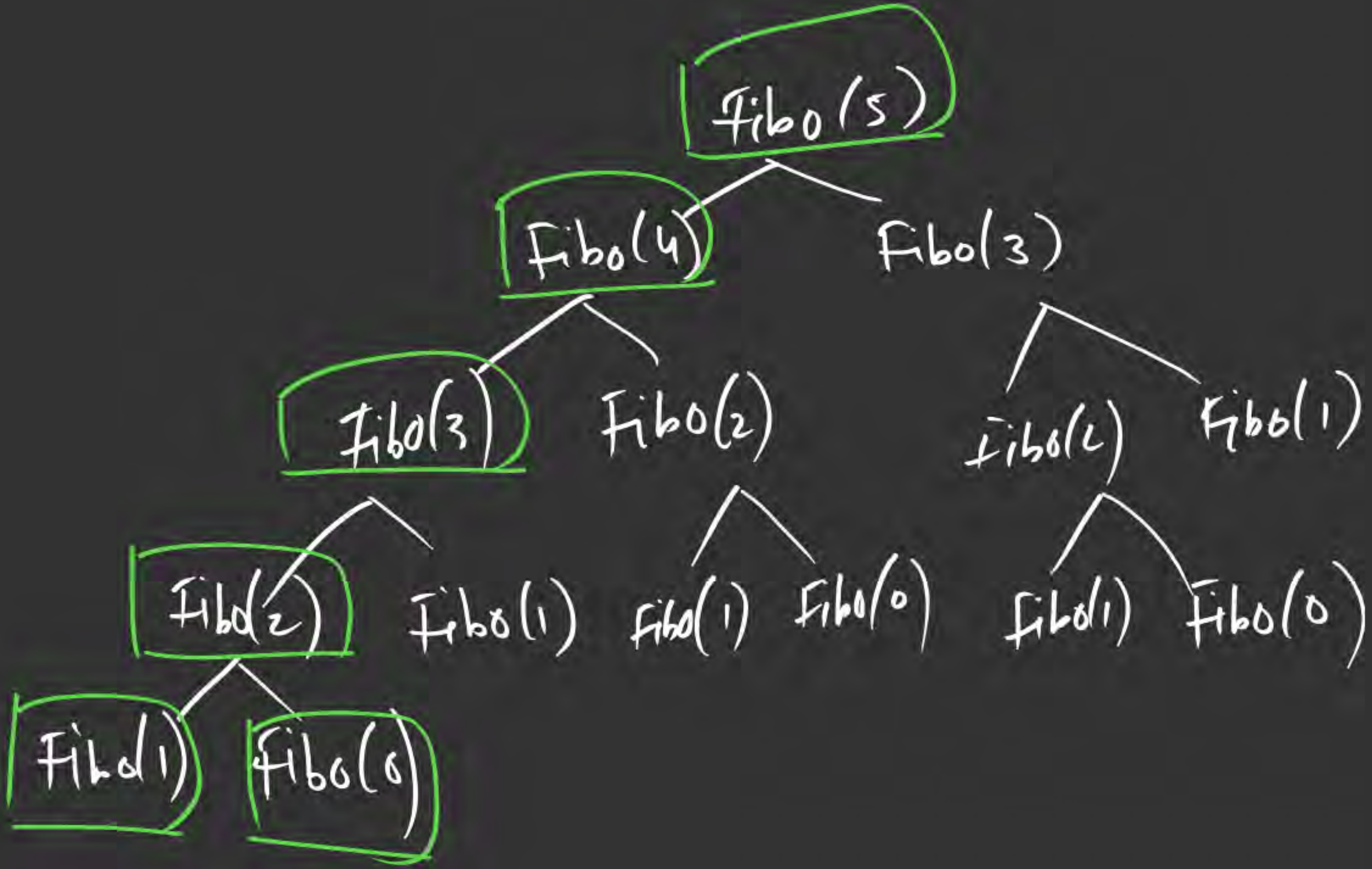
$F(0) = 0$
$F(1) = 1$

Normal Recursive Code

```
Algo fibo(n)
{
    if(n <= 1)
        return n

    else return (fibo(n-1) + fibo(n-2))
}
```

Fibo(5)

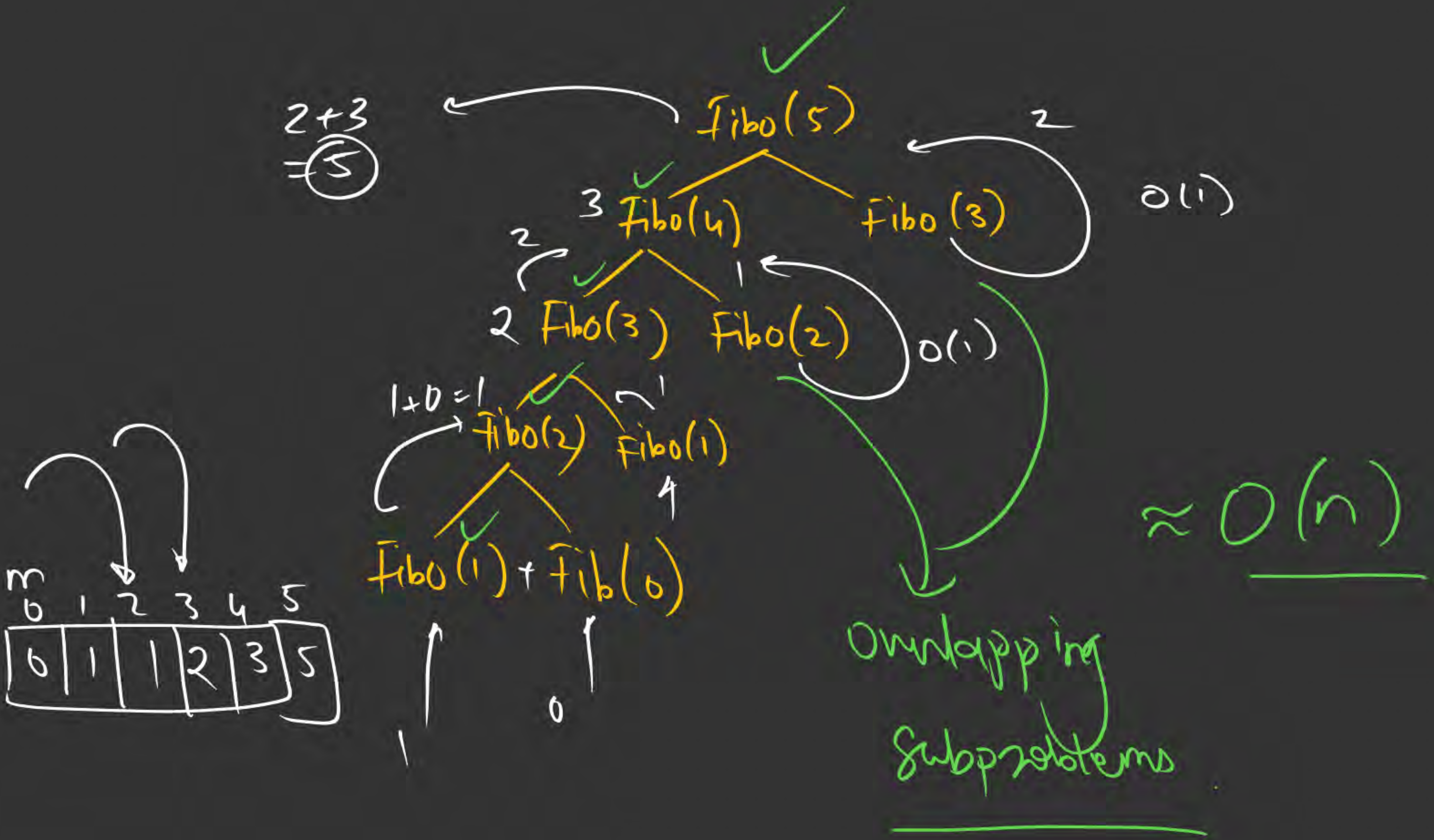


$\approx O(2^n)$

Non - DP

Top-down - Memoized implementation of Fib(n)

```
Algo memiFb(n)
{
    if (m [n] is undefined)
    {
        if (n <=1) result = n;
        else
        result = memFib (n-1) + memFib(n-2);
        m[n] = result; //memorizing (caching)
    }
    return (m[n]);
}
```



2+3
= 5
Fibo(5)
Fibo(4)
Fibo(3)
Fibo(3)
Fibo(2)
1+0=1
Fibo(2)
Fibo(1)
Fibo(1) + Fib(0)
O(1)
O(1)
m
0 1 2 3 4 5
0 1 1 2 3 5
Overlapping subproblems
≈ O(n)

Topic : Dynamic Programming: (DP)

Bottom-up approach of D.P for Fib (n)

```
Algo memFib (n)
{
    M [0] = 0
    M [1] = 1
    for i = 2 to n
    {
        M [i] = M [i-1] + M [i-2];
    }
    return (M[n]);
}
```

$O(n)$

$m[2] = m[1] + m[0] = 1$

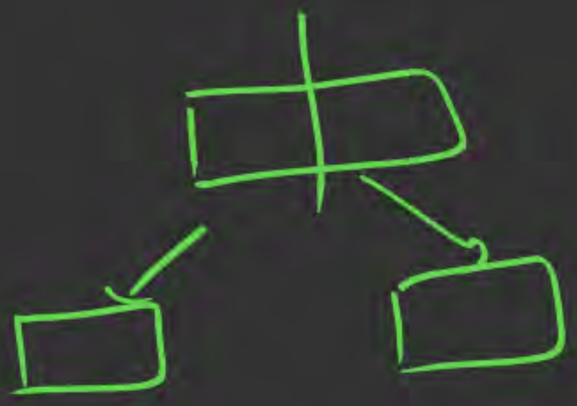
$m[3] = m[2] + m[1]$

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

Dynamic Programming vs Greedy Method vs Divide & Conquer:

- In all methods the problem is divided into subproblem;
- Greedy Method: Building up of the solution to the problem is done in step-wise manner (incrementally) by applying local options only (local optimality).
- Divide & conquer: Breaking up a problem into separate problems (independent), then solve each subproblem separately (i.e. independently) & combine the solution of subproblems to get the solution of original problem.
- Dynamic Programming: Breaking up of a problem into a series of overlapping subproblems & building up solution of larger & larger subproblems.

(global optimality)



Divide & Conquer → independent (non-overlapping) subproblems

Dynamic Programming → Overlapping subproblems

1) Single Source Shortest Paths (SSSP)

1) Dijkstra

2) Bellman Ford

Imp points :-

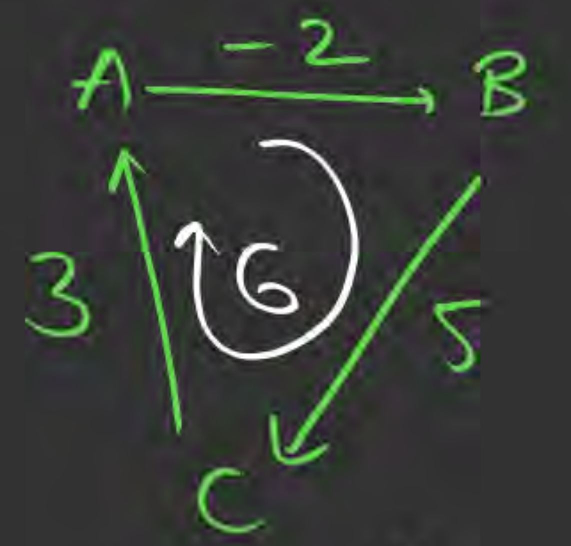
1) G ⇒ every edge +ve wt ⟶ Dijkstra ✓

2) G ⇒ even if one −ve wt edge ⟶ Dijkstra may or may not work.

3) G ⇒ if −ve wt edges are there but no −ve wt cycle reachable from src ⟶ Bellman Ford ✓

4) G ⇒ −ve wt cycle that is reachable from Source ⟶ BF ✗, Dijkstra ✗ } no algo

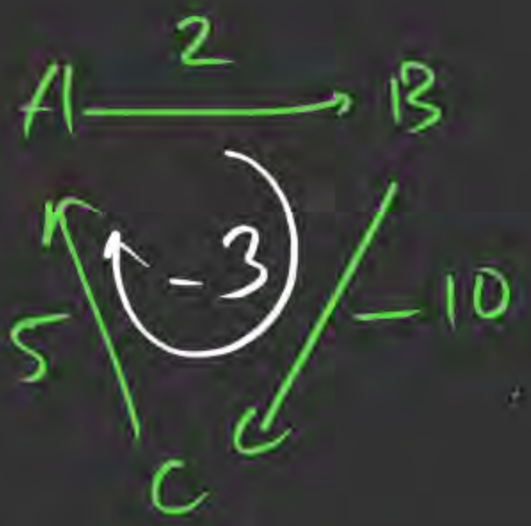
eg:



- ve wt edge ✓
- ve wt cycle ✗

} BF ✓

eg 2)



- ve wt edge ✓
- ve wt cycle ✓

} no algo.

$A \rightarrow B$?

$2 \rightarrow -1 \rightarrow -4$

$\vdots$

$-\infty$

V. Imp   A → Source   Dijkstra SSSP

eg:

V	A	B	C	D	E
{A}	0	5	∞	6	∞
{A,B}	0	5	9	6	?
{A,B,E}	0	5	8	6	2
{A,B,E,D}	0	5	9	6	2
{A,B,E,D,C}	0	5	4	6	2

A

1) A → B ?

A → D → C → B

$6 + (-2) + (-1)$

$= 6 - 3 = \boxed{3}$ ✓

2) A → E ?

A → D → C → B → E

$6 + (-2) + (-1) + (-3) = 0$

V.Imp

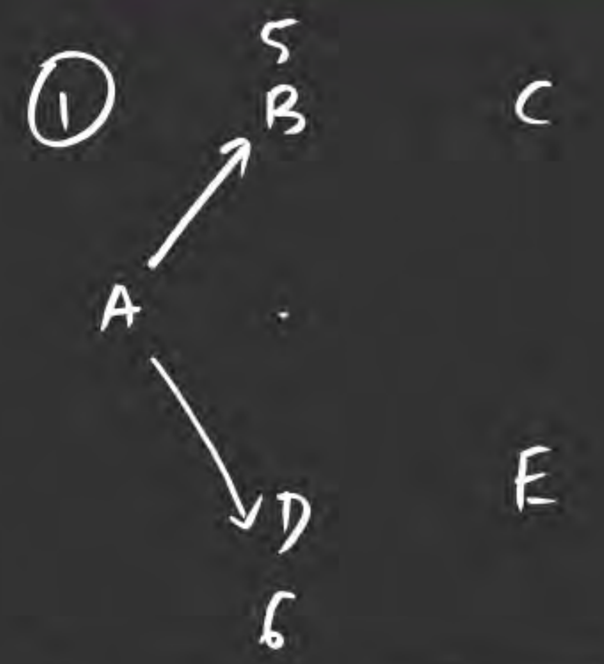
A → Source

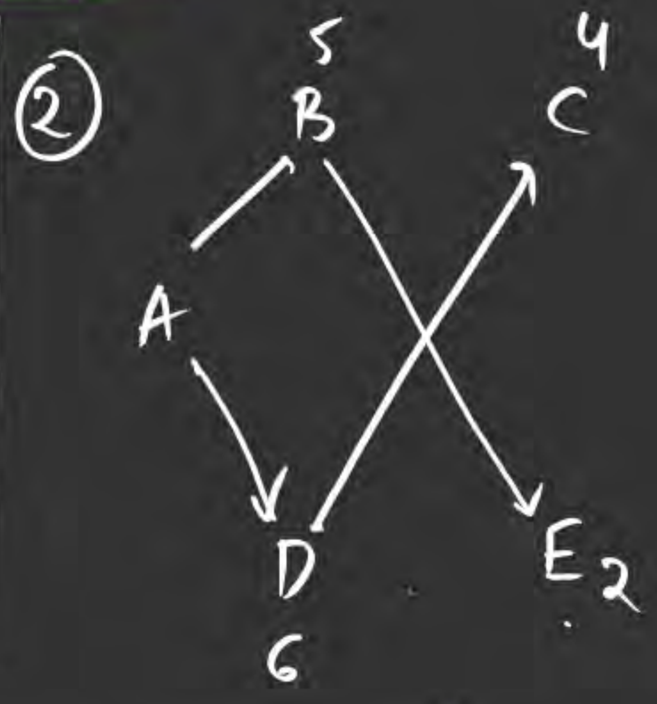
Bellman Ford SSSP

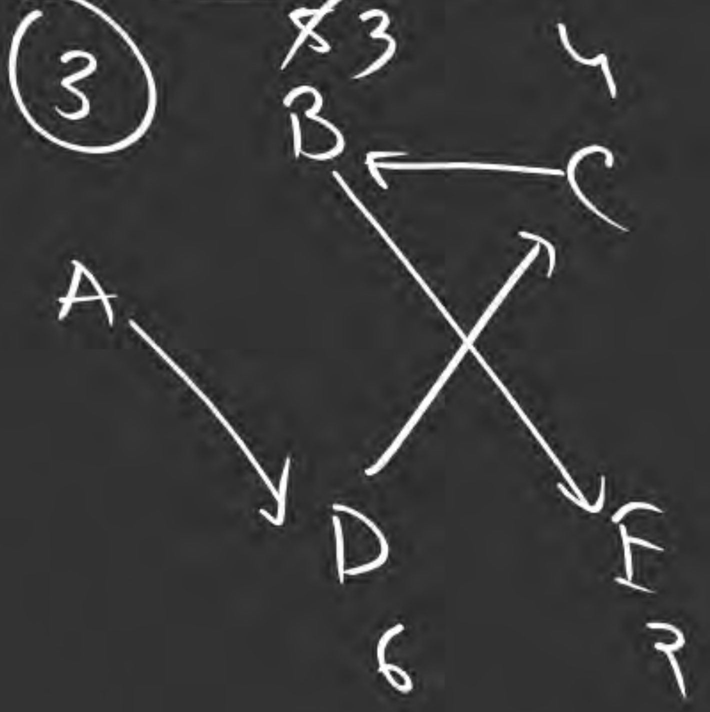
eg:

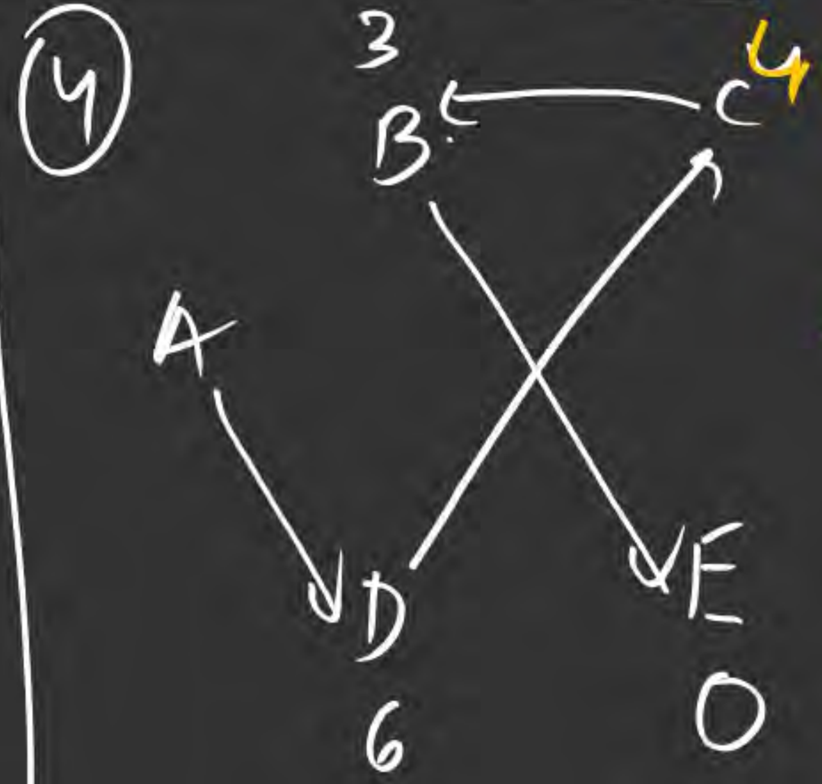
$n = 5$

$n-1 = 4$









final SSSP soln using Bellman Ford

Dijkstra SSSP

$A \longrightarrow B : 5$

$A \longrightarrow C : 4$

$A \longrightarrow D : 6$

$A \longrightarrow E : 2$

Bellman Ford SSSP

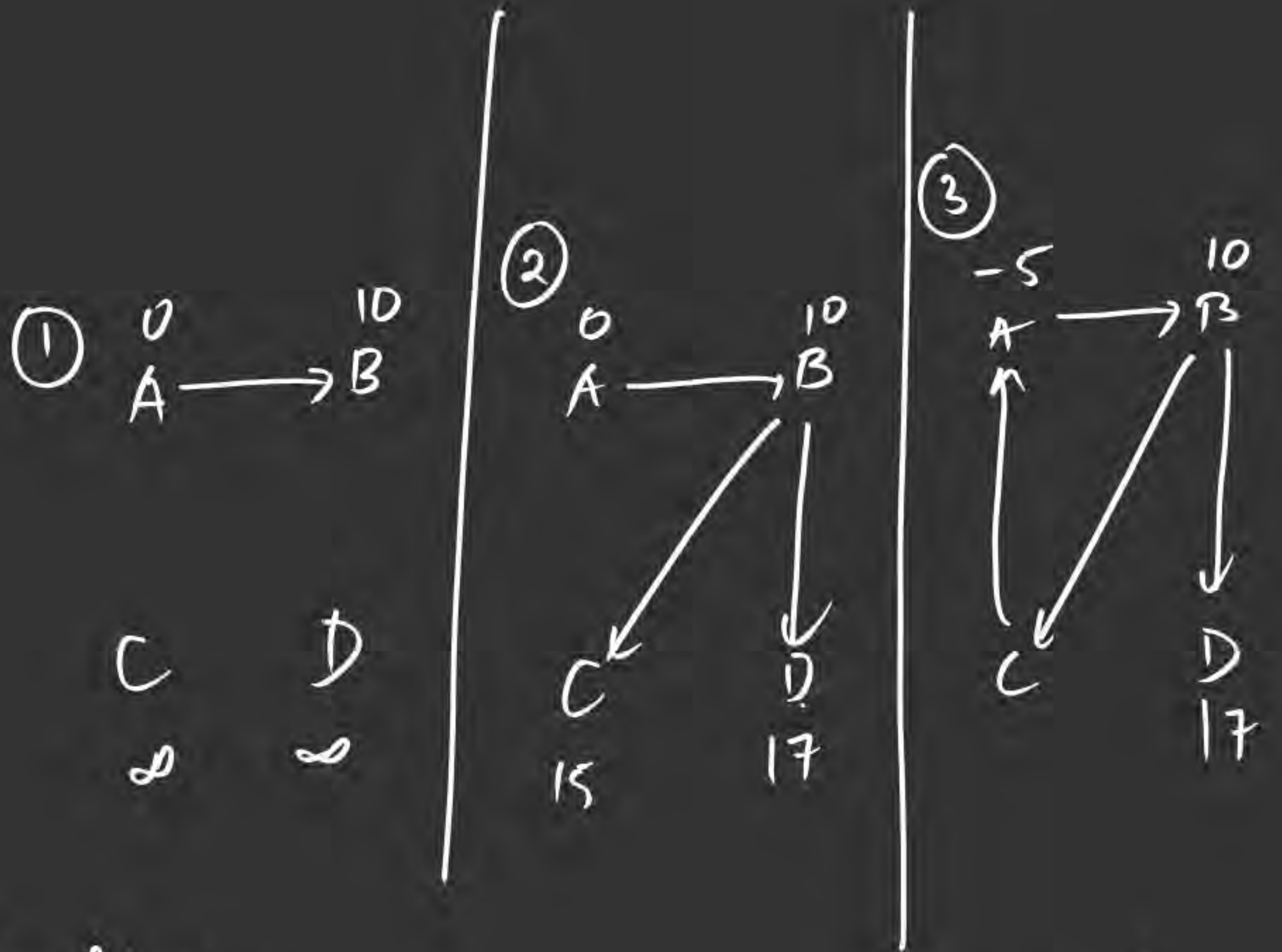
$A \longrightarrow B : 3$

$A \longrightarrow C : 4$

$A \longrightarrow D : 6$

$A \longrightarrow E : 0$

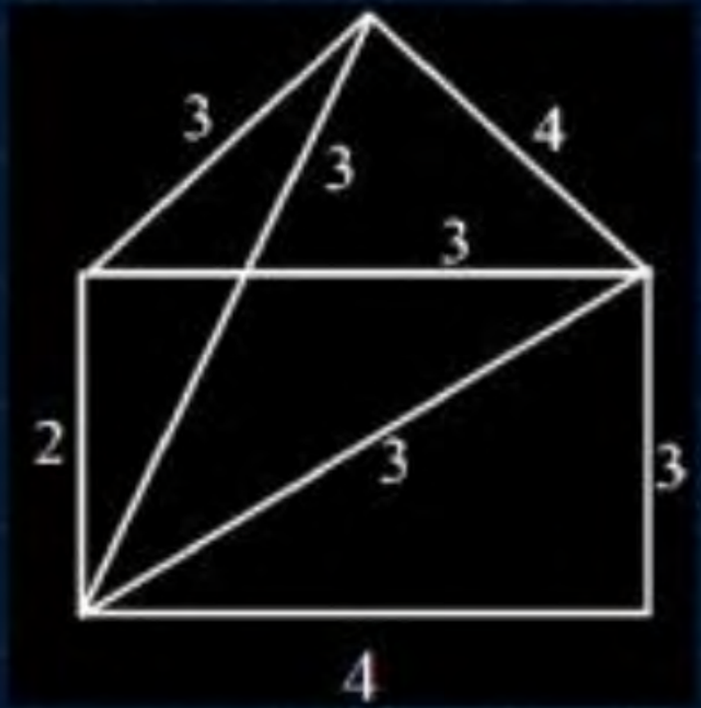
3)



after (n-1) iter, more iterations will result in further relaxation. $\longrightarrow$ -ve wt cycle reachable from Source

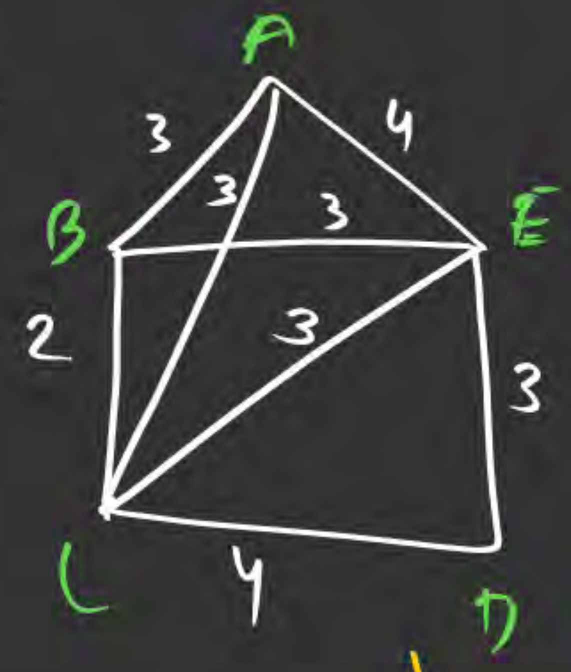
Question

#Q. Consider is the weighted graph G given by

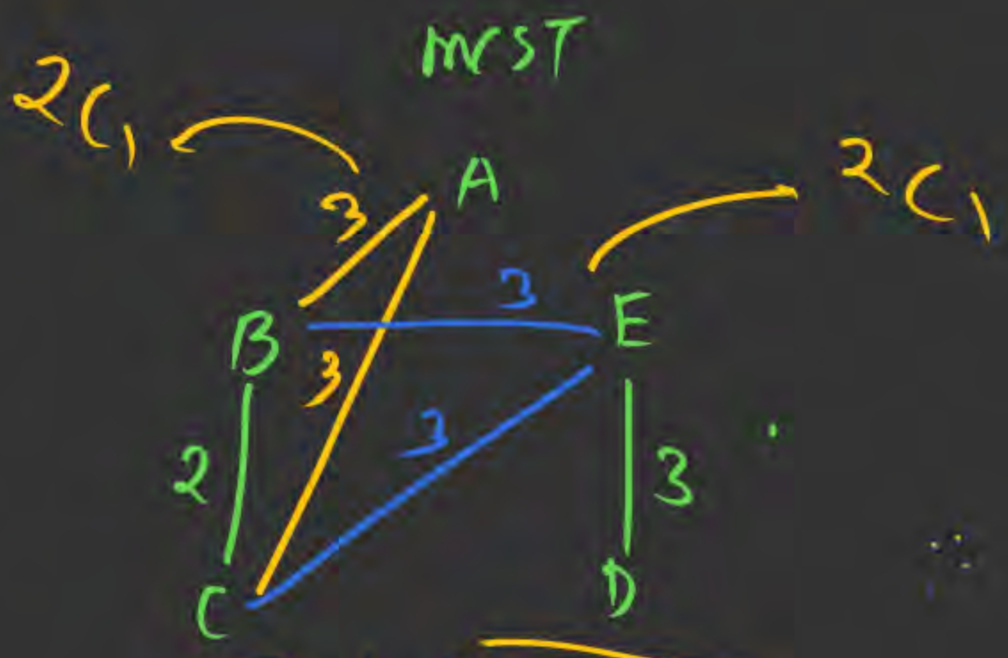


If total MST of G is X and Weight of MST is Y then the value of X + Y is ____.

Soln:-



MST

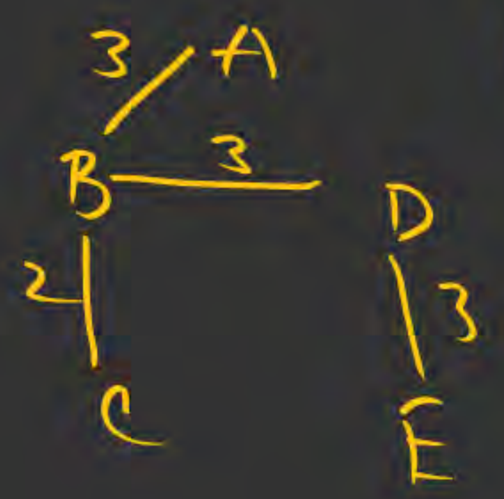


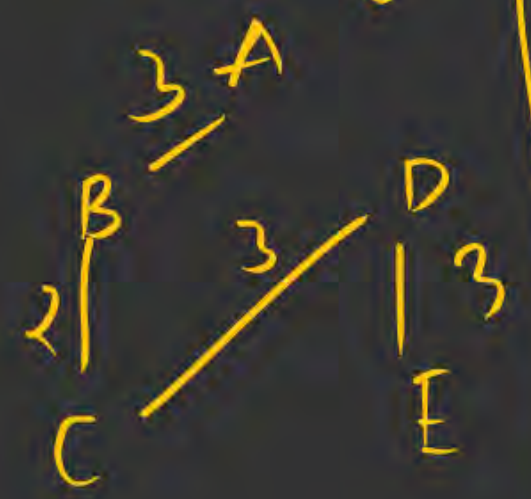
4 MCSTs

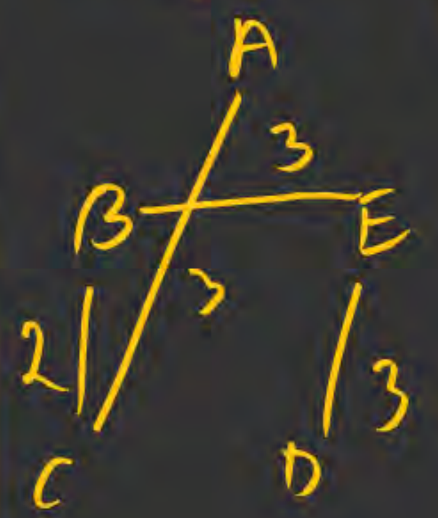
$X = 4$

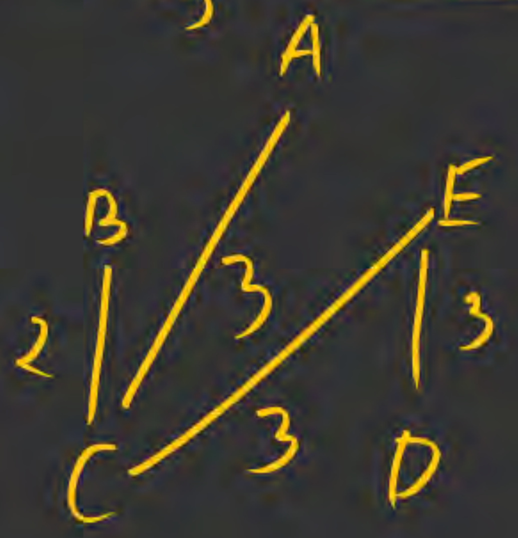
$Y = 3+3+3+2$

$= 11$









$X+Y = 4+11$

$= 15$

Thank

THANK

Keep Hustling!