

TIME COMPLEXITY AND ASYMPTOTIC NOTATION

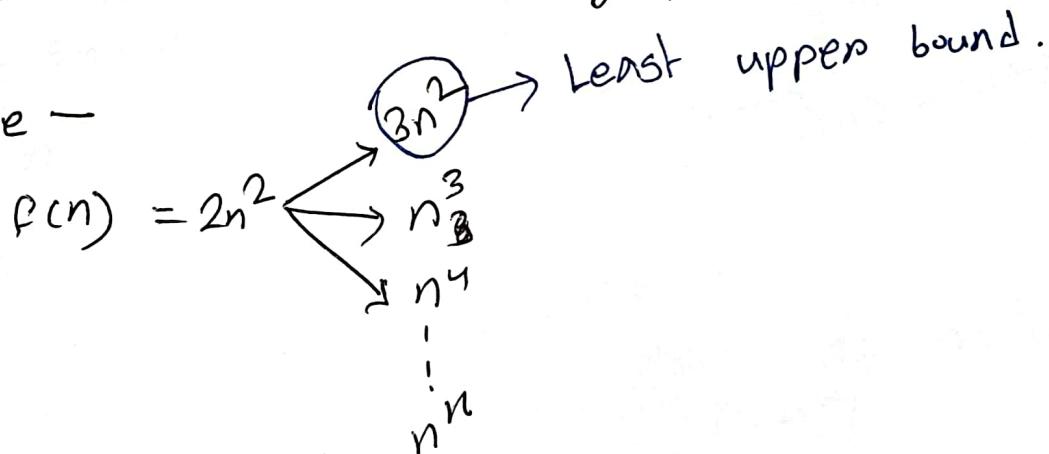
$O \rightarrow f(n) \rightarrow$ upper bound $(n) f \cdot g \geq (n) g$

$\Omega \rightarrow f(n) \rightarrow$ lower bound

$\Theta \rightarrow f(n) \rightarrow$ upper + lower bound $(n) f \leq (n) g$

Big O → If $f(n)$ and $g(n)$ are two functions,
then $f(n) = O(g(n))$, IF there exists constants c
and n_0 such that $f(n) \leq c \cdot g(n)$ for all $n \geq n_0$.

Example —



$f(n)$ is least upper bound, $g(n)$ is $2n^2$

Big O.

* If $f(n) = 4n+3$. Is $f(n) = O(g(n))$?

$$\Rightarrow f(n) \leq c \cdot g(n)$$

$$\Rightarrow 4n+3 \leq 5n$$

$c \cdot n \leq n$ $\Rightarrow c \leq 1$

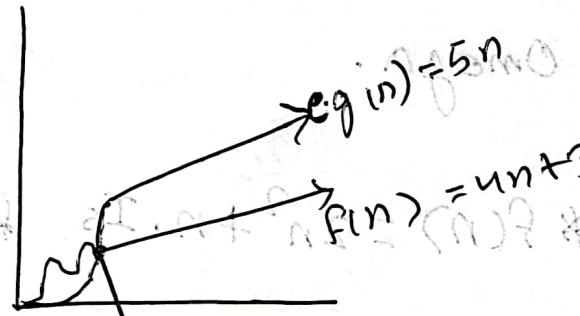
$$f(n) = O(g(n)) = O(n)$$

Now,

$$4n+3 \leq 5n$$

$$\Rightarrow 3 \leq 5n - 4n$$

$$\Rightarrow 3 \leq n$$



$n_0 = 3 \leq n$
 [n_0 point $\Rightarrow g(n) > f(n)$]

* $f(n) = 2n^2 + n$. Find $O(g(n))$.

$$\Rightarrow f(n) \leq c \cdot g(n)$$

$$\Rightarrow 2n^2 + n \leq 3n^2$$

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

$$(f_n) \Rightarrow n^2$$

$$f(n) \leq n^2$$

$$n^2 \leq n^2$$

$\Omega(g(n))$ — If $f(n) = g(n)$ and $g(n)$

are two functions then $f(n) = \Omega(g(n))$ if

there exists constants c and n_0 such that $f(n) \geq c \cdot g(n)$ for all $n \geq n_0$.

Example -

$$f(n) = 2n^2 \quad \text{greatest lower bound } g(n)$$

$f(n)$ is greatest lower bound, $g(n)$ is $\Omega(n^2)$

$\Omega(n^2)$

~~# $f(n) = 2n^2 + n$. Is $f(n) \geq \Omega(g(n))$?~~

$$\Rightarrow f(n) \geq c \cdot g(n)$$

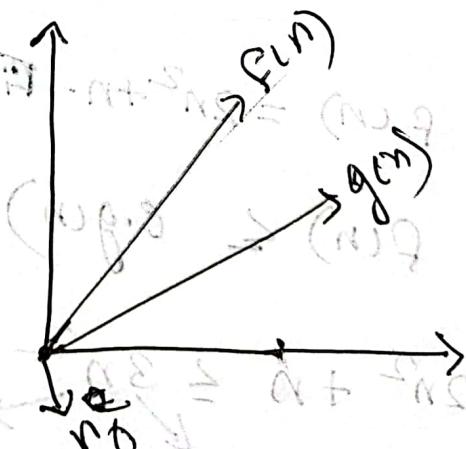
$$\Rightarrow 2n^2 + n \geq 2n^2$$

$$f(n) = \Omega(n^2)$$

Now,

$$2n^2 + n \geq 2n^2$$

$$\Rightarrow n \geq 0 \rightarrow n_0$$



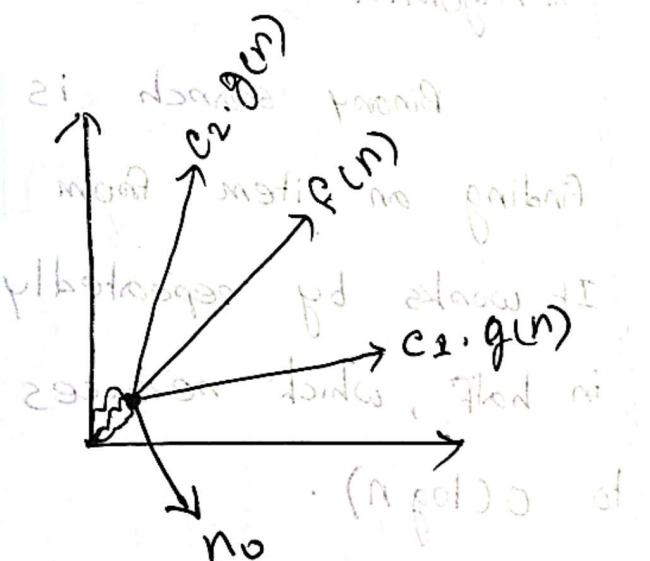
Theta (θ) \Rightarrow If $f(n)$ and $g(n)$ are two functions then $f(n) = \theta(g(n))$, if there exists constants c and n_0 such that $c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0$.
 * $f(n) = 2n^2 + n$, $f(n) = \theta(g(n)) = ?$

$$c_1 g(n) \leq f(n) \leq c_2 g(n)$$

$$\frac{2n^2}{c_1 g(n)} \leq 2n^2 + n \leq \frac{3n^2}{c_2 g(n)}$$

and multiplying this we get

$$\therefore f(n) = \theta(g(n)) \\ = \theta(n^2)$$



Recurrence Relation -

* Substitution

Method

→ void Test (int n) —

(α-tert)-Tetrahydro-

5

if($n > 0$)

```
{ printf ("%d", n )
```

Test (r)

3

$$T(n) =$$

$$T(n) = T(n-1) + 1$$

$$= [\tau(n-2) + 1] + 1$$

$$= \left[\left[T(n-3) + 1 \right] + 1 \right] + 1$$

$$= T(n-3) + 3$$

$$= T(n-k) + k$$

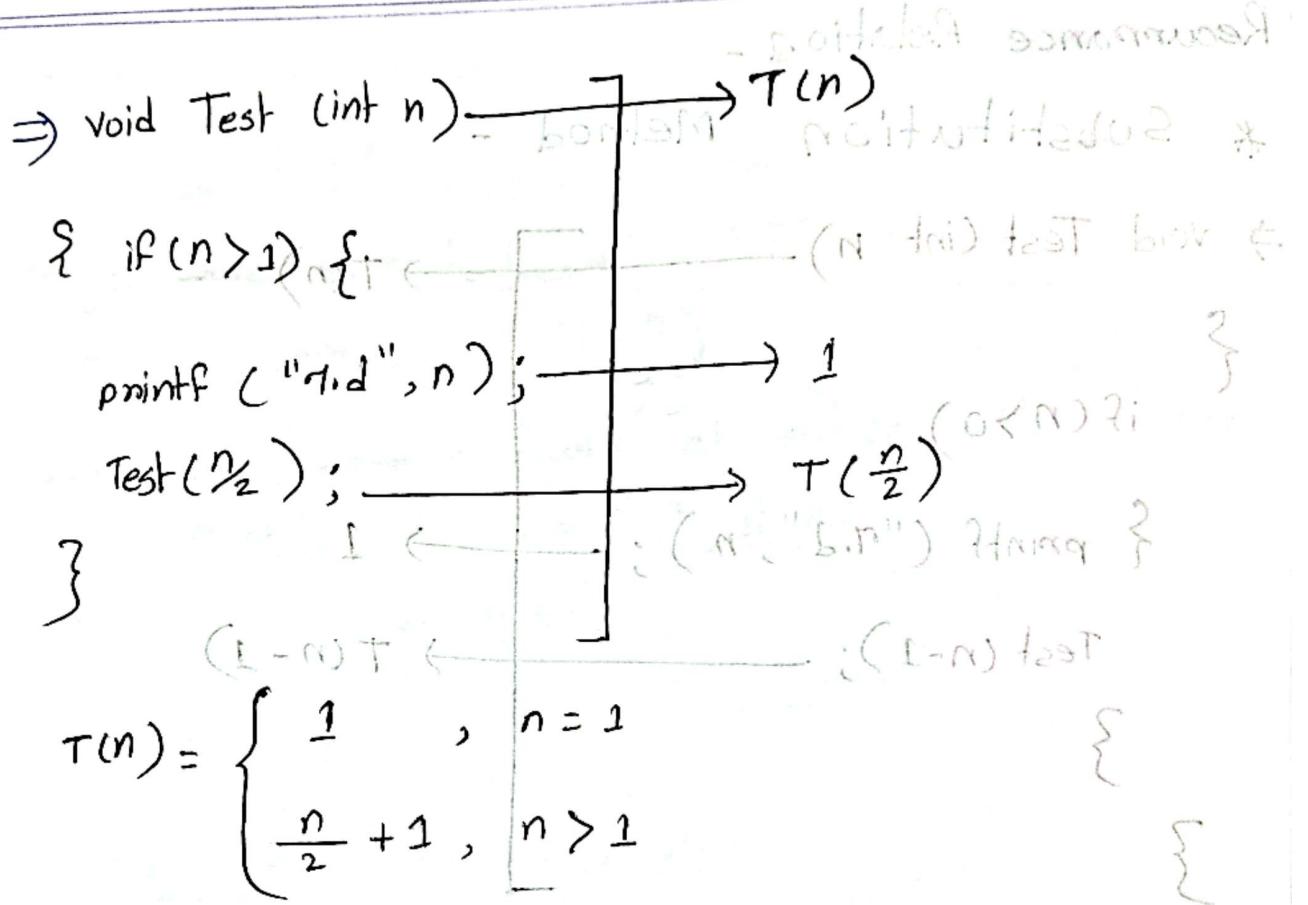
IF $k = n$

$$= \tau(n-n) + n$$

$$= T(0) + n$$

$$= 1 + n$$

• Time complexity = $O(1)$



$$\begin{aligned}
 T(n) &= T\left(\frac{n}{2}\right) + 1 \\
 &= \left[T\left(\frac{n}{4}\right) + 1 \right] + 1 && \vdots = (n)T \\
 &= \left[\left[T\left(\frac{n}{8}\right) + 1 \right] + 1 \right] + 1 && \vdots + (L-n)T = (n)T \\
 &= T\left(\frac{n}{2^3}\right) + 3 && \vdots + [L + (L-n)T] = \\
 &= T\left(\frac{n}{2^k}\right) + k && \text{when } \frac{n}{2^k} = 1 \\
 &= \vdots + [L + [L + (L-n)T]] =
 \end{aligned}$$

when,

$$\frac{n}{2^k} = 1$$

$$\Rightarrow 2^k = n$$

$$\Rightarrow \log_2^{2^k} = \log_2^n$$

$$\Rightarrow k \log_2 2 = \log_2^n$$

$$100 \Rightarrow k = \log_2^n$$

$$k + (k-n)T =$$

$$n + (n-n)T = n = 4 \cdot 21$$

$$n + (n-n)T =$$

$$n + (0)T =$$

wee millionaire said recursive division starts

$\therefore k = \log_2^n$ said no base branches if

$$T(1) = T(1) + k$$

$\Rightarrow k < b$ if (base case)

$\Rightarrow k = b$ if ($T(b)$) O (c)

$$= 1 + \log_2^n \cdot \Rightarrow k > b \text{ if } (T(n)) O (c)$$

M Master Method \Rightarrow

\Rightarrow Master theorem - millionaire draws pyramid and

$$T(n) = a^k T(n/b) + O(n^d)$$

$$(D) + (S) T^k 1 = (N) T$$

Here $a \geq 1, b > 1$

[a = no. of recursive steps for T(n)] $1 = a$ result

[b = size of subproblem]

Here,

$T(n)$ = Time complexity of the algorithm on an input
of size n .

a = number of recursive subproblems in each
division steps.

$\frac{n}{b}$ = size of each subproblem

$O(n^d)$ = The time complexity of combining the results
of all subproblems and any additional work
done outside the recursive calls.

⇒ Time complexity ($T(n)$) of the algorithm can be determined based on three cases -

(1) $O(n^d)$ if $d > \log_b^a$

(2) $O(n^d \log_b^n)$ if $d = \log_b^a$

(3) $O(n \log_b^a)$ if $d < \log_b^a$

↳ Both M algorithm

For binary search algorithm -

$$T(n) = 1 * T(\frac{n}{2}) + O(1)$$

Hence, $a=1$ [Microarray array $\left[\begin{matrix} 2 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{matrix} \right]$ \Rightarrow $n=8$ \Rightarrow $\frac{n}{2}=4$ \Rightarrow $d=0$ \Rightarrow $T(n)=O(n)$ \Rightarrow $T(n)=O(8)$ \Rightarrow $T(n)=O(1)$]

$$\text{From } \frac{n}{b} \Rightarrow \frac{n}{2}, b=2 \quad \text{So, } O(n) \text{ operations unit} = O(nT)$$

$$nd = 1, \text{ so, } d = 0$$

$$\text{Now, } \log_b^a = \log_2^1 = 0 = d \Rightarrow \text{medium} = 0$$

$$\therefore T(n) = O(\log_2^n) \quad [\text{if } d = \log_b^a, T(n) = O(n^d \log_b^n)]$$

These all conditions to get time complexity unit $= O(n)$
above conditions are true analogous to
all the situations will satisfy above

For Ternary search algorithm -

$$T(n) = 1 * T\left(\frac{n}{3}\right) + O(1)$$

Hence, $a=1$

$b=3$

$$\text{Now, } \log_b^a = \log_3^1 = 0 = d$$

$$\therefore T(n) = O(\log_3^n)$$

For merge sort algorithm -

$$T(n) = 2 * T\left(\frac{n}{2}\right) + O(n)$$

Hence, $a=2$

$b=2$

$$n^d = n, \text{ so, } d = 1$$

$$\text{Now, } \log_b^a = \log_2^2 = 1 = d$$

$$T(n) = O(n \log_2^n)$$

CT QUESTION

→ Derive recursive relation on equation & find the time complexity using master method if possible.

from the following function and find the time complexity using master method if possible.

void Test (int n) → $T(n)$

{ if ($n > 1$)

{ for (int i=0; i<n; i++)

{ printf ("*"); } → $n = (n)T$

Test ($\frac{n}{3}$); → $T(\frac{n}{3})$

}

$L = b, \alpha = b, \beta = b_n$

Recursive relation ⇒

$$T(n) = \begin{cases} 1, & \text{when } (n \leq 1) \\ T\left(\frac{n}{3}\right) + n, & \text{when } n > 1 \end{cases}$$

$$T(n) = \cancel{2} T\left(\frac{n}{3}\right) + n$$

$$= 1 \times T\left(\frac{n}{3}\right) + n$$

smaller part multiplies $n^d = n$, $b=3$, $d=1$

Now, this part work due to divide & conquer

$$\log_b a = \log_3 1 = 0$$

Hence, $d > \log_b a$.

$$d > \log_b a \Rightarrow O(n^d) = O(n)$$

$$\text{So, } T(n) = O(n^d)$$

(using i, f2st, f3rd, [max, min]) take apart

(f1g1 > f2g1) if

$$\{ \{ f1(f2g1 - f1g1) + f2g1 = bim \}$$

;(bim, f2g1, max) take apart

; (f1g1, 1 + bim, max) take apart

; (f1g1, bim, f2g1, max) take apart

DYNAMIC PROGRAMMING

→ Probleme

Divide and Conquer — Disjoint subproblems.

Dynamic Programming — Overlapping subproblems.

Fibonacci Series —

Recursive Approach —

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

int fib (int n) {

 if ($n \leq 1$)

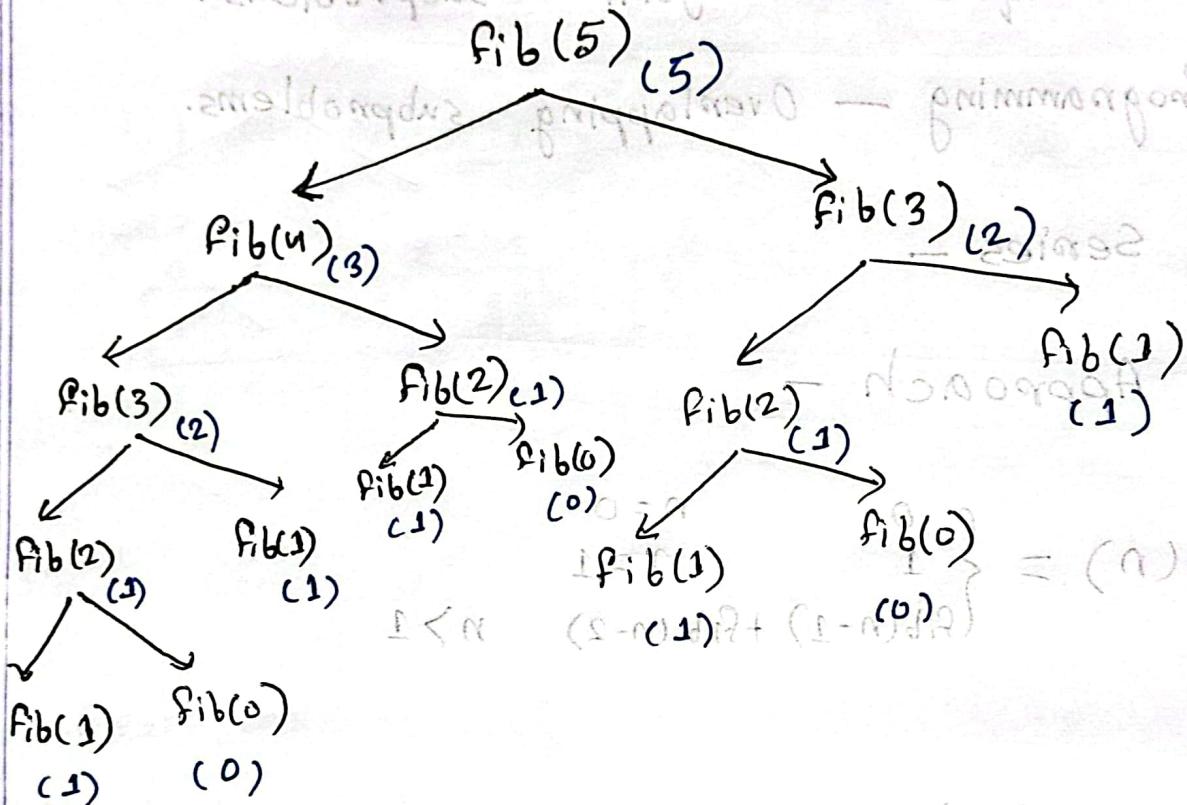
 return n;

 return fib(n-1) + fib(n-2); } $\leq n$; $L=d$ & $S=0$

}

Effizienz mitmemoisiert $\leftarrow C^N S = CNT$

Simulation —



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

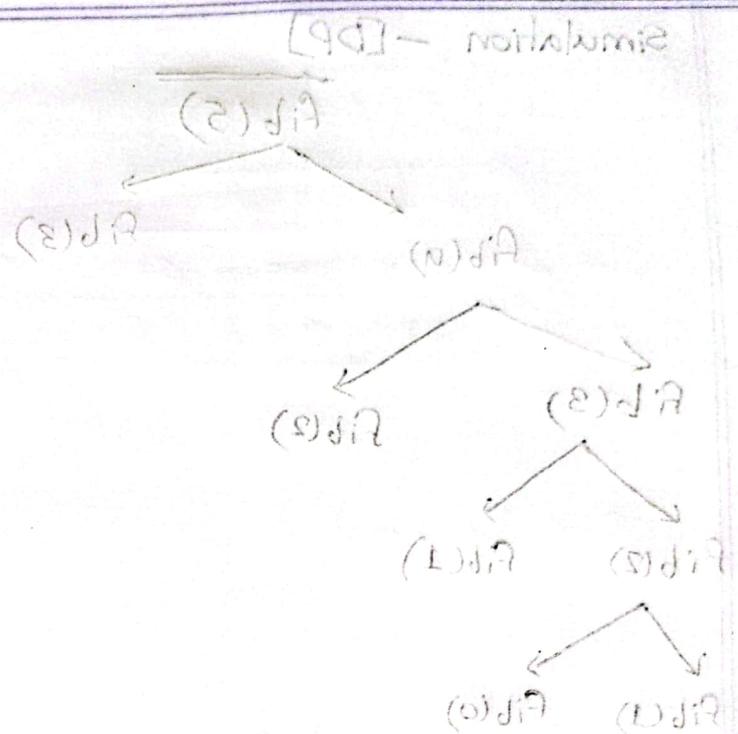
$$\approx 2T(n-1) + 1$$

$$a=2, b=1; a \geq 1, b \neq 1$$

$$T(n) = O(2^n) \rightarrow \text{Exponential complexity.}$$

DP Approach -

```
int dp[n+1];  
int fibonacci(int n)  
{  
    if (n == 0)  
    { return 0;  
    }  
    else if (n == 1)  
    { return 1;  
    }  
    else if (dp[n] == 0) ———————  
    {  
        dp[n] = fibonacci(n-1) + fibonacci(n-2);  
    }  
    return dp[n];  
}
```

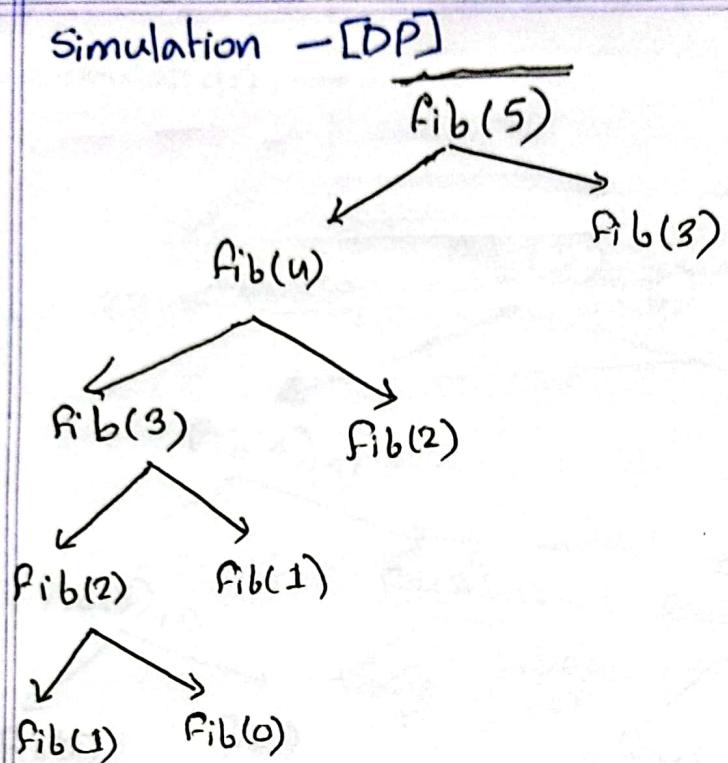


{

(1-10) linked * n recursive

{

Simulation - [DP]



Factorial —

Recursive approach —

Factorial(n) :

{ If $n=0$ or $n=1$

return 1;

return $n * \text{Factorial}(n-1)$

}

DP Approach - ~~Naive~~ Normal Solution - 291

An array of size $n+1$ to store the results
dp[n+1] # An array of size $n+1$ to store the results
• primitive ev. recursive approach

Factorial(n)

{
 If $n=0$ or $n=1$ then - return 1
 return 1

Else If $dp[n] = 0$ then *
 $dp[n] = n * \text{Factorial}(n-1)$

return $dp[n]$.

}

base case normal

else if 0

{0, 1} = x

{1, 2} = x

Factorial ← {0, 1, 2, 3, 4, 5} = x

LCS - Longest Common Subsequence

Subsequence vs substring.

"programming"

grammm - both subsequence and substring

gammg - subsequence, not substring.

* Substring is main sequence

same sequence, element

Now,

$$S = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

$$E = \{2, 4, 6, 8, 10\}$$

$$X_1 = \{2, 4\}$$

$$X_2 = \{2, 4, 6\}$$

$$X_3 = \{2, 4, 6, 8, 10\}$$

Common Subsequence
of S and E

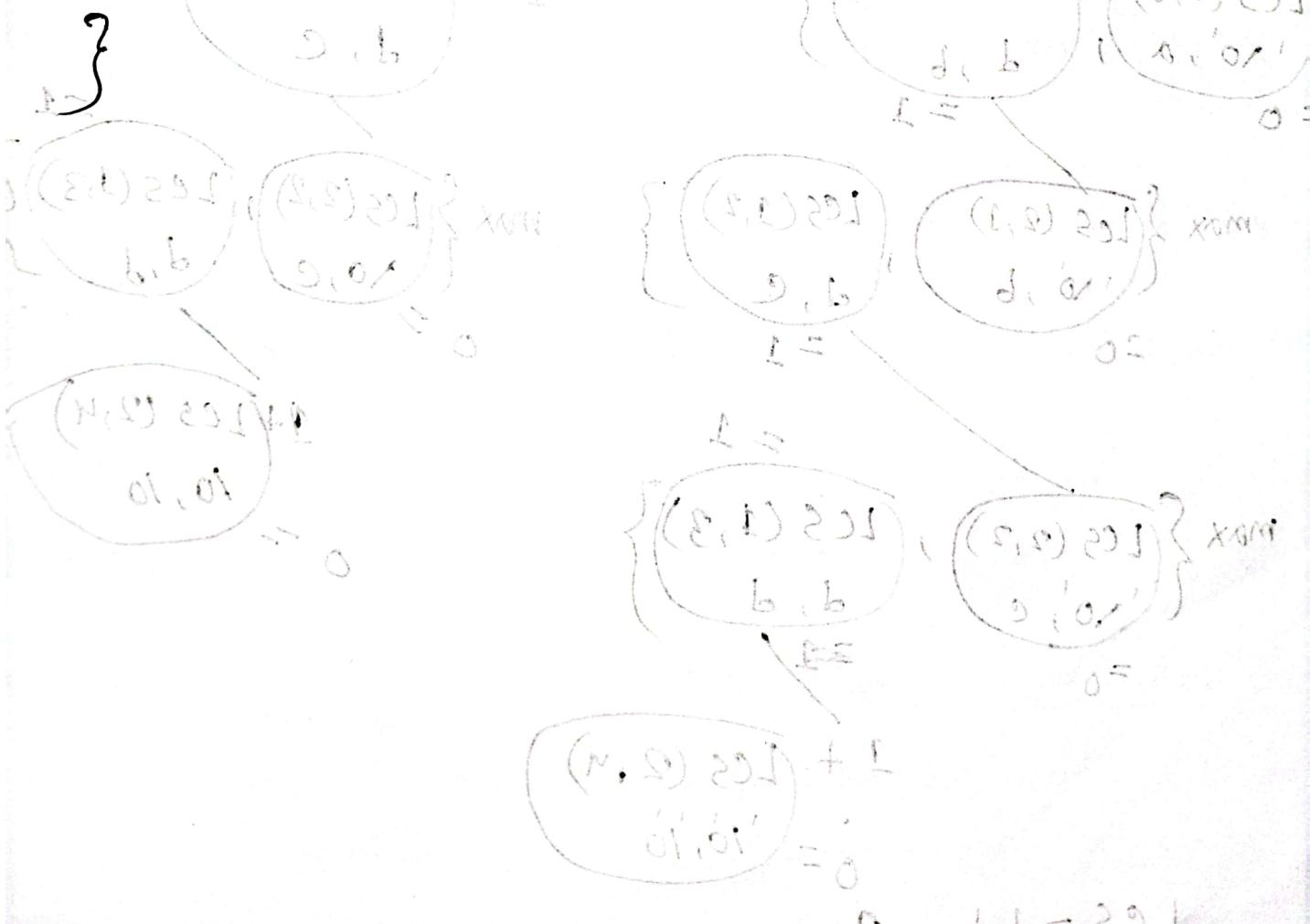
longest.

LCS - Recursive Approach -

- profit/loss

$LCS(i, j)$

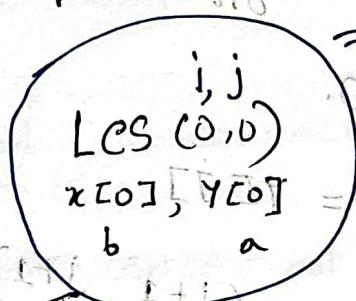
```
{
    if  $x[i] = \text{NULL}$  or  $y[j] = \text{NULL}$  or  $y[j] < i$ 
        return 0
    else if  $x[i] = y[j]$ 
        return  $1 + LCS(i+1, j+1)$ 
    else
         $L = \max(LCS(i+1, j), LCS(i, j+1))$ 
        return  $L$ 
}
```



Simulation -

i → X [0 | 1 | 2 | b | d | 10]

j → Y [a | b | c = d] → 10



$$\max \{ \text{Les}(1,0) \} = 1$$

$$\max \{ \begin{cases} LCS(2,0) \\ '10', a \end{cases}, \begin{cases} LCS(1,1) \\ d, b \end{cases} \} = 1$$

$$\max \left\{ \begin{array}{l} \text{Lcs}(2,1) \\ \text{d}, b \end{array} , \begin{array}{l} \text{Lcs}(1,2) \\ \text{c} \end{array} \right\} = 1$$

$$\max \left\{ \begin{array}{l} \text{LCS}(2,2) \\ '10', c \end{array}, \begin{array}{l} \text{LCS}(1,3) \\ d, d \end{array} \right\}$$

$$1 + \overbrace{LCS(2, 4)}^{10, 10}$$

$$LCS = bd = 2$$

$$= \sum_{i=1}^n x_i q_i$$

$\lim_{n \rightarrow \infty} a_n = 0$

$$1 + \overbrace{2cs(1,2)}^{d,c} = 1$$

$$\max \{ \text{LCS}(2,2), \text{Les}(1,3) \}$$

$$\text{LCS}(2,4)$$

LCS [DP Approach] -

LCS(i, j)

To find the ~~seq~~ LCS of 2 sequences, we need to follow the steps given below -

>Create a matrix dp of $m+1$ by $n+1$ size where m and n are sizes of the given sequences.

$[dp[i, j]] = \text{LCS of } x_i \text{ and } y_j]$ and traverse each cell row-wise for column-wise.

If $i=0$ or $j=0$, then set $dp[i, 0] = dp[0, j] = 0$

If $x_i = y_j$, set $dp[i, j] = dp[i-1, j-1] + 1$

If $x_i \neq y_j$, set $dp[i, j] = \max(dp[i-1, j], dp[i, j-1])$

If $x_i \neq y_j$, set $dp[i, j] = \max(dp[i-1, j], dp[i, j-1])$

$$Q = [Q_{i,j}]_{m \times n} = [w_i]_{m \times 1} [h_j]_{n \times 1}, Q = \sum_{i=1}^m w_i \otimes h_j^T$$

size-matters and dimension matters does not matter

but classes i mostly answers to higher QL (n)

with $w_i \otimes h_j^T$ minimizes phinges function

$$[Q_{i,i-j}]_{m \times m} = [w_i]_{m \times 1} [h_{i-j}]_{m \times 1}$$

such encoding for order such that the most frequent soft q's

Knapsack -

- [Nonogram 90] 20.1

Algorithm —



To solve the 0/1 knapsack problem with n items, each with a weight and a value, and a knapsack with a maximum capacity w , we need to follow the steps given below.

(1) Create a matrix dp of size $n+1$ by $w+1$. [where $dp[i][w]$ = the maximum value of items $1 \dots i$ with weight limit w].

(2) If $i=0$ or $w=0$, set $dp[i,0] = dp[0,w] = 0$.

(3) Traverse each cell row-wise or column-wise.

(4) If weight of current item i exceeds the remaining capacity w ,

set $dp[i,w] = dp[i-1,w]$.

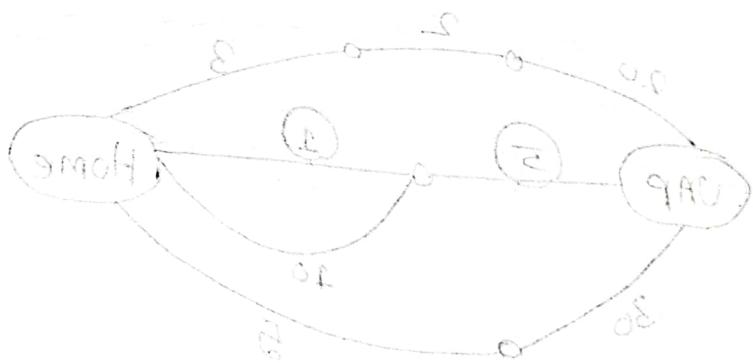
[skip the current item and take value of previous row].

(5) If the current item can fit ($\text{weight}[i-1] \leq w$):
~~set $dp[i, w] = \max(dp[i-1, w], dp[i-1, w - \text{weight}[i]] + v(i))$~~

~~set $dp[i, w] = \max(dp[i-1, w], dp[i-1, w - \text{weight}[i]] + v(i))$~~

[choose the maximum between skipping the item or taking it)

(6) The value at $dp[n, w]$ will be the maximum value ~~to~~ that can be achieved with the given capacity.

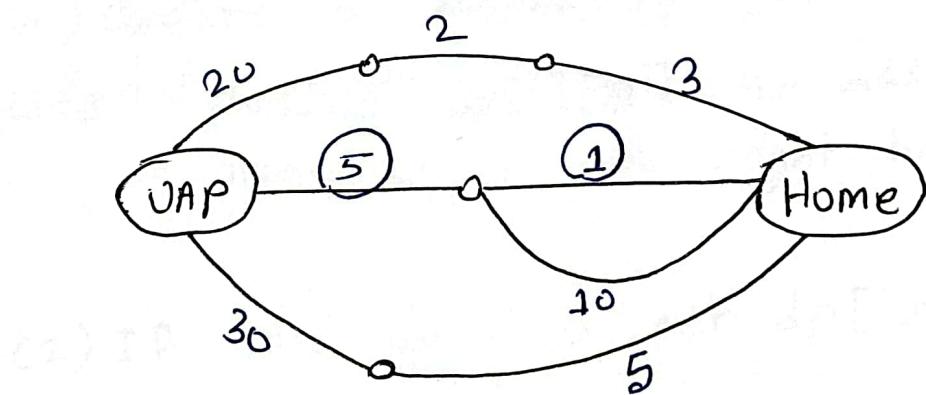


Greedy Algorithm

④ Makes locally optimal choice

④ Once a decision made never reconsidered.

④ do not always give optimal solution.



Example : 0/1 knapsack problem in greedy approach

Item	Value	Weight	Ratio	note
1	\$100	20	5	
2	\$120	30	4	
3	\$60	10	6	Capacity 50.

Objective : Max profit
constraint : capacity

Step - 1 : calculate value by weight for each item

Step - 2 : sort items based on the above ratio in descending order.

Step - 3 : Take item on notes (0/1) . note (S)

Solution \Rightarrow

Item	Value	Weight	Value/Weight
1	\$100	20	5
2	\$120	30	4
3	\$60	10	6

1 note + 2 note = 120 \$ x M

60 \$ + 120 \$ = 180 \$
[Free space] 00L\$ =

After sorting in descending order -

Item	Value	Weight	Value/Weight
3	\$60	10	6
1	\$100	20	5
2	\$120	30	4

If weight \leq capacity, pick item.

Then find remaining capacity.

(1) Item 3 \rightarrow $10 \leq 50$ pick item 3. capacity = $50 - 10 = 40$

Profit = \$60.

(2) Item 1 \rightarrow $20 \leq 40$ pick item 1. capacity = $40 - 20 = 20$

Profit = \$60 + \$100 = \$160

(3) Item 2 \rightarrow $30 > 20$. Item 2 rejected.

Value	Weight	Value/Weight	Max I
60	10	6	6
100	20	5	5
120	30	4	4

\therefore Max Profit = Item 3 + Item 1

$$= \$60 + \$100$$

$$= \$160 [\text{Ans}]$$

Hence, we can see that, ~~maximum~~ if we took item 1 and 2, then ~~the~~ the maximum profit would be \$220 within the given capacity. If we took item 2 and 3, then the maximum profit would be \$180 within given capacity. \$220 and \$180 are greater than \$160. So, we can say that greedy algorithm doesn't give optimal solution of 0/1 knapsack.

Fractional knapsack -

0/1 Knapsack		Fractional Knapsack	
Item	Profit	Weight	Profit/Weight
1	100	20	5
2	120	30	4
3	100	20	5
4	100	20	5
5	100	20	5
6	100	20	5
7	100	20	5
8	100	20	5
9	100	20	5
10	100	20	5
11	100	20	5
12	100	20	5
13	100	20	5
14	100	20	5
15	100	20	5
16	100	20	5
17	100	20	5
18	100	20	5
19	100	20	5
20	100	20	5
21	100	20	5
22	100	20	5
23	100	20	5
24	100	20	5
25	100	20	5
26	100	20	5
27	100	20	5
28	100	20	5
29	100	20	5
30	100	20	5
31	100	20	5
32	100	20	5
33	100	20	5
34	100	20	5
35	100	20	5
36	100	20	5
37	100	20	5
38	100	20	5
39	100	20	5
40	100	20	5
41	100	20	5
42	100	20	5
43	100	20	5
44	100	20	5
45	100	20	5
46	100	20	5
47	100	20	5
48	100	20	5
49	100	20	5
50	100	20	5
51	100	20	5
52	100	20	5
53	100	20	5
54	100	20	5
55	100	20	5
56	100	20	5
57	100	20	5
58	100	20	5
59	100	20	5
60	100	20	5
61	100	20	5
62	100	20	5
63	100	20	5
64	100	20	5
65	100	20	5
66	100	20	5
67	100	20	5
68	100	20	5
69	100	20	5
70	100	20	5
71	100	20	5
72	100	20	5
73	100	20	5
74	100	20	5
75	100	20	5
76	100	20	5
77	100	20	5
78	100	20	5
79	100	20	5
80	100	20	5
81	100	20	5
82	100	20	5
83	100	20	5
84	100	20	5
85	100	20	5
86	100	20	5
87	100	20	5
88	100	20	5
89	100	20	5
90	100	20	5
91	100	20	5
92	100	20	5
93	100	20	5
94	100	20	5
95	100	20	5
96	100	20	5
97	100	20	5
98	100	20	5
99	100	20	5
100	100	20	5
101	100	20	5
102	100	20	5
103	100	20	5
104	100	20	5
105	100	20	5
106	100	20	5
107	100	20	5
108	100	20	5
109	100	20	5
110	100	20	5
111	100	20	5
112	100	20	5
113	100	20	5
114	100	20	5
115	100	20	5
116	100	20	5
117	100	20	5
118	100	20	5
119	100	20	5
120	100	20	5
121	100	20	5
122	100	20	5
123	100	20	5
124	100	20	5
125	100	20	5
126	100	20	5
127	100	20	5
128	100	20	5
129	100	20	5
130	100	20	5
131	100	20	5
132	100	20	5
133	100	20	5
134	100	20	5
135	100	20	5
136	100	20	5
137	100	20	5
138	100	20	5
139	100	20	5
140	100	20	5
141	100	20	5
142	100	20	5
143	100	20	5
144	100	20	5
145	100	20	5
146	100	20	5
147	100	20	5
148	100	20	5
149	100	20	5
150	100	20	5
151	100	20	5
152	100	20	5
153	100	20	5
154	100	20	5
155	100	20	5
156	100	20	5
157	100	20	5
158	100	20	5
159	100	20	5
160	100	20	5
161	100	20	5
162	100	20	5
163	100	20	5
164	100	20	5
165	100	20	5
166	100	20	5
167	100	20	5
168	100	20	5
169	100	20	5
170	100	20	5
171	100	20	5
172	100	20	5
173	100	20	5
174	100	20	5
175	100	20	5
176	100	20	5
177	100	20	5
178	100	20	5
179	100	20	5
180	100	20	5
181	100	20	5
182	100	20	5
183	100	20	5
184	100	20	5
185	100	20	5
186	100	20	5
187	100	20	5
188	100	20	5
189	100	20	5
190	100	20	5
191	100	20	5
192	100	20	5
193	100	20	5
194	100	20	5
195	100	20	5
196	100	20	5
197	100	20	5
198	100	20	5
199	100	20	5
200	100	20	5
201	100	20	5
202	100	20	5
203	100	20	5
204	100	20	5
205	100	20	5
206	100	20	5
207	100	20	5
208	100	20	5
209	100	20	5
210	100	20	5
211	100	20	5
212	100	20	5
213	100	20	5
214	100	20	5
215	100	20	5
216	100	20	5
217	100	20	5
218	100	20	5
219	100	20	5
220	100	20	5
221	100	20	5
222	100	20	5
223	100	20	5
224	100	20	5
225	100	20	5
226	100	20	5
227	100	20	5
228	100	20	5
229	100	20	5
230	100	20	5
231	100	20	5
232	100	20	5
233	100	20	5
234	100	20	5
235	100	20	5
236	100	20	5
237	100	20	5
238	100	20	5
239	100	20	5
240	100	20	5
241	100	20	5
242	100	20	5
243	100	20	5
244	100	20	5
245	100	20	5
246	100	20	5
247	100	20	5
248	100	20	5
249	100	20	5
250	100	20	5
251	100	20	5
252	100	20	5
253	100	20	5
254	100	20	5
255	100	20	5
256	100	20	5
257	100	20	5
258	100	20	5
259	100	20	5
260	100	20	5
261	100	20	5
262	100	20	5
263	100	20	5
264	100	20	5
265	100	20	5
266	100	20	5
267	100	20	5
268	100	20	5
269	100	20	5
270	100	20	5
271	100	20	5
272	100	20	5
273	100	20	5
274	100	20	5
275	100	20	5
276	100	20	5
277	100	20	5
278	100	20	5
279	100	20	5
280	100	20	5
281	100	20	5
282	100	20	5
283	100	20	5
284	100	20	5
285	100	20	5
286	100	20	5
287	100	20	5
288	100	20	5
289	100	20	5
290	100	20	5
291	100	20	5
292	100	20	5
293	100	20	5
294	100	20	5
295	100	20	5
296	100	20	5
297	100	20	5
298	100	20	5
299	100	20	5
300	100	20	5
301	100	20	5
302	100	20	5
303	100	20	5
304	100	20	5
305	100	20	5
306	100	20	5
307	100	20	5
308	100	20	5
309	100	20	5
310	100	20	5
311	100	20	5
312	100	20	5
313	100	20	5
314	100	20	5
315	100	20	5
316	100	20	5
317	100	20	5
318	100	20	5
319	100	20	5
320	100	20	5
321	100	20	5
322	100	20	5
323	100	20	5
324	100	20	5
325	100	20	5
326			

Example -

capacity = 15

Item	1	2	3	4	5	6	7
Value	11	6	16	8	7	19	4
Weight	3	4	6	8	2	5	2
value/ weight	3.67	1.5	2.67	1	3.5	3.8	2

$$x \begin{pmatrix} 1 & 0 & \frac{5}{6} & 0 & 1 & 1 & 0 \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \end{pmatrix}$$

Item	Value	Weight	Remaining capacity	Total Profit
5215	6	19	15 - 6 = 10	19
3210	1	3	10 - 3 = 7	19 + 1 = 20
227	5	2	7 - 2 = 5	20 + 5 = 25
675	$\frac{5}{6} \times 16$ $= 13.3$	$\frac{5}{6} \times 6$ $= 5$	5 - 5 = 0	25 + 13.3 = 38.3

Algorithm:

for ($i=0$; $i < n$; $i++$)
 if ($\text{arr}[i].weight \leq \text{capacity}$)
 {
 $c = c - \text{arr}[i].weight$;
 total-profit += arr[i].value;
 }
 else {
 frac = $c / \text{arr}[i].weight$;
 total-profit += total-profit + frac * arr[i].value;
 }

Complexity :

For sorting $\Rightarrow O(n \log n)$

Comparison $\Rightarrow O(n)$

Overall $\Rightarrow O(n \log n) + O(n)$

Since $O(n \log n)$ is prominent \rightarrow Time complexity

Time complexity = ~~of~~ $O(n \log n)$.

Coin Change Problem \Rightarrow

Objective - Minimum number of coins.

constraints - Coins are available in infinite numbers,

Example -

$$n = 140$$

$$C = \{5, 10, 20, 25, 50\}$$

After sorting -

$$C = \{50, 25, 20, 10, 5\}$$

IF $C[i] \leq n$, pick coin

	Coin used	Total coins	Remaining amount
$50 \leq 140$	(50)	1	$140 - 50 = 90$
$50 \leq 90$	(50) (50)	2	$90 - 50 = 40$
$25 \leq 40$	(50) (50) (25)	3	$40 - 25 = 15$
$10 \leq 15$	(50) (50) (25) (10)	4	$15 - 10 = 5$
$5 = 5$	(50) (50) (25) (10) (5)	5	$5 - 5 = 0$

Number of coins used = 5

Algorithm :-

* "Greedy doesn't give optimal solution of coin change problem every time" - Explain with example.

→ Assume, $n=6$ and $C = \{1, 3, 4\}$.

After sorting, $C = \{4, 3, 1\}$

Coins	Total	Remaining
4	1	$6-4=2$
3 1	2	$2-1=1$
3 1 1	3	$1-1=0$

Hence, we can see that, ~~if we took~~ two coins of amount 3. That would be the minimum number of coins to make 6 with the available coins. But, using greedy algorithm, we found that 3 coins are needed for making 6 with the available coins. So, greedy doesn't always give optimal solution.

(a) \in golden ratio faster

(b) $a + (m \cdot g(a)) \in$ linear

$m < n$ best

fastest in $O(n)$

Algorithm :

```
int greedyCoinChange (int c[], int n, int i) {
```

```
    if (n == 0) return 0;
```

```
    if (c[i] <= n)
```

```
        return 1 + greedyCoinChange (c, n - c[i], i);
```

```
    else
```

```
        return greedyCoinChange (c, n, i + 1);
```

minimise	start	status
$\sum N = j$	1	①
$L = L - 1$	2	②
$G = L - 1$	3	③

Complexity \Rightarrow Time complexity $O(n^m)$

Time complexity $O(n^m)$ is dominant because $n \gg m$.

Overall complexity $\Rightarrow O(n^m)$

For sorting $\Rightarrow O(m \log m)$

Total amount making $\Rightarrow O(n)$

Overall $\Rightarrow O(m \log m) + O(n)$

Hence, $n \gg m$.

So, $O(n)$ is prominent.

Time complexity $= O(n)$.

Job sequencing with deadline \Rightarrow

Job ID	1	2	3	4	5	6	7	8
Benefit	10	5	20	1	15	5	4	5
Deadline	1	3	2	3	2	3	2	2
Completion Time	5:00	10:00	15:00	19:00	20:00	21:00	22:00	23:00

$$0 - \frac{1}{2} - 2 \left(\frac{5}{3} \right) = \text{Northwest}$$

এটি JobID আর কোন সময় Timeslot:

For Timeslot \geq only 1st Job is prioritized

example 2 Timeslot 3Bt - 0 to 1, 1 to 2, 2 to ?

After sorting \Rightarrow $(n) \circ = \text{sorted}$

JobID → 3(a_0) $500 + 1$ (a_0^2) = 4110000

Probit value \rightarrow 20 15 10 ($\frac{5}{10}$) 1

Deadline → 2 2 1 3 3
(apolo) 0 = pfixalgmos smit

JobID	Profit	Deadline	Slot assigned
J ₃	20	2	[1, 2]
J ₅	15	2	[0, 1]
J ₁	10	1	Rejected.
J ₂	5	3	[2, 3]
J ₄	1	3	Rejected

Solution - (J₃, J₅, J₂)

$$\text{Profit} \Rightarrow 20 + 15 + 5 = 40$$

Complexity \Rightarrow Job slot plan \propto Job slot time
 Sorting n jobs $\propto O(n \log n)$. signature

$$\text{choice} = O(n)$$

$$\text{Overall P} = O(n) + O(n \log n)$$

Since, $O(n \log n)$ is dominant

$$\text{Time complexity} = O(n \log n)$$

GRAPH

↳ Non linear data structure

Types of Graph :

— edge connects vertices

Directed \Rightarrow Direction ONCE

Undirected \Rightarrow Direction TWICE

Half directed (1)

Normal weighted (2)

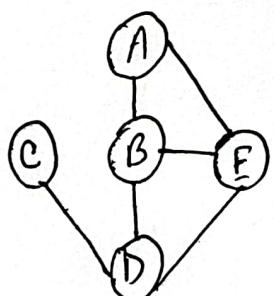
Weighted \Rightarrow

Weight ONCE

Unweighted \Rightarrow

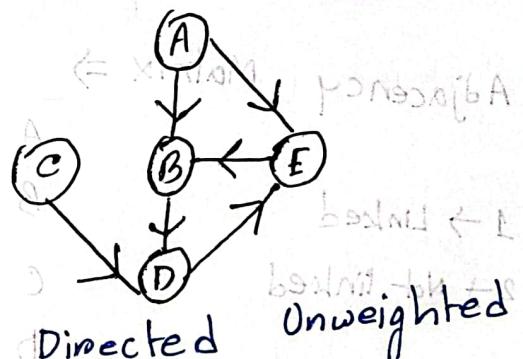
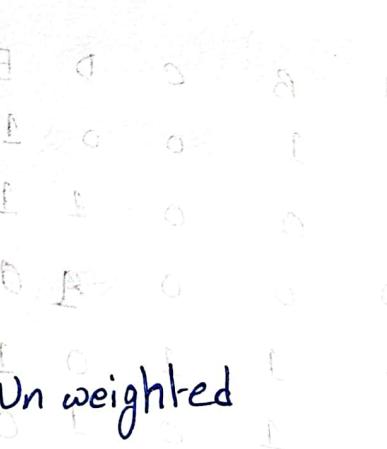
Weight TWICE

Half weighted (A)



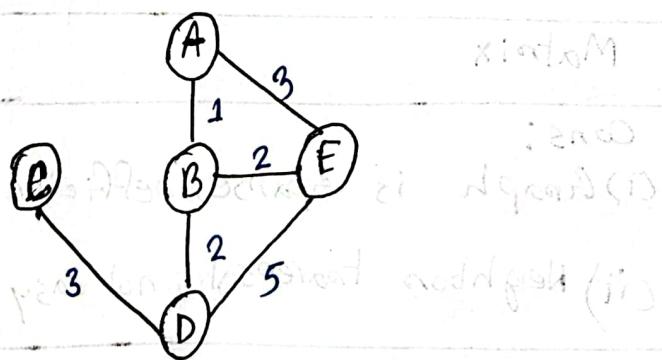
Undirected

Unweighted

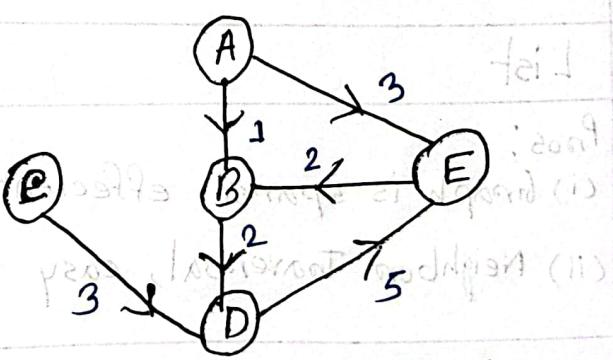


Directed

Onweighted



Undirected Weighted



Directed weighted

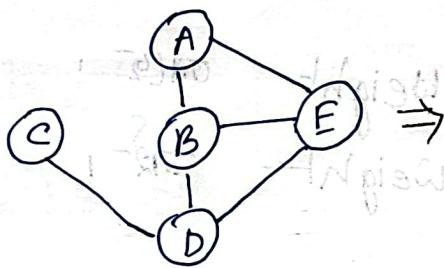
Graph Representation \Rightarrow 119A85

Two standard ways —

(1) Adjacency List

(2) Adjacency Matrix

Adjacency List \Rightarrow



$A \rightarrow B, E$

$B \rightarrow A, D, E$

$C \rightarrow B$

$D \rightarrow B, C, E$

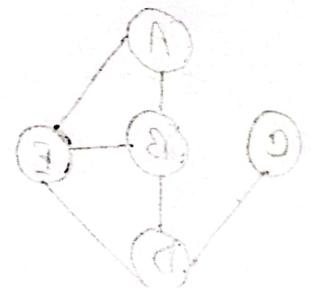
$E \rightarrow A, B, D$

Adjacency Matrix \Rightarrow

1 \rightarrow Linked

2 \rightarrow Not linked

	A	B	C	D	E
A	0	1	0	0	1
B	1	0	0	1	1
C	0	0	0	1	0
D	0	1	1	0	1
E	1	1	0	1	0

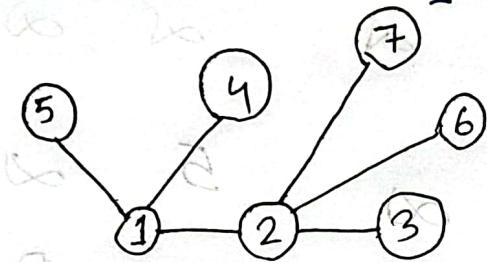


Differences:

List	Matrix
<p>Pros:</p> <ul style="list-style-type: none"> (i) Graph is sparse, efficient. (ii) Neighbors traversal, easy <p>Cons:</p> <ul style="list-style-type: none"> (i) Graph is dense, inefficient. (ii) Whether edge is present or not, not easy. 	<p>Cons:</p> <ul style="list-style-type: none"> (i) Graph is sparse, inefficient. (ii) Neighbors traversal, not easy <p>Pros:</p> <ul style="list-style-type: none"> (i) Graph is dense, efficient. (ii) Whether edge is present or not, easy

Graph Traversal Algorithm :

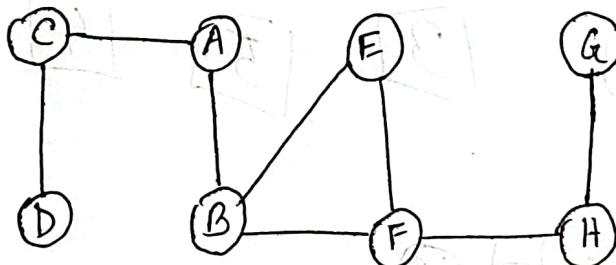
BFS [Breadth First Search] :



Queue :

1	2	4	8	3	6	7
0	1	1	1	2	2	2

Visited nodes: 1, 2, 4, 5, 3, 6, 7

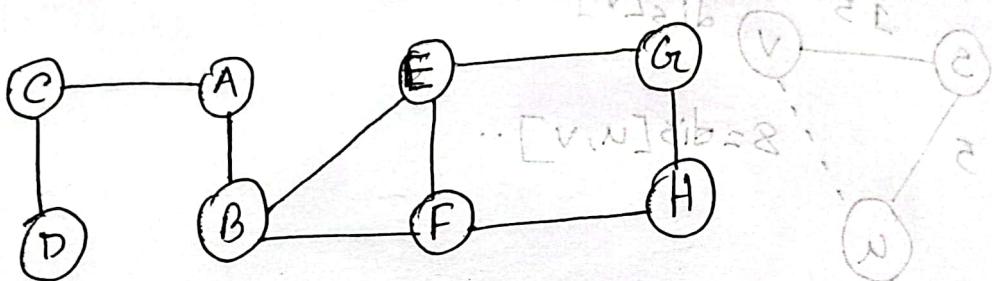


Queue :

A	B	C	E	F	D	A	G
0	1	1	2	2	2	3	4

Visited: A, B, C, E, F, D, H, G

DFS (Depth First Search):



Stack :

G	5
H	4
E	3
F	2
B	1
A	0

V	1
A	0

Visiting Nodes : A , B , E , F , H , G , V , C , D

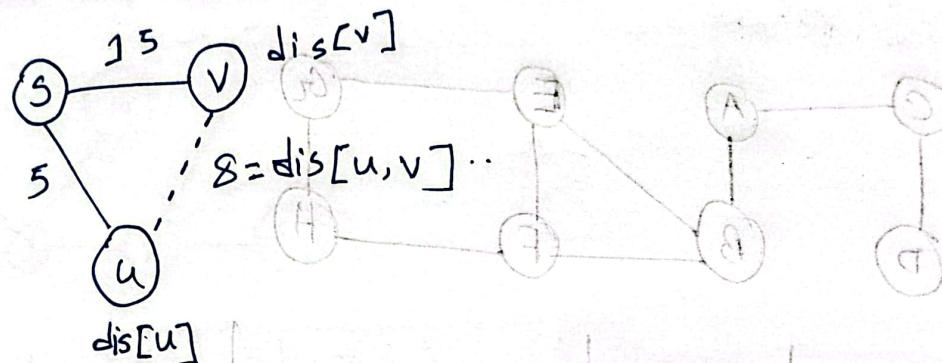
$$[V]_{\text{zib}} + [G]_{\text{zib}} = [V]_{\text{zib}}$$

Complete Nodes : G , H , F , E , B , D , C , A

Head \leftarrow Head *

Head \leftarrow Stack *

Dijkstra's Shortest Path Algorithm - (Page) 270



Relaxation condition -

$$dis[u] + dis[u, v] < dis[v]$$

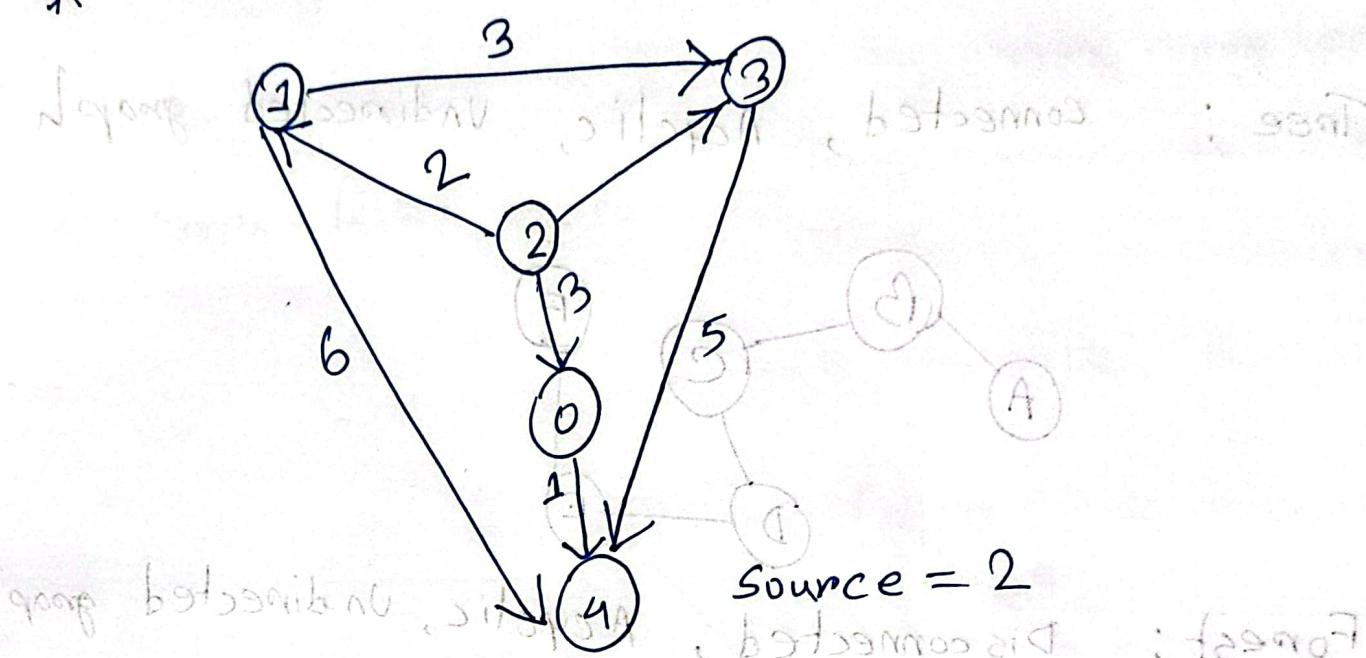
$$\Rightarrow 5 + 8 < 15$$

$$\Rightarrow 13 < 15 \quad \text{Hence } E \rightarrow A \text{ is established}$$

$$dis[v] = dis[u] + dis[u, v]$$

* Parent \rightarrow Path

* Distance \rightarrow cost



Parent

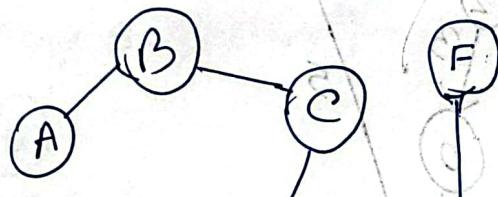
0	1	2	3	4
-2 2	-2 2	-1 3	-2 1	-2 0

Distance

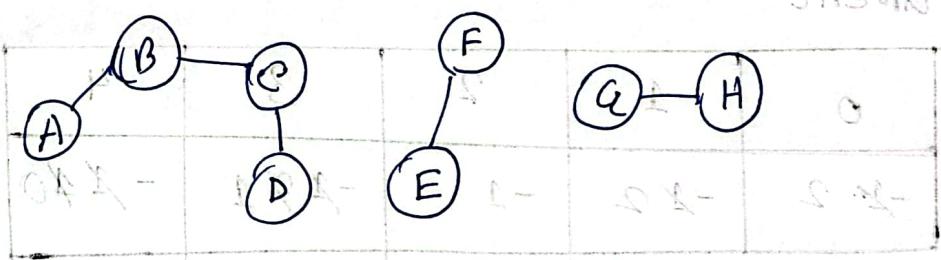
0	1	2	3	4
2	∞	∞	0	∞
1	$\min(\infty, 0+3)$ 3	$\min(\infty, 0+2)$ 2	0	$\min(6\infty, 0+7)$ 7
0	3	2	0	$\min(7, 2+3)$ 5
4	3	2	0	$\min(5, 2+6)$ 8
3	3	2	0	$\min(8, 3+1)$ 4

Trees

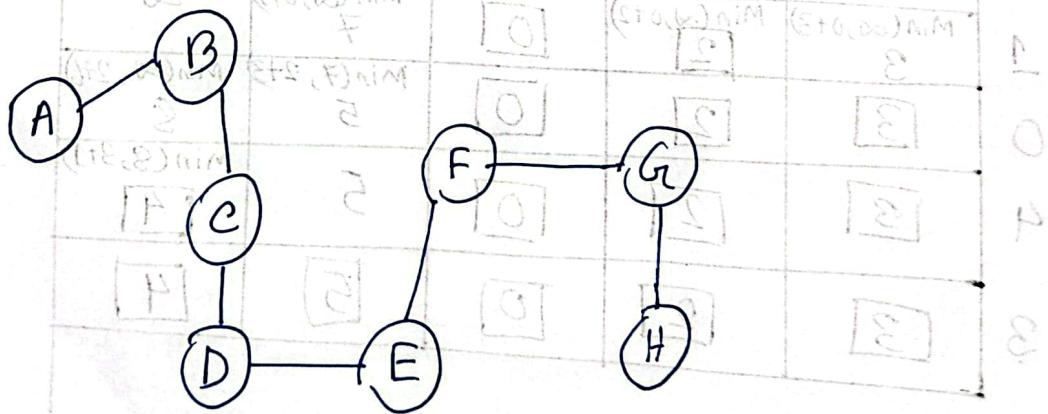
Tree : connected, Acyclic, Undirected graph



Forest : Disconnected, Acyclic, Undirected graph



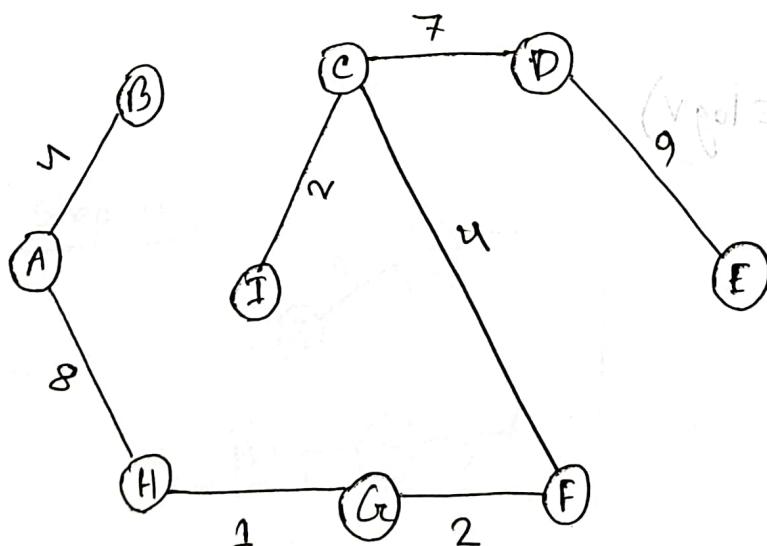
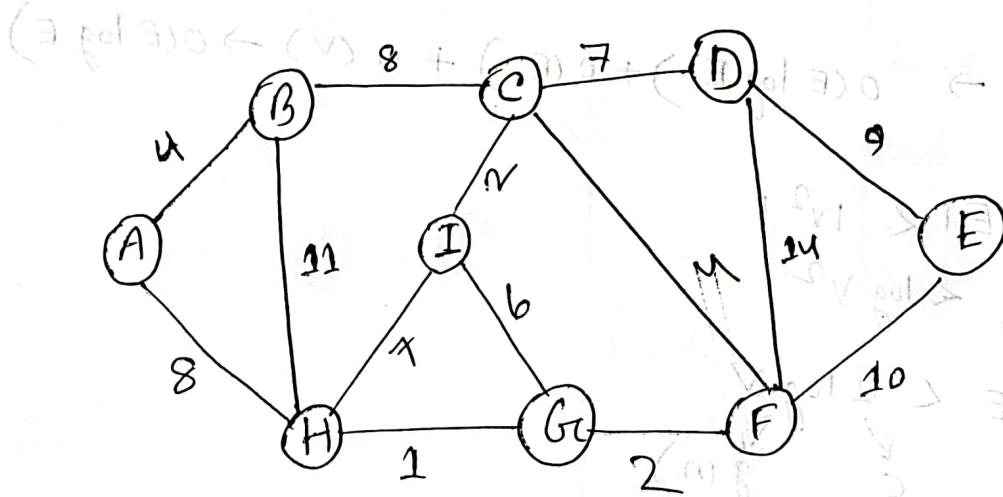
Spanning tree : A tree which includes all the vertices of a graph.



Kruskal Algorithm: Minimum weight spanning tree

Ascending order of sort Then, same edge in smaller cycle

Simulation:



Edge	Weight
G-H	1 ✓
G-F	2 ✓
C-I	2 ✓
A-B	4 ✓
C-F	4 ✓
G-I	6 x [cycle]
C-D	7 ✓
H-I	7 x [cycle]
A-H	8 ✓
B-C	8 x [cycle]
D-E	9 ✓
F-F	10 ✗
B-H	1 ✗
I-F	1 ✗

Found MST with Total weight = $4+8+1+2+4+2+7+9 = 37$

Time Complexity -

Sorting E edges $\rightarrow O(E \log E)$

Selecting E edges $\rightarrow O(E)$

Checking V nodes if is in the same component

(Disjoint sets) $\rightarrow O(V)$

Overall $\rightarrow O(E \log E) + O(E) + O(V) \rightarrow O(E \log E)$

$$\text{As, } |E| < V^2$$

$$\Rightarrow \log E < \log V^2$$

$$\Rightarrow \log E < 2 \log V$$

$$f(n)$$

$$c$$

$$g(n)$$

$$\log E = O(\log V)$$

$$O(E \log E) = O(E \log V)$$

$$V-P$$

$$V-H$$

$$V-F$$

$$V-S$$

$$V-E$$

$$V-O$$

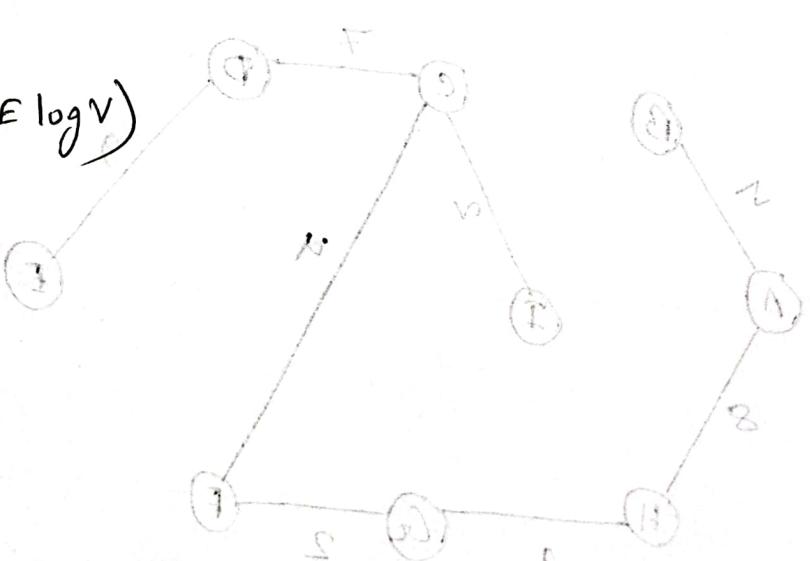
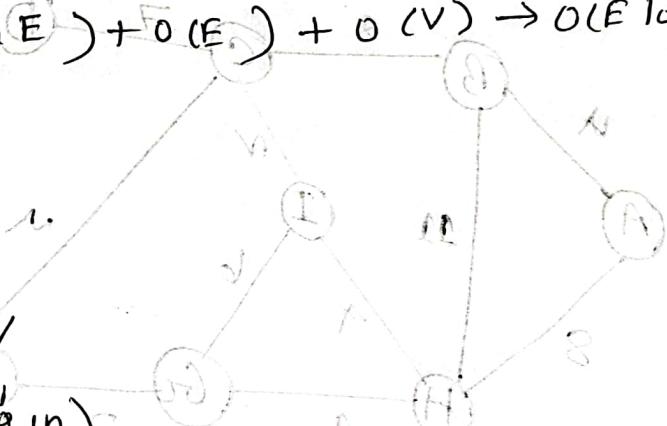
$$V-L$$

$$V-P$$

$$V-D$$

$$V-T$$

$$V-B$$

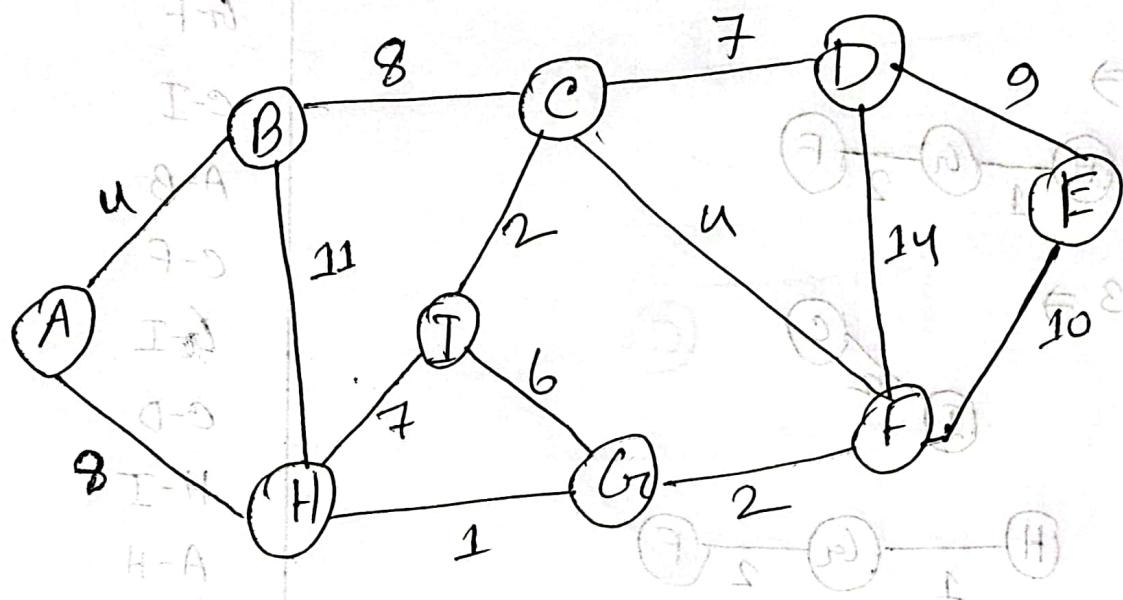


$$H+S+L+T+V+N = \text{Total weight of tree}$$

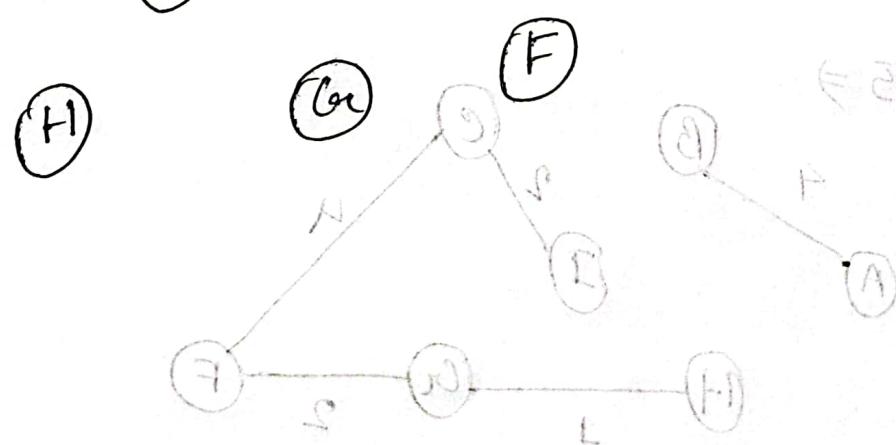
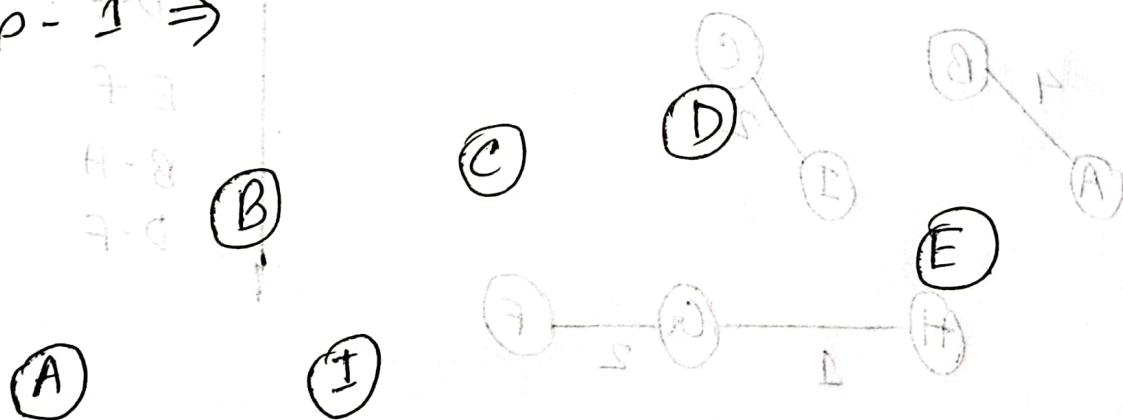
$$48$$

Kruskal Algorithm :

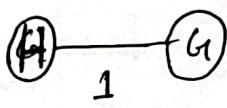
Simulation \Rightarrow



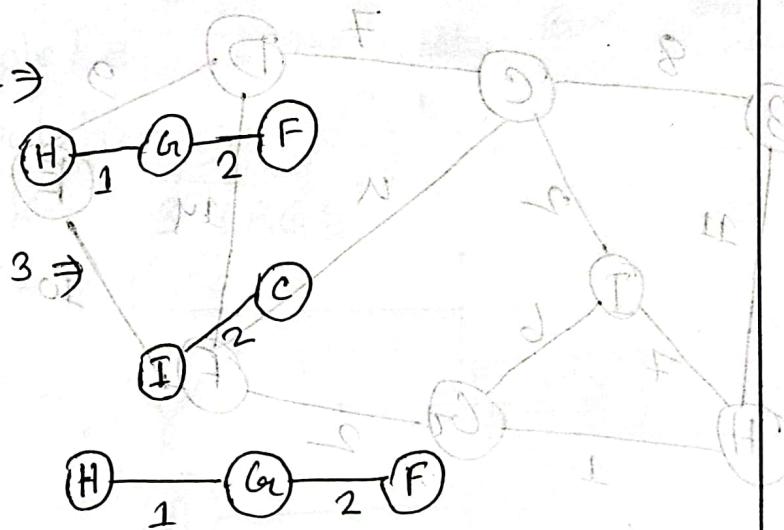
Step - 1 \Rightarrow



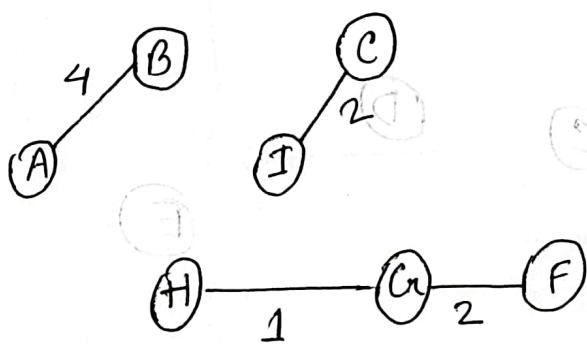
Step-1 \Rightarrow



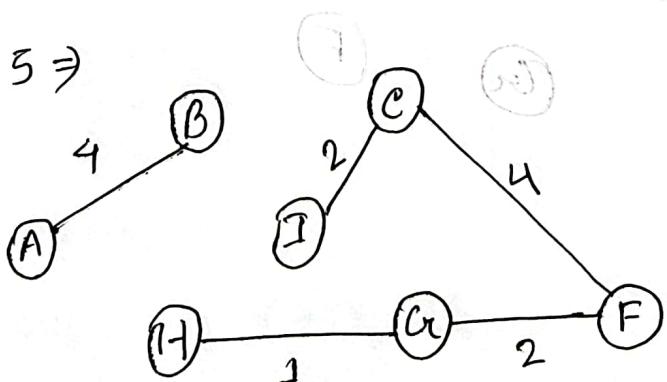
Step-2 \Rightarrow



Step-4 \Rightarrow

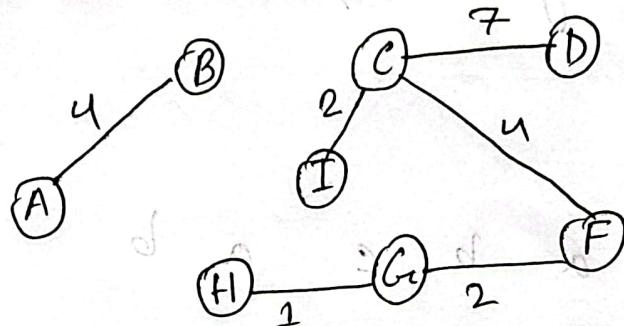


Step-5 \Rightarrow



Edge	Weight
G-H	1 ✓
G-F	2 ✓
C-I	2 ✓
A-B	4 ✓
C-F	4 ✓
G-I	6 X [cycle]
C-D	7 ✓
H-I	7 X [cycle]
A-H	8 ✓
B-C	8 X [cycle]
D-F	9 ✓
E-F	10 X
B-H	11 X
D-F	14 X

Step - 6 \Rightarrow



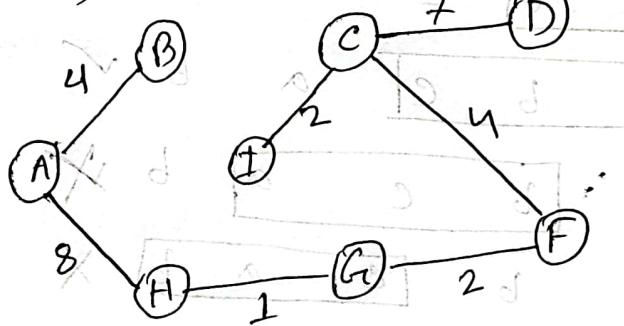
smallest grid size prime

\Leftarrow select stand / stick

\Rightarrow ST fix \rightarrow

S + d \Rightarrow E.g. max

Step - 7 \Rightarrow



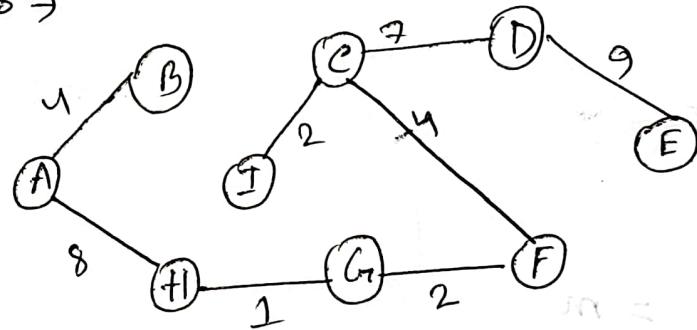
\Rightarrow E - gate

\Rightarrow E - gate

\Rightarrow S - gate

\Rightarrow N - gate

Step - 8 \Rightarrow



L : twigs

E : pixels/gates

N = size fix

Found MST with total weight = $4+8+1+2+4+2+7+9 = 36$

(M-N) at 0 file

L + M - N file total

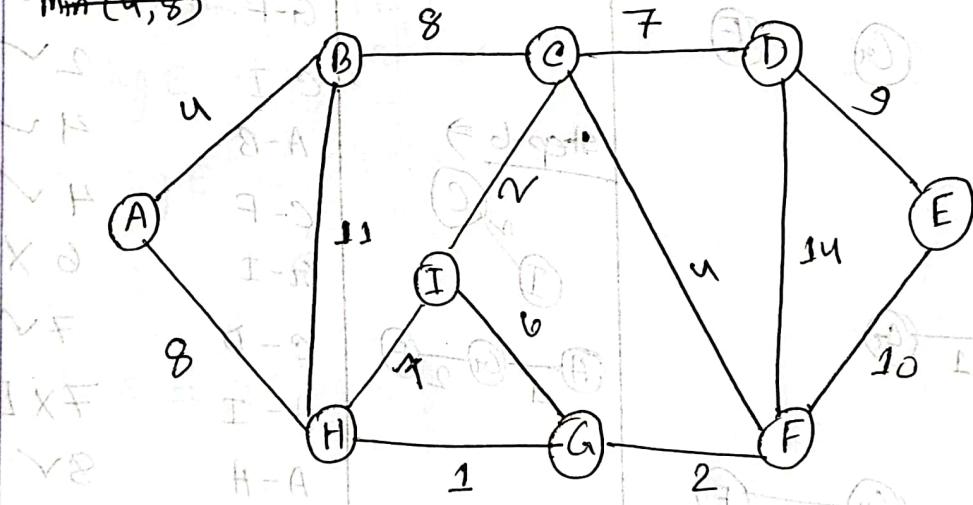
M * (L + M - N) = pixels/gates

Prim's Algorithm -

Source node : ~~start~~, Then minimum weight Arc
 First source node add ~~2nd~~, then minimum cost
 প্রথম করে কোন রাস্তা দ্বিতীয় রাস্তা
 দ্বিতীয় রাস্তা

simulation -

Adding A to MST
 $\min(4, 8)$



Steps \Rightarrow (1) Add A to MST.

(2) $\min(4, 8)$, Add B, A-B to MST

(3) $\min(8, 11, 8)$, Add C, B-C to MST

(4) $\min(8, 11, 2, 6, 7)$, Add I, C-I to MST.

(5) $\min(8, 11, 4, 7, 6, 7)$, Add F, C-F to MST.

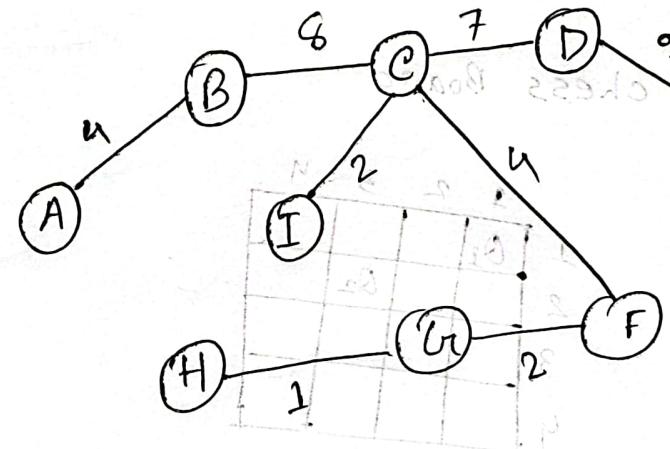
(6) $\min(8, 11, 7, 6, 7, 2, 10, 14)$, Add G, F-H to MST.

(7) $\min(8, 11, 7, 7, 10, 14)$, Add H, G-H to MST.

(8) $\min(7, 10, 14)$, Add D, C-D to MST.

(g) Min(9, 10), Add E, D-E to MST.

Found MST with total weight = $4+8+2+4+2+1+7+9 = 37$



Complexity \Rightarrow

Using Adjacency

Matrix - $O(V^2)$

Using Adjacency List

List - $O(E \log V)$

Backtracking - All Possible solutions [23/2021]

N QUEEN Problem -

same row, same column ~~row~~ Diagonally

2x10 max at 1

Simulation - 4x4 chess Board

	1	2	3	4
1	Q ₁			
2				
3				
4				

	1	2	3	4
1	Q ₁			
2		Q ₂		
3				
4				

	1	2	3	4
1	Q ₁			
2			Q ₂	
3	X	X	X	X
4				

	1	2	3	4
1	Q ₁			
2		Q ₂		
3				
4				

	1	2	3	4
1	Q ₁			
2				
3				
4				

	1	2	3	4
1	Q ₁			
2				
3				
4				

	1	2	3	4
1	Q ₁			
2				
3				
4				

	1	2	3	4
1	Q ₁			
2				
3				
4				

Q₁ Q₂
2 4

Q₃ Q₄
1 3

	1	2	3	4
1				Q ₁
2				
3				
4				

	1	2	3	4
1				Q ₁
2		Q ₂		
3				
4				

	1	2	3	4
1			Q ₁	
2		Q ₂		
3				Q ₃
4				

	1	2	3	4
1				Q ₁
2		Q ₂	(Q ₂)	
3				Q ₃
4				Q ₄

Q₁
3

Q₂
1

4

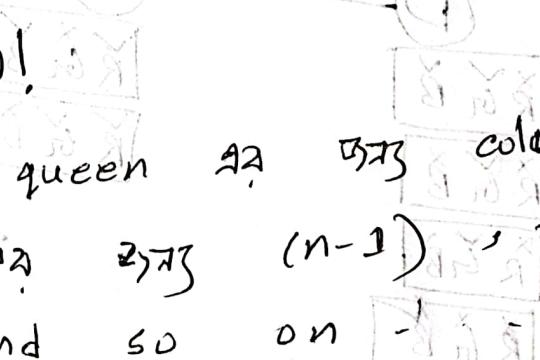
Q₄
2

complexity -

Backtracking

$$T(n) = n!$$

Because 1st queen (1), 2nd queen (2), 3rd queen (3) and so on



then 3rd
column n.

→ an algorithm

क्वीन (n)

क्वीन (2)

क्वीन (1)

क्वीन (3)

क्वीन (2)

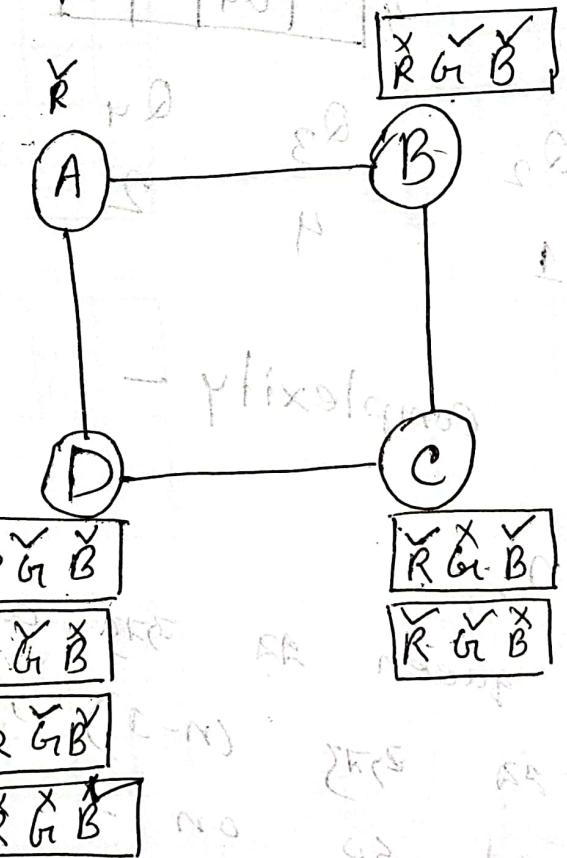
क्वीन (1)

Graph Coloring -

Adjacent node same color
complete $2^{(n)}$ Bt. Box

Simulation \Rightarrow :

$$k = 3 \text{ (RGB)}$$



Combinations —

- (1) RGRG
- (2) RGRR
- (3) RGBG
- (4) RBRG
- (5) RBRB
- (6) RBGB

String Matching Algorithm \Rightarrow

Naive / Brute Force \Rightarrow

* Text $T \Rightarrow x \ a \ b \ . \ c \ a \ b$

Pattern $P \Rightarrow a \ b \ - \ c$

Step - 1 :

$x \ a \ \boxed{b}$

$c \ a$

$b \ x$

Step - 2 :

$a \ \boxed{b} \ - \ c$

$b \ \checkmark$

Step - 3 :

$b \ c \ - \ a$

$b \ \times$

Step - 4 :

$c \ - \ a \ b$

\times

Output : 1

complexity \Rightarrow

Text size = m

Pattern size = n

shift 0 to $(n - m)$

Total shift $n - m + 1$

Complexity = $O(n - m + 1) \times m$

Rabin-Karp Algorithm \Rightarrow straightforward and faster

Rabin Karp Algorithm \Rightarrow finds all occurrences

Keywords \Rightarrow Hash function, Hash code, Rolling Hash,
Match hit, Spurious hit.

* Text T : $x \ a \ b$ (no $a \in b$)

Pattern P : $a \ b \ c$

\Rightarrow Let,

$$a=1, b=2, c=3 \quad x = (L+m-1) \cdot 25 + 2 = 26$$

Text T : $x \ a \ b \ c \ a \ b$

Pattern P :

$$\begin{array}{c|ccc} a & b & c \\ \hline 1 & 2 & 3 \end{array}$$

$$1+2+3 = 6 \Rightarrow \text{Hash code}$$

Text T :

$$\begin{array}{c|ccc} x & a & b \\ \hline 2 & 1 & 2 \end{array}$$

$$2+1+2 = 5$$

a

$b \Rightarrow \text{Match Hit}$
[Hash code and pattern same]

$$\begin{array}{c|ccc} x & a & b & c \\ \hline 2 & 3 & 1 & 2 \end{array}$$

$$2+3+1 = 6$$

$b \Rightarrow \text{Spurious Hit}$

$$\begin{array}{c|ccc} x & a & b & c \\ \hline 3 & 1 & 2 & 2 \end{array}$$

$$3+1+2 = 6$$

[Hash code same, pattern different]

Output : 1

Rabin Karp Algorithm ~~drawback~~

spurious hit ~~last~~ $O(m)$ performance ~~last~~

spurious hit ~~last~~ way m Hash Function

strong ~~last~~ $O(n)$ Next window hash \leftarrow approach

Complexity: $O(n+m)$ \rightarrow find substring, find pattern

Initial Hash Calculation /

Processing time $\Rightarrow O(m) \times n : T \text{ Text}$

shifting - $O(n-m+1)$ \rightarrow $n-m+1$ matches

Worst - $O(n-m+1) \times m$

Avg / Best - $O(n-m+1) \Rightarrow O(n)$

Total complexity -

Worst complexity = $O(n-m+1) \times m + O(m)$

Avg / Best = $O(n) + O(m)$

String match simulation [Rabin Karp] \Rightarrow Text

way - 2

Text T : $x \ a \ b \ c \ a \ b$

Pattern P : $\underline{a \ b = L + C}$

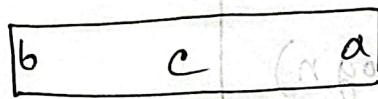
$$1 + 2 + 3 = 6$$

$$\begin{array}{r} |x \ a \ b| + c + 8a \ b \ x \\ 2n + 1 + 2 = 27 \end{array}$$

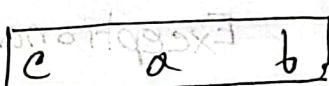
L: length



$$2^7 - 2^4 + 3 = 6$$



$$6 - 1 + 1 = 6$$



$$6 - 2 + 2 = 6$$

Output: 1

Approximation Algorithm is a algorithm which deals with NP-hard problems.

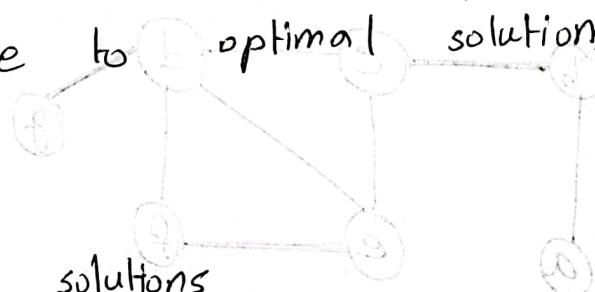
An Approximation Algorithm is a

way of dealing with NP completeness

optimization problem to determine such solutions

which are close to optimal solutions in polynomial

time.



$C \rightarrow$ cost of solutions

$C^* \rightarrow$ cost of optimal solutions.

$P(n) \rightarrow$ Approximation Ratio

Maximization $\Rightarrow \frac{C^*}{C} \leq P(n)$

Minimization $\Rightarrow \frac{C}{C^*} \leq P(n)$

Merge sort $\Rightarrow O(n \log n)$ } Solving process & time complexity known.

Bubble sort $\Rightarrow O(n^2)$ } Deterministic + Polynomial

Binary search $\Rightarrow O(\log n)$ } time complexity

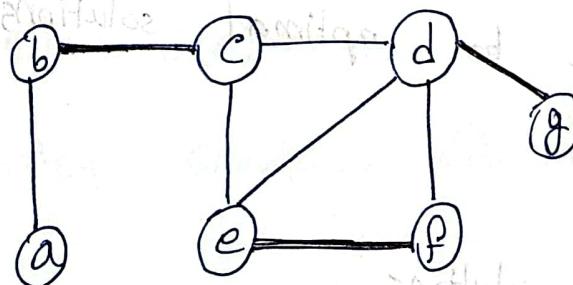
NP Hard \Rightarrow Exceptional Problem \Rightarrow solving process fully यात्रा वर इसत प्र०

Non deterministic Polynomial

\Rightarrow Exponential time complexity $\Rightarrow 2^n$ [Graph coloring]

NP Hard problem solve 20112 CSE 7
time complexity यानान्तरे तय कर रखा,

VERTEX COVER PROBLEM \Rightarrow [edge के लिए 322r bold 2012r]



V: φ

E: { (b,c), (a,b), (c,d), (e,f), (d,e), (c,e),
(d,g), (d,f) }

$$V: \emptyset \cup \{b, c\}$$

$$: \emptyset \cup \{b, c\} \cup \{e, f\}$$

$$: \{b, c, e, f\} \cup \{d, g\}$$

$$: \{b, c, e, f, d, g\} = \boxed{6} \Rightarrow C$$

एक vertex cover problem solve करते हैं तो इसका ans

प्रत्येक edge के 2 vertices पर approximation algorithm

$$V: \emptyset$$

$$V: \emptyset \cup \{b, c\}$$

$$: \{b, c\} \cup \{d, f\}$$

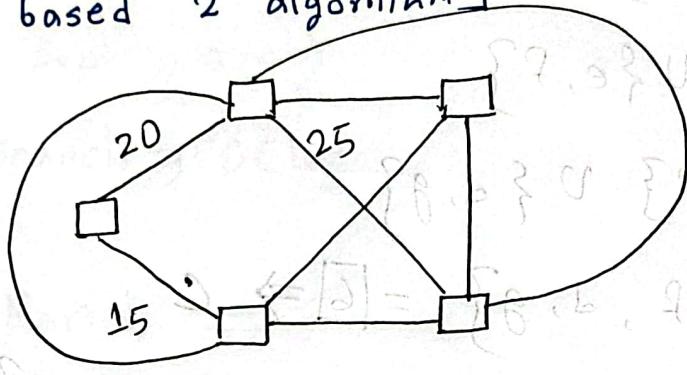
$$: \{b, c, d, f\} = \boxed{4} \Rightarrow C^*$$

इसका ans

प्रत्येक edge के 2 vertices पर approximation algorithm

■ TRAVELLING SALESMAN PROBLEM

MST based 2 algorithm



20(2) salesman for city cost visit
 cost A. back track minimizes

Steps \Rightarrow

(1) Combine triangle inequality [2nd step question A part 2020 CA 1]

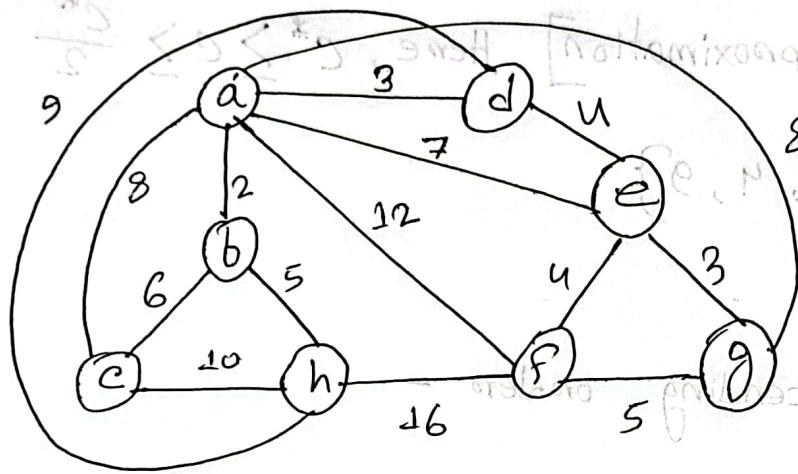
Example - $a \rightarrow (a, c) \rightarrow (b, c)$

$$a \rightarrow b \rightarrow c \\ \text{dis}(a, c) \leq \text{dis}(a, b) + \text{dis}(b, c)$$

(2) Apply Kruskal Algorithm

(3) Apply DFS

(4) Add starting node at the end of DFS Traversal.



: mohamed mohamed fedouz

Applying Kruskal Algorithm -

ad
ac
af
ab
ac
ag

ab ✓

~~ad~~ 3 ✓

eg ✓

de ✓

ef ✓

bh ✓

~~fg~~ 5 X

bc ✓

~~ae~~ 7 X

ac 8 X

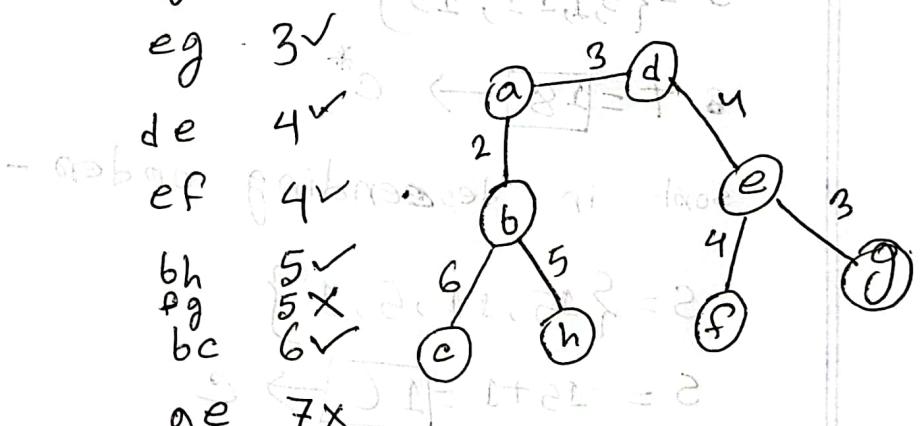
ag 8 X

dh 9 X

ch 10 X

af 12 X

hf 16 X



Applying DFS -

visited - a, b, c, h, d, e, f, g, a

TSP approximate solution

Subset Sum Problem :

$\left[\frac{1}{2} \text{ Approximation} \right]$ Hence, $C^* \geq C \geq \frac{C^*}{2}$

$$* S = \{2, 5, 1, 4, 9\}$$

$$T = \{10\}$$

Sort in descending order -

$$S = \{9, 5, 4, 2, 1\}$$

$$\text{sum} = 9 + 1 = 10$$

→ mithilfe einer Tabelle ermittelt

$$* S = \{5, 1, 11, 15\}$$

$$S + T = \boxed{18} \rightarrow C^*$$

Sort in descending order -

$$S = \{15, 11, 5, 1\}$$

$$S = 15 + 1 = \boxed{16} \rightarrow C$$

$$C^* \geq C \geq \frac{C^*}{2}$$

$$\Rightarrow 18 \geq 16 \geq 9$$



XOL
XLB
XLE