Topic Name : _____ Day : _____

Time : Date : / /

Algorithm Analysis and Design

Introduction to Algorithms:

Algorithm → step-by-step procedure for solving a computational problem.

Criteria:

1. Input - 0 or more
2. Output - 1 or more
3. Definiteness - clear instructions and unambigious
4. Finiteness - certain time / space
5. Effectiveness.

Area of study

1. How to devise algorithms.
2. How to validate algorithms.
   → If works for generalized i/p
3. How to analyze algo. (Performance analysis)
4. How to test a program (Performance Measurement)

Why study?

1. Efficiency
2. Compare the techniques (space + time complexity)

Topic Name : _____ Day : _____

Time : Date : / /

3. Knowing different techniques.
4. For dynamic algorithm.
5. It's programming pre step.

Application:

1. Multimedia
2. Internet → clouding / routing
3. Communication
4. Transportation
5. science

Different problems

1. Sorting
2. Searching
3. String processing
4. Graph problems
5. Geometric "
6. Numerical "
7. Probabilistic problems.

Algorithm Design strategy

1. Divide and conquer
2. Incremental
3. Dynamic programming
4. Randomized "
5. Probabilistic "
6. Backtracking
7. Branch and bound.

Topic Name : _____ Day : _____

Time : Date : / /

How to write an algorithm?

Algorithm swap(a,b)

Begin

temp ← a or temp := a

a ← b

b ← temp

end

↳ for assignment operator

How to analyze an algorithm?

Criterias:

1. Time (Must be faster)
2. Space (Memory space)
3. Network
4. Power consumption
5. CPU Registers

```
Algorithm swap(a,b)
{
    temp = a;
    a = b ;
    b = temp;
}
```

	taking units of time	space
temp = a; ⟶	1	a − 1
a = b ; ⟶	1	b − 1
b = temp; ⟶	1	temp − 1
	f(n) = 3 constant time.	s(n) = 3 word

x = 5 * a + 6 * b ⟶ 1 (also take 1 unit time)

Topic Name : _____ Day : _____

Time : Date : / /

Frequency count method

(1) Algorithm sum (A, n)

```
{
    s = 0;                          ——— 1
    for( i=0; i<n; i++)             ——— n+1
    {                                   (1, n+1, n)
      s = s + A[i];                 ——— n
    }
    return s;                       ——— 1
}
```

Time	
	1
	n+1
	n
	1
$f(n) = 2n+3$	
$O(n)$ [order of ...]	

Space	
A	n
n	1
s	1
i	1
$s(n) = n+3$	
$O(n)$	

Space : $n+3$, $O(n)$

Time : $2n+3$, $O(n)$

(2) Algorithm Add (A, B, n) {

```
for (i=0; i<n; i++){            ——— n+1
  for (j=0; j<n; j++){          ——— n x (n+1)
    c[i,j] = A[i,j] + B[i,j];   ——— n x n
  }
}
}
```

Time	
	$n+1$
	$n \times (n+1)$
	$n \times n$
$f(n) = 2n^2+2n+1$	
$O(n^2)$	

Space	
A	n^2
B	n^2
C	n^2
n	1
i	1
j	1
$s(n) = 3n^2+3$	
$O(n^2)$.	

Topic Name : _____ Day : _____

Time : Date : / /

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

for (j = 0; j < i ; j ++) {

// stmt;

}

}

i	j	no of time
0	0 x	0
1	0 1 x	1
2	0 1 2 x	2
3	0 1 2 3 x	3
⋮	⋮	⋮
n	n (n+1) x	n

Time complexity,

$1 + 2 + 3 + \cdots + n$

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

$\therefore f(n) = \frac{n^2 + n}{2}$ $O(n^2)$.

(4) p = 0;

for (i = 1; p <= n ; i ++) {

p = p + i;

}

i	p
1	0+1
2	1+2
3	1+2+3
4	1+2+3+4
⋮	
K	1+2+3+⋯+K

Assume $p > n$

$\because p = \frac{K(K+1)}{2}$

$\frac{K(K+1)}{2} > n$

$\Rightarrow K^2 > n$ $O(\sqrt{n})$

$\therefore K > \sqrt{n}$

Topic Name : _____ Day : _____

Time : Date : / /

⑤ for (i = 1; i < n; i = i * 2)

{

stmt;

}

Assume, $i > K = n$

$\therefore i = 2^K$

so, $2^K >= n$

$\therefore 2^K = n$

$\Rightarrow K = \log_2 n$ $O(\log_2 n)$.

Value will be taken as $= \lceil \log n \rceil$

i
1
$1 \times 2 = 2$
$1 \times 2 \times 2 = 2^2$
$2 \times 2 \times 2 = 2^3$
⋮
2^K

⑥ for (i = n; i >= 1; i = i/2)

{

stmt;

}

Assume $i < 1$

$\therefore n/2^K < 1$

$\Rightarrow n/2^K = 1$

$\Rightarrow n = 2^K$

$\therefore K = \log_2 n$ $O(\log_2 n)$.

i
n
$n/2$
$n/2^2$
$n/2^3$
⋮
$n/2^K$

Topic Name : _____ Day : _____

Time : Date : / /

(7) for (i=0; i*i<n ; i++)

{

stmt;

}

i*i < n → executes while this is true

i*i >= n → Terminates when this is true

∴ $i^2 = n$

⇒ $i = \sqrt{n}$ $O(\sqrt{n})$

(8) for (i=0; i<n ; i++)

{

... ——————— n

}

for(j=0; j<n ; j++)

{

... ——————— n

}

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

(9) p = 0;

for (i=1; i<n; i = i*2) { ... } → $p = \log n$

for (j=1; j<p; j = j*2) { ... } → $\log p = \log \log (n)$

∴ $O(\log \log n)$.

(10) for (i=0; i<n ; i++){ ——— n

for (j=0; j<n ; j=j*2){ — n x log n

stmt; ——————— n x log n

}

}

$2n \log n + n$, $O(n \log n)$

Topic Name : _____ Day : _____

Time : Date : / /

'If' and 'While' analysis

(11)
```
i = 0;          ——— 1
while (i<n){    ——— n+1
  stmt;         ——— n
  i++;          ——— n
}
```

$f(n) = 3n+2$

$O(n)$

(12)
```
a = 1;
while (a<b)
{
  stmt;
  a = a*2;
}
```

a
1
$1 \times 2 = 2$
$2 \times 2 = 2^2$
$2 \times 2 \times 2 = 2^3$
:
2^K

Terminate $a \geq b$

$\therefore a = 2^K$

$2^K \geq b$

$2^K = b$

$\therefore K = \log_2 b$

$O(\log n)$

(13)
```
i = n;
while (i>1)
{
  stmt;
  i = i/2;
}
```

i
n
$n/2$
$n/2^2$
:
$n/2^K$

Terminate $\frac{n}{2^K} \geq 1$

$\Rightarrow \frac{n}{2^K} = 1$

$\Rightarrow n = 2^K$

$\Rightarrow \therefore K = \log_2 n$

$\therefore O(\log n)$.

Topic Name : _____ Day : _____

Time : Date : / /

(14)
```
i=1;
K=1;
While (K<n)
{
 stmt;
 K=K+i;
 i++;
}
```

i	K
1	1
2	1+1 = 2
3	2+2
4	2+2+3
5	2+2+3+4
	⋮
m	2+2+3+4+---m
	$\frac{m(m+1)}{2}$

Terminate,

$K \geq n$

$\Rightarrow \frac{m(m+1)}{2} \geq n$ $\quad O(\sqrt{n})$

$\Rightarrow m^2 \geq n \quad \therefore m = \sqrt{n}$

(15)
```
while (m!=n)
{
  if (m>n)
    m=m-n;
  else
    n=n-m;
}
```

execute for $\frac{n}{2}$ time maximum

$\therefore O(n) \rightarrow$ max

$O(1) \rightarrow$ min

Topic Name : _____ Day : _____

Time : Date : / /

'if' is present on a statement or algorithm then it maybe the best case or worst case depending upon the condition given/fulfilled.

```
Algorithm Test(n)
{
    if (n<5)
    {
      cout << n;
    }
    else
    {
      for (i=0; i<n; i++)
      {
        cout << i;
      }
    }
}
```

1, O(1) if this condition is fulfilled → so, best case

n, O(n) if condition is not fulfilled. → so, worst case

Best case: O(1)

Worst case: O(n)

Topic Name : _____ Day : _____

Time : Date : / /

Classes of functions and comparison

$O(1) \longrightarrow$ constant

$O(\log n) \rightarrow$ logarithmic

$O(n) \longrightarrow$ Linear

$O(n^2) \longrightarrow$ Quadratic

$O(n^3) \longrightarrow$ Cubic

$O(2^n) \longrightarrow$ exponential

$O(3^n)$

$O(n^n)$

$f(n) = 2 \longrightarrow O(1)$

$f(n) = 5$

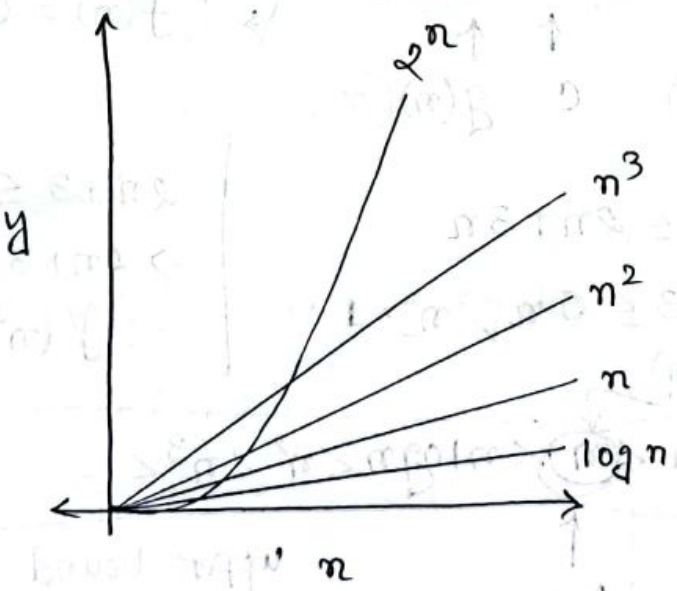
$f(n) = 5000$

$f(n) = 2n+1 \longrightarrow O(n)$

$f(n) = 500n + 700$

$f(n) = \frac{n}{50} + 6$

$1 < \log n < \sqrt{n} < n < n\log n < n^2 < n^3 < \cdots < 2^n < 3^n < \cdots < n^n$



Topic Name : _____ Day : _____

Time : Date : / /

Asymptotic Notations

O big-oh upper bound of function

Ω big-omega Lower bound "

θ theta ~~Avegar~~ Average "

Big-oh

The function $f(n) = O(g(n))$ iff $\exists$ +ve constant c, n_0 such that $f(n) \le c * g(n)$ $\forall\ n \ge n_0$

Ex: $f(n) = 2n+3$

$2n+3 \le 10n,\ n \ge 1$

$f(n)$ c $g(n)$

$\therefore f(n) = O(n)$

Trick

$2n+3 \le 2n+3n$

$\Rightarrow 2n+3 \le 5n,\ n \ge 1$

$f(n)$

$2n+3 \le 2n^2+3n^2$

$\Rightarrow 2n+3 \le 5n^2$

$\therefore f(n) = O(n^2)$.

$1 < \log n < \sqrt{n} < n < n\log n < n^2 < n^3 < \dots < 2^n < 3^n < \dots < n^n$

lower bound — big-Ω

Average bound — θ

upper bound — big-O

Topic Name : _____ Day : _____

Time : Date : / /

For the big-oh we should write the closest function. If we write bigger function then it's true but not useful.

Omega - Ω

$$f(n) \geq c * g(n)$$

ex: $f(n) = 2n + 3$

$2n + 3 \geq 1 \times n \quad \forall n \geq 1$

$f(n)$ c $g(n)$

$\therefore f(n) = \Omega(n)$ ← Nearest

also we can say, $f(n) = \Omega(\log n)$

Not true, $\Omega(n \log n)$

Theta - notation

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

ex: $f(n) = 2n + 3$

$1 \times n \leq 2n + 3 \leq 5n$

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

$\therefore f(n) = \Theta(n)$

N:B:

We can use any notation for best or worst case.

Topic Name : _____ Day : _____

Time : Date : / /

Examples with some functions:

1. $f(n) = 2n^2 + 3n + 4$

$2n^2 + 3n + 4 \geq 1 \times n^2 \quad \Omega(n^2)$

and, $2n^2 + 3n + 4 \leq 2n^2 + 3n^2 + 4n^2$

$\Rightarrow 2n^2 + 3n + 4 \leq 9n^2 \quad O(n^2)$

and, $1 \times n^2 \leq 2n^2 + 3n + 4 \leq 9n^2$

$\therefore \Theta(n^2)$.

2. $f(n) = n^2 \log n + n$

$1 \times n^2 \log n \leq n^2 \log n + n \leq 10\, n^2 \log n$

$\therefore O(n^2 \log n)$

$\Omega(n^2 \log n)$

$\Theta(n^2 \log n)$

3. $f(n) = n! = n(n-1)(n-2) \cdots 3 \times 2 \times 1$

$1 \times 1 \times 1 \times 1 \times 1 \cdots \leq 1 \times 2 \times 3 \times \cdots \times n \leq n \times n \times n \times n \cdots \times n$

$1 \leq n! \leq n^n$

$O(n^n)$

$\Omega(1)$

} cannot find the average bound.

Θ can't be defined.

Topic Name : _____ Day : _____

Time : Date : / /

4. $f(n) = \log n!$

$\log(1 \times 1 \times 1 \ldots) \leq \log(1 \times 2 \times 3 \times \ldots n) \leq \log(n \times n \times n \ldots n)$

$1 \leq \log n! \leq \log n^n$

$O(n \log n)$

$\Omega(1)$

Can't find any Θ (Average bound).

Properties of asymptotic notations

General properties:

1. If $f(n)$ is $O(g(n))$ then $a * f(n)$ is $O(g(n))$
2. " $f(n)$ " Ω ----
3. " $f(n)$ " Θ ---- → same for Ω, Θ

Reflexive properties:

1. if $f(n)$ is given then $f(n)$ is $O(f(n))$

Transitive properties:

if $f(n)$ is $O(g(n))$ and $g(n)$ is $O(h(n))$

then, $f(n) = O(h(n))$.

Topic Name : _____ Day : _____

Time : Date : / /

Symmetric :

↳ True only for θ notations

if $f(n)$ is $\theta(g(n))$ then $g(n) = \theta(f(n))$

Transpose symmetric :

↳ True for O, Ω

if $f(n) = O(g(n))$ then $g(n)$ is $\Omega(f(n))$

ex: $f(n) = n$ $g(n) = n^2$

then n is $O(n^2)$ and

n^2 is $\Omega(n)$

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

$f(n) = \Omega(g(n))$

then, $f(n) = \theta(g(n))$

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

$d(n) = O(e(n))$

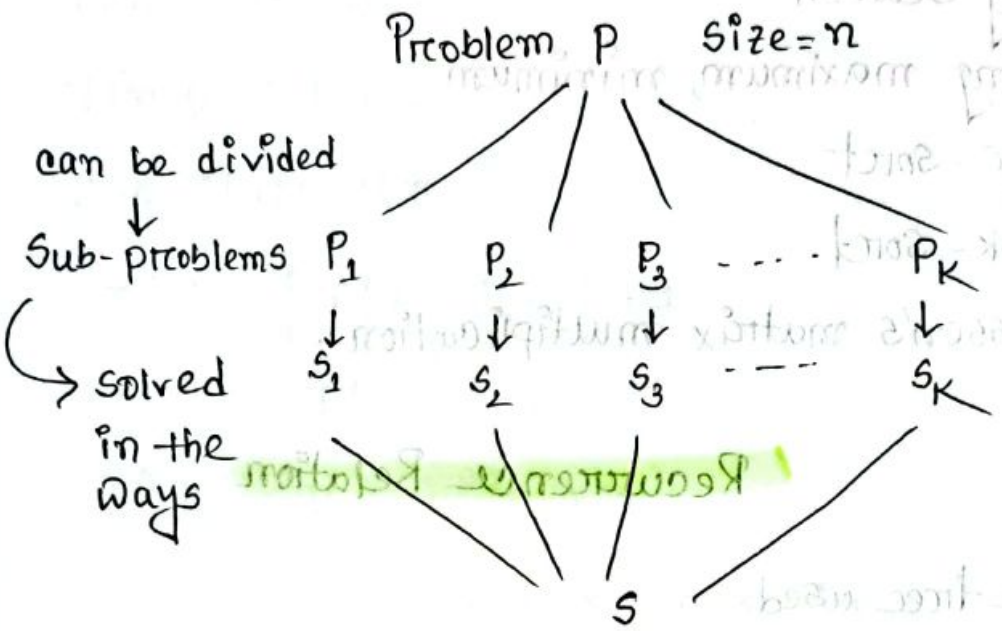
then $f(n) + d(n) = O(\max(g(n), e(n)))$

$f(n) * d(n) = O(g(n) * e(n))$

Topic Name : _____ Day : _____

Time : Date : / /

Divide and Conquer



combine the solutions to find main solution.

Technique:

```
DAC (P) {
  if (small P) {
      S(P);
  }
  else {
    divide P into P1, P2, P3 .... PK
    Apply DAC(P1), DAC(P2) .... DAC(PK)
    combine (DAC(P1), DAC(P2) .... DAC(PK))
  }
}
```

Topic Name : _____ Day : _____

Time : Date : / /

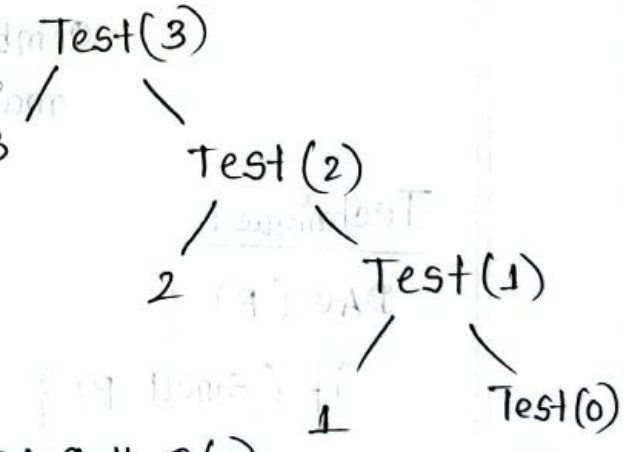
Problems:

1. Binary search
2. Finding maximum, minimum
3. Merge-sort
4. Quick-sort
5. Strassen's matrix multiplication.

Recurrence Relation

Recursive tree used.

(1)
```
void Test(n){ — T(n)
    if(n>0){
        cout << n; — 1
        Test(n-1); — T(n-1)
    }
}
```



$f(n) = n+1$ call $O(n)$ for, n

Total:

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

solving this,

$$T(n) = \begin{cases} 1 & n = 0 \\ T(n-1) + 1 & , n > 0 \end{cases}$$

Topic Name : _____ Day : _____

Time : Date : / /

Using substitution method:

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

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

Substitute $T(n-1)$

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

$= T(n-2) + 2$

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

⋮ continue for K times

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

Assume, $n - K = 0$

$\therefore n = K$

so, $T(n) = T(0) + n$

$= 1 + n$

Complexity / uppere bounds, $O(n)$.

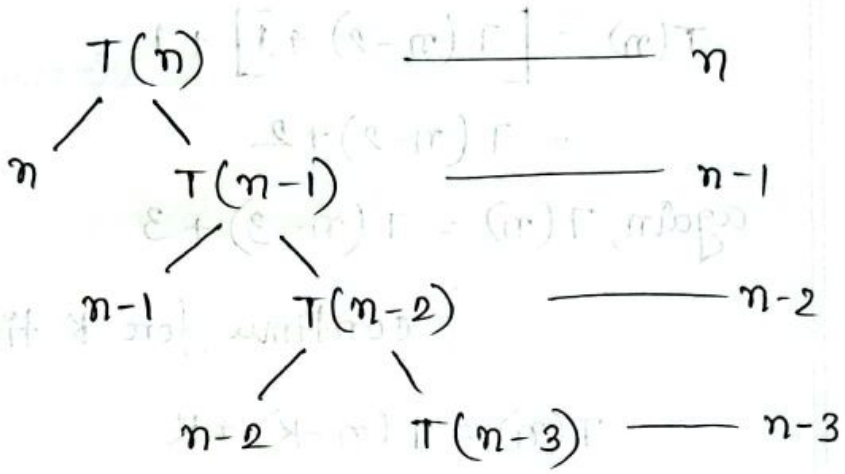
Topic Name : _____ Day : _____

Time : Date : / /

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

$$T(n) = \begin{cases} 1 & , n = 0 \\ T(n-1) + n & , n > 0 \end{cases}$$

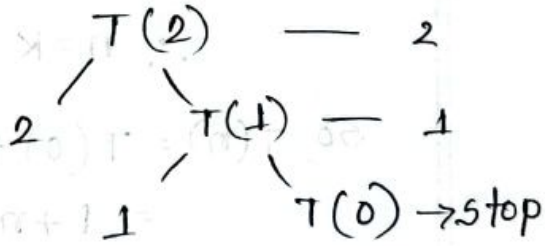
Recursion tree:



$0 + 1 + 2 + \cdots + (n-1) + n$

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

$= O(n^2)$.



using tree method to solve the problem of recurrance relation.

Topic Name : _____ Day : _____ Time : Date : / /

Solving with substitution:

$T(n) = T(n-1) + n$ —— (1)

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

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

$\Rightarrow T(n) = T(n-2) + (n-1) + n$ —— (2)

again, $T(n) = T(n-3) + (n-2) + (n-1) + n$ —— (3)

$\vdots$

$T(n) = T(n-K) + (n-K+1) + (n-K+2) + \cdots + (n-1) + n$ —— (4)

Assume, $n-K = 0$

$\therefore n = K$

$\therefore T(n) = T(n-n) + (n-n+1) + (n-n+2) + \cdots + (n-1) + n$

$= T(0) + 1 + 2 + \cdots + (n-1) + n$

$= T(0) + \frac{n(n+1)}{2}$

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

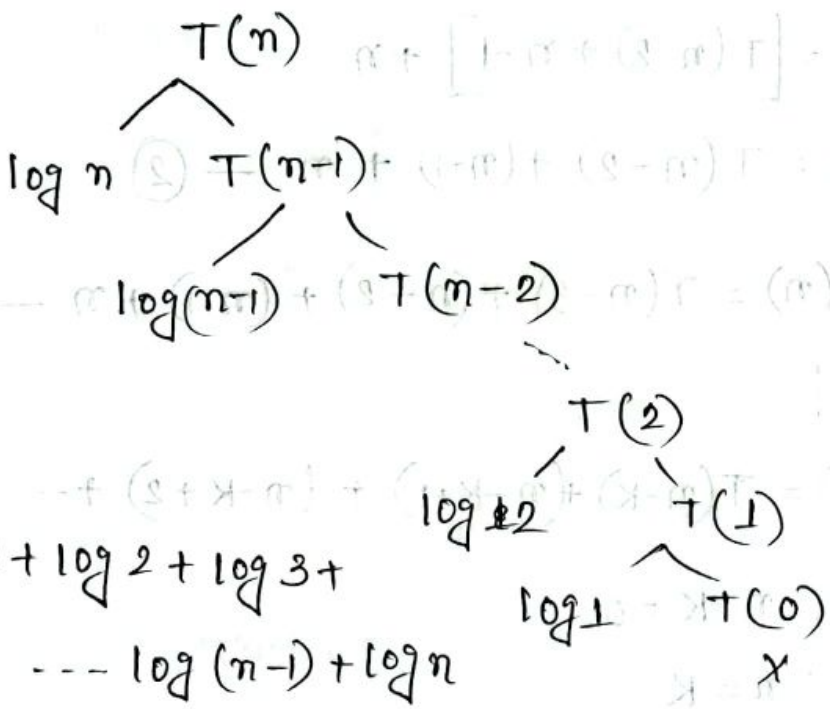
$\therefore$ upper bound $= O(n^2)$.

Topic Name : _____ Day : _____

Time : Date : / /

(3) $T(n) = T(n-1) + \log n$

$$T(n) = \begin{cases} 1, & n = 0 \\ T(n-1) + \log n, & n > 0 \end{cases}$$



$\log 1 + \log 2 + \log 3 + \cdots \log(n-1) + \log n$

$= \log(1 \times 2 \times 3 \times \cdots (n-1) \times n)$

$= \log n!$

$\Theta(n \log n)$.

Induction / substitution method (Same as before)

$T(n) = T(n-1) + 1$ — $O(n)$

$T(n) = T(n-1) + n$ — $O(n^2)$

$T(n) = T(n-1) + \log n$ — $O(n \log n)$

$T(n) = T(n-1) + n^2$ — $O(n^3)$

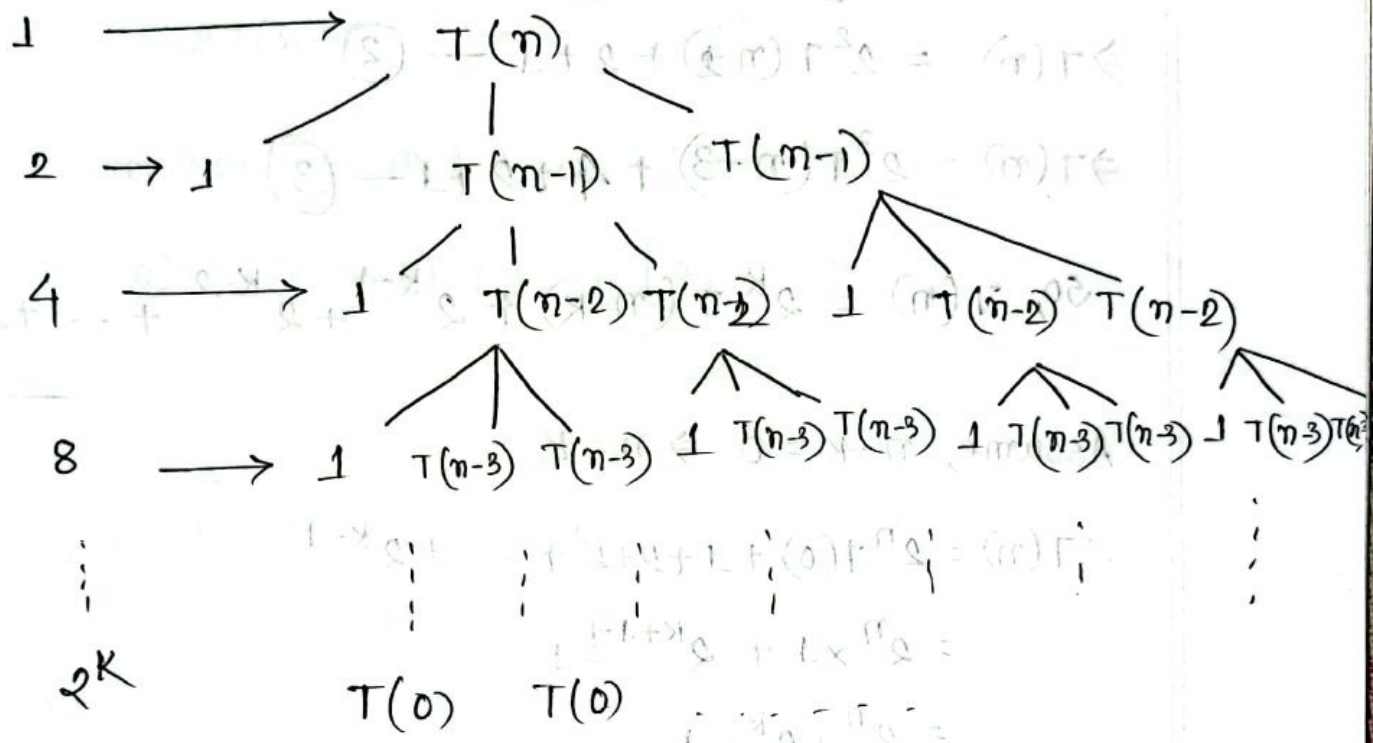
} Multiplied by n ↳ Trick

Topic Name : _____ Day : _____

Time : Date : / /

④ $T(n) = 2T(n-1) + 1$

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



Total: $1 + 2 + 4 + 8 + \cdots + 2^K$

$= 1 + 2 + 2^2 + 2^3 + \cdots + 2^K$

$= 2^{K+1} - 1$

Formula: $a + ar + ar^2 + \cdots + ar^K = \dfrac{a(r^{K+1} - 1)}{r - 1}$

Here, $a = 1$, $r = 2$. So, $\dfrac{1(2^{K+1} - 1)}{2 - 1} = (2^{K+1} - 1)$.

Assume, $n - K = 0 \Rightarrow n = K$

So, $2^{n+1} - 1$, $O(2^n)$

Topic Name : _____ Day : _____

Time : Date : / /

By substitution method:

$$T(n) = 2T(n-1) + 1 \quad \text{——} (1)$$

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

$$\Rightarrow T(n) = 2^2 T(n-2) + 2 + 1 \quad \text{——} (2)$$

$$\Rightarrow T(n) = 2^3 T(n-3) + 4 + 2 + 1 \quad \text{——} (3)$$

$$\text{So, } T(n) = 2^K T(n-K) + 2^{K-1} + 2^{K-2} + \cdots + 2^2 + 2 + 1 \quad \text{——} (4)$$

Assume, $n - K = 0 \Rightarrow n = K$

$$\therefore T(n) = 2^n T(0) + 1 + 2 + 2^2 + \cdots + 2^{K-1}$$

$$= 2^n \times 1 + 2^{K+1-1} - 1$$

$$= 2^n + 2^K - 1$$

$$= 2^n + 2^n - 1$$

$$= 2^{n+1} - 1$$

$\therefore O(2^n)$.

Topic Name : _____ Day : _____

Time : Date : / /

Master theorem for decreasing function

$T(n) = 2T(n-1) + 1$ —— $O(2^n)$

$T(n) = 3T(n-1) + 1$ —— $O(3^n)$

$T(n) = 2T(n-1) + n$ —— $O(n2^n)$

So, General ~~solution~~: Form of reccurance solution :

$T(n) = aT(n-b) + f(n)$ [$a>0$, $b>0$ and $f(n) = O(n^k)$ where $k \geq 0$]

Case: 1

if $a = 1$, $O(n^{k+1})$

$O(n * f(n))$

Case: 2

if $a > 1$, $O(n^k a^n)$

Case: 3

if $a < 1$, $O(n^k)$

$O(f(n))$

These three conditions are called Master's theorem

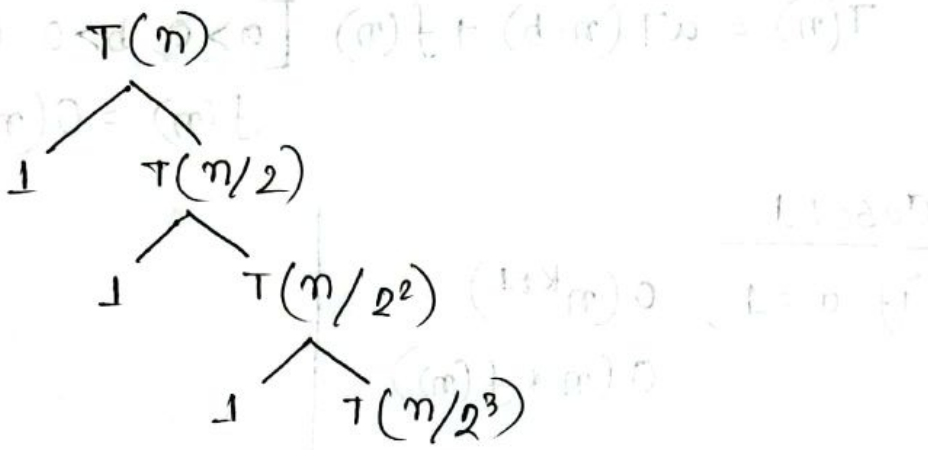
Topic Name : _____ Day : _____

Time : Date : / /

Recurrence Relation for Dividing function :

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

$$T(n) = \begin{cases} 1 & , n = 1 \\ T(n/2) + 1 & , n > 1 \end{cases}$$



K steps

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

$\Rightarrow k = \log_2 n \quad O(\log n)$

solving with substitution

$T(n) = T(n/2) + 1$ —— (1)

$T(n) = T(n/2^2) + 2$ —— (2)

$T(n) = T(n/2^3) + 3$ —— (3)

$\therefore T(n) = T(n/2^k) + k$ —— (4)

Topic Name : _____ Day : _____

Time : Date : / /

Assume, $\frac{n}{2^K} = 1$

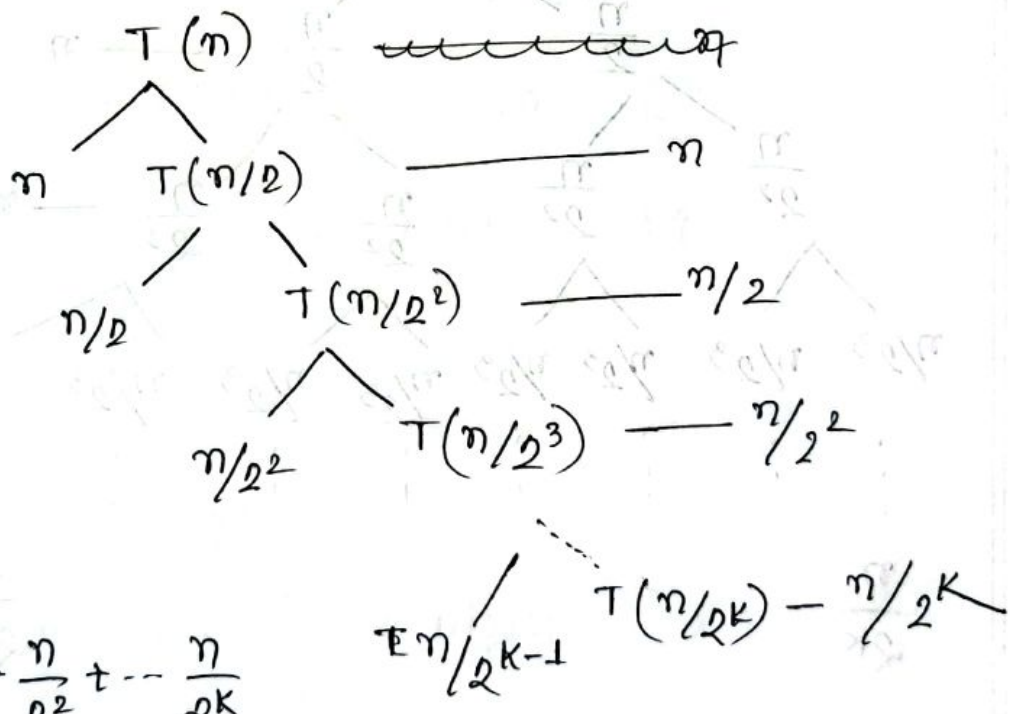
$\therefore n = 2^K \Rightarrow K = \log n$

$\therefore T(n) = T(1) + \log n$

$= 1 + \log n$

$O(\log n)$.

(2) $T(n) = \begin{cases} 1, & n = 1 \\ T(n/2) + n, & n > 1 \end{cases}$



$\therefore n + \frac{n}{2} + \frac{n}{2^2} + \cdots \frac{n}{2^K}$

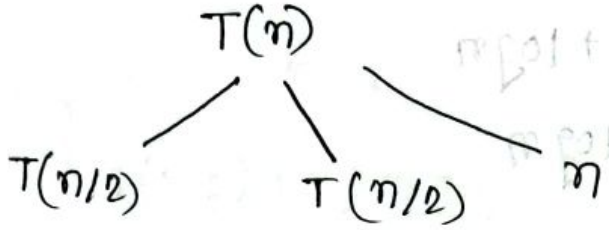
$= n\left[1 + \frac{1}{2} + \frac{1}{2^2} + \cdots + \frac{1}{2^K}\right]$

$= n \sum_{i=0}^{K} \frac{1}{2^i} = n \times 1 = n$, $O(n)$.

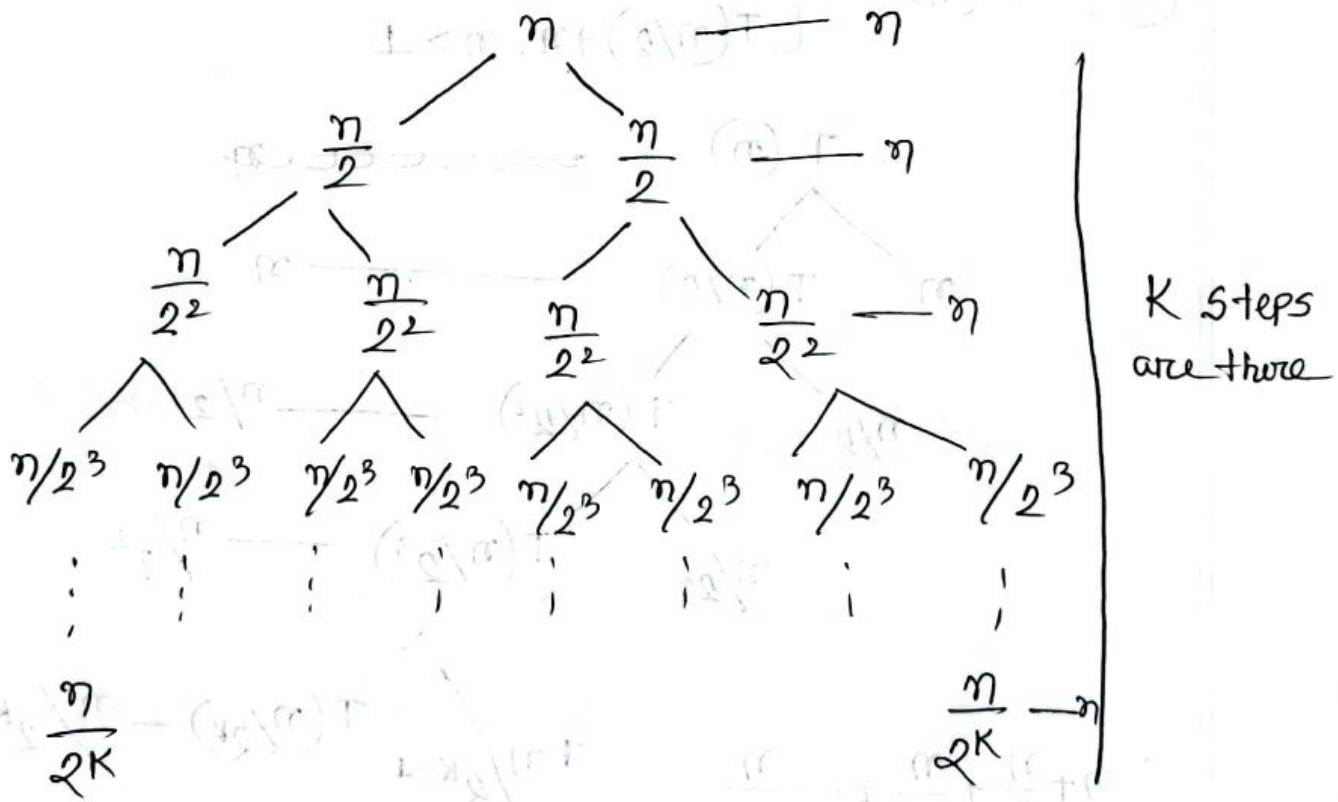
Topic Name : _____ Day : _____

Time : Date : / /

(3) $T(n) = \begin{cases} 1 & , n = 1 \\ 2T(n/2) + n & , n > 1 \end{cases}$



This will be big so we can write shortly,



Total time = nK

Assume, $\frac{n}{2^K} = 1 \Rightarrow K = \log n$

$\therefore$ Total time = $n \log n$. $O(n \log n)$.

Topic Name : _____ Day : _____

Time : Date : / /

Solving with substitution:

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

$T(n) = 2[2T(n/2^2) + n/2] + n$

$= 4T(n/2^2) + n + n$

$\therefore T(n) = 2^2 T(n/2^2) + 2n$ --(2)

Again, $T(n) = 2^2\left[2T(n/2^3) + \frac{n}{2^2}\right] + 2n$

$= 2^3 T(n/2^3) + n + 2n$

$= 2^3 T(n/2^3) + 3n$ --(3)

$\therefore T(n) = 2^k T(n/2^k) + kn$ --(4)

Assume, $n/2^k = 1$

$\Rightarrow n = 2^k$

$\therefore k = \log n$

(4) $\Rightarrow T(n) = n \cdot T(1) + n\log n$

$= n + n\log n$

So, $O(n\log n)$.

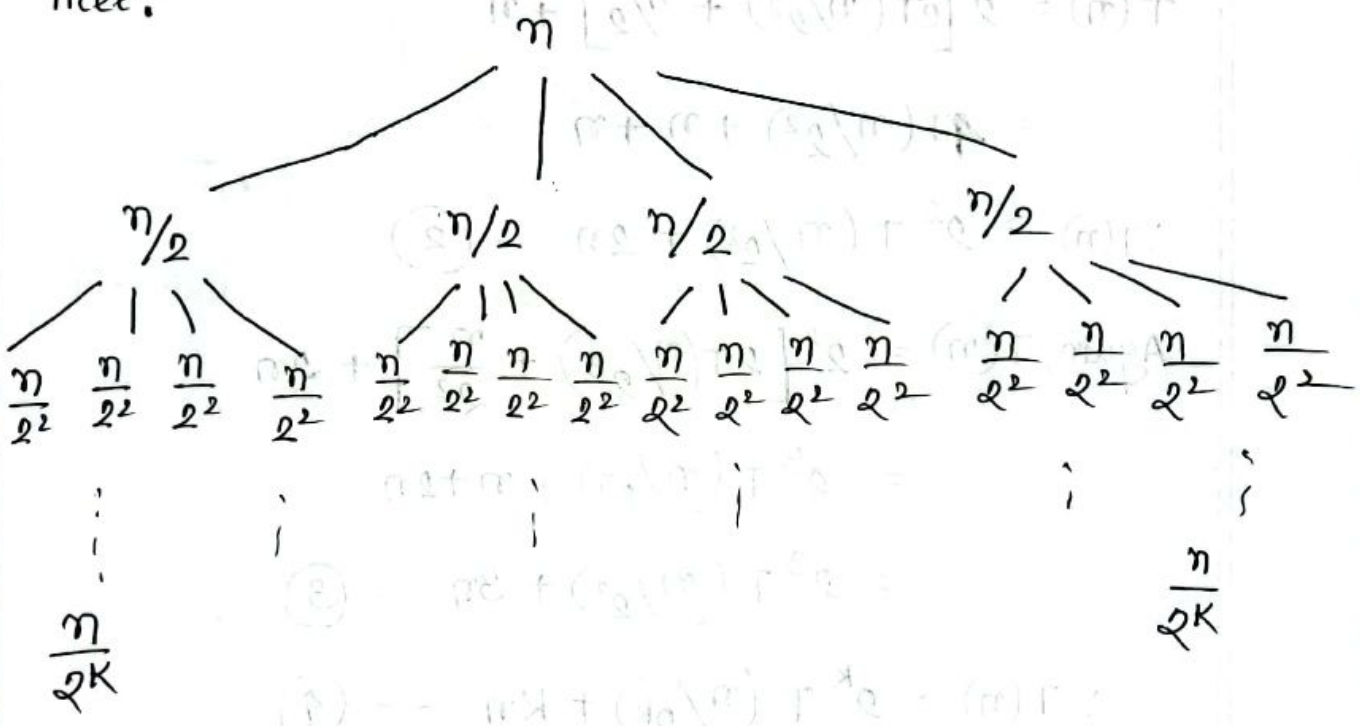
Topic Name : _____ Day : _____

Time : Date : / /

(4) $T(n) = 4T(n/2) + cn$ ---- (1)

~~$T(n) = 4$~~

Tree:



Total steps = K

Total Time = nK

Assume, $\frac{n}{2^K} = 1$

$\Rightarrow K = \log n$

So, Total time = $n \log n$

By Master's theorem,

$T(n) = 4T(n/2) + cn$

here, $\log_2 4 = 2 > K = 1, P = 0$

$\therefore$ Time = n^2

Asymptotic notation for upper bound = ~~$O(n \log n)$~~.

$O(n^2)$.

Topic Name : _____ Day : _____

Time : Date : / /

Solving with substitution:

$T(n) = 4T(n/2) + cn$ ---- (1)

$\therefore T(n) = 4\left[4T(n/2^2) + \frac{cn}{2}\right] + cn$

$= 4^2 T(n/2^2) + 2cn + cn$

$= 4^2 T(n/2^2) + 3cn$ --- (2)

$2cn + cn$

$T(n) = 4^2\left[4T(n/2^3) + \frac{cn}{2^2}\right] + 3cn$

$= 4^3 T(n/2^3) + 4cn + 2cn + cn$

$= 4^3 T(n/2^3) + (2^2 + 2^1 + 2^0)cn$

$\therefore T(n) = 4^K T(n/2^K) + (2^{K-1} + \cdots + 2^2 + 2^1 + 2^0)cn$

$= 4^K T(n/2^K) + (2^K - 1)cn$

$= 4^K T(n/2^K) + 2^K \cdot cn - cn$

Now, $\frac{n}{2^K} = 1 \Rightarrow 2^K = n \Rightarrow K = \log n$

$\therefore T(n) = n^2 + n \cdot cn - cn = n^2 + cn^2 - cn$

$\therefore O(n^2)$.

Topic Name : _____ Day : _____

Time : Date : / /

Masters theorem for dividing functions

General form, $T(n) = aT(n/b) + f(n)$

$a \geq 1$ $\quad f(n) = \theta(n^k \log^p n)$

$b > 1$

① $\log_b a$

② k

Case-1 if $\log_b a > k$ then $\theta(n^{\log_b a})$

Case-2 if $\log_b a = k$

→ if $p > -1$ $\quad \theta(n^k \log^{p+1} n)$

→ if $p = -1$ $\quad \theta(n^k \log \log n)$

→ if $p < -1$ $\quad \theta(n^k)$

Case-3 if $\log_b a < k$

→ if $p \geq 0$ $\quad \theta(n^k \log^p n)$

→ if $p < 0$ $\quad \theta(n^k)$

↓

$f(n)$ ans it is.

Topic Name : _____ Day : _____

Time : Date : / /

Examples:

$T(n) = 2T(n/2) + 1$ — $\theta(n^1)$

$T(n) = 4T(n/2) + 1$ — $\theta(n^2)$

$T(n) = 4T(n/2) + n$ — $\theta(n^2)$

$T(n) = 8T(n/2) + n^2$ — $\theta(n^3)$

$T(n) = 16T(n/2) + n^2$ — $\theta(n^4)$

Case - 1

$T(n) = T(n/2) + n$ — $\theta(n)$

$T(n) = 2T(n/2) + n^2$ — $\theta(n^2)$

$T(n) = 2T(n/2) + n^2 \log n$ — $\theta(n^2 \log n)$

$T(n) = 4T(n/2) + n^3 \log^2 n$ — $\theta(n^3 \log^2 n)$

$T(n) = 2T(n/2) + \frac{n^2}{\log n}$ — $\theta(n^2)$

→ no need to include

Case - 3

$T(n) = T(n/2) + 1$ — $\theta(\log n)$

$T(n) = 2T(n/2) + n$ — $\theta(n \log n)$

$T(n) = 2T(n/2) + n \log n$ — $\theta(n \log^2 n)$

$T(n) = 4T(n/2) + n^2$ — $\theta(n^2 \log n)$

$T(n) = 4T(n/2) + (n \log n)^2$ — $\theta(n^2 \log^3 n)$

$T(n) = 2T(n/2) + \frac{n}{\log n}$ — $\theta(n \log \log n)$

$T(n) = 2T(n/2) + n/\log^2 n$ — $\theta(n)$

case - 2

Topic Name : _____ Day : _____

Time : Date : / /

Recurrence Relation for root functions:

$$T(n) = \begin{cases} 1 & , n = 2 \\ T(\sqrt{n}) + 1 & , n > 2 \end{cases}$$

$T(n) = T(n^{\frac{1}{2}}) + 1$ ---- (1)

$T(n) = T(n^{\frac{1}{2^2}}) + 2$ ---- (2)

$T(n) = T(n^{\frac{1}{2^3}}) + 3$ ---- (3)

$\vdots$

$T(n) = T(n^{\frac{1}{2^k}}) + k$ ---- (4)

Assume, $n = 2^m$

$T(2^m) = T(2^{\frac{m}{2^k}}) + k$

Assume, $T(2^{m/2^k}) = T(2^1)$

$\therefore \frac{m}{2^k} = 1$

$m = 2^k$ and, $k = \log_2 m$

$\because n = 2^m$, $m = \log_2 n$

$\therefore k = \log\log_2 n$

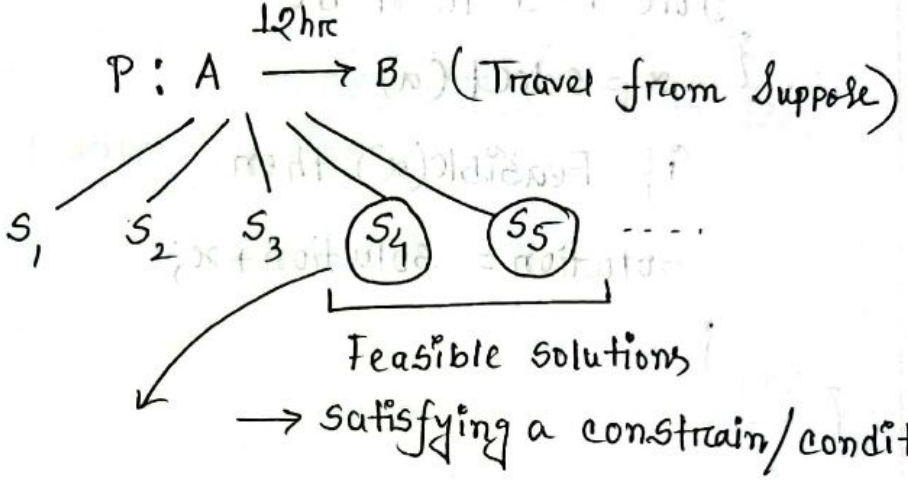
$\Theta(\log\log_2 n)$.

Topic Name : _____ Day : _____

Time : Date : / /

Greedy Method

→ used for solving optimization problem



→ satisfying a constrain/condition

If minimal cost

→ Then optimal solution

→ There can be only 1 optimal solution

→ optimization problem

↳ which requires a minimum/maximization.

Strategy use for optimization problems:

1. Greedy
2. DP
3. Branch and Bound.

Topic Name : _____ Day : _____

Time : Date : / /

Approaches :

```
Algorithm Greedy (a, n) {
    for i = 1 to n do
    {
        x = select (a)
        if Feasible(x) then
        solution = solution + x;
    }
}
```

Topic Name : _____ Day : _____

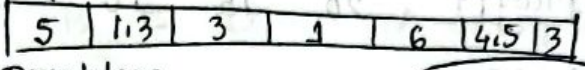
Time : Date : / /

Knapsack Problem → Fractional

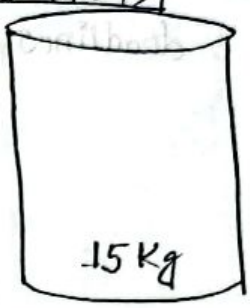
$n = 7$ Object : 0 1 2 3 4 5 6 7

$m = 15$ profit : P 10 5 15 7 6 18 3

weight : w 2 3 5 7 1 4 1



→ container loading problem



$\frac{P}{w}$ is required.

x (1 2/3 1 0 1 1 1)

x_1 x_2 x_3 x_4 x_5 x_6 x_7

In the bag : 15 − 1 = 14

14 − 2 = 12

12 − 4 = 8

8 − 5 = 3

3 − 1 = 2

2 − 2 = 0 ✓

constraint,

$\sum x_i w_i \leq m$

objective

$\max \sum x_i p_i$

Total profit : $1 \times 2 + 2/3 \times 3 + 1 \times 5 + 0 \times 7 + 1 \times 1 + 1 \times 4 + 1 \times 1$

$\sum x_i w_i = 2 + 2 + 5 + 0 + 1 + 4 + 1 = 15$ → satisfy constrain

$\sum x_i p_i = 1 \times 10 + \frac{2}{3} 5 + 1 \times 15 + 1 \times 6 + 1 \times 18 + 1 \times 3$

$= 54.6$

Topic Name : _____ Day : _____

Time : Date : / /

Job Sequencing With Deadlines

$n=5$

Jobs	J_1	J_2	J_3	J_4	J_5
profit	20	15	10	5	1
deadlines	2 ✓	2 ✓	1 x	3 ✓	3 x

J_2 J_1 J_4

0 — 1 — 2 — 3

Machine + unit

Have to complete in these unit only

$20+15+5 = 40$ → Total profit

Have to complete $\{J_2, J_1, J_4\}$

$J_1 \rightarrow J_2 \rightarrow J_4$

$J_2 \rightarrow J_1 \rightarrow J_4$

Job consider	Slot asign	solution	profit
–	–	ϕ	0
J_1	[1, 2]	J_1	20
J_2	[0,1] [1,2]	J_1, J_2	20+15
J_3 x	[0,1] [1,2]	J_1, J_2	20+15
J_4	[0,1] [1,2] [2,3]	J_1, J_2, J_4	20+15+5
J_5 x	[0,1] [1,2] [2,3]	J_1, J_2, J_4	40

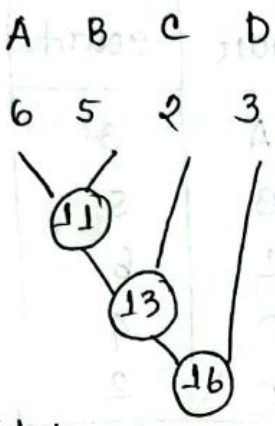
Topic Name : _____ Day : _____

Time : Date : / /

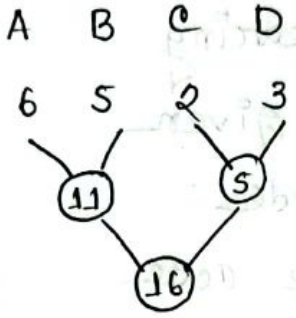
Optimal Merge Pattern

List → A B C D

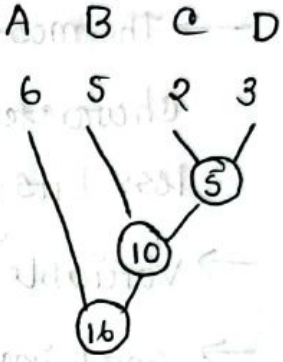
Size → 6 5 2 3



Total = 40



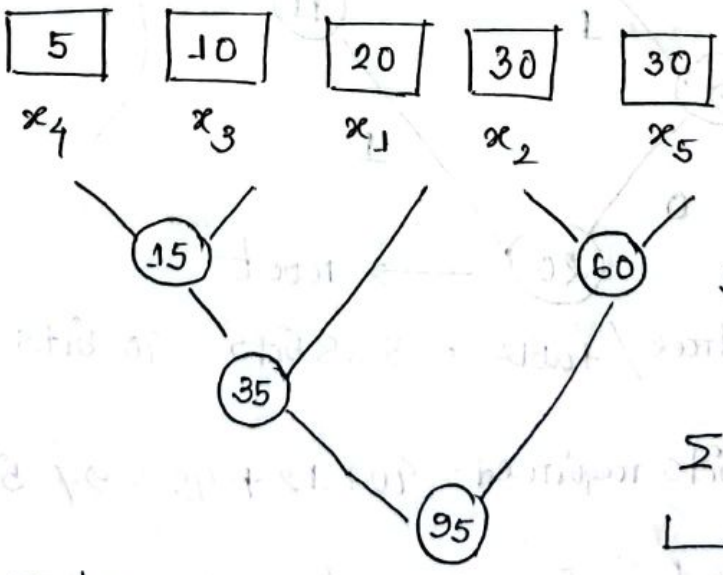
Total = 32



Total = 31

Solving: lists → x_1 x_2 x_3 x_4 x_5

Sizes → 20 30 10 5 30



15 + 35 + 45 + 60 = 205

or,

3×5 + 3×10 + 2×20 + 2×30 + 2×30

= 205

$\Sigma d_i * x_i$

↳ Formula

Mainly use Huffman coding

Topic Name : _____ Day : _____

Time : Date : / /

Huffman Coding

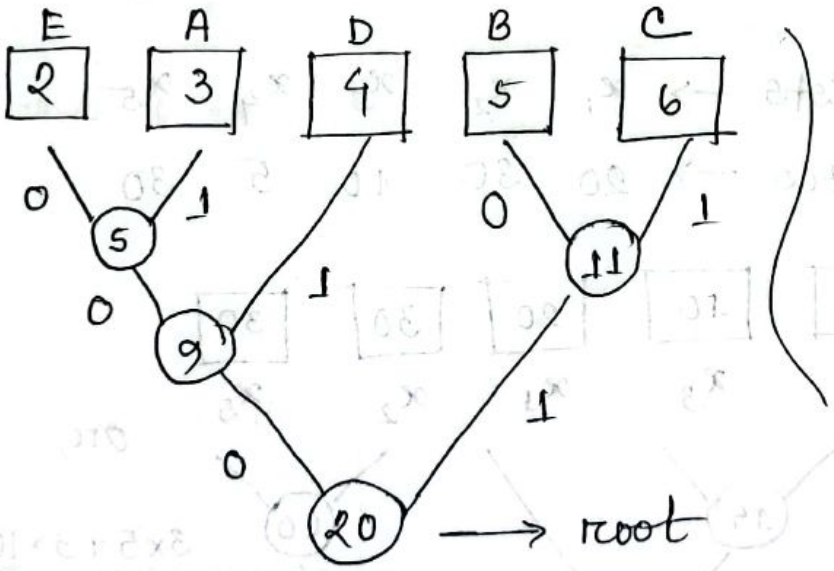
Message - BCCABBDDAECCBBAEDDCC

→ The most appearing character will given less bits of code.

→ Variable size code

→ Sort increasing no of count

Char	count	code	
A	3	001	3x3=9
B	5	10	5x2=10
C	6	11	6x2=12
D	4	01	4x2=8
E	2	000	2x3=6
	20		12bit



So, message size is 45 bits

For this tree / table = 5 x 8 bits = 40 bits

∴ Total bits required = 40 + 12 + 45 = 97 bits.

$\Sigma d_i * f_i$ for size of code

Topic Name : _____ Day : _____

Time : Date : / /

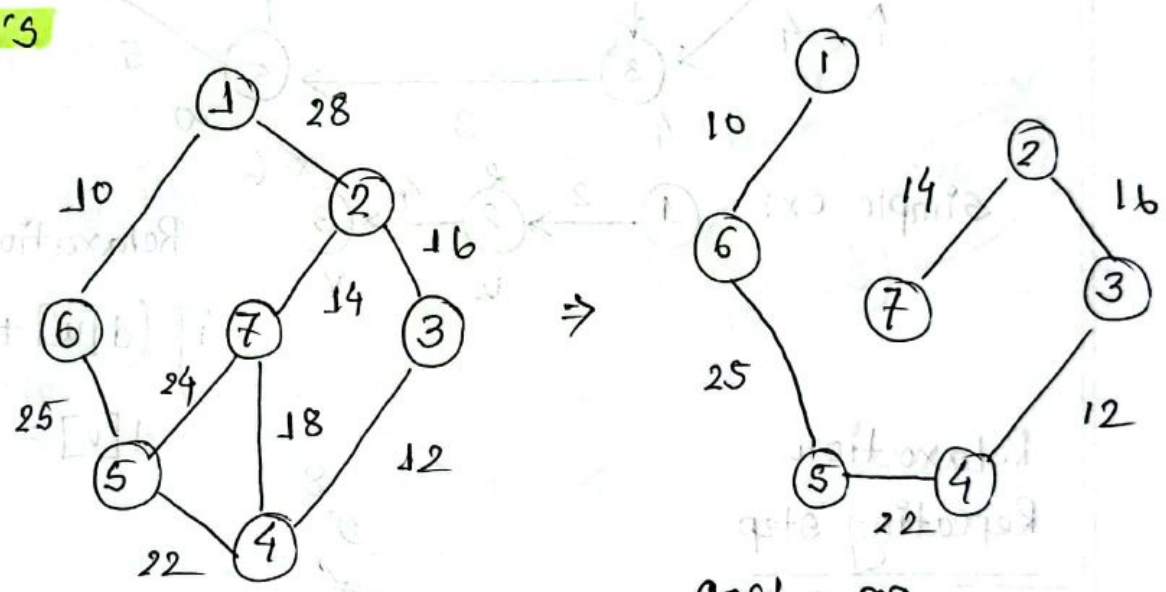
Minimum Cost Spanning Tree

→ Spanning tree, $|E'| = |V| - 1$

No of spanning trees, $|E| C_{|V|-1}$ − no. of cycles

1. Prim's
2. Kruskal's

Prim's



Cost = 99.

Kruskal's

① Sort all the ve edges by weight

② Add untill find any cycle formed.

→ $\theta(n, e)$ or $\theta(n^2)$

→ min heap is used then $\theta(n \log n)$

↳ It make faster.

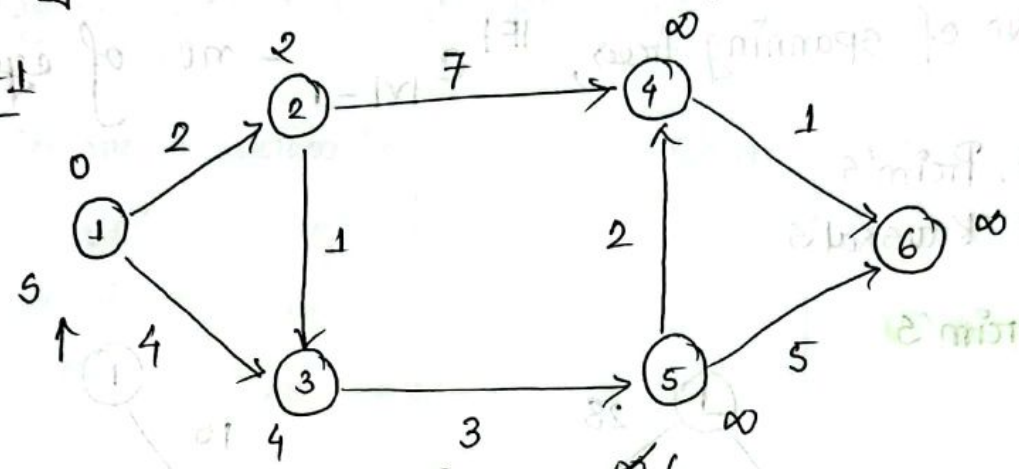
Topic Name : _____ Day : _____

Time : Date : / /

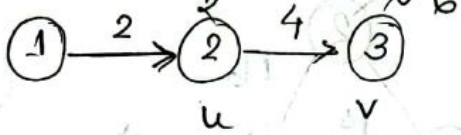
Dijkstra Algorithm

→ Single source shortest path algorithm

Step-1



Simple ex:



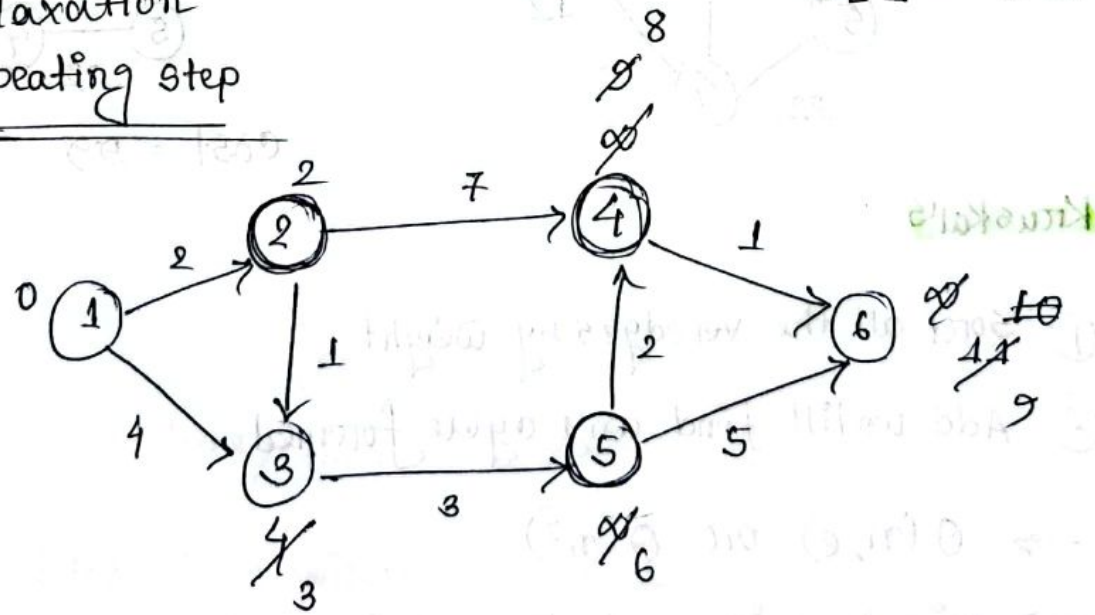
Relaxation

if $(d[u] + c(u,v) < d[v])$

$d[v] = d[u] + c(u,v)$

Relaxation

Repeating step



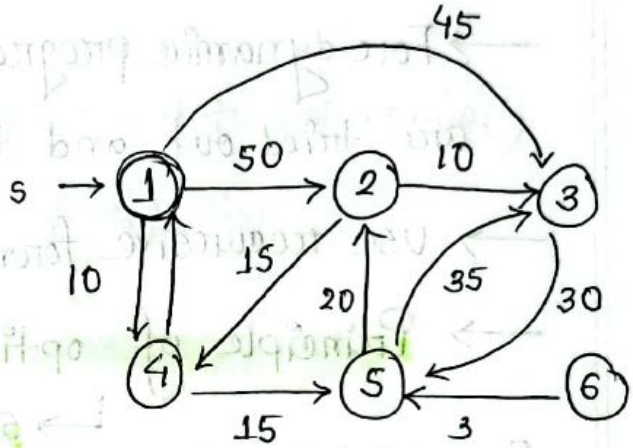
Topic Name : _____ Day : _____

Time : Date : / /

v	d[v]
2	2
3	3
4	8
5	6
6	9

$n = |V|$

$O(n^2)$ → worst case time



Solving problem:

starting vertex : 1

Selected vertex	2	3	4	5	6
4	50	45	(10)	∞	∞
5	(50)	50	45	(10)	(25) ∞
2	(45)	45	(10)	(25)	∞
3	(45)	(45)	(10)	(25)	∞
6	(45)	(45)	(10)	(25)	∞

Drawback: Negative weight can't be calculated/or can be calculated.

Topic Name : _____ Day : _____

Time : Date : / /

Dynamic programming

1. Greedy method
2. Dynamic programming

↓ Both use to solve optimization problems.

→ For dynamic programming Feasible all solutions are tried out and then we get only best one

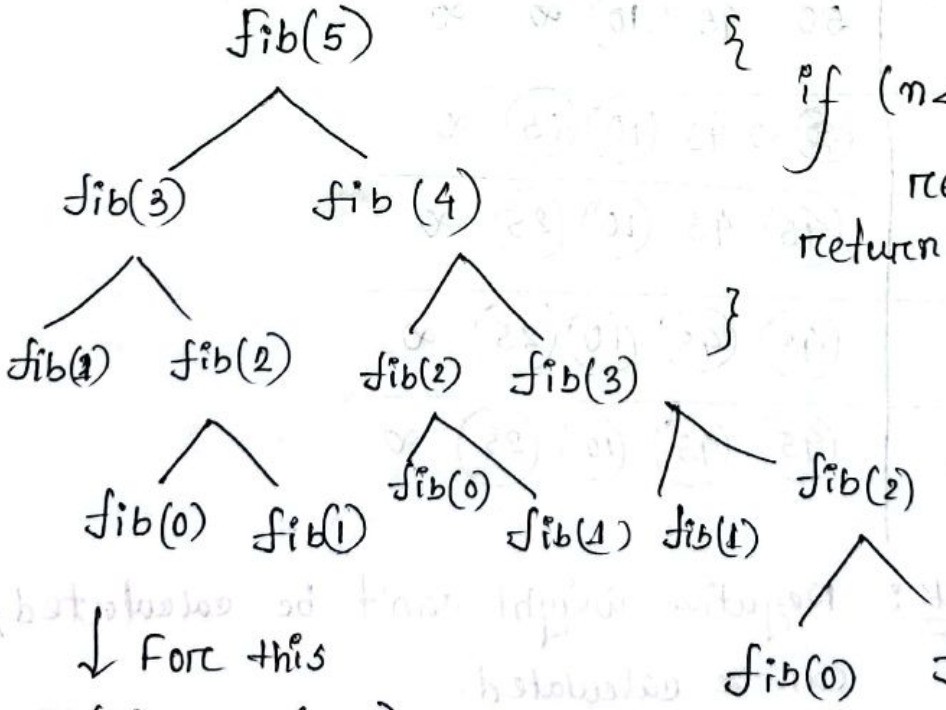
→ use recursive formulas / iterations

→ Principle of optimality.

↳ says that a problem can be solved by taking sequence of decisions

Ex: 0, 1, 1, 2, 3, 5 - - -

```
int fib(int n)
{
    if (n<=1)
        return n;
    return fib(n-2)+fib(n-1);
}
```



↓ For this

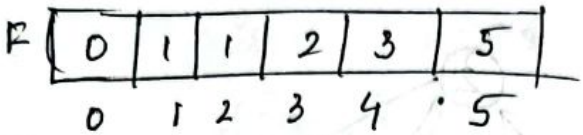
$T(n) = 2T(n-1) + 1$

$\theta(2^n)$.

Topic Name : _____ Day : _____

Time : Date : / /

If use a global array,



fib(n) = n + 1 call

$\theta(n)$

changes $2^n \rightarrow n$ (exponential to polynomial)

↳ This is because of memorization

↓

Top down approach.

If use iteration,

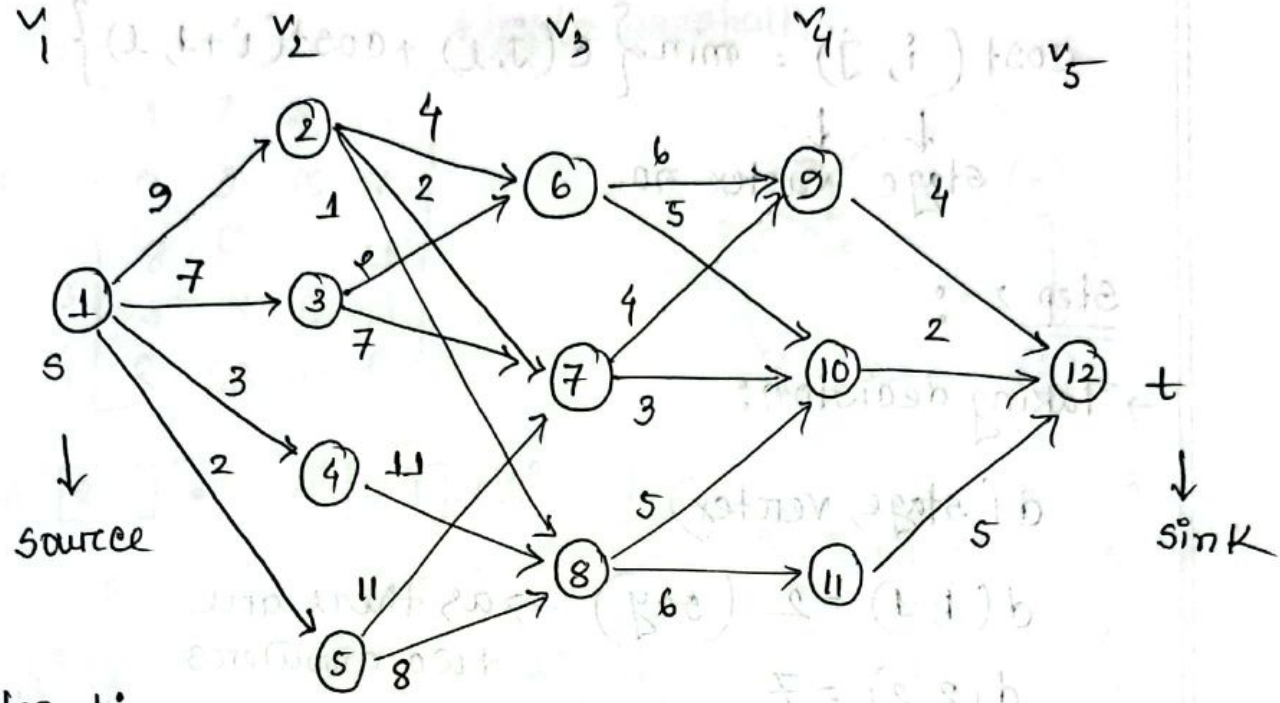
using 'for' loop → Tabulation method

→ Bottom up approach.

Topic Name : _____ Day : _____

Time : Date : / /

Multi Stage Graph → optimization p.
→ DP.



Step-1:

v	1	2	3	4	5	6	7	8	9	10	11	12
cost	16	7	9	18	15	7	5	7	4	2	5	0
d	2/3	7	6	8	8	10	10	10	12	12	12	12

stage ↓

cost(5, 12) = 0

↓ vertex

cost(4, 9) = 4

cost(4, 10) = 2

cost(4, 11) = 5

cost(3, 6) = min{c(6, 9) + cost(4, 9), c(6, 10) + cost(4, 10)}

= min(6+4, 5+2)

= 7

cost(3, 7) = 5

cost(3, 8) = 7

Similarly others.

Topic Name : _____ Day : _____

Time : Date : / /

Formula :

$$cost(i, j) = \min\{c(j, l) + cost(i+1, l)\}$$

↓ ↓

stage vertex-no.

Step-2 :

→ Taking decision :

d (stage, vertex)

d (1, 1) = 2 (say) → as there are two answers

d (2, 2) = 7

d (3, 7) = 10

d (4, 10) = 12

Path, 1 → 2 → 7 → 10 → 12 (shortest path 1)

d (1, 1) = 3

d (2, 3) = 6

d (3, 6) = 10

d (4, 10) = 12

1 → 3 → 6 → 10 → 12 (shortest path 2)

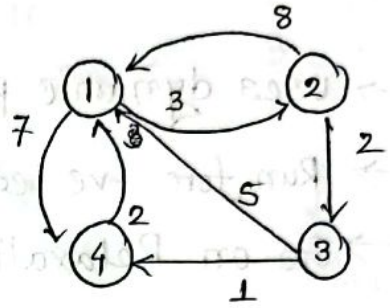
Topic Name : _____ Day : _____

Time : Date : / /

All pairs shortest path

Floyd - Warshall

$$A^1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & \infty & 7 \\ 8 & 0 & 2 & 15 \\ 5 & 8 & 0 & 1 \\ 2 & 58 & \infty & 0 \end{bmatrix} \end{array}$$



$A^0[2,3] \quad A^0[2,1] + A^0[1,3]$

$2 \quad < \quad 8 + \infty$

$A^0[2,4] \quad A^0[2,1] + A^0[1,4]$

$\infty \quad > \quad 8 + 7 = 15$

$A^0[3,2] \quad A^0[3,1] + A^0[1,2]$

$\infty \quad > \quad 5 + 3 = 8$

$$A^0 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & \infty & 7 \\ 8 & 0 & 2 & \infty \\ 5 & \infty & 0 & 1 \\ 2 & \infty & \infty & 0 \end{bmatrix} \end{array}$$

$$A^2 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & 5 & 7 \\ 8 & 0 & 2 & 15 \\ 5 & 8 & 0 & 1 \\ 2 & 85 & 7 & 0 \end{bmatrix} \end{array}$$

--- A^3 ---

A^4 ...

↳ This is required matrix

Formula :

$$A^K[i,j] = \min\{A^{K-1}[i,j], A^{K-1}[i,K] + A^{K-1}[K,j]\}$$

$O(n^3)$

Topic Name : _____ Day : _____

Time : Date : / /

Single source shortest path

Bellman - Ford

→ uses dynamic programming strategy

→ Run for -ve edges

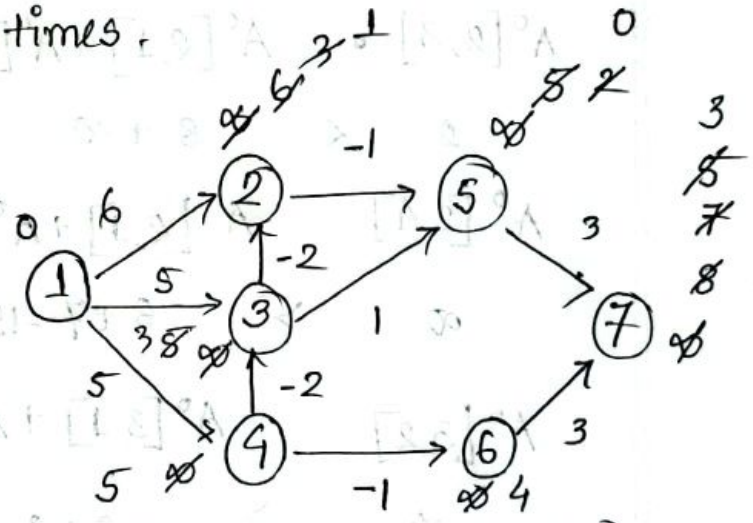
→ Go on Relaxation for all edges

→ Relaxation for (n-1) times.

Relaxation:

if (d[u] + c(u,v) < d[v])

d[v] = d[u] + c(u,v)



edgeList → (1,2), (1,3), (1,4), (2,5), (3,2), (3,5), (4,3), (4,6), (5,7), (6,7)

1 - 0
2 - 1
3 - 3
4 - 5
5 - 0
6 - 4
7 - 3

Time compexity, O(|V| |E|)

$O(n^2)$

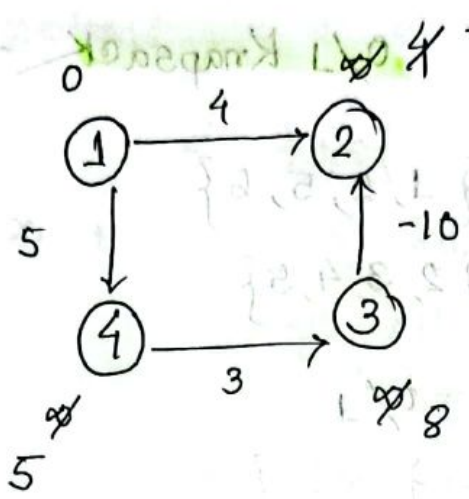
In case of complete graph, $|E| = \frac{n(n-1)}{2}$

$\therefore O(n^3)$.

Topic Name : _____ Day : _____

Time : Date : / /

Drawback:



1 – 0
2 – (-2)
3 – 8
4 – 5

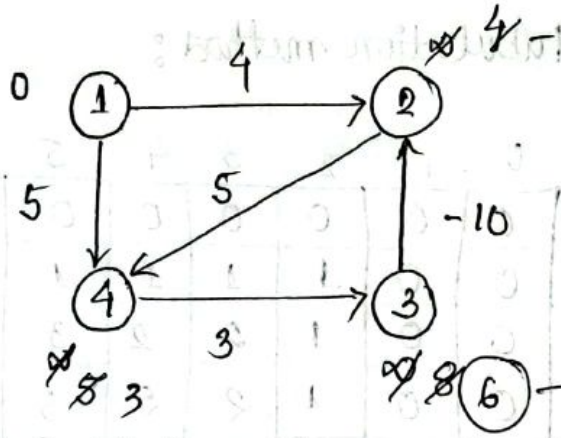
↓

Result after 3 Relaxation

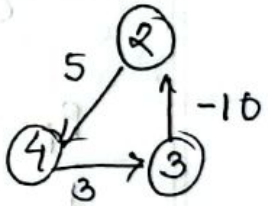
edges → (3,2), (4,3), (1,4), (1,2)

There Relaxation = n – 1 = 4 – 1 = 3 times

But,



⇒



5 + 3 – 10 = -2

→ after n-1 times

edges → (3,2), (4,3), (1,4), (1,2), (2,4)

If there present a negative weighted cycle, then the value will reduced drastically after every iteration.

So, it fails in this case.

Topic Name : _____ Day : _____

Time : Date : / /

0/1 Knapsack — non-Fractional

$m = 8 \qquad P = \{1, 2, 5, 6\}$

$n = 4 \qquad w = \{2, 3, 4, 5\}$

Solution, x. $x_i = 0/1$

$x = \{ \dots \}$

$\sum P_i x_i \longrightarrow$ maximized (optimization).

$\sum w_i x_i \leq m$

Solving with tabulation method:

P_i	w_i		0	1	2	3	4	5	6	7	8 ← m
1	2	0	0	0	0	0	0	0	0	0	0
2	3	1	0	0	1	1	1	1	1	1	1
5	4	2	0	0	1	2	2	3	3	3	3
6	5	3	0	0	1	2	5	5	6	7	7
		4	0	0	1	2	5	6	6	7	8

↓ no of object

Formula, $v[i,w] = \max\{v[i-1,w], v[i-1, w-w[i]] + P[i]\}$

Step-2

x_1	x_2	x_3	x_4
0	1	0	1

Maximum profit, $x = \{0, 1, 0, 1\}$

Topic Name : _____ Day : _____

Time : Date : / /

Solving with set method:

$S^0 = \{0, 0\}$

$S_1^0 = \{(1, 2)\}$

$S^1 = \{(0, 0), (1, 2)\}$

$S_1^1 = \{(2, 3), (3, 5)\}$

$S^2 = \{(0, 0), (1, 2), (2, 3), (3, 5)\}$

$S_1^2 = \{(5, 4), (6, 6), (7, 7), (8, 8)\}$

$S^3 = \{(0, 0), (1, 2), (2, 3), (3, 8), (5, 4), (6, 6), (7, 7)\}$

$S_1^3 = \{(6, 5), (7, 7), (8, 8), (11, 9), (12, 11), (13, 12)\}$

$\therefore S^4 = \{(0, 0), (1, 2), (2, 3), (5, 4), (6, 6), (6, 5), (7, 7), (8, 8)\}$

① $(8, 8) \in S^4$

$(8, 8) \in S^3 \therefore x_4 = 1$

$(8-6, 8-5) = (2, 3)$

② $(2, 3) \in S^3$

$(2, 3) \in S^2 \therefore x_3 = 0$

③ $(2, 3) \in S^2$

$(2, 3) \notin S_1^1 \therefore x_2 = 1$

$(2-2, 3-3) = (0, 0)$

④ $(0, 0) \in S^2$

$(0, 0) \in S^1 \therefore x_1 = 0$

solution, $x = \{0, 1, 0, 1\}$

$x_1\ x_2\ x_3\ x_4$

0 → excluded.

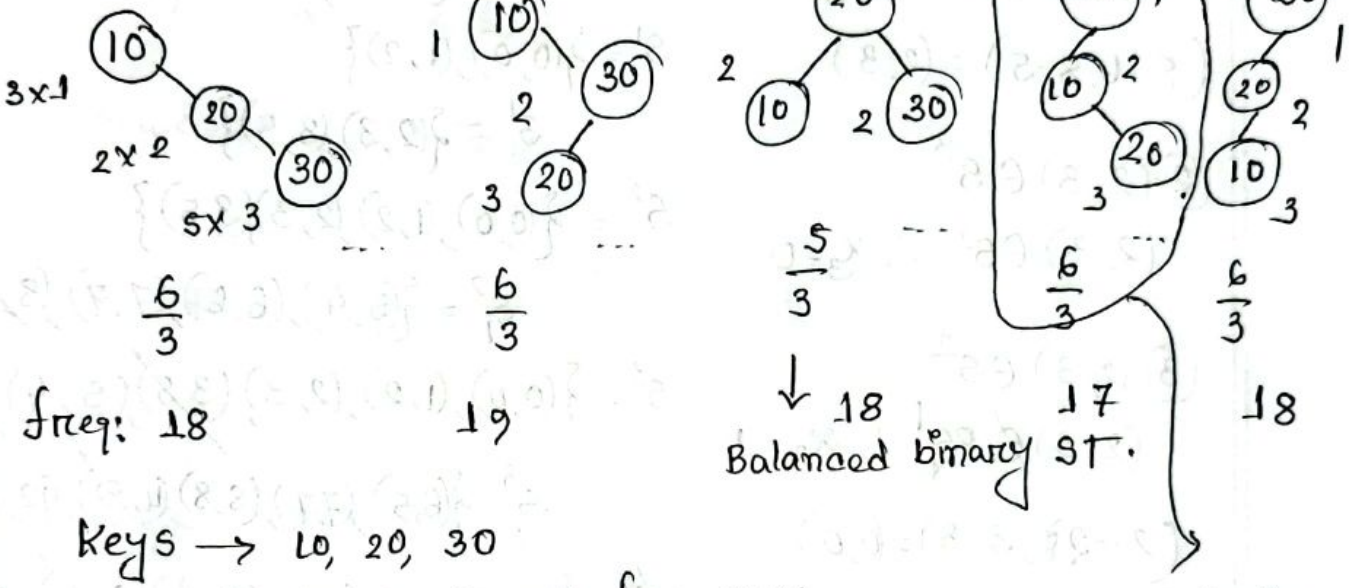
1 → included.

Topic Name : _____ Day : _____

Time : Date : / /

Optimal Binary Search Tree

Keys → 10, 20, 30



Balanced binary ST.

Keys → 10, 20, 30

3 2 5 → frequency

This is OBST.

* Problem solving with DP:

	1	2	3	4
Keys →	10	20	30	40
Freq. →	4	2	6	3

Topic Name : _____ Day : _____

Time : Date : / /

$l = j - i = 0$

$0-0=0$
$1-1=0$
$2-2=0$
$3-3=0$
$4-4=0$

$\therefore c[0,0] = c[1,1] \cdots = 0$

i \ j	0	1	2	3	4
0	0	4	8^1	20^3	26^3
1		0	2	10^3	16^3
2			0	6	12^3
3				0	3
4					0

$l = j - i = 1$

$1-0=1 \ (0,1)$
$2-1=1 \ (1,2)$
$3-2=1 \ (2,3)$
$4-3=1 \ (3,4)$

$c[0,1]$ $c[1,2]$ $c[2,3]$ $c[3,4]$

10 20 30 40

4 2 6 3

$$w(0,4) = \sum_{i=1}^{4} f(i)$$

$l = j - i = 2$

$2-0=(0,2)$
$3-1=(1,3)$
$4-2=(2,4)$

$c[0,2]$

$\frac{10}{4}$ $\frac{20}{2}$

4×1 (10) (20) 2×1

2×2 (20) (10) 4×2

$= 8$ $= 10$

$\therefore c[0,2] = 8^1$

Form: $c[0,0] + c[1,2] + w[0,2]$ 10

$= 8$

$l = j - i = 3$

$3-0=(0,3)$
$4-1=(1,4)$

Topic Name : _____ Day : _____

Time : Date : / /

Now, c[0,3]

1	2	3
10	20	30
4	2	6

w[0,3] = 4+2+6 = 12

$\min(c[0,0]+c[1,3]+\underbrace{w[0,3]}_{12}, c[0,1]+c[2,3]+12, c[0,2]+c[3,3]+12)$

22 22 20

$\therefore c[0,3] = 20^{3}$.

c[1,4]

2	3	4
20	30	40
2	6	3

w[1,4] = 2+6+3 = 11

$\min(c[1,1]+c[2,4]+11, c[1,2]+c[3,4]+11, c[1,3]+c[4,4]+11)$

0+12+11 = 23 | 2+3+11 = 16 | 10+0+11 = 21

$\therefore c[1,4] = 16^{3}$

$l = j-i = 4$

c[0,4]

1	2	3	4
10	20	30	40
4	2	6	3

w[0,4] = 15

Topic Name : _____ Day : _____

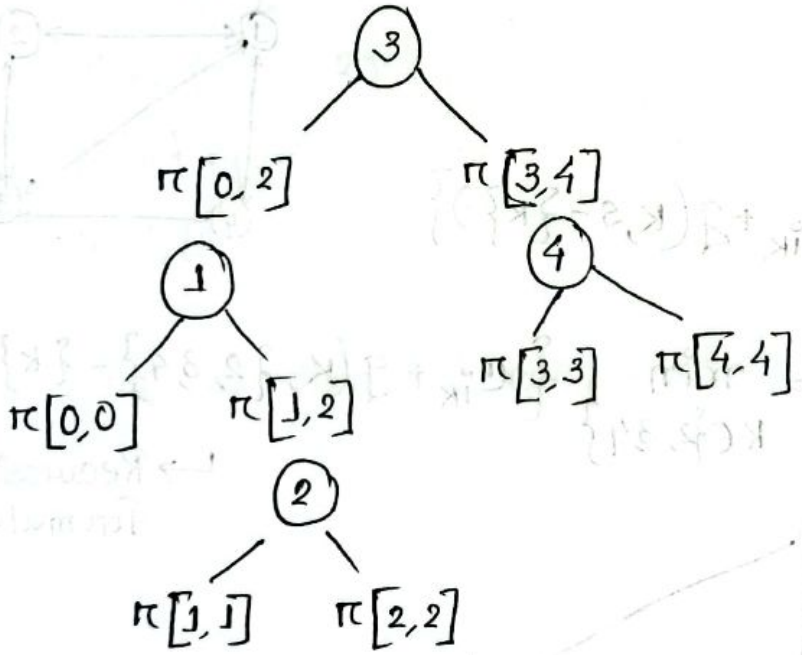
Time : Date : / /

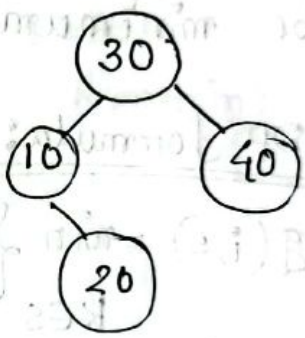
$$c[0,4] = \min \begin{cases} c[0,0] + c[1,4] \\ c[0,1] + c[2,4] \\ c[0,2] + c[3,4] \\ c[0,3] + c[4,4] \end{cases} + 15 \begin{cases} \rightarrow 0+16 \\ \rightarrow 4+12 \\ \rightarrow 8+3 \\ \rightarrow 20+3 \end{cases}$$

$$\text{root} = 11 + 15 = 26$$

$\downarrow$

So, $r[0,4] = 3$





$\downarrow$

Optimal BST

Ans.

Formula $c[i,j] = \min_{i < k \le j} \{ c[i,k-1] + c[k,j] \} + w(i,j)$

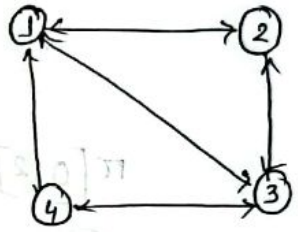
Topic Name : _____ Day : _____

Time : Date : / /

Travelling Salesman Problem

Problem: starting through a vertex we have to travel each node & and come to the same node. cost of traveling has to be minimum.

	1	2	3	4
1	0	10	15	20
2	5	0	9	10
3	6	13	0	12
4	8	8	9	0



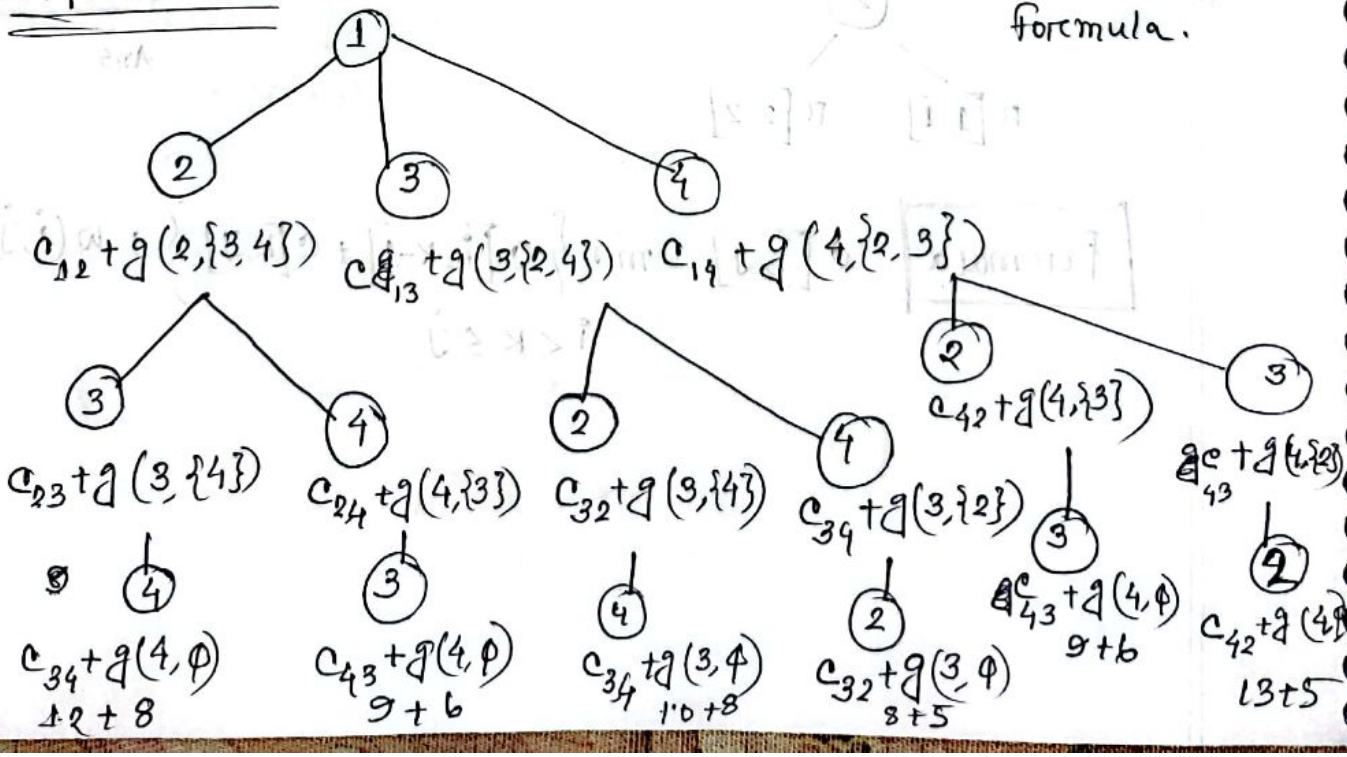
DP formula:

$$g(i,s) = \min_{k \in s} \{ c_{ik} + g(k, s-\{k\}) \}$$

$$g(1,\{2,3,4\}) = \min_{k \in \{2,3,4\}} \{ c_{ik} + g(k, \{2,3,4\} - \{k\}) \}$$

↳ Recursive formula.

Explanation:



Topic Name : _____ Day : _____

Time : Date : / /

→ DP, mainly solving program in tabulare values / Table.

$g(2, \phi) = 5$
$g(3, \phi) = 6$
$g(4, \phi) = 8$

} no vertex Remaining

2 V. Rem:

$g(2, \{3,4\}) = 25$
$g(3, \{2,4\}) = 25$
$g(4, \{2,3\}) = 23$

$g(2, \{3\}) = 15$
$g(2, \{4\}) = 18$
$g(3, \{2\}) = 18$
$g(3, \{4\}) = 20$
$g(4, \{2\}) = 13$
$g(4, \{3\}) = 15$

} 1 Vertex remaining

$g(1, \{2,3,4\}) = 35$

1 vertex Remaining

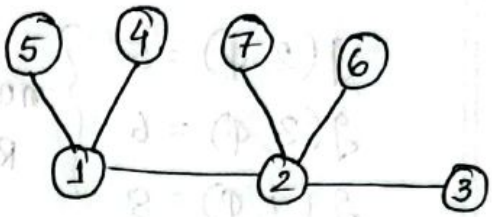
So, the shortest cost = 35.

Topic Name : _____ Day : _____

Time : Date : / /

BFS and DFS

1. Visiting a Vertex
2. Exploration of Vertex

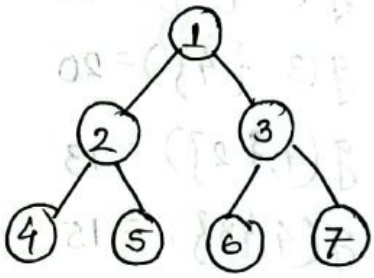


Ex:1

BFS: 1, 2, 4, 5, 7, 3, 6

DFS: 1, 2, 3, 6, 7, 4, 5

Ex:2



BFS: 1, 2, 3, 4, 5, 6, 7

DFS: 1, 2, 4, 5, 3, 6, 7

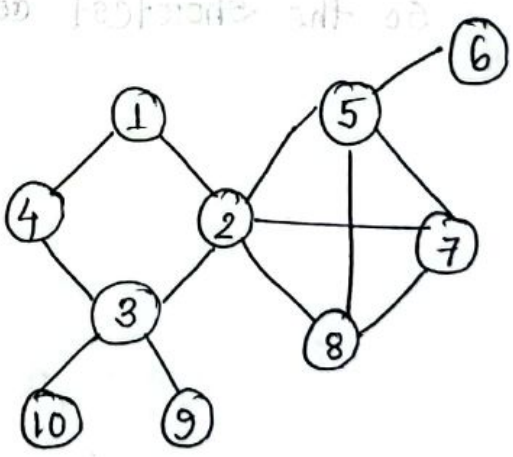
→ Level order

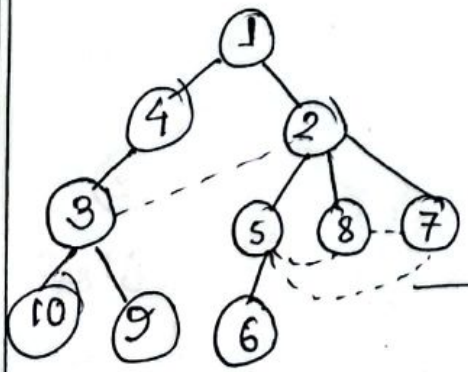
→ Pre-order

Ex:3

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

Q | 1 | 4 | 2 | 3 | 5 | 8 | 7 | 10 | 9 | 6 |





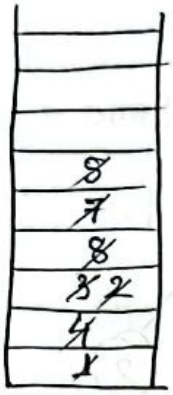
→ Cross edges.

BFS spanning tree.

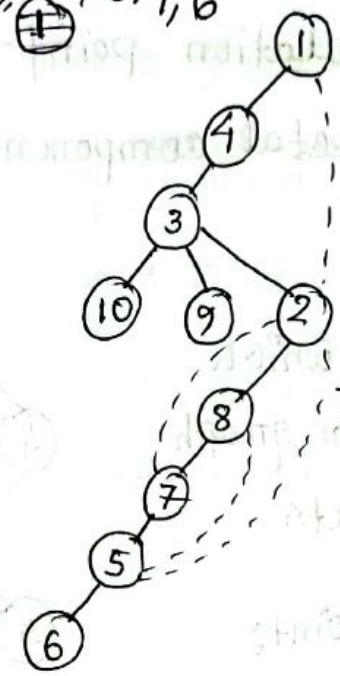
Topic Name : _____ Day : _____

Time : Date : / /

DFS: 1, 4, 3, 10, 9, 2, 8, 7, 6



Stack



DFS spanning tree

→ Back edges

Topic Name : _____ Day : _____

Time : Date : / /

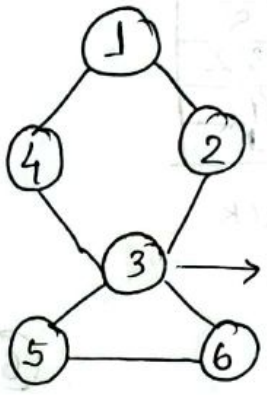
~~Articulare Po~~

Articulation point

Biconnected components

→ Articulation point:

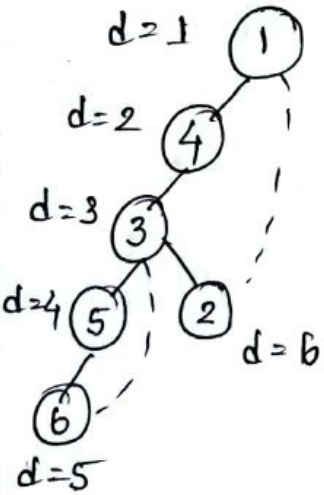
The component for which if it removed then the graph will break into two parts.



→ Articulation point

To Find Articulation point:

Step-1 DFS



	1	2	3	4	5	6
Discovery time =	1	6	3	2	4	5
L =	1	1	1	1	3	3

→ **Step-2**

Step-3

u , v

↓ parent ↓ child

If for any, u,v $L[v] \geq d[u]$

then u is an Articulation Point [except root].

Topic Name : _____ Day : _____

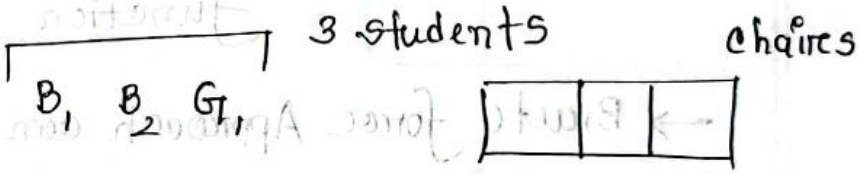
Time : Date : / /

Backtracking

Brute force Approach

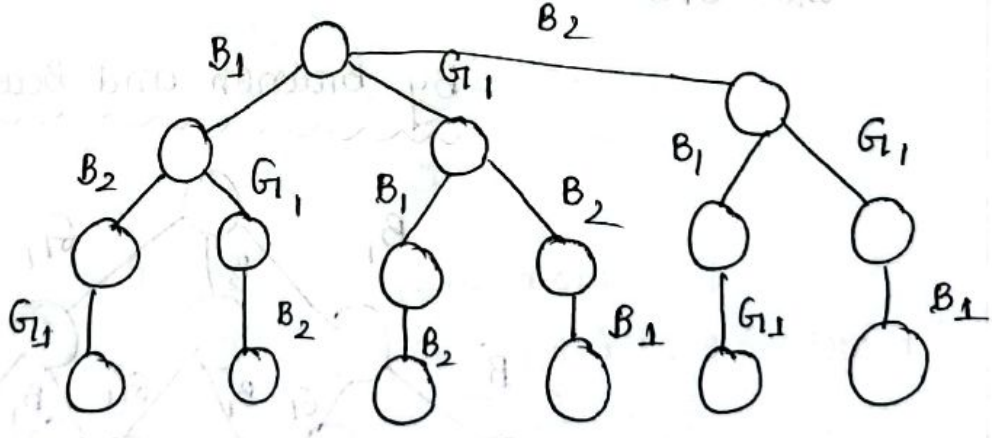
→ Backtracking is used for finding multiple solutions but DP is used to find one among every solution

→ DP is used for optimization problem, backtracking is not for optimization.



Brute force Algorithm uses: n=3 3! → possibility

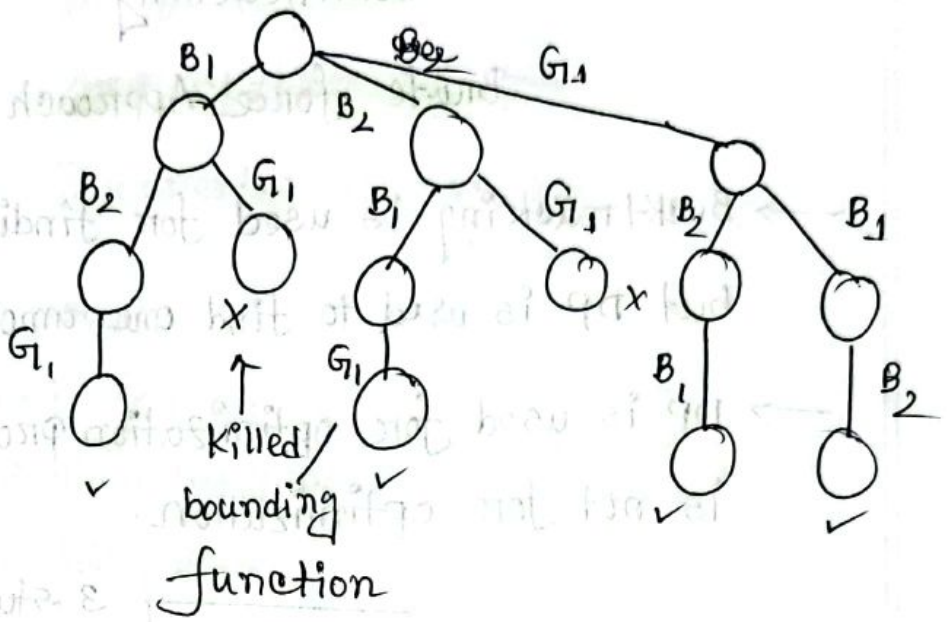
State space tree:



→ For backtracking we usually have some constraints.

For the example : constrain is, Girl can't seat in middle.

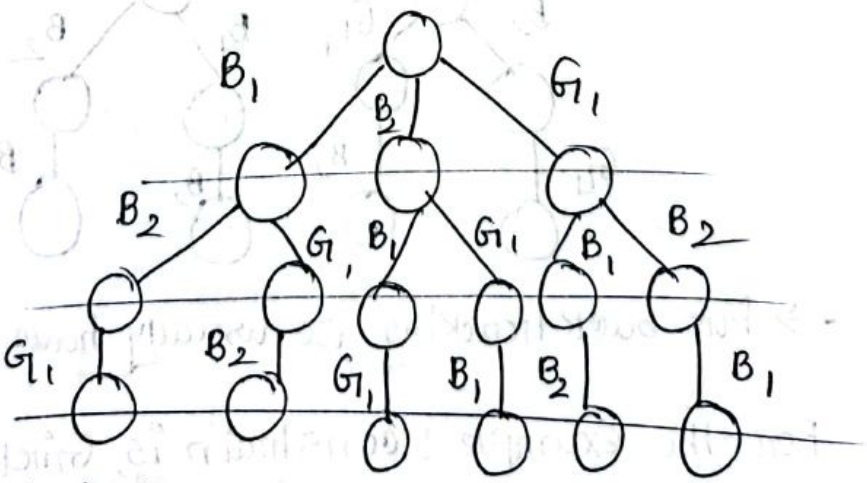
Then the SST will looks like:



→ Brute force Approach can also be solved with Branch and Bounds

→ Backtracking used DFS but Branch and Bound uses BFS.

By Branch and Bound



Problem solved Level Wise.

Topic Name : _____ Day : _____

Time : Date : / /

Backtracking Problems

N-Queens

Arranging Queens in not-attack

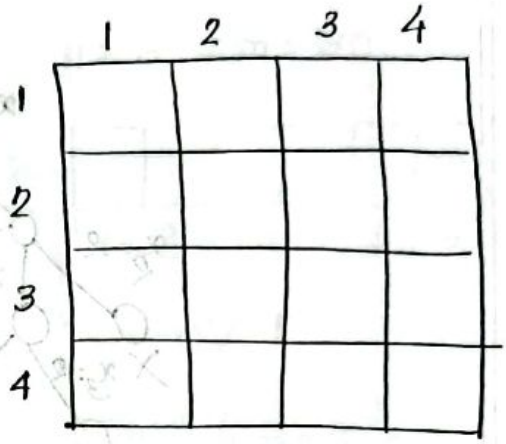
Avoid same

→ row

→ column

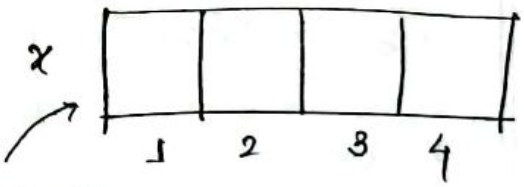
→ Diagonal

$^{16}C_4$ ↓ ways to arrange 4 Q.

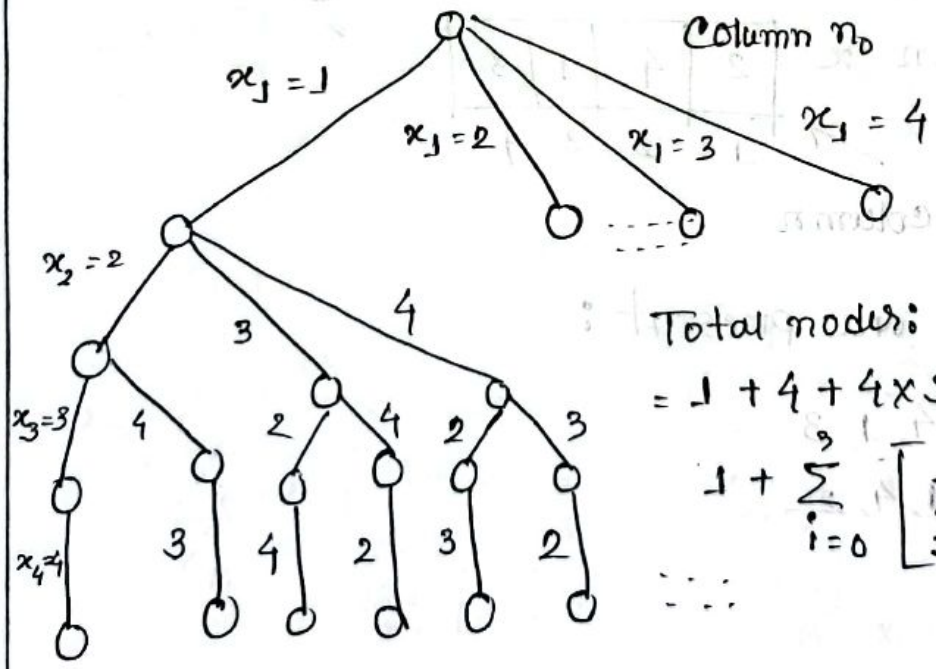


Solving with Backtracking:

State space Tree



Column n_0



Total nodes:

$= 1 + 4 + 4\times3 + 4\times3\times2 + 4\times3\times2\times1$

$1 + \sum_{i=0}^{3} \left[\prod_{j=0}^{i} (4-j) \right] = 65$

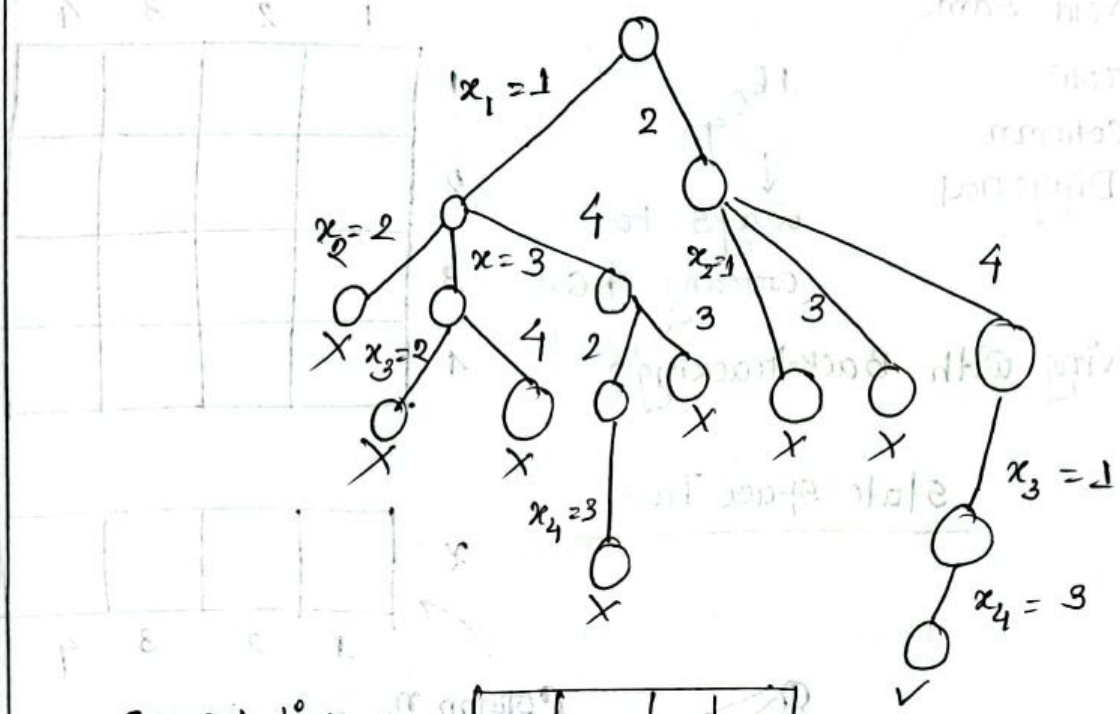
↓

Formula for maximum no. of nodes.

Topic Name : _____ Day : _____

Time : Date : / /

Bounding function = row
col
diag



So, solution x

2	4	1	3
1	2	3	4

column

2 solutions are present :

1. 2, 4, 1, 3
2. 3, 1, 4, 2

Topic Name : _____ Day : _____

Time : Date : / /

Sum of Subset

→ The problem is to find set that the sum of those are total equals to m.

→ solving this with backtracking it will take 2^n times.

So, constraint is required for Bounding function.

$w[1:6] = \{5, 10, 12, 13, 15, 18\}$

$n = 6 \quad m = 30$

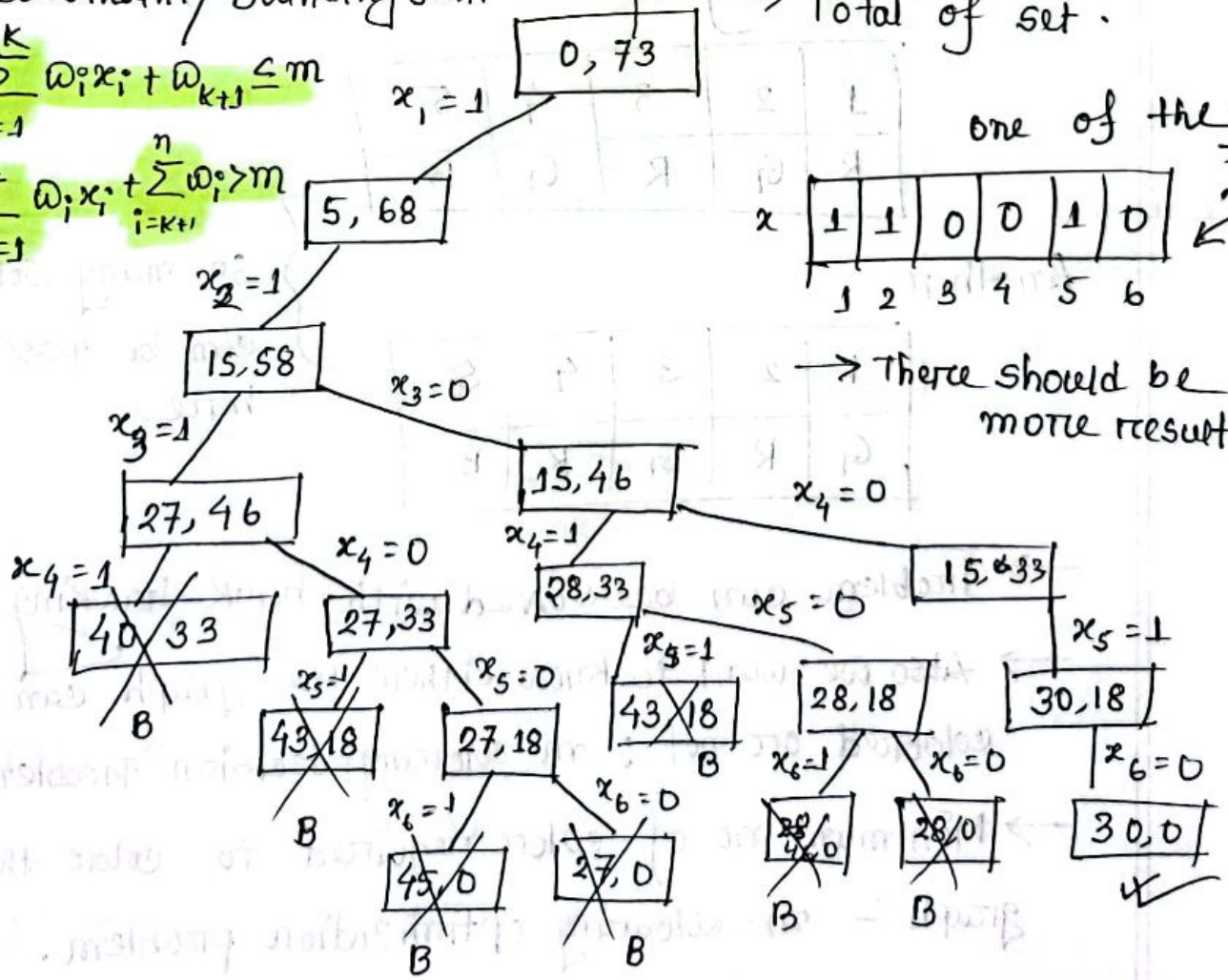
x: 1 2 3 4 5 6

$x_i = 0/1$

constraint / Bounding fun:

$$\sum_{i=1}^{k} w_i x_i + w_{k+1} \leq m$$

$$\sum_{i=1}^{k} w_i x_i + \sum_{i=k+1}^{n} w_i > m$$



one of the result

x: 1 1 0 0 1 0

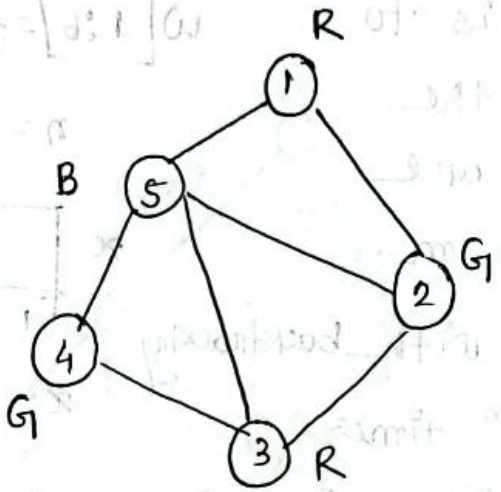
1 2 3 4 5 6

→ There should be more results.

Topic Name : _____ Day : _____

Time : Date : / /

Graph Coloring



colors

↓

m = 3

R, G, B

→ vertices should color such that no 2 adjacent vertices are of same colors.

1	2	3	4	5
R	G	R	G	B

Another

1	2	3	4	5
G	R	G	R	B

} So many solutions can be possible here.

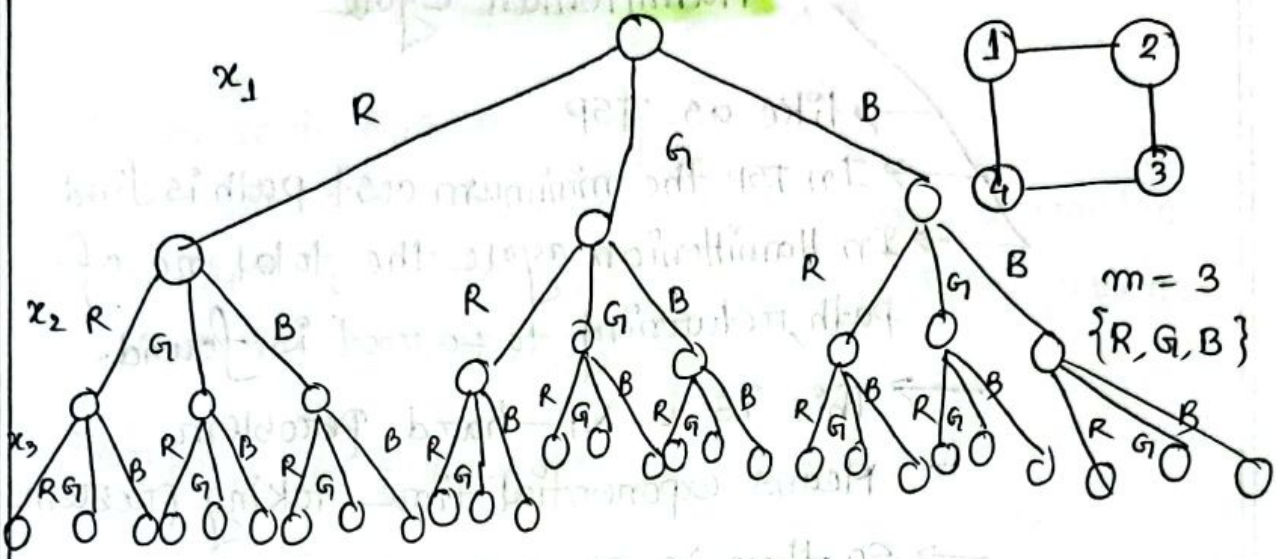
→ Problem can be solved with back tracking

→ ~~Also~~ we want to know either the graph can be coloured or not : m coloring decision problem

→ Minimum no. of color required to color the graph — m-colouring optimization problem.

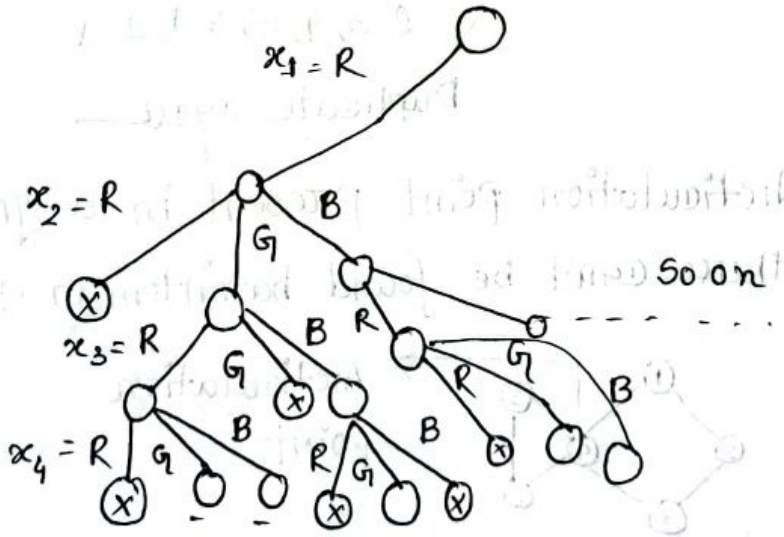
Topic Name : _____ Day : _____

Time : Date : / /



Statespace tree without any bounding function.

With bounding function: Two adjacent vertex can't be of same color



So, the time complexity will 3^n.

1) R, G, R, B
2) R, G, R, B
3) R, G, B, G
4) R, B, R, G
5) R, B, R, B

and more solutions.

Topic Name : _____ Day : _____

Time : Date : / /

Hamiltonian Cycle

→ like as TSP

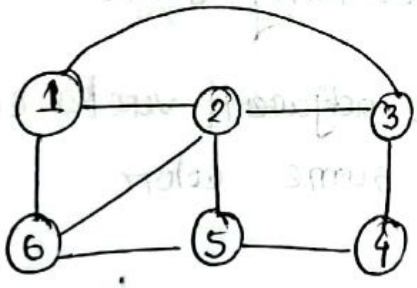
→ In TSP the minimum cost path is find

→ In Hamiltonian cycle the total no of path, returning to ~~po~~ root is found.

→ This is a NP-hard problem

Means exponential time taking problem

→ So, there is no easy way to find.



Paths:

1, 2, 3, 4, 5, 6, 1 ✓

1, 2, 6, 5, 4, 3, 1 ✓

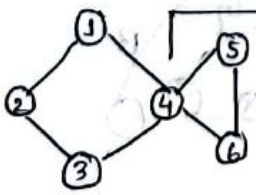
1, 6, 2, 5, 4, 3, 1 ✓

2, 3, 4, 5, 6, 1, 2 ✗

Duplicate cycle

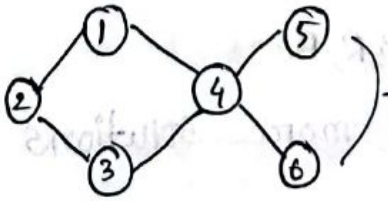
Note: if Articulation point present in a graph then there can't be found hamiltonian cycle.

ex:



→ Articulation point

if pendant ~~graph~~ vertices are present then can't found--



→ Pendent Vertices.

Topic Name : _____ Day : _____

Time : Date : / /

Branch and Bound

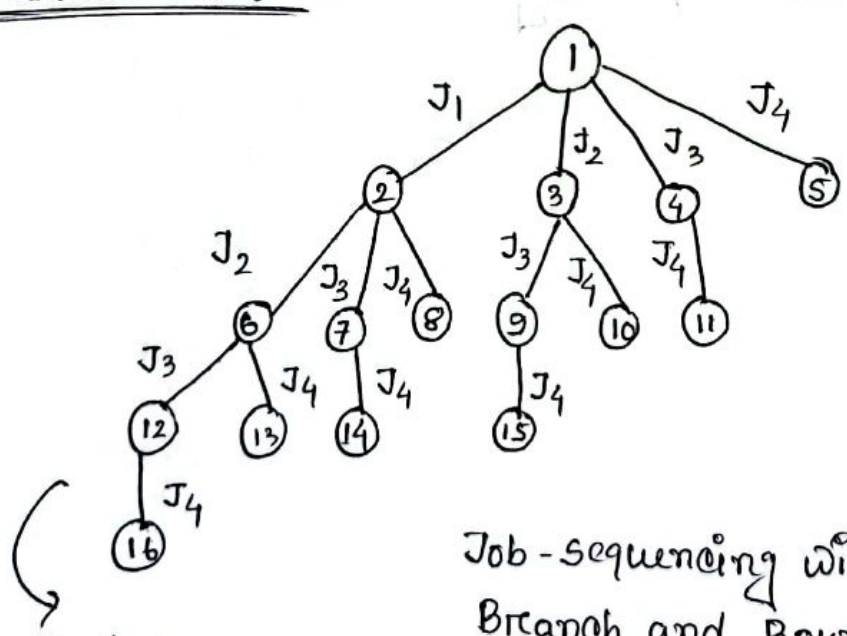
→ Similare to Backtracking
→ also use a state space Tree to solve a problem
→ But it is used for solving optimization problem
→ And only minimization problem.

Example Job sequencing

Jobs = { J_1, J_2, J_3, J_4 }
P = { 10, 5, 8, 3 }
d = { 1, 2, 1, 2 }

Say, solution:
$S_1 = \{J_1, J_4\}$
$S_2 = \{1, 0, 0, 1\}$
→ Variable size
→ Fixed size.

Variable size:



Job-sequencing with Branch and Bound.

Method - 1

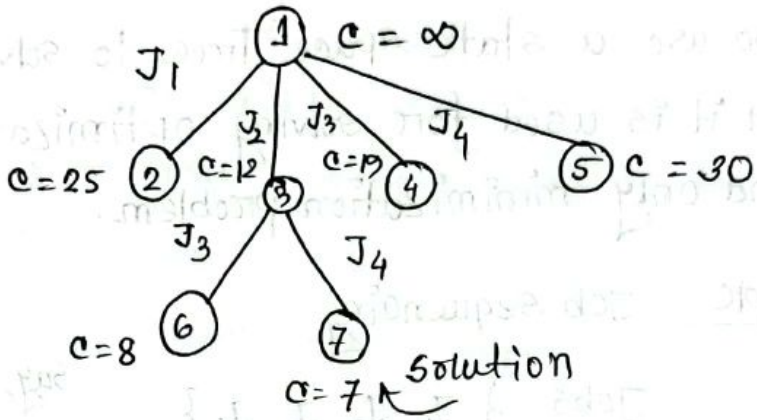
This used Queue for the solving → FIFO

M-2 used Stack " " " → LIFO

Topic Name : _____ Day : _____

Time : Date : / /

Least cost - BB



→ Quickly go to the result by the cost

→ Much faster than the other two.

Solving Job sequencing with deadline

Topic Name : _____ Day : _____

Time : Date : / /

0/1 Knapsack

↳ using LC - BB.

upper = ~~∞~~ -32 -38

	1	2	3	4
P.	10	10	12	18
W.	2	4	6	9

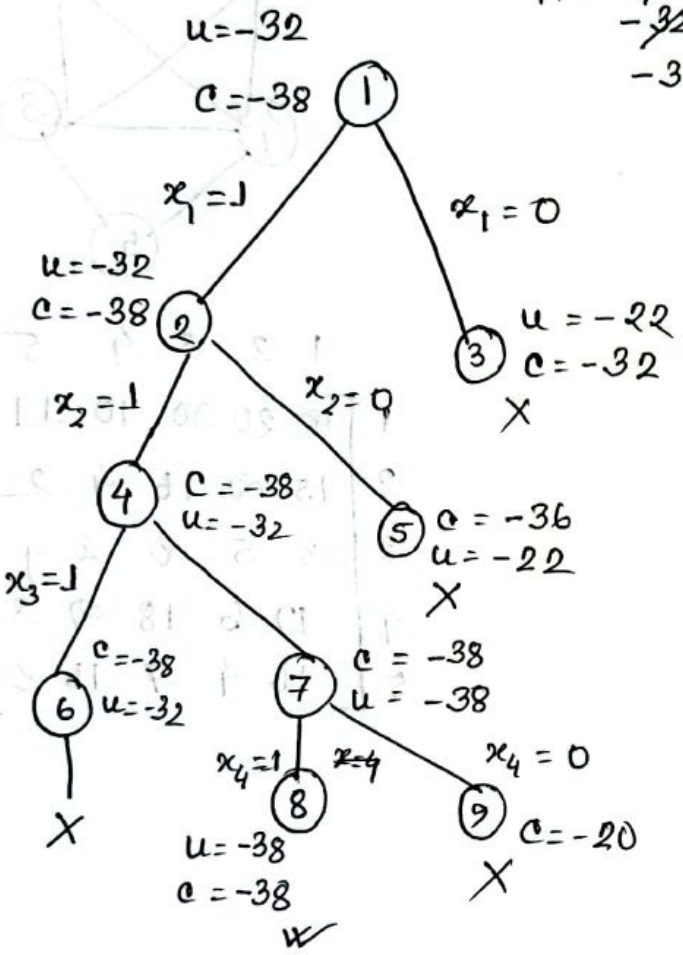
$m = 15$

$n = 4$

LC - BB

$u = \sum_{i=0}^{n} P_i x_i \leq m$

$c = \sum_{i=1}^{n} P_i x_i$ (With fraction)



Solution, $x = \{1, 1, 0, 1\}$

Profit = 10 + 10 + 18 = 38

2 + 4 + 9 = 15.