

## Backtracking Previous Year Question

# Backtracking problem with 2018-19 question : Problem Solving

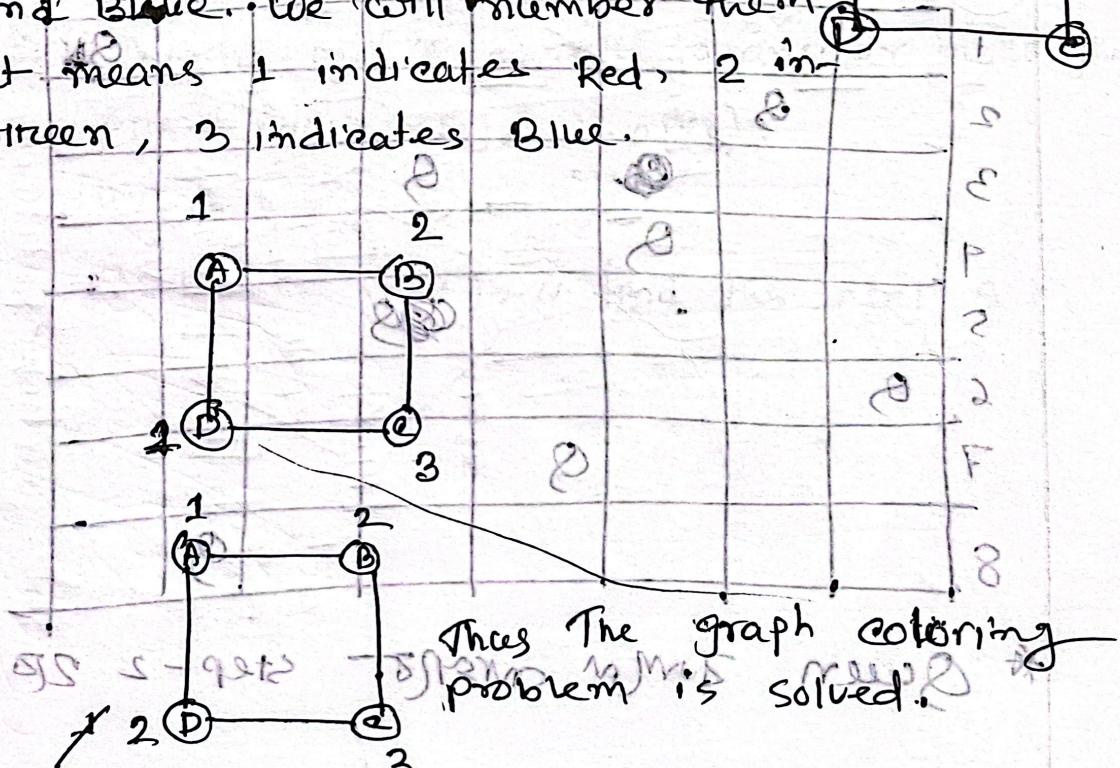
a) Graph coloring problem involves assigning colors to certain elements of a graph. In other words, the process of assigning colors to the vertices such that no two adjacent vertices have the same color is called graph coloring. This is also known as vertex coloring.

If the degree of given graph is  $d$ , then we can color it with  $d+1$  colors. The least number of colors needed to color the graph is called chromatic number.

We can use backtracking approach for assigning different colors to adjacent vertices as follows:-

Step-1: A graph  $G_1$  consists of vertices A, B, C, D. There are three colors used Red, Green and Blue. We will number them out. That means 1 indicates Red, 2 indicates Green, 3 indicates Blue.

Step-2:



Step-3:

Thus The graph coloring problem is solved.

- solving many working problems

b) Backtracking: Backtracking is the most general problem technique to solve a problem or finding the set of solutions of a problem which satisfies some ~~strat~~ constraints.

Principle of Backtracking:

i) find a set of soln from the all feasible soln.

ii) Use recursive calling to find a soln.

iii) Use Brute force approach.

Q) 8-queens problem for a feasible sequence (8, 2, 1, 5, 3).

As the feasible sequence is given, we will place the queens accordingly and then try out the other remaining place.

	1	2	3	4	5	6	7	8
1	Q							Q
2		Q						
3			Q					
4				Q				
5					Q			
6	Q							
7				Q				
8						Q		

\* Queen sum constraint step - 2 20 abr

a) Backtracking: { Backtracking is a general method to finding a set of solutions from the all possible solutions of a problem. It uses recursive calling to solve a problem and finding the solutions.

The general recursive algo for backtracking:—

~~Backtrack()~~

// This is a recursive backtracking algorithm.

//  $a[k]$  is a sol~~ut~~ vector. On entering  $(k-1)$  remaining values can be computed.

//  $T(a_1, a_2, \dots, a_k)$  be the set of all values for  $a_{(i+1)}$  such that  $(a_1, a_2, \dots, a_i)$  is a path to problem state.

{ for (each  $a_k$  that belong to  $T(a_1, a_2, \dots, a_{k-1})$ ) do

{ if  $((a_1, a_2, \dots, a_k)) = \text{true}$ ) then feasible sequence

{ if  $((a_1, a_2, \dots, a_k))$  is a path to answer node then  
print  $(a[1], a[2], \dots, a[k])$ ;

if  $(k < n)$  then

Backtrack  $(k+1)$ ; // find the next set.

}

backtracking  
current node  
out

1) Components of elements from 1 to 30

Elements 1 to 30 have the following components:

- 1st period has 2 elements.
- 2nd period has 8 elements.
- 3rd period has 8 elements.
- 4th period has 18 elements.

The number of components in each period is given below:

Period	Number of Components
1	2
2	8
3	8
4	18

2) Elements of groups 1 to 18

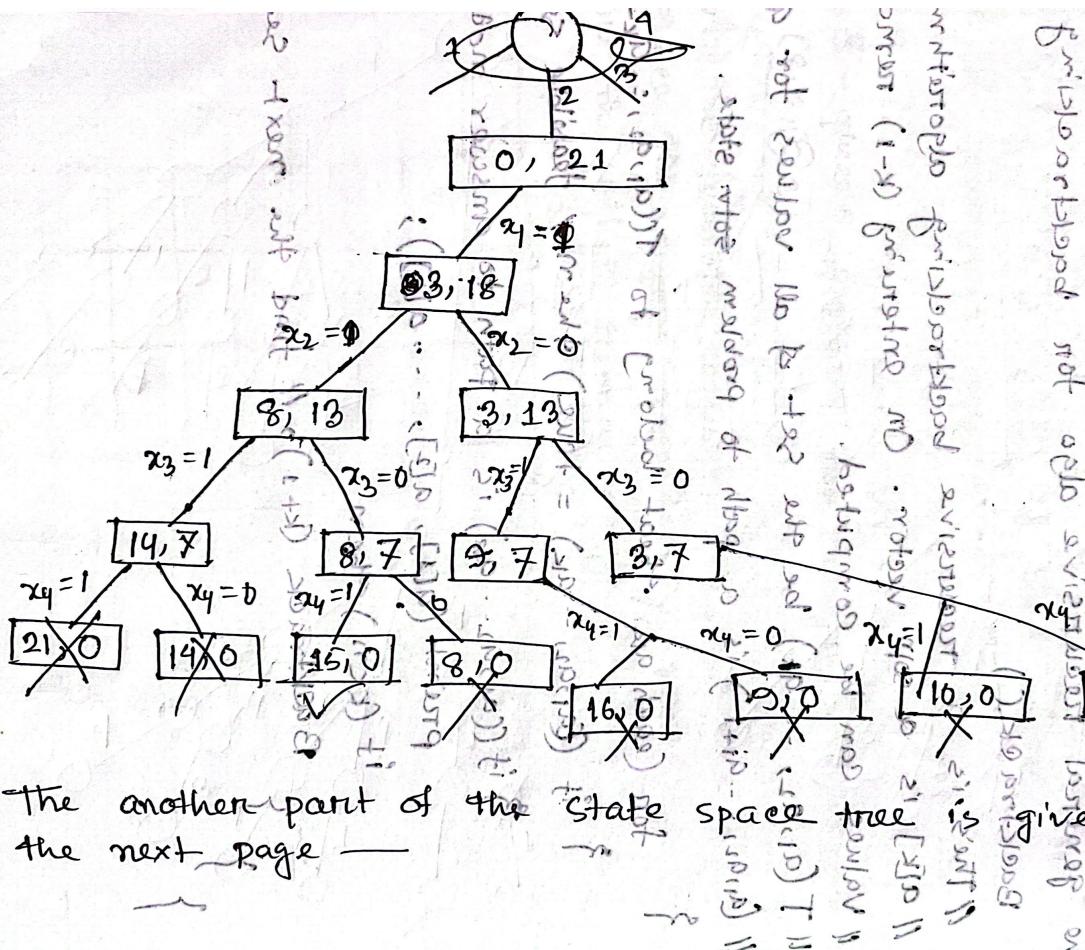
Elements 1 to 18 are divided into groups as follows:

- Group I: Hydrogen
- Group II: Helium
- Groups III, IV, V, VI, VII, VIII: Seven groups of two elements each.

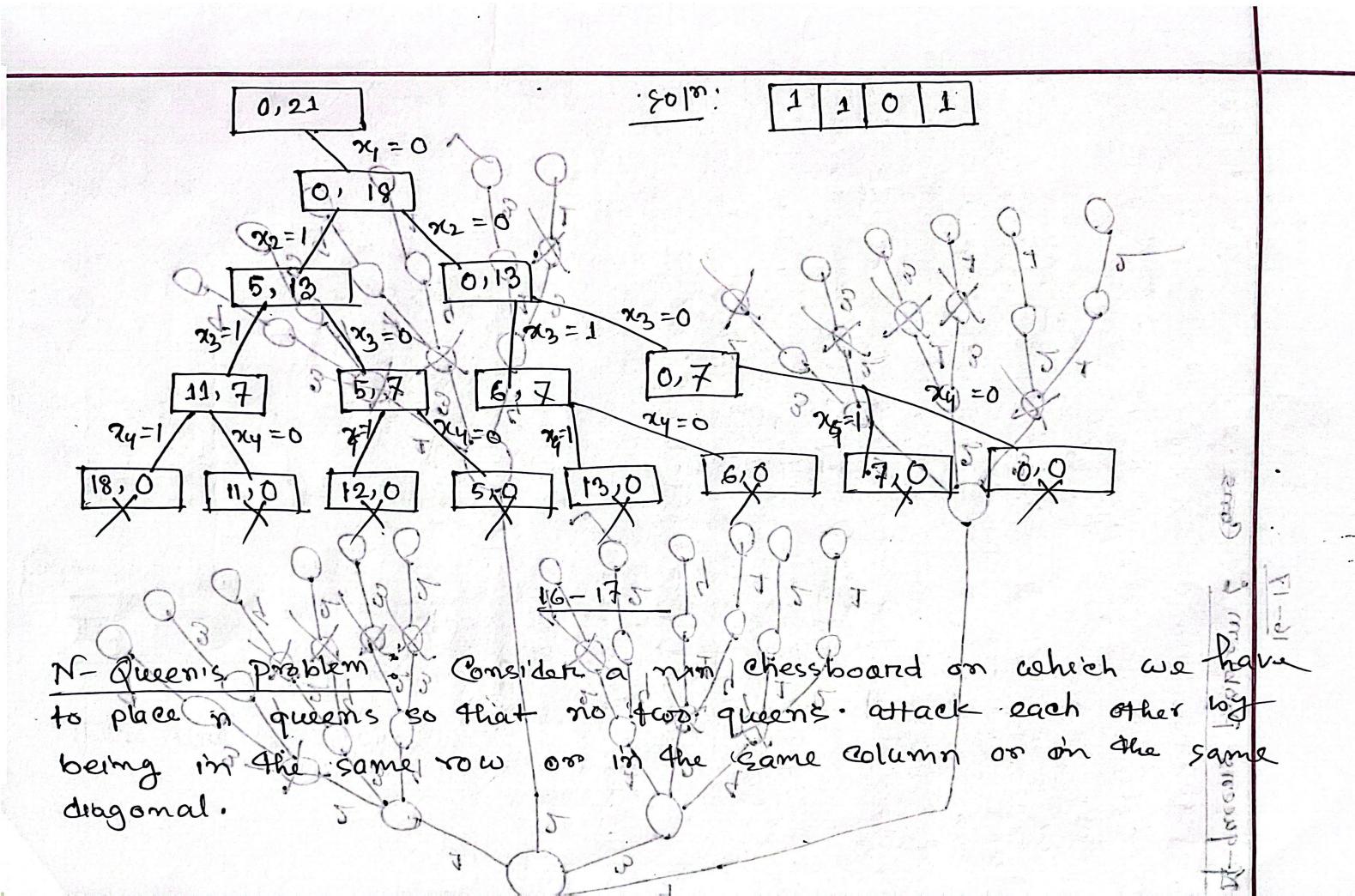
3) Groups of elements

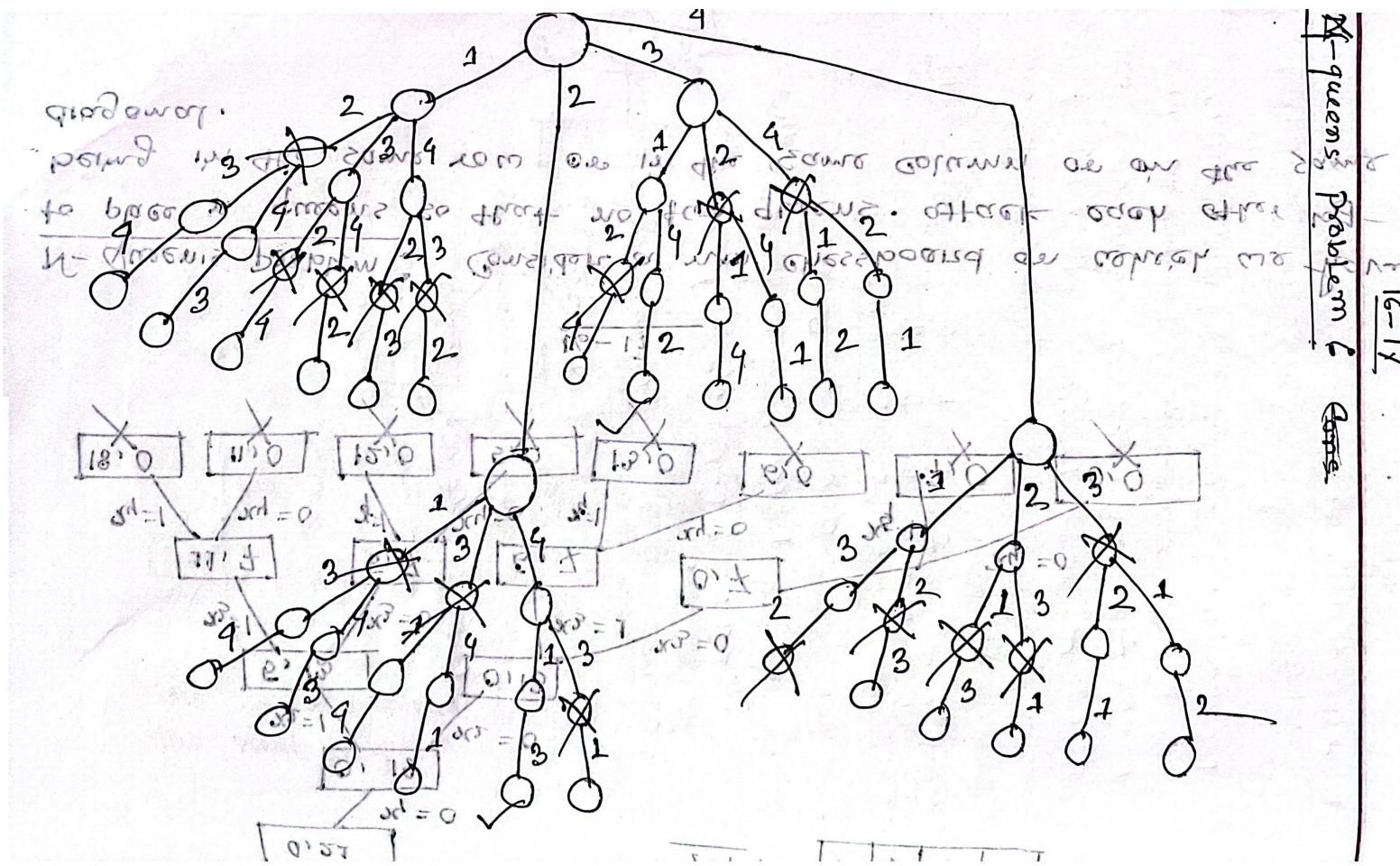
Groups of elements are as follows:

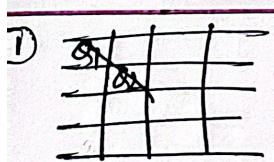
- Group I: Hydrogen
- Group II: Helium
- Group III: Boron, Nitrogen, Phosphorus, Sulfur, Chlorine, Fluorine
- Group IV: Carbon, Silicon, Germanium, Tin, Lead
- Group V: Nitrogen, Phosphorus, Arsenic, Antimony, Bismuth
- Group VI: Oxygen, Sulfur, Selenium, Tellurium, Polonium
- Group VII: Chlorine, Fluorine, Bromine, Iodine, Astatine
- Group VIII: Helium, Neon, Argon, Krypton, Xenon, Radon



The another part of the  
the next page —







A handwritten musical staff consisting of five horizontal lines and four vertical bar lines. In the top left corner, there is a circled '3'. Above the second vertical line from the left, there is a 'B1'. Above the fourth vertical line, there is a 'D2'. Above the fifth vertical line, there is an 'E2'. The notes are represented by short vertical strokes on the lines.

(4)	B1			B2
		B3		
	B4	B5	B6	B7

(5)	81	1	<del>83</del>	82
also	one	c	n.	o.

B)  
81  
81

n. ②	21	81	83
	-	-	-
Tg.	45	83	83
	-	-	-

(10)	
	81
82	
	83
	84

A handwritten musical staff consisting of five horizontal lines. In the top left corner, there is a circled "12". In the top right corner, there is a circled "B1". The staff ends with a curved brace at the bottom right.

✓

(13)				
	7		84	
	-			
	-			
	5			

→ 3 ← (14) → 82 → 09 → 16

81      15      82      83

A handwritten musical staff consisting of five horizontal lines. In the top left corner, there is a circled '16'. Above the staff, there is a short vertical line with a dot above it. To the right of the staff, there are three circled notes: 'B3' at the top, 'B2' in the middle, and 'B1' at the bottom.

F	2	3	4	5
2	3	4	5	6
3	4	5	6	7
4	5	6	7	8
5	6	7	8	9


base  
top

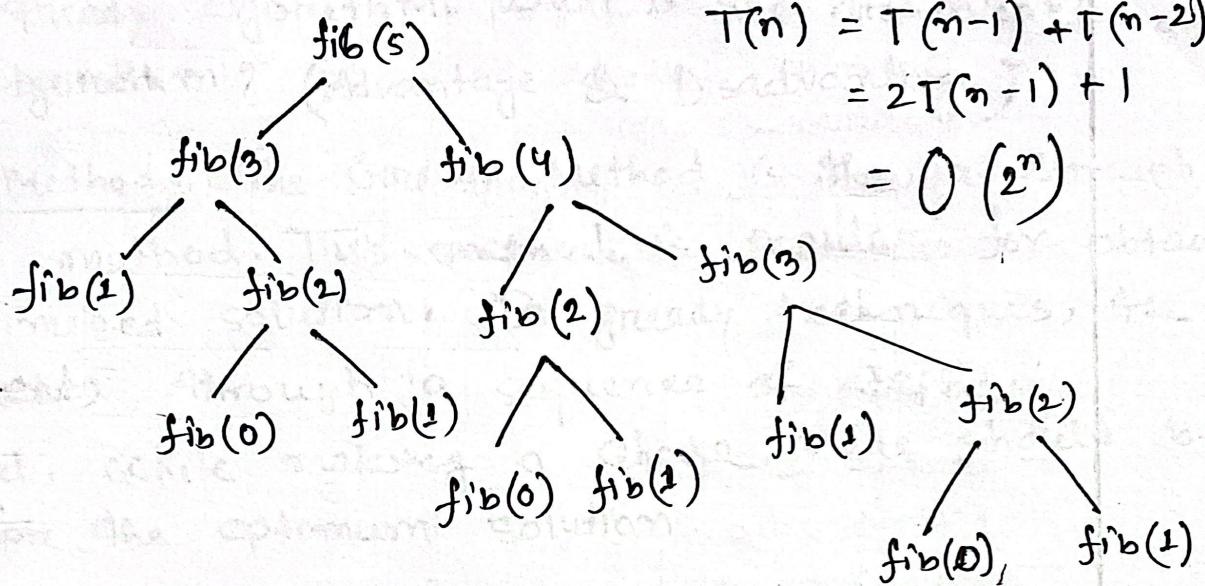
There are two solutions!

2	4	1	3
---	---	---	---

3	1	4	2
---	---	---	---



$\text{fib}(5) = ?$



Time complexity

$$\begin{aligned}T(n) &= T(n-1) + T(n-2) + \\&= 2T(n-1) + 1 \\&= \mathcal{O}(2^n)\end{aligned}$$

## Greedy Method

Q1. What is greedy algorithm? If state the generalized principle of the greedy algorithm. What is the drawback of greedy algorithm? (Advantage & Disadvantage)

Greedy Method: The Greedy Method is the a straight forward method. This method is popular for obtaining the optimized solution. In greedy techniques, the solution is constructed through a sequence of steps. In short, while making a choice there should be greed for the optimum solution.

General Method: Algorithm

Greedy(D,n)

// In Greedy approach D is a domain

// from which soln is to be obtained of size

// Initially assume

solution  $\leftarrow \emptyset$

for i  $\leftarrow 1$  to n do

    if solution + item i is feasible then

        Check if the solution  
        is feasible or not

        Make a  
        feasible  
        and select  
        the optimal  
        soln.

return soln.

General principles of greedy algorithms are given below.

- Activity selection problem → Minimum spanning tree
- Optimal substructure → Greedy Algo.
- Recursive soln → Prim's Algo.
- Greedy choice property → Dijkstra's Algo.
- Not recursive algorithm. direct. bottom-up approach
- Greedy algorithm is not suitable for problems involving overlapping subproblems.

Applications to some problems of computer science

- Knapsack Algo. → forward states, backtracking
- Prim's Algo., mst/mcs mst/mcs w.r.t not best.
- Kruskal's Algo.
- finding shortest path → Dijkstra's Algo.
- Job Sequencing w.r.t deadlines.
- Optimal storage with tapes.

to benefit from it in most cases.

Advantage:

- Easier to implement.
- Much faster to execute.
- Used to finding optimum solution.

Disadvantage / Drawback

- Not always find the optimal soln.
- It may not produce the best possible outcome.
- It is very difficult to understand.

Most important

## Step's of Greedy methods :

1. Select some soln from input domain.
2. Check the soln is feasible or not.
3. from a set of feasible soln — particular soln that satisfy or nearly satisfy to the object is called the optimal soln.
4. As greedy method work in stage. At each stage only one input is considered at each time. Based on the's input it is decided whether particular soln gives the optimal soln or not.

Optimization problem : Optimization problem is to find out either min or max result. To solve optimization problem , there are several method , like —

① Greedy Method ② Dynamic programming ③ Branch-N-

16.2.1 : Suppose that instead of always selecting the first activity to finish. We instead ~~not~~ select the last activity to start that is compatible with all previously selected activities. Describe how this approach is greedy algorithm and prove that it yields an optimal soln.

Knapsack Problem: Suppose there are  $n$  objects from  $i = 1, 2, 3, \dots, n$ . Each object have some positive weight  $w_i$  and some profit associated with the object which denotes as  $P_i$ . The knapsack carry out the most weight

$$W_{\text{max}} = W$$

\* Chose only those object which gives maximum profit.

\* Total weight of object should be  $\leq W$ ; then we can obtained feasible soln -

$$\begin{aligned} \text{maximized } & \sum P_i x_i \\ \text{subject to } & \sum w_i x_i \leq W; \\ & 0 \leq x_i \leq 1 \end{aligned}$$

\* Fractional :  $m = 15$

Object :  $\frac{1}{12}, \frac{2}{5}, \frac{3}{16}, \frac{4}{7}, \frac{5}{9}$

Profit :  $12, 5, 16, 7, 9$

Weight :  $3, 1, 4, 2, 3$

$$\max \left( \frac{P}{w} \right) = 4, 5, 4, 3.5, 1, 2.75, 2$$

Object	Profit	Weight	Remaining weight
2	5	1	$15 - 1 = 14$
1	12	3	$14 - 3 = 11$
3	16	4	$11 - 4 = 7$ maximum
4	$7 \left( \frac{10}{8} \times 2 \right)$	2	$7 - 2 = 5$
5	11	4	$5 - 4 = 1$
7	$(6/3) = 2$	1	$1 - 1 = 0$

$N = 28$ ,  $M = 110$  event 2099923 : molded > 100gms  
 $P_f = \{11, 21, 31, 33, 43, 53, 55, 65\}$   
 $W = \{1, 1, 1, 21, 23, 33, 43, 45, 55\}$  before filtering  
 $\frac{P}{W} = \underline{11}, 1.9, 1.47, 1.43, 1.303, 1.23, 1.22, 1.18$

now we have to calculate remaining weight

Item	Profit	Weight	Remaining Weight
1	11	11	$110 - 11 = 109$
2	21	11	$109 - 11 = 98$
3	31	21	$98 - 21 = 77$
4	33	23	$77 - 23 = 54$
5	43	21	$54 - 21 = 33$
6	53	21	$21 - 21 = 0$
			$= \left(\frac{w}{W}\right) \times M$

43 (maximum profit)  
21 (maximum weight)  
 $\frac{21}{43}$

$$\text{Maximum Profit} = 11 + 21 + 31 + 33 + 43 + \left(53 \times \frac{21}{43}\right)$$

$$F = 1 - P$$

$$P = 1 - F$$

$$P = 1 - \frac{21}{43}$$

$$P = \frac{22}{43}$$

$$P = 0.511$$

state 0/1 knapsack problem:

The 0/1 knapsack problem is a classic optimization problem where a set of items with given weights and profits are to be packed in a knapsack with a limited capacity. It is called 0/1 knapsack because if the weight count then the fill with 1 otherwise 0. So, it is 0/1 knapsack.

Given the 0/n knapsack prob.  $\begin{cases} \max = 5 \\ n = 4 \end{cases}$

Item : 1 2 3 4

Weight: 3 2 5 6

Profit: 4 3 5 6

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

→ weight

Item	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	0	0	3	4	4	5	6	7
3	0	0	3	4	4	5	6	7
4	0	0	3	4	4	5	6	7
5	0	0	25	25	25	25	25	25

DP =  $2 + 2 + 25 : \text{fibonacci M}$

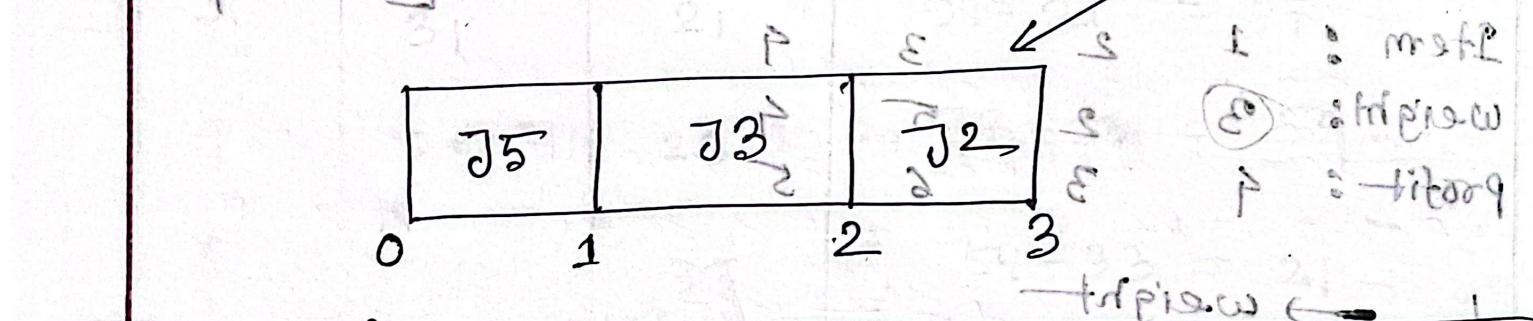
## Job sequencing problem

Given 5 jobs with 3 machines. Deadlines: 1/5, 2/5, 3/5  
 Machine 1 profit: 10, 15, 20, 5, 10. Machine 2 profit: 15, 20, 10, 10, 15.

Machine 3 profit: 5, 10, 15, 20, 10. Deadlines: 1/5, 2/5, 3/5.  
 Job 1: Profit 10, Deadline 1/5. Job 2: Profit 15, Deadline 2/5. Job 3: Profit 20, Deadline 3/5. Job 4: Profit 5, Deadline 1/5. Job 5: Profit 10, Deadline 1/5.

Deadline: 1/5, 2/5, 3/5. Machine 1 profit: 10, 15, 20, 5, 10. Machine 2 profit: 15, 20, 10, 10, 15. Machine 3 profit: 5, 10, 15, 20, 10.

Deadlines: 1/5, 2/5, 3/5. Machine 1 profit: 10, 15, 20, 5, 10. Machine 2 profit: 15, 20, 10, 10, 15. Machine 3 profit: 5, 10, 15, 20, 10.



Job id	Profit	Deadline	Slot assign
J3	20	2	[1, 2]
J5	15	2	[0, 1]
J1	10	1	Rejected
J2	5	3	[2, 3]
J4	1	3	Rejected

Solution: J3, J5, J2  
 Max profit:  $20 + 15 + 5 = 40$

Job id: J1 to J20  
 profit: 30, 25, 20, 15, 18, 12, 10, 8, 6, 4, 3, 2, 1, 0, 9, 7, 5, 3, 2, 1  
 deadlines: 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, 9, 8, 7, 6, 5, 4, 3, 2, 1

Q find the max profit from max deadlines.

Sol: Sort the jobs based on profit and then

Job id	Profit	Deadline
J1	30	10
J2	25	9
J3	20	8
J4	18	7
J5	15	6
J6	12	5
J7	10	4
J8	8	3
J9	6	2
J10	4	1
J11	3	0

deadlines: 10, 9, 8, 7, 6, 5, 4, 3, 2, 1

Job id	Profit	Deadline	Job id	Profit	Deadline
J1	30	10	J2	25	9
J3	20	8	J4	18	7
J5	15	6	J6	12	5
J7	10	4	J8	8	3
J9	6	2	J10	4	1
J11	3	0			

Job id	Profit	Deadline	Eligible assignment of false
J5	30	8	[7, 8]
J10	25	5	[4, 5]
J1	20	4	[3, 4]
J8	18	3	[2, 3]

Max profit = 126

Rejected: J6, J4, J7, J2, J3

\* Explain how job sequencing with deadline problems can be solved using the greedy approach

- Sort the jobs in decreasing order of their profit.
- find the highest deadlines and draw a gantt chart up to that deadline.

- Now we need to assign time slots to individual job id.

- Now - pick each job one by one and check if the max possible time slot for the job i.e its deadline is assigned to another job or not. if it is not filled yet, assign the slot to the current job.

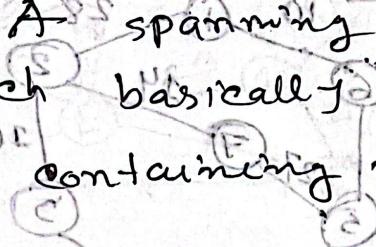
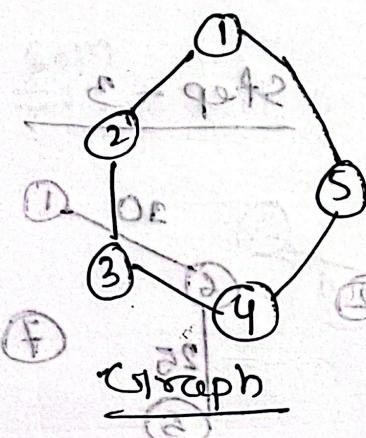
- Otherwise, search for any empty time slot less than the deadline of the current job if such a slot is found, assign it to the current job id and move to the next job.

- Continue this process until all the feasible jobs are allocated to their time slot.

- In the end, we can calculate the profit for all allocated feasible jobs.

## Minimum Cost Spanning Tree

Spanning tree: A spanning tree is a subgraph of a graph  $G$  which basically is a tree that contains all the vertices containing no circle.



1 - qet 2

Spanning tree.

For single cycle spanning tree  
for multistage  $n = 10$

$$= n c_{n-1}$$

Total edge

$|V-1|$   
no. of cycles

$$= \sum c_{n-1}$$

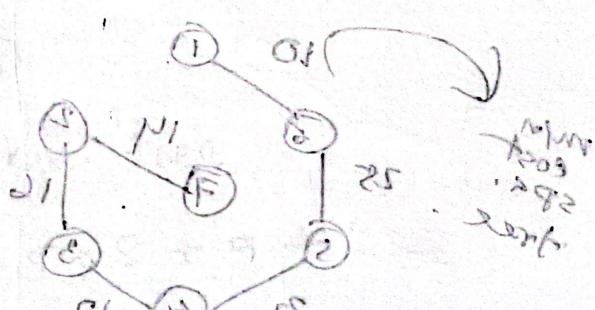
Amina Khatun  
Dept. of ICE, PUST

$$55 + 25 + 01 = 0$$

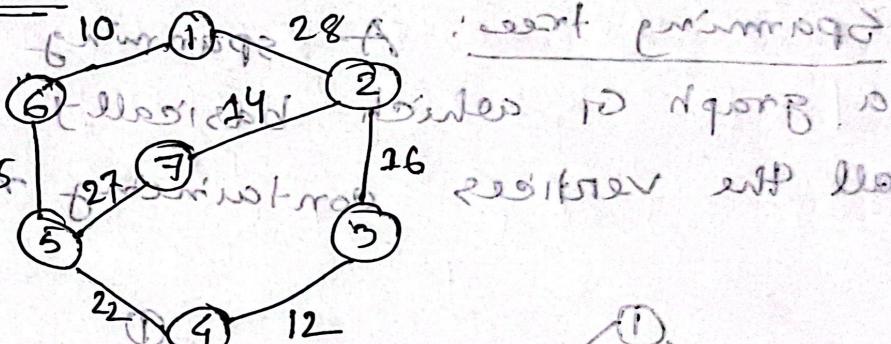
$$F2 =$$

$$P1 + 1 + 11 + 15 + 25 + 01 =$$

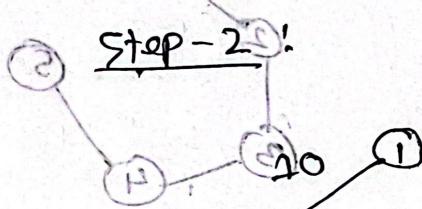
$$PP =$$



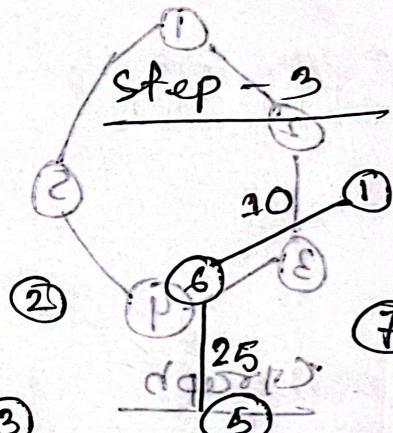
## Prim's Algorithm with Priority Queue Using Min-Heap



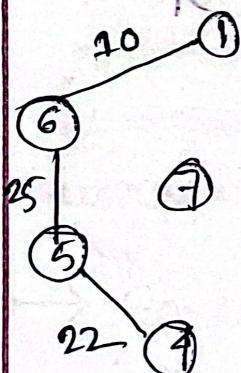
### Step - 1 :



### Step - 3

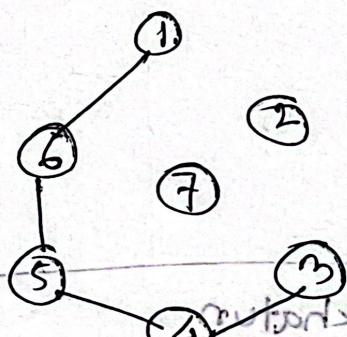


step-4

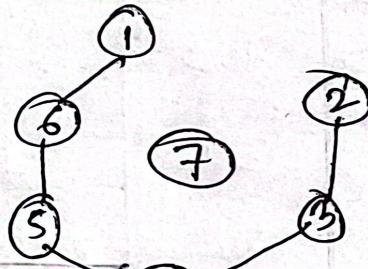


$$W = 10 + 25 \neq 22 \\ \equiv 57$$

Step-S



## Step - 6



## Step-7

A hand-drawn graph diagram showing nodes 1 through 9 connected by edges with weights. The graph consists of the following edges and weights:

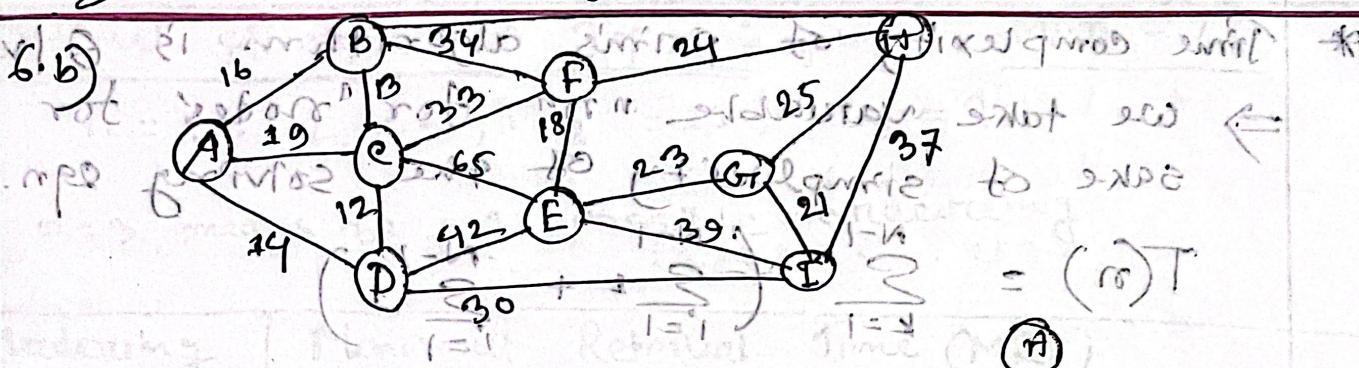
- Edge (1, 6) with weight 10
- Edge (2, 3) with weight 16
- Edge (2, 7) with weight 14
- Edge (3, 9) with weight 12
- Edge (4, 9) with weight 8
- Edge (5, 6) with weight 25
- Edge (5, 7) with weight 7
- Edge (6, 7) with weight 10
- Edge (7, 9) with weight 11

Handwritten note on the left: "min cost spa agree".

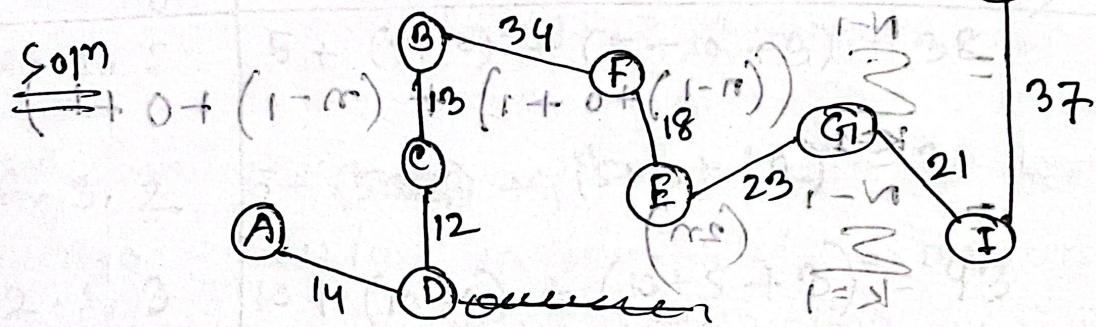
$$\begin{array}{l}
 \text{Min Cost} \\
 \cancel{\text{Selling price}} \\
 = 10 + 25 + 12 + \\
 = 47
 \end{array}$$

17-18

36



$$= (r)T$$



Minimum Spanning Tree.

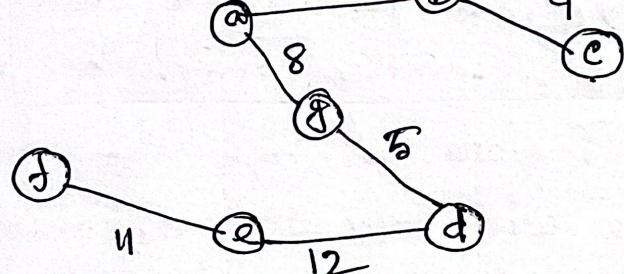
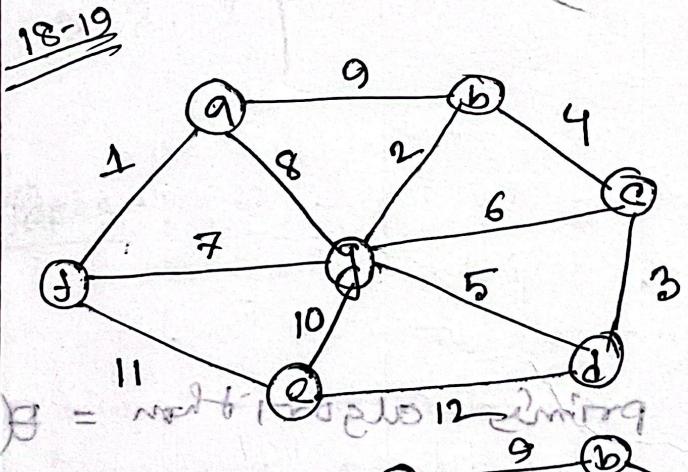
$$\text{Min cost} = 14 + 12 + 13 + 34 + \underbrace{(1-m)18}_{(1-m)} + 23 + 21 + 37$$

$$(1-m) mS =$$

$$(mS - mS) =$$

$$m = (r)T$$

$$(m) \theta = (r)T$$



Maximum spa. tree

$$\text{Max cost} = 11 + 12 + 5 + 8 + 9 + 4 =$$

28

\* Time complexity of prim's algorithm is  $\Theta(n^2)$

$\Rightarrow$  we take variable "n" for "nodes" for the sake of simplicity of the solving eqn. Then

$$T(n) = \sum_{k=1}^{N-1} \left( \sum_{i=1}^{N-1} + \sum_{i=1}^{N-1} \right)$$

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

$$= \sum_{k=1}^{N-1} (2n)$$

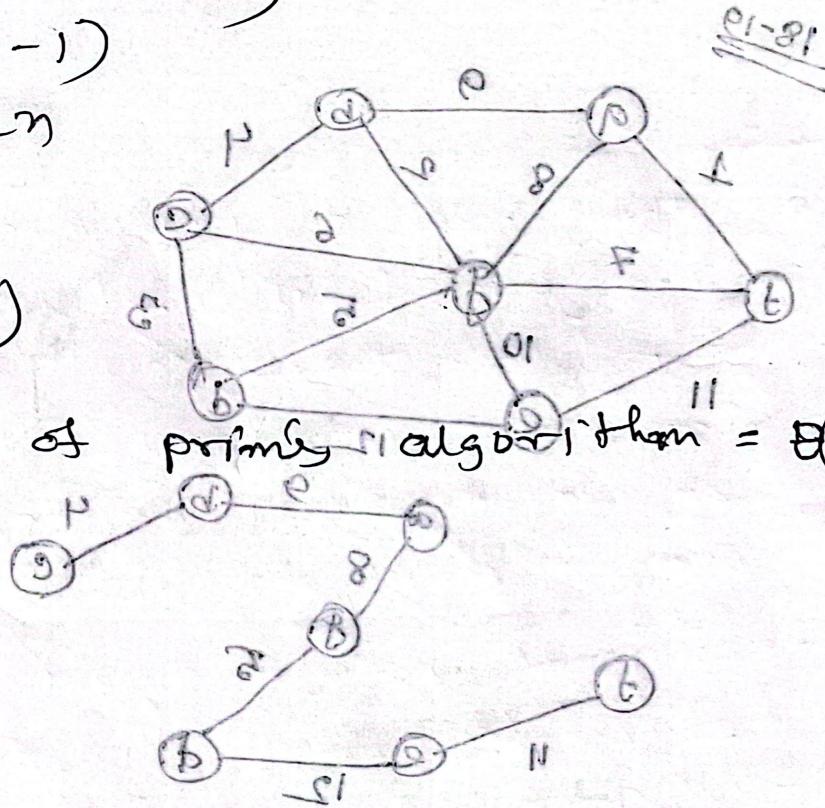
$\therefore T(n) = 2n \sum_{k=1}^{N-1} 1$

$$T(n) = 2n(n-1) = 2n^2 - 2n$$

$$T(n) = n^2$$

$$T(n) = \Theta(n^2)$$

The complexity of prim's algorithm =  $\Theta(n^2)$



gymnasiums Simons (1)

Optimal storage on Tapes  $\rightarrow$  SMRT  $\leftarrow$

$$n = 3, L_1 = 5, L_2 = 10, L_3 = 3$$

$n=3$  means  $3! = 6$  possible ordering

Ordering	Minimun Retrieval Time (MRT)
1, 2, 3	$5 + (5+10) + (5+10+3) = 38$
1, 3, 2	$5 + (5+10) + (5+3+10) = 31$
2, 1, 3	$10 + (10+5) + (10+5+3) = 43$
2, 3, 1	$10 + (10+3) + (10+3+5) = 32$
3, 1, 2	$10 + (10+5) + (10+5+3) = 43$
3, 2, 1	$10 + (10+3) + (10+3+5) = 32$

PC neben PW ↙

# Dynamic Programming

→ Dynamic programming is a technique for solving a problem with overlapping subproblems.  
In this method, each subproblem is solved only once. The result of each subproblem is recorded in a table which we can obtain soln to original subproblems.

→ General procedure of dynamic programming.

- ① Characterize the structure value of optimal soln.
- ② Recursive define the value of optimal soln.
- ③ Compute the value of optimal soln typically bottom up fashion.
- ④ Compute the optimal soln from the computed information.

→ Why needed DP?

The dynamic programming is used where we have problems, which can be divided into similar subproblems, so that their result can be reused. Mostly the algorithm used for solved optimization problems. Optimization problems means that we are trying to find out the min or max soln of the problem. Before solving the in hand subproblem, the algo will try to examine the result of the previously solved subproblems.

\* State the principle of optimality. 20192 100 marks

- The dynamic programming algo obtain the soln using principle of optimality.
- The principle of optimality states that, In an optimal sequence of decision or choice, each subsequence must also be optimal.

$$K(n) = (1-m) \text{bit} + (c-m) \text{bit}$$

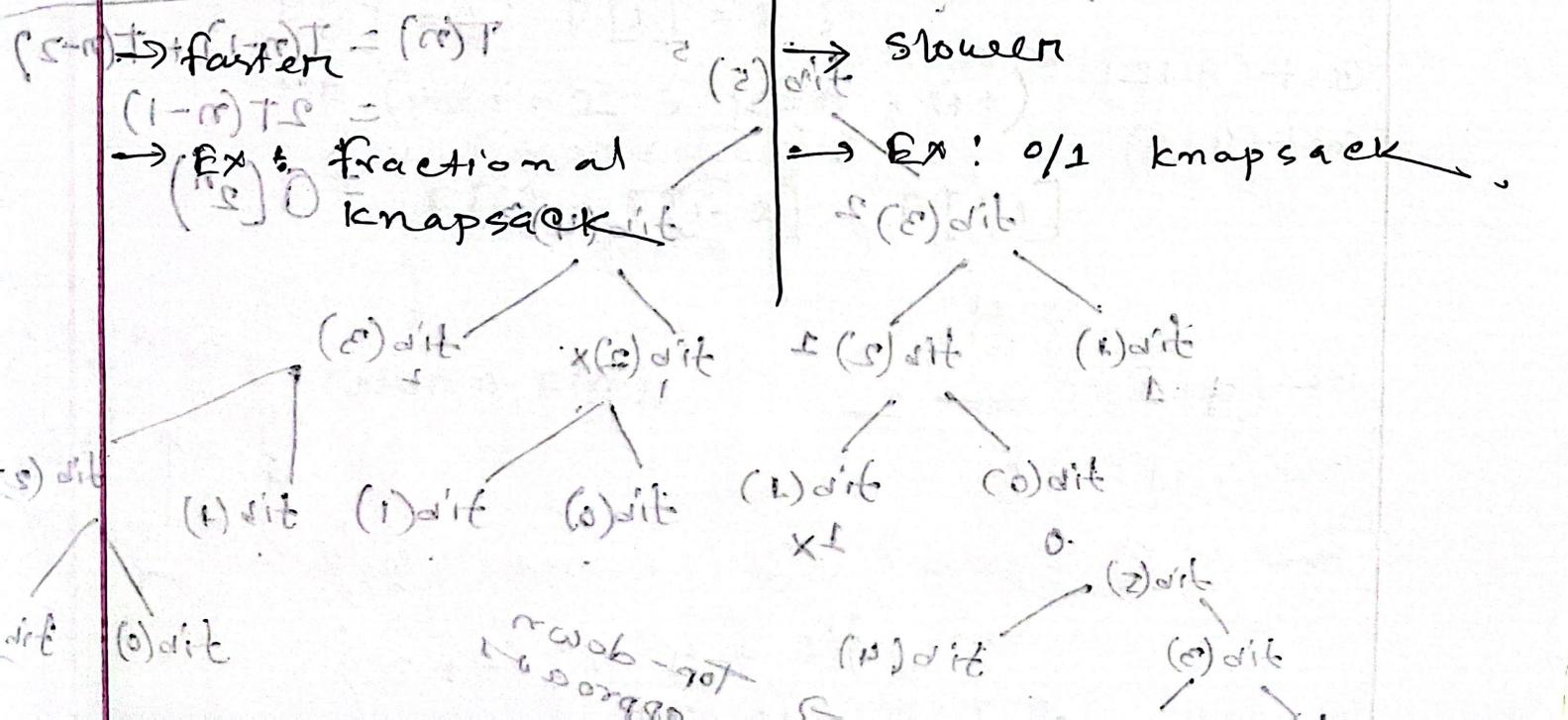
\* Greedy                          (n+m) Dynamic

- |  |  |
|--|--|
| <ul style="list-style-type: none"> <li>→ used for obtaining optimal soln.</li> <li>→ It has a set of feasible soln and picks up optim soln.</li> <li>→ There is no guarantee of getting optimal soln.</li> </ul> | <ul style="list-style-type: none"> <li>→ same <math>\Rightarrow m</math> bit</li> <li>→ No feasible set of soln.</li> <li>→ Guaranteed.</li> </ul> |
|--|--|

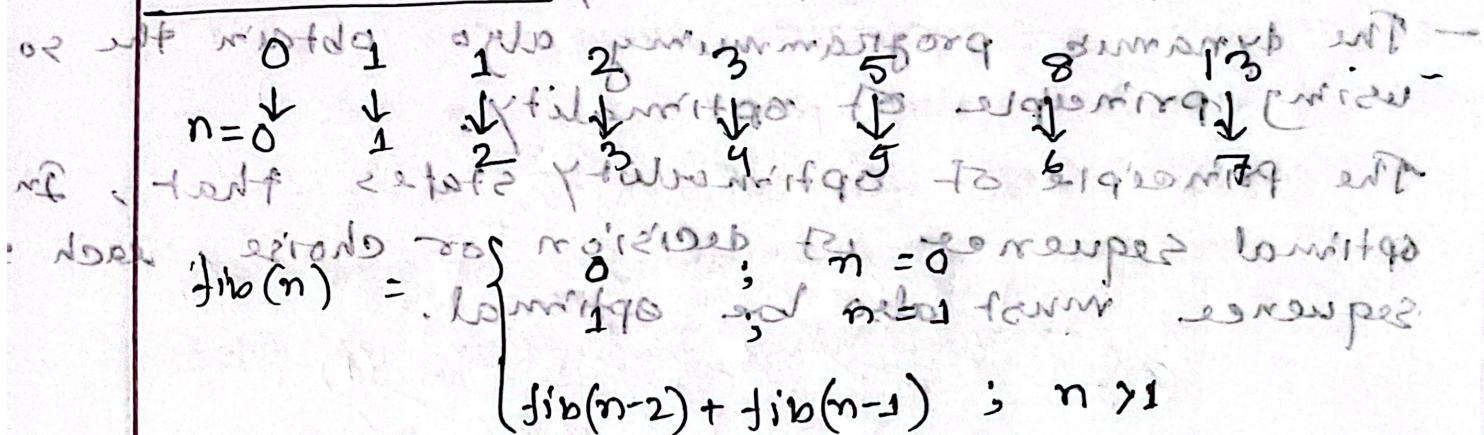
$$S = (z) \text{bit}$$

→ slower

→ Ex: 0/1 knapsack.



## Fibonacci Series



int fib(int n)

```
{
  if (n <= 1)
    return n;
}
```

fib(n)

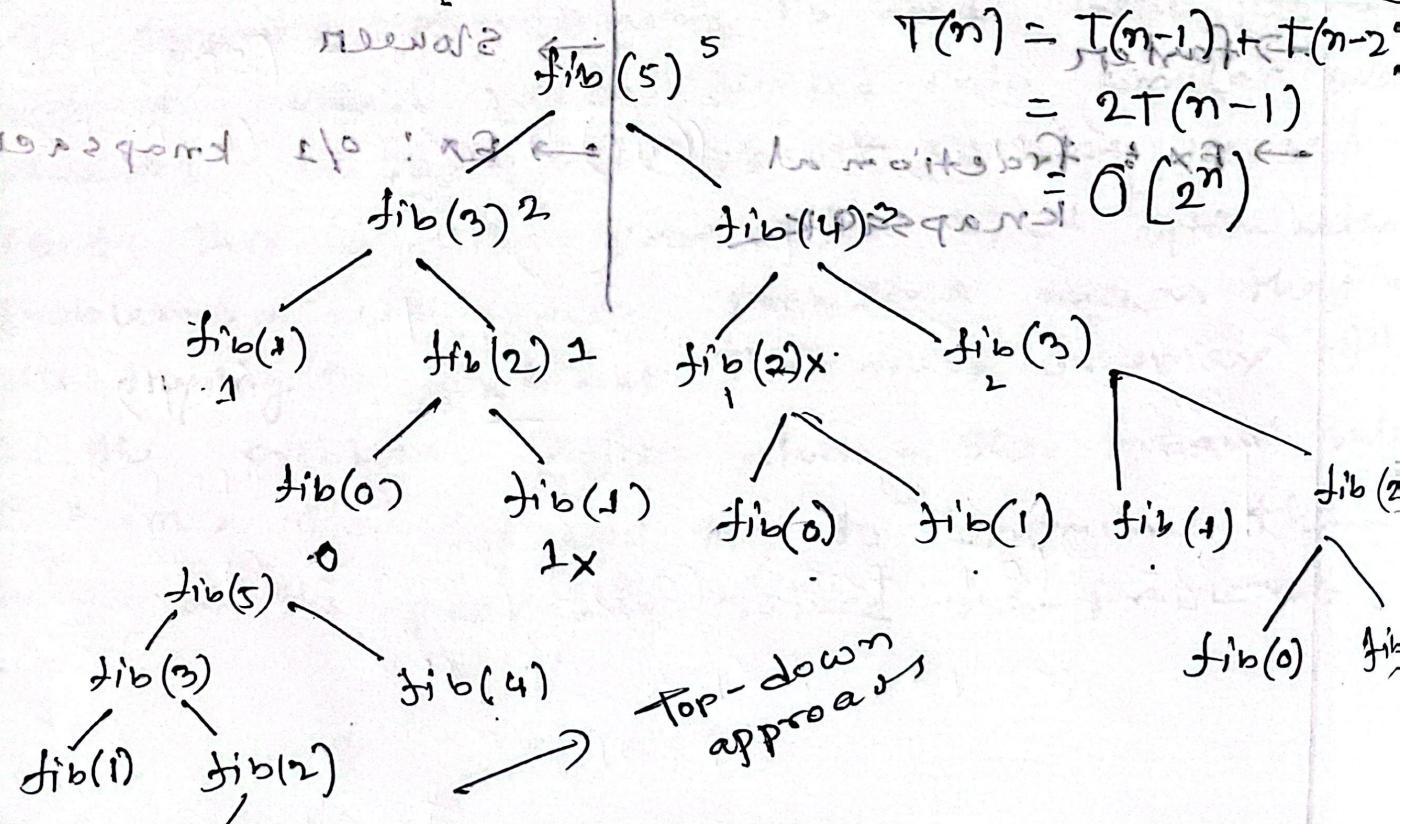
for numbers not less than 1  
more than 1

```

  else
    return fib(n-2) + fib(n-1);
}

```

\*  $\text{fib}(5) = ?$



## Memorization Process (follow top-down approach)

Step-1:

	0	1	2	3	4	5
$f(0)$	0	1	2	3	4	5
$f(1)$						
$f(2)$						

Step-2:

	-1	0	1	2	3	4	5
$f(0)$	-1	-1	-1	-1	-1	-1	-1
$f(1)$	0	1	2	3	4	5	
$f(2)$							
$f(3)$							
$f(4)$							
$f(5)$							

Step-3:

	0	1	2	3	4	5
$f(0)$	0	1	1	2	3	5
$f(1)$						
$f(2)$						
$f(3)$						
$f(4)$						
$f(5)$						

$$T(n) = O(n)$$

## Tabulation method (follow bottom-up approach)

$$fib(n) = \begin{cases} 0 & n=0 \\ 1 & n=1 \\ fib(n-1) + fib(n-2) & otherwise \end{cases}$$

$(l, i+j) \rightarrow fib(n-2) + fib(n-1) \rightarrow fib(n-1) + fib(n-2) \rightarrow 1$  summa

int fib(int n) {

    F[0] = 0; F[1] = 1;

    for (int i = 2; i <= n; i++)

        F[i] = F[i-2] + F[i-1];

}

return F[n];

x2 b/w app

return (0, 1);

= (0, 1) + 20

= (0, 1) + 20

= (0, 1) + 20

= (0, 1) + 20

= (0, 1) + 20

= (0, 1) + 20

= (0, 1) + 20

= (0, 1) + 20

= (0, 1) + 20

= (0, 1) + 20

## Multistaged graph (not well) working notes

→ find the minimum cost path

→ find the shortest path.

stage 2

Stage-3

L-gec2

14

11

10

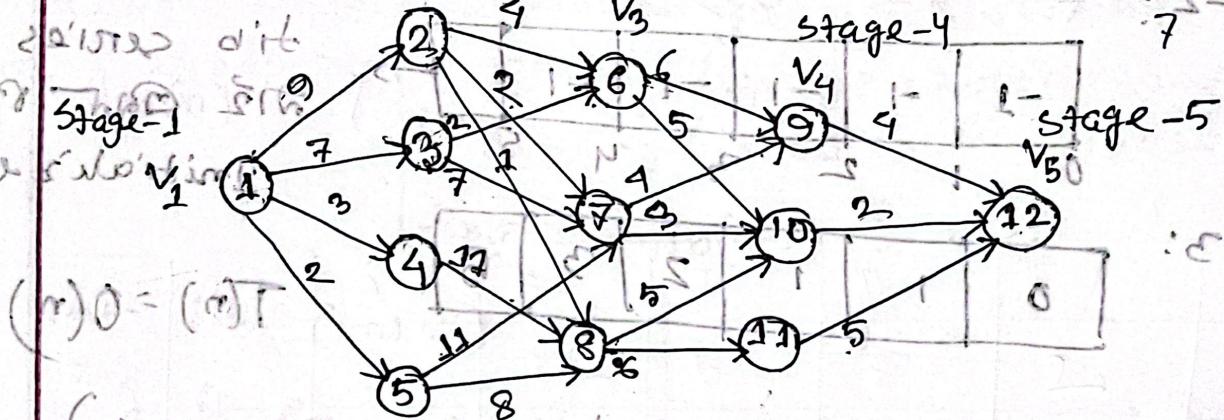
7

5

3

1

0



C-gec2

Vertex	1	2	3	4	5	6	7	8	9	10	11	12	
Cost	16	7	9	18	15	7	5	7	4	2	(10)	5	0
d	2,3	7	6	8	17	10	10	10	12	12	12	12	12

formula :  $\text{Cost}(i, j) = \min \{ C(i, l) + \text{Cost}(l, j) \}$

↓      ↓      ↓      ↓      ↓      ↓      ↓      ↓      ↓      ↓      ↓      ↓

stage vertex      2 vertex      3 vertex      4 vertex      5 vertex      6 vertex      7 vertex      8 vertex      9 vertex      10 vertex      11 vertex      12 vertex

$$\begin{aligned} \text{Cost}(5, 12) &= 0 \\ \text{Cost}(4, 9) &= \min \{ C(9, 12) + \text{Cost}(4, 9) \} \\ &= \min \{ 5 + 0 \} = 5 \\ &= 5 \end{aligned}$$

$$\text{Cost}(4, 10) = 2$$

$$\text{Cost}(4, 11) = 5$$

$$\begin{aligned} \text{Cost}(3, 6) &= \min \{ C(6, 9) + \text{Cost}(3, 9) \}, C(6, 10) + \text{Cost}(3, 10) \\ &= \min \{ 10 + 7, 10 + 2 \} \\ &= \min \{ 17, 12 \} \end{aligned}$$

l गे अप्पे 10  
अगले min cost  
अगले cost = 7  
d = 10

$$\text{Cost}(3,7) = \min \{ c(7,9) + \text{cost}(4,9), c(7,10) + \text{cost}(4,10), \\ = \min \{ (9+4), (3+2) \} \\ = 5$$

$$\text{Cost}(3,8) = \min \{ c(8,10) + \text{cost}(4,10), c(8,11) + \text{cost}(4,11) \} \\ = \min \{ (5+2), (6+5) \}$$

T	= 7	1	2	3	4	5	6	7	8	9	10	11	12
---	-----	---	---	---	---	---	---	---	---	---	----	----	----

$$\text{Cost}(2,2) = \min \{ c(2,6) + \text{cost}(3,6), c(2,7) + \text{cost}(3,7), \\ c(2,8) + \text{cost}(3,8) \}$$

$$= \min \{ (4+7), (2+5), (1+7) \} \\ = \min \{ 11, 7, 8 \} \\ = 7$$

$$\text{Cost}(2,3) = \min \{ c(3,6) + \text{cost}(3,6), c(3,7) + \text{cost}(3,7) \} \\ = \min \{ (2+7), (2+5) \} \\ = 9$$

$$\text{Cost}(2,4) = \min \{ c(4,8) + \text{cost}(3,8) \} \\ = 11 + 7$$

$$= 18$$

Path: Start - 2

$$d(1,2) = 2$$

$$d(2,2) = 7$$

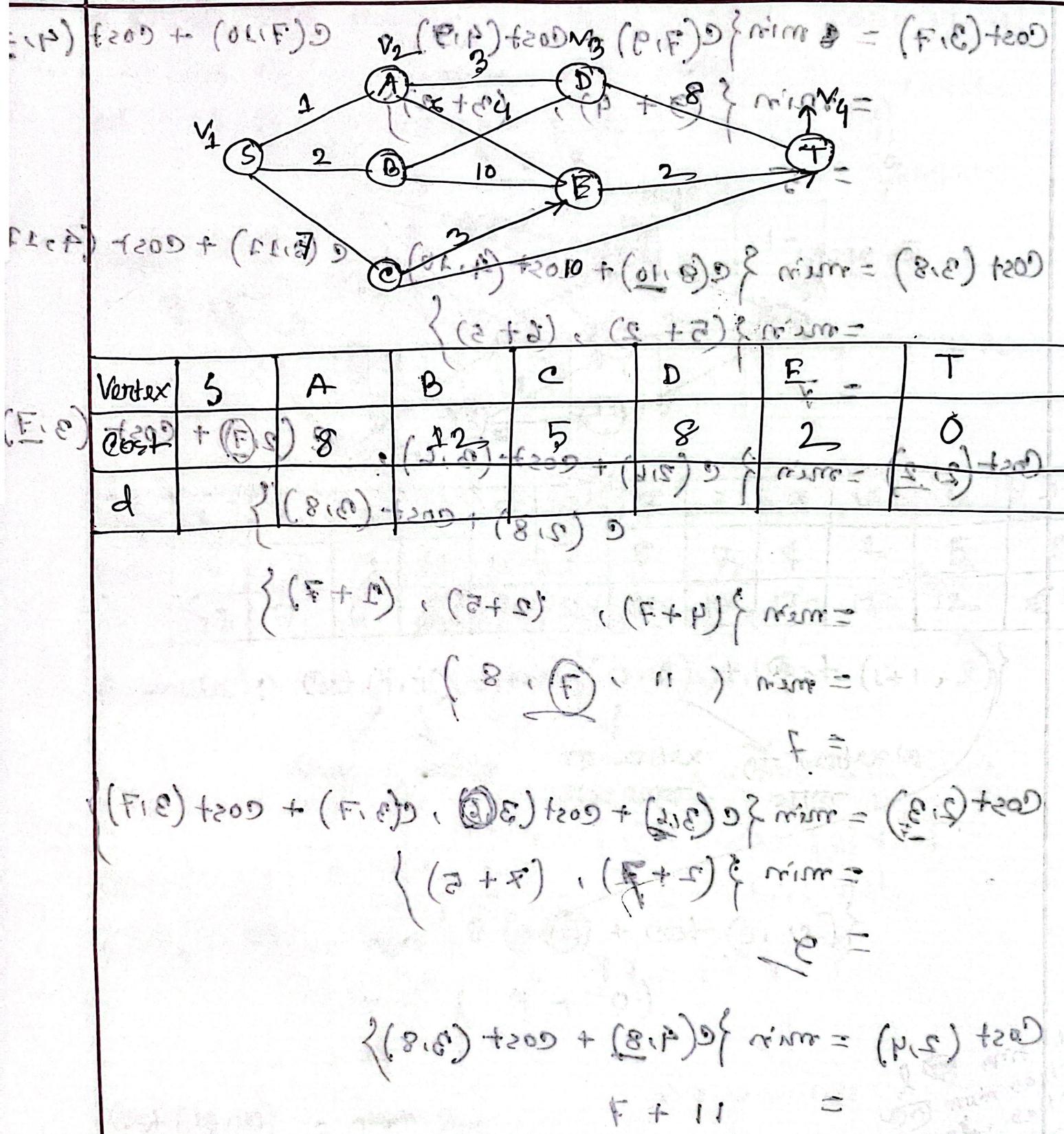
$$d(3,7) = 10$$

$$d(1,1) = 11$$

$$\begin{cases} \text{Start-3} \\ d(1,1)=3 \\ d(2,3)=6 \\ d(3,6)=10 \\ d(4,10)=12 \end{cases}$$

$$\begin{array}{ccccccc} 1 & \rightarrow & 2 & \rightarrow & 7 & \rightarrow & 10 \rightarrow 12 \\ 1 & \rightarrow & 3 & \rightarrow & 6 & \rightarrow & 10 \rightarrow 12 \end{array}$$

$$\min \text{ cost} = 16$$



$\{ (E, F) + f(v_1) + (E, F) \} \min = (E, S) + f(v_1)$   
 $\{ (E + F), (F + E) \} \min =$   
 $E + F =$   
 $\leftarrow 01 \leftarrow F \leftarrow S \leftarrow 1$   
 $\leftarrow 01 \leftarrow 01 \leftarrow 01 \leftarrow S \leftarrow 1$   


---

 $\leftarrow 01 \leftarrow \text{flip} \min \leftarrow 1$

$c \rightarrow \text{state} \quad 81 =$   
 $c = (E, F) b \quad | \quad c = (E, S) b$   
 $d = (E, F) b \quad | \quad d = (E, S) b$