9/7/18 UNIT-1 ALGORITHM DEFINITION

* ALGORITHM:-

An algorithm is a finite set of instructio-ns that it follows accomplishions a particu-lar task.

All algorithms must satisfy following cond-itions:-

(i) Input (zero (or) more i/ps quantities are externally supplied.

ii) output (an atleast one quantity is produ-ced)

iii) Definiteness:-

Each instruction is clear & unambigu-ous (unrepeated)

iv) Finiteness:-

The algorithm terminates after finite number of steps.

v) Effectiveness:

Algorithm must be physical (und-erstandable)

The study of algorithms has 4 distinct areas

(1) How to device algorithm? (Applying an appropriate strategy)

(2) How to validate algorithm? (The process of validation is checking whether approach is exactly applicable for expected o/p)

(3) How to analyze algorithms (performance analysis)

(4) How to test a program?

After debugging we perform profiling (or) performance analysis.

It is a process of executing program on data sets (variables) & measuring time & space, it takes to computer results.

Algorithm specifications:-

Pseudo code conventions:-

Pseudo code is a template.

Algorithms are designed using a pseudo code that resembles c & pascal.

(1) comments begin with // and continue till end of line.

(2) Blocks (compound statements) are indicated with {, }.

(3) statements are delimited by semicolon.

(4) An identifier begin with a letter (alphabet)

(5) Data types of variables are not explicitly declared.

(6) compound data types can be formed with record = node.

```
node = record
{
  datatype1 data1;
  :
  :
datatypen datan;
}
```

* Assignment of values is done using :=.

  <Variable> := <expression>

* There are 2 boolean values are true & false.

  while (true)

* logical operators are and, or, not.

  while ((a>b) and (b>c))

* Relational operators are $<, \leq, >, \geq, =, \neq$

* Elements of multi-dimensional arrays are accessed. using A[i,j]

* loop statements are for, while, repeat until.

syntax:-

```
while (condition)do
 {
 <statement 1>;
  ¦
  ¦
 <statement n>;
 }
```

```
for variable := value1 to value2 step
                step do
{
<statement 1>;
  ¦
  ¦
<statement n>;
}
```

The statements are executed as long as condi-tion is false.

```
repeat
{
<statement 1>;
  ¦
  ¦
<statement n>;
```

3 until (condition);

---

if <condition> then <statement>

if <condition> then <statement> else
<statement2>

---

case
{
: <condition 1> : <statement1>
|
|
|
: <condition n> : <statement n>
: else : <statement n+1>
}

For incrementing 2 steps, we use :-

for (i:=1 to 10 step+2 do)
{
}

The unconditional statements are break, control, goto, continue, return.

I/p & O/p statements are done using read & write. There is only one type of procedure called as algorithm.

Algorithm Name (parameter list)

A, i

A, n

A, m, n

write an algorithm to find maximum elements in given array.

```
Algorithm Max (A,n)
{
x:=a[0];
for i:=1 to n-1 do
{
if x<A[i] then
 {
  x=a[i];
 }
}
return x; write "max value" x;
}
```

Recursive algorithm :-

A function calling itself is recursion.

```
Algorithm fact(n)
{
 if (n=1) then
     return 1;
```

```
else
  return n*fact(n-1);
}
```

Towers of Hanoi :-

```
Algorithm TOH (n,s,D,A)
{
 if (n=1) then
write "move from",s,"to"D;
return;
}
TOH (n-1,s,A,D);
write "move","n","from",s,"to",D;
TOH (n-1,A,D,s);
}
```

Performance analysis :-

Performance analysis is finding computing time & storage requirements. Performance evaluation can be done in 2 major phases :-

(i) priori estimate (performance analysis)

ii) posteriori testing (performance measurement)

Space complexity of algorithm :-

Space complexity of algorithm is amount of memory it needed to run to completion.

$$S(P) = c + S_p$$

S_p → Instance characteristic

c → constant

Time complexity of algorithm is amount of computer time it needs to run to completion.

$$T(P) = c + t_p$$

```
Algorithm abc(a,b,c)
{                    | | |
return a+b+b*c+(a+b-c)/(a+b)+4.0;
}
```

S(P) = 3 (No. of variables)

T(P) = 10 (No. of operations)

```
Algorithm sum (A,n)
{                                          ——1
 s:=0.0;                                   ——1
 for i:=1 to n do ——→ 1 }                  2(n)+1
   s:=s+A[i];           }
 return s;  ——→ 1
```

S(P) = 2+n

T(P) = 2(n)+1+1+1

= 2(n)+3

```
Algorithm sum (A,n)
 {
 s:=0.0;  ——→1

 count:=count+1;  ——→ 2
for i:=1 to n do ——→1 }
{                      } 2n
count:=count+1;        }
s:=s+A[i], count:=count+1;
}    2n                2n
count:=count+1; 2
 count:=count+1;2
 return s; 1
 }
```

T(P) = 1+2+2+2n+2n+2n+1+2+1

= 6n+9

$S(P) = 3+n$

* Algorithm sum (a, n)

{

for i := 1 to n do count := count + 2; → 1, → 2n

count := count + 2; → 2

}

$SP = 2+n$

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

$= 3+2n$

* Algorithm add (a, b, c, m, n)

{

for i := 1 to m do → 1+m

for j := 1 to n do → m(n+1)

c[i, j] := a[i, j] + b[i, j]

} ↓ mn ↓ mn

$T(P) = 2mn+mn+m+m+1$

$= 3mn+2m+1$

$S(P) = 3(m \times n)+2$

$= 3mn+2$

* Algorithm add (a, b, c, m, n)

```
{
for i:=1 to m do → 1+m
{
count:=count+1; → 2m
for j:=1 to n do → m(n+1)
{
count:=count+1; → 2mn
c[i,j]=a[i,j]+b[i,j]; → 2mn
count:=count+1; → 2mn
}
count:=count+1; → 2m
}
count:=count+1; → 2
}
```

$S(P) = 3mn+3$

$T(P) = 6m+3+7mn$

```
Algorithm add(a,b,c,m,n)
{
for i:=1 to m do
{
count:=count+2;
for j:=1 to n do
count:=count+2;
}
count:=count+1;
}
```

$T(P) = 4n+1$

```
Algorithm fibanocci(n)
{
if (n≤1) then
write(n); → 1
else {
fnm2:=0; fnm1:=1; 2
for i:=2 to n do { ①
fn=fnm1+fnm2; 2(n-1)
fnm2:=fnm1; fnm1:=
fnm; n-1 n-1
}
write(fnm); } }
```

16/7/18

Assymptotic notation :-

To shorten representation of time complexity & space complexity we use asymptotic notations

(1) O (or) worst case (or) upper bound

(2) ω (or) best case (or) lower bound

(3) θ (or) average case/average bound

(4) little ω $\longrightarrow$

(5) little o $\rightarrow$ very worst case

(1) Big oh :-

The function $f(n) = O(g(n))$ if & only if, there exists positive constants c & n_0 such that $f(n) \leq c \times g(n)$ for all n, $n \geq n_0$.

Prove that $3n+2 = O(n)$

$3n+2 \leq 4 \times n$

$3n+2 \leq 4n$ for $n=1$

$5 \leq 4$

for $n=2$,

$8 \leq 8$

for $n \geq 2$, $3n+2 = O(n)$

for $n=3$,

$11 \leq 12$ ✓

Represent $10n^2+4n+2$ in Big oh notation.

$10n^2+4n+2 \leq 11n^2$

for $n=1$

$16 \leq 11$

for $n=2$

$50 \leq 44$

for $n=3$

$104 \leq 99$

for $n=4$

$178 \leq 176$

for $n=5$, $272 \leq 275$ ✓

So, for $\forall\ n \geq 5$, $10n^2+4n+2 = O(n^2)$.

Prove that $6 \times 2^n + n^2 = O(2^n)$

for $n=1$

$6 \times 2^n + n^2 \leq 6+1 \cdot 2^n$

$6 \times 2 + 1 \leq 7 \cdot 2$

$13 \leq 14$

for, $n \geq 1$, $6 \times 2^n + n^2 = O(2^n)$

Prove that $3n+3 = O(n^2)$

for $n=1$ $\quad [0+1=1]$

$3+3 = 1(n^2)$ $\quad$ n^2 coefficient

$6 \leq 1$

for $n=2$,

$9 \leq 1(4)$

$9 \leq 4$

for $n=3$, $12 \leq 9$

→ Prove that $4n+2 = O(n)$

for $n=1$

$4+2 \leq 4(n)$

$6 \leq 4+1 \Rightarrow 6 \leq 5$

for $n=2$

$8+2 \leq 5(n)$

$10 \leq 10$

omega :- The function $f(n) = \omega(g(n))$ iff $\exists$ positive constants c & n_0 such that $f(n) \geq c \times g(n)$ for all n, $n \geq n_0$.

Ex :- $10n^2+4n+2 = \omega(n^2)$

for $n=1$

$10+4+2 \geq 10 \times n^2$

$16 \geq 10$

For $n=1$, $10n^2+4n+2 = \omega(n^2)$

(2) $6 \times 2^n + n^2 = \omega(2^n)$

for $n=1$,

$6 \times 2 + 1 \geq 6 \times 2^1$

$13 \geq 12$

for $n=2$,

$6 \times 2^2 + 4 \geq 6 \times 2^2$

θ (Theta) :- The function $f(n) = \theta(g(n))$ iff $\exists$ positive constants c_1, c_2 & n_0 such that $c_1(g(n)) \leq c_2(g(n)) \leq f(n)$ for all n, $n \geq n_0$;

Ex:- prove that $3n+2 = \theta(n)$

$3 \times n \leq 3n+2 \leq 4 \times n$

for $n=1$,

$3 \leq 5 \leq 4$

for $n=2$,

$6 \leq 8 \leq 8$

prove that $10n^2+4n+2 = \theta(n^2)$

$10 \times n^2 \leq 10n^2+4n+2 \leq 11 \times n^2$ for $n=1$,

$10 \leq 16 \leq 11$

for n=2,

$10 \times 4 \leq 10 \times 4 + 8 + 2 \leq 11 \times 4$

$40 \leq 50 \leq 44$

for n=3,

$10 \times 9 \leq 10 \times 9 + 12 + 2 \leq 11 \times 9$

$90 \leq 104 \leq 99$

for n=4, $160 \leq 178 \leq 176$

4) Little 'o' :-

The function $f(n) = o(g(n))$ iff $\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$.

5) Little 'ω' :-

The function $f(n) = \omega(g(n))$ iff $\lim_{n \to \infty} \frac{g(n)}{f(n)} =$

1.

Divide & conquer :-

General method (or) control abstraction :-

Given function to compute on 'n' i/ps, divide & conquer strategy suggests splitting i/ps into 'k' distinct subsets, $1 < k \leq n$, yielding 'k' subprob-ms, these sub-problems must be solved & a me-thod must be found to combine sub-solutions into soln's of whole.

Algorithm DAndc(P)

{

if small(P) then return s(P);

else

{

divide P into smaller instances $P_1, P_2 \text{----}$,

$k \geq 1$;

Apply DAndc to each of these subproblems;

return combine (DAndc(P_1), DAndc(P_2), ---

--- DAndc(P_k));

}

}

$$T(n) = \begin{cases} g(n) \text{ ---------}, & n \text{ small otherwise} \\ T(n_1)+T(n_2)+\text{---}+T(n_k)+f(n) \end{cases}$$

↳ time for dividing & combining

$$T(n) = \begin{cases} T(1) & n=1 \\ aT\left(\frac{n}{b}\right)+f(n) & n>1, n=b^k \end{cases}$$

$T(n_1)$ → time taken for solving sub-problem P_1.

→ Time-complexity of merge-sort $= 2T\left(\frac{n}{2}\right)+c(n)$

sub-stitution method :-

To shorten time complexity of recursive algorithms.

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

$a=2, b=2, f(n)=n, n=2^k \rightarrow k=\log_2 n$

$$= 2\left[2T(n/4) + n/2\right] + n$$

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

$$= 4\left[2T(n/8) + n/4\right] + \frac{3n}{3}$$

$$= 8T(n/8) + 3n$$

$$\vdots$$

$$= 2^k T(n/2k) + kn$$

$$= nT(n/n) + n\log_2 n$$

$$= T(n + n\log_2 n)$$

$$\cong O(n\log n)$$

Method-2 :- [Master's theorm]

$$T(n) = n^{\log_a^b}\left[T(1) + \mu(n)\right]$$

$$= n^{\log_b^a}\left[T(1) + \mu(n)\right]$$

$$\mu(n) = \sum_{j=1}^{k} h(b^j)$$

$$h(n) = \frac{f(n)}{n^{\log_b^a}}$$

$h(n)$	$\mu(n)$
$O(n^r) r<0$	$O(1)$
$\theta((\log n)^i)_{i \geq 0}$	$\theta(\log n)^{i+1}/i+1$
$\Omega(n^r) r>0$	$\theta(h(n))$

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

$$a=2, b=2, f(n)=n$$

$$T(n) = n^{\log_2^2}[T(1) + \theta(\log n)]$$

$$h(n) = n/n^{\log_2^2}$$

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

$$a=1, b=2, f(n)=c$$

$$T(n) = n^{\log_2^1}[T(1)+$$

$$h(n) = c/n^{\log_2^1} = c/n^0 = c(1)$$

$$= c(\log n)^0$$

$$=$$

* $T(n) = 7T(n/2) + 18n^2$

$a = 7, b = 2, f(n) = 18n^2$

$T(n) = n^{\log_2^7}[T(1) + \theta \log n]$

$= n^2[T(1) + \theta \log n]$

$h(n) = \frac{18n^2}{n^{\log_2^7}}$

$= \frac{18n^2}{n^2} = 18(1) =$

$h(n) = 18(\log n)^0$

$= 18(\log n)^0$

$H(n) = \theta(\log n)/_1$

$= \theta(\log n)$

$T(n) = \theta(n^2 \log n)$

* $T(n) = 9T(n/3) + 4n^6$

$a=9, b=3, f(n)=4n^6$

$T(n) = n^{\log_3^9}[T(1) +$

$= n^2[T(1) + \theta(\Omega(n^4))] = n^2[T(1) + \theta(n^4)]$

$= \theta(n^6)$

$h(n) = \frac{4n^6}{n^2}$

$= 4n^4$

$h(n) = \Omega(n^4)$

$\mu(n) = \theta(h(n))$

$= \theta(\Omega(n^4))$

$= \theta(n^4)$

Binary-search :-

Algorithm Binsrh(a, i, l, x) (l → high)

```
{
 if (l=i) then
  {
    if (x=a[i]) then return i;
  else return 0;
 }
 else
 {
   mid := [(i+l)/2];
```

```
if (x = a[mid]) then return mid;
elseif (x < a[mid] then return Binsrh(a, l,
                                    mid-1, x);
else return Binsrh (a, mid+1, l, x);
}
}
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13
-15	-6	0	7	9	23	54	82	101	112	125	131	142	151

search of 151

i	l	mid
0	13	6
7	13	10
11	13	12
13	13	

search of 101

i	l	mid
0	13	6
7	13	10
9	13	9
10	13	17

Iterative algorithm:-

```
Algorithm Binsrch (a, n, x)
{
  low := 1; high := n;
  while (low ≤ high)
{
  mid := [(low+high)/2];
  if (x < a[mid]) then high := mid-1;
elseif (x > a[mid]) then low := mid+1;
```

else return mid;

} → return 0;
}

$T(n) = T(n/2) + c$ $\quad f(n) = c$

$= [T(n/4) + c] + c$ $\quad n = 2^k \; [n = b^k]$

$k = \log_2 n$

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

$= [T(n/8) + c] + 2c$

$= [T(n/8) + 3c]$

⋮

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

$= T(1) + c \log_2 n$

$\cong O(\log n)$

Merge-sort algorithm:-

```
Algorithm Mergesort (low, high)
{
 If (low < high) then
 {
 mid := [(low+high)/2];
```

```
Mergesort (low, mid);
Mergesort (mid+1, high);
Merge (low, mid, high);
}}
Algorithm Merge (low, mid, high) {
h:=low; i:=low; j:=mid+1;
while (h<=mid) and (j<=high) do {
if (a[h]<=a[j]) then{
b[i]:=a[h]; h:=h+1; }
else{
  b[i]:=a[j]; j:=j+1; }
i:=i+1;
}
if (h>mid) then
   for k:j to high do
  {
  b[i]:=a[k]; i:=i+1;
}
else
  for k:=h to mid do
  {
  b[i]:=a[k]; i:=i+1;
}
for k:=low to high do a[k]:=b[k]; }
```

Quick-sort (partition sort) :-

```
Algorithm Pattern (a,m,p)
{
V= a[m]; i:=m; j:=p;
repeat
{
 repeat
     i:=i+1;
   until (a[i]≥V);
 repeat
     j:=j-1;
   until (a[j]≤V);
 if (i<j) then interchange (a,i,j);
}until (i≥j);
a[m]:=a[j]; a[j]:=V; return j;
}
Algorithm Interchange (a,i,j)
 {
   p:=a[i];
 a[i]:=a[j]; a[j]:=p;
}
 Algorithm Quicksort (P,q)
  {
   if (p<q) then
   {
```

```
j = partition (a, p, q+1);
Quicksort (p, j-1);
Quicksort (j+1, q);
}
}
```

Time complexity :- (Best case) = $O(n \log n)$

① ② 3 4 5

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

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

$= T(n-2) + 2cn - c$

$= [T(n-3) + c(n-2)] + 2cn - c$

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

$= [T(n-4) + c(n-3)] + 3cn - 3c$

$= T(n-4) + 4cn - 6c$

$= [T(n-5) + c(n-4)] + 4cn - 6c$

$= T(n-5) + 5cn - 10c$

⋮

$= T(n-k) + kcn - (1+2+---+(k-1))c$

$= T(n-k) + kcn - c\frac{k(k-1)}{2}$ [k can repeat upto 'n' numbers]

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

$= T(0) + cn\left(n - \frac{n-1}{2}\right)$

$= T(0) + cn\left(\frac{2n-n+1}{2}\right)$

$= T(0) + cn^2 + \frac{cn}{2}$

$T(n) = O(n^2)$ for worst case

$S(p) = O(n)$

23/7

~~Disjoint sets :-~~ STRASSENS MATRIX MULTIPLICATION

Method-1 :-

NORMAL METHOD :-

let A, B be 2 $n \times m$ matrices, product matrix $C = AB$ is also $n \times m$ matrix. $c(i,j) = \sum_{1 \le k \le n} A(i,k)\, B(k,j)$ for all i and j between 1 and n with a time complexity of $O(n^3)$ to reduce time complexity we go for conventional method.

1) conventional method :-

we used divide and conquer approach where n is a power of 2 i.e., $n = 2^k$. In case if $n \neq 2^k$ then add enough 0's to rows & columns.

Formulaes :-

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

$C_{11} = A_{11}B_{11} + A_{12}B_{21}$

$C_{12} = A_{11}B_{12} + A_{12}B_{22}$

$C_{21} = A_{21}B_{11} + A_{22}B_{21}$

$C_{22} = A_{21}B_{12} + A_{22}B_{22}$

$$T(n) = \begin{cases} b & n \le 2 \\ 8T(n/2) + c^2 & n > 2 \end{cases}$$

$a = 8, b = 2, f(n) = cn^2$

By Masters theorm,

$$h(n) = \frac{cn^2}{n^{\log_2 8}} = \frac{cn^2}{n^3} = cn^{-1}$$

$h(n) = O(n^1)$

$M(n) = O(1)$

$T(n) = n^3 [T(1) + O(1)]$

$= O(n^3)$

METHOD-3 :-

strassen's Matrix multiplication :-

$P = (A_{11} + A_{22})(B_{11} + B_{22})$

$Q = (A_{21} + A_{22}) B_{11}$

$R = A_{11} (B_{12} - B_{22})$

$S = A_{22} (B_{21} - B_{11})$

$T = (A_{11} + A_{12}) B_{22}$

$U = (A_{21} - A_{11})(B_{11} + B_{12})$

$V = (A_{12} - A_{22})(B_{21} + B_{22})$

$$T(n) = \begin{cases} b & n \le 2 \\ 7T(n/2) + cn^2 & n > 2 \end{cases}$$

$C_{11} = P + S - T + V$

$C_{12} = R + T$

$C_{13} = Q + S = C_{21}$

$C_{22} = P + R - Q + U$

$a=7, b=2, f(n)=cn^2$

$$h(n)=\frac{cn^2}{n\log_2^7}=\frac{cn^2}{n^{2.81}}=cn^{-0.81}$$

$$=O(n^{-0.81})$$

$$H(n)=O(1)$$

$$T(n)=n^{2.81}[T(1)+O(n)]$$

$$T(n)=O(n^{2.81})$$

Eg:-

$$\begin{bmatrix}1&2&3\\1&2&1\\1&4&1\end{bmatrix}\begin{bmatrix}1&3\\2&1\\1&1\end{bmatrix}$$

$$=\left[\begin{array}{cc|cc}1&2&3&0\\1&2&1&0\\\hline1&4&1&0\\0&0&0&0\end{array}\right]\left[\begin{array}{cc|cc}1&3&0&0\\2&1&0&0\\\hline1&1&0&0\\0&0&0&0\end{array}\right]$$

A_{11} A_{12} B_{11} B_{12}

A_{21} A_{22} B_{21} B_{22}

$$P=\begin{bmatrix}6&8\\5&5\end{bmatrix},\ Q=\begin{bmatrix}10&10\\0&0\end{bmatrix},\ T=\begin{bmatrix}0&0\\0&0\end{bmatrix}$$

$$R=\begin{bmatrix}0&0\\0&0\end{bmatrix}\qquad U=\begin{bmatrix}0&0\\0&2\end{bmatrix}$$

$$S=\begin{bmatrix}1&0\\0&0\end{bmatrix}\begin{bmatrix}0&-2\\-2&-1\end{bmatrix}\qquad V=\begin{bmatrix}2&0\\0&0\end{bmatrix}$$

$$=\begin{bmatrix}0&-2\\0&0\end{bmatrix}$$

$$C_{11}=\begin{bmatrix}8&8\\5&5\end{bmatrix},\ C_{22}=\begin{bmatrix}-4&-2\\9&7\end{bmatrix}$$

$$C_{21}=\begin{bmatrix}10&8\\0&0\end{bmatrix}\qquad C_{12}=\begin{bmatrix}0&0\\0&0\end{bmatrix}$$

RANDOMIZED ALGORITHM:

Randomly choosing number of for operation is randomization.

For ex :- flipping a coin.

Randomized algorithm follows two approaches:-

(1) LAS VEGAS :- These are randomized algorithms which always gives correct answer. The running time is non-deterministic or not fixed for same I/P. Ex:- Randomized Quicksort.

we make system to choose random no. of as pivot, it may result in best case or worst case.

(2) MONTECARLO ALGORITHM:

These are randomized algorithm which may not gives correct answer but running time of algorithm is fixed.

Ex:- 15 dice puzzle problem

The empty slide may be moved randomly.

25/7/18

UNIT-2

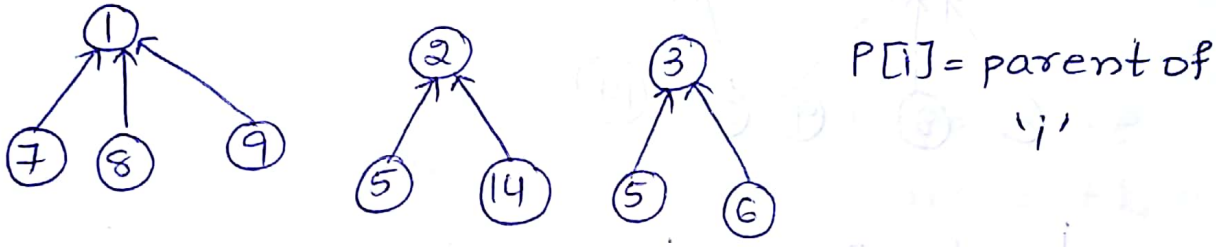
Disjoint sets :-

The elements of sets are numbers which can be represented as s_i & s_j, where $i \neq j$ i.e, there is no element same in both s_i, s_j.

$S_1 = \{1, 7, 8, 9\}$

$S_j = \{2, 5, 14\}$

$S_3 = \{3, 5, 6\}$



P[i] = parent of 'i'

i	1	7	8	9
P[i]	-1	1	1	1

i	2	5	14
P[i]	-1	2	2

i	3	5	6
P(i)	-1	3	3

The operations we perform on sets are :-

i) Union :- If s_i, s_j are 2 disjoint sets, there union equals to all elements that are in s_i & s_j.

$s_1 \cup s_2$.

ii) Find (i) :- Gives element i, finds set containing i. i.e, return overall root of tree.

simple Union & simple find() :-

Algorithm simpleUnion (i, j)

{

P[i]:=j, (or) P[j]:=i;

}

Algorithm simpleFind (i)

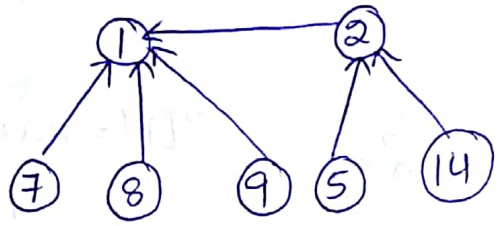
{

while (P[i]≥0) do i:=P[i];

return i;

}

Ex:-



i	1	2	7	8	9	5	14
P(i)	-1	1	1	1	1	2	2

overall time complexity of a forest = O(n).

For simpleFind if for n nodes, Time complexity is $n+n-1+\_\_\_+1$

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

suppose Each set has one elements (forest) is given where parent of each element = -1, we apply simple union then it leads to degenerate tree of (n-1) levels. Therefore T(n)=O(n).

Forest :-

	S_1	S_2	S_3	----	S_n
i	1	2	3	----	n
P(i)	2	3	4	----	-1

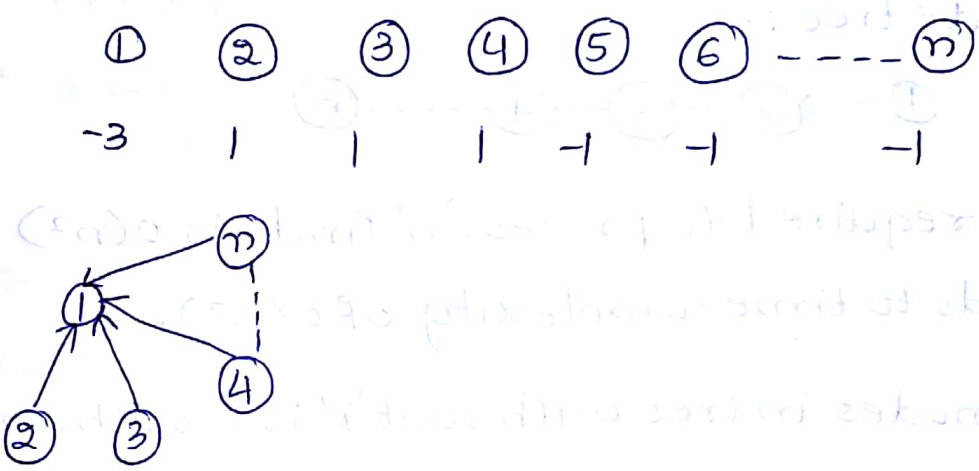
Degenerate tree :-

The time required to process 'n' finds is $O(n^2)$ which leads to time complexity of $O(n^2)$.

If no. of nodes in tree with root 'i' is less than number in tree with root j, make j parent of i, otherwise make i parent of j.

```
Algorithm weightedUnion(i,j)
//weighting rule P[i]=-count[i]and P[j]=
                    -count[j]
  {
  temp := p[i]+p[j];
if(P[i]>p[j]) then
{
   P[i]:=j , P[j]:=temp;
}
else
 {
   p[j]:=i, p[i]:=temp;
}}
```

In weighted union to construct a tree for a forest, time complexity = O(1). To process 'n' finds O(n) each find operation requires one jump, T(n) = O(n-1) = O(n).



collapsing rule for find :-

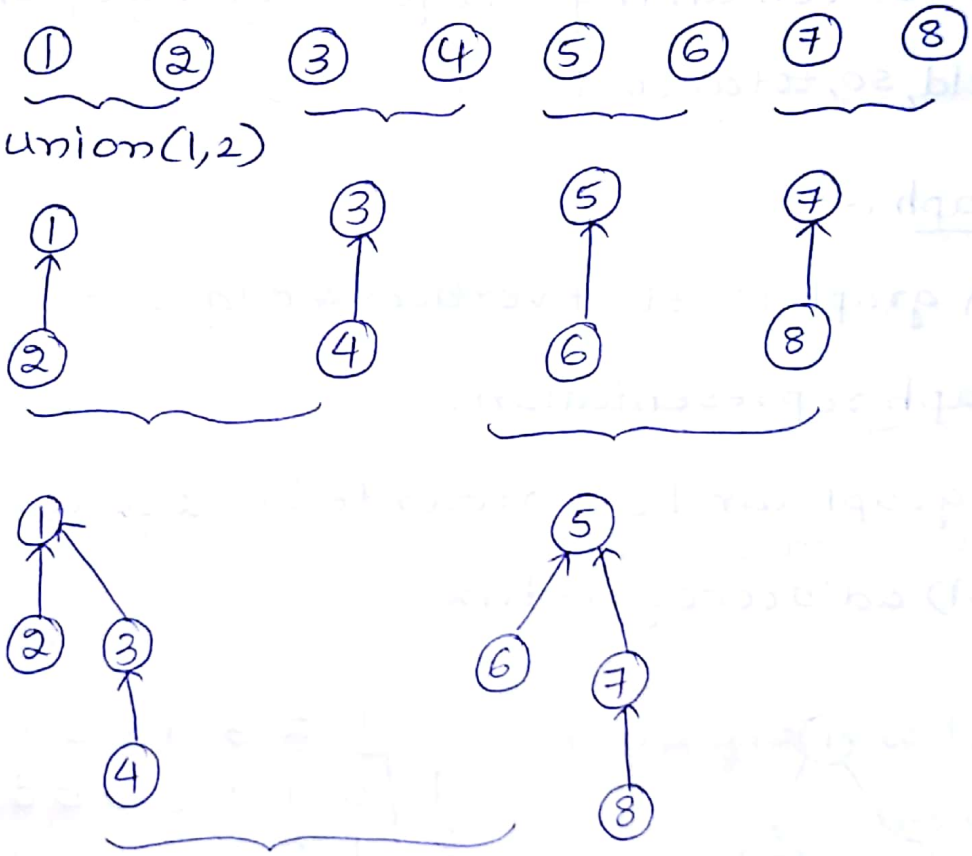
If j is a node on path from i to its root & p(i) ≠ root of i, then set p[j] to root of i.

```
Algorithm simpleFind (i)
{
while (p[i] ≥ 0) do i := p[i];
return i;
}
Algorithm collapsingFind (i)
{
r := i;
while (p[r] > 0) do r := p[r];
```

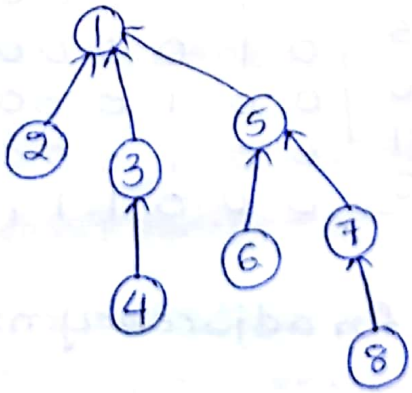
i	1	2	3	4	5	6	7	8
p(i)	-1	1	1	3	1	5	5	1

```
while (i ≠ r) do
 {
 s:= P[i]; P[i]:= r; i:= s;
}
return r;
}
```

Ex:-



union (1,5)



i	1	2	3	4	5	6	7	8
P(1)	-1	1	1	3	1	5	1	1

In simplefind to process 8 find(8) in simplefind each find(8) requires going up 3 parent leaf fields for total of 8x3=24 moves.

when collapsing find is used, 1st find(8) requires going up 3 leaf fields, resetting 3 leaf fields, each of remaining 7 requires going up only 1 link field, so, total cost = 3+3+7 = 13

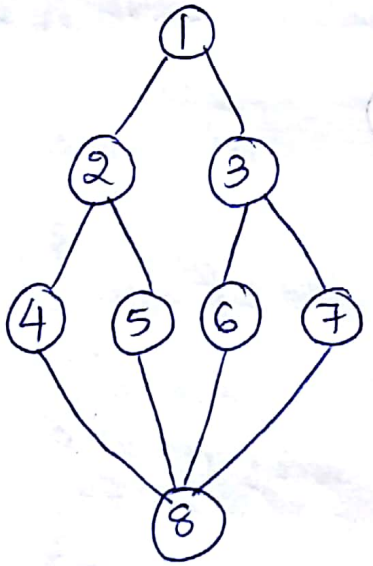
Graph :-

A graph is set of vertices & edges. G = (V, E)

Graph representation :-

A graph can be represented in 2 ways :-

(1) adjacency matrix

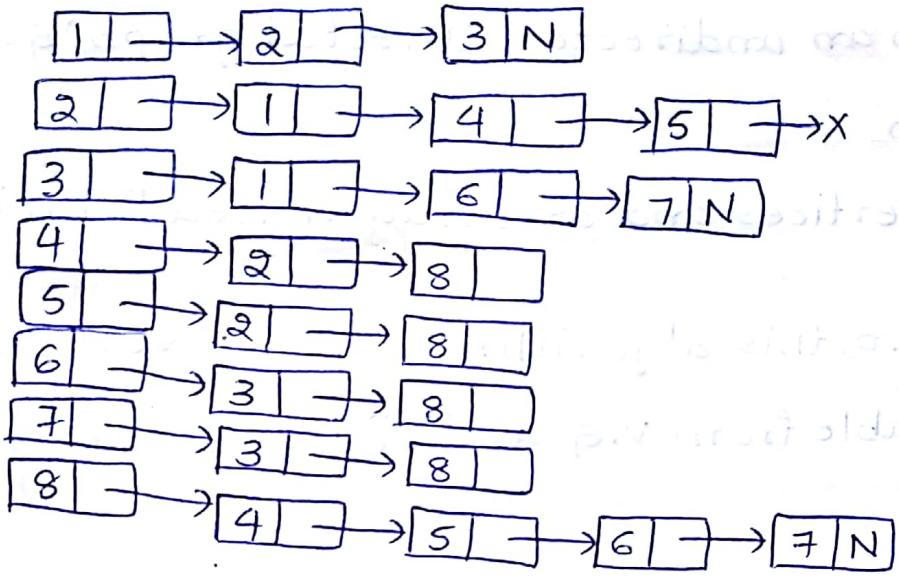


	1	2	3	4	5	6	7	8
1	0	1	1	0	0	0	0	0
2	1	0	0	1	1	0	0	0
3	1	0	0	0	0	1	1	0
4	0	1	0	0	0	0	0	1
5	0	1	0	0	0	0	0	1
6	0	0	1	0	0	0	0	1
7	0	0	1	0	0	0	0	1
8	0	0	0	1	1	1	1	0

$T(n) = O(n^2)$ for adjacency matrix

$S(P) = O(n)$

ii) Adjacency list :-



$T(n) = O(n+e)$

Graph traversals :-

(1) Breadth first traversal (Queue) (connected)

(2) Depth first tranversal (stack) (bi-connected)

Spanning tree :-

A tree derived/constructed from a graph with n vertices & (n-1) edges using BFS & DFS technique.

(2) Depth first search :-

Vertex 'v' is suspended as new vertex is reached. At this time, exploration of new ver-tex begins. When this new vertex has been explored, exploration of 'v' continues. The search terminates when all reached vertices are fully explored.

Algorithm DFS(v)

// given an undirected (directed graph) G = (V, E) with

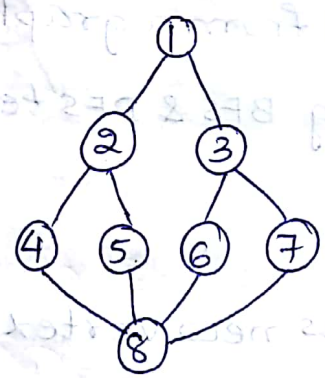
// n vertices and an array visited [ ] initially set

// to zero, this algorithm visits all vertices

// rechable from v.G and visited [ ] are global

```
{
   visited[v] := 1;
  for each vertex w adjacent from v do
 {
  if (visited[w] < 0) then DFS(w);
 }
}
```

Ex:-



[top to bottom]
left to right

DFS(1) = 2✓, 3

DFS(2) = 4✓, 5, 1✗

DFS(4) = 2✗, 8✓

DFS(8) = 4✗, 5✓, 6✓, 7

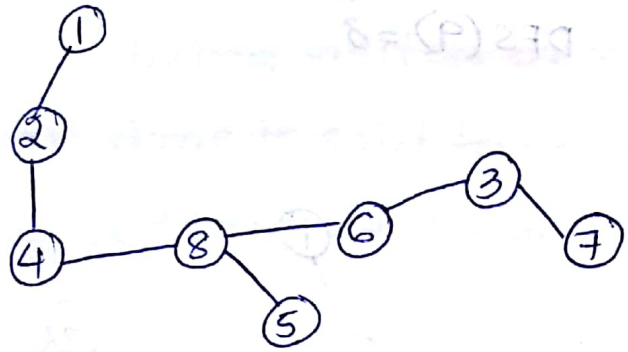
DFS(5) = 2, 8
✗ ✗

DFS(6) = 3✓, 8✗

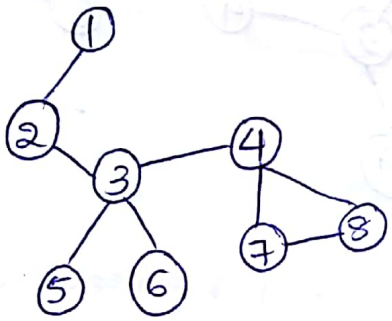
DFS(3) = 1✗, 6✗, 7✓

DFS(7) = 3✗, 8✗

spanning tree :-



Ex :-



DFS(1) = 2✓

DFS(2) = 1×, 3✓

DFS(3) = 2×, 5✓, 6✓, 4✓

DFS(5) = 3×

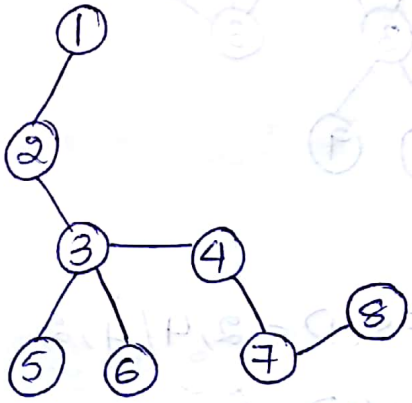
DFS(6) = 3×

DFS(4) = 3×, 7✓, 8

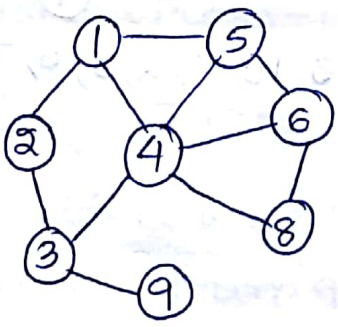
DFS(7) = 4×, 8✓

DFS(8) = 4×, 7×

spanning tree :-



Ex :-



DFS(1) = 2, 4, 5

DFS(2) = 1, 3

DFS(3) = 2, 4, 9

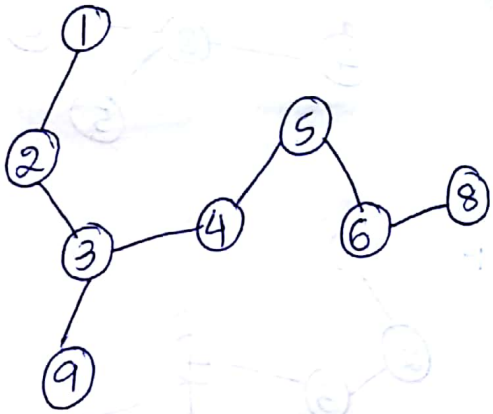
DFS(4) = 1, 5, 6, 8, 3

DFS(5) = 1, 6, 4

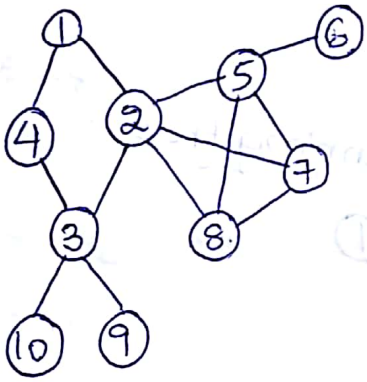
DFS(6) = 4, 5, 8

DFS(8) = 4, 6

DFS(9) = 3



Ex:-



DFS(1) = 2, 4/4, 2

DFS(4) = 1, 3

DFS(3) = 4, 10, 9, 2

DFS(10) = 3

DFS(9) = 3

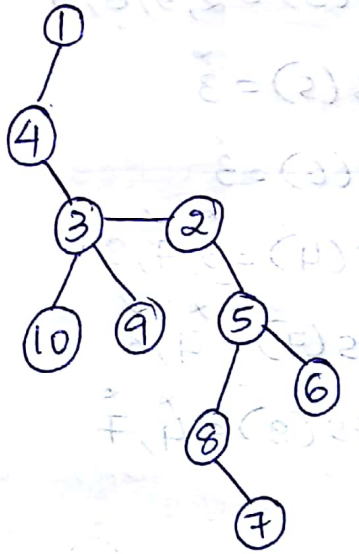
DFS(2) = 5, 8, 7

DFS(5) = 2, 8, 7, 6

DFS(8) = 2, 5, 7

DFS(7) = 2, 5, 8

DFS(6) = 5



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

Breadth First SEARCH (Queue) :-

In breadth first search we start at a vertex 'v' mark it as being visited or reached. The vertex 'v' at this time is said to be unexplored. If a vertex is said to be explored if we find all the adjacent vertices from it.

All these adjacent vertices are visited next and explored.

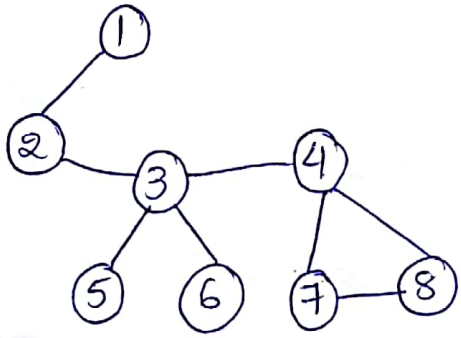
```
Algorithm BFS (V)
  //visited [] is initialized to zero
  {
    u := v;
   visited [v] := 1;
   repeat
   {
     for all vertices w adjacent from u do
  {
    if (visited [w] = 0) then
    {
     Add w to q;
     visited [w] := 1;
    } }
  if q is empty then return;
    Delete u from q;
  } until (false); }
```

* Algorithm BFT(G,n)

```
{
 for i:=1 to n do
   visited[i]:=0;
 for i:=1 to n do
   if (visited[i]=0)
   then BFS(i);
}
```

Example :-

i)



BFS :-

~~1~~	~~2~~	~~3~~	4	5	6

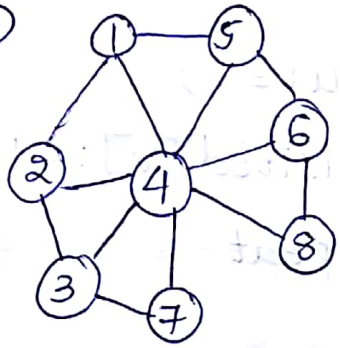
BFS(1) = 2

BFS(2) = 3

BFS(3) = 4, ~~5~~, 6, ~~2~~

BFS(5) = 5

ii)



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

BFS(1) = ~~2~~, 4, 5

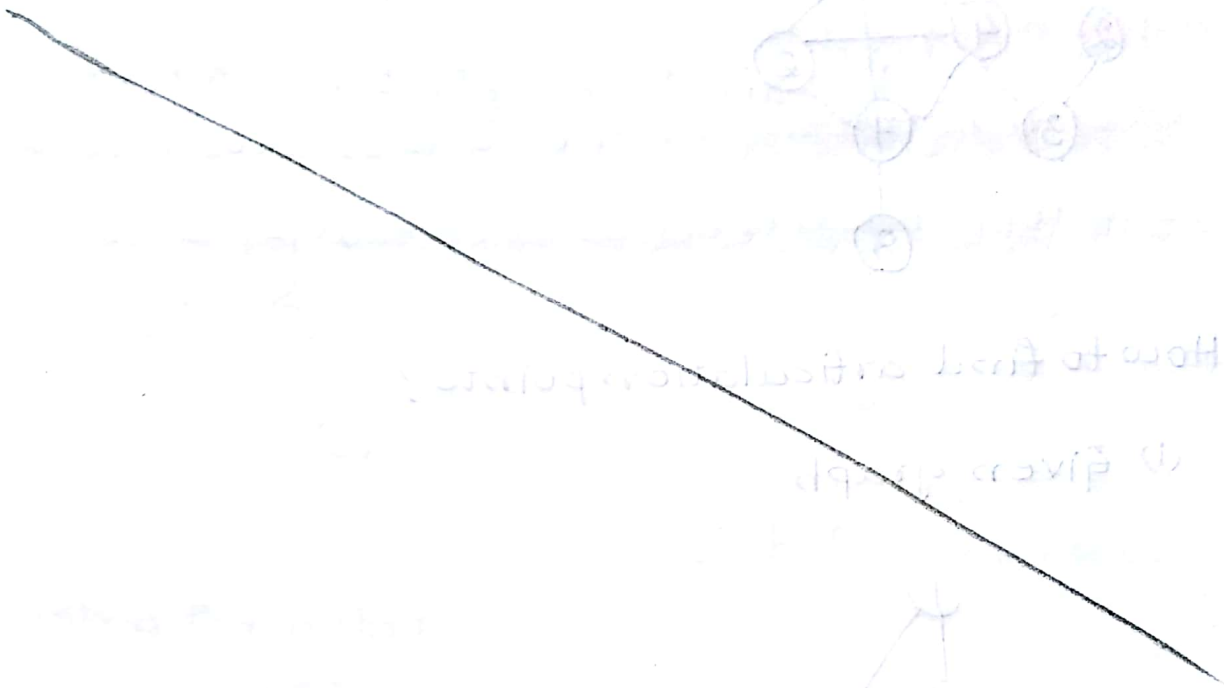
BFS(2) = ~~1~~, ~~3~~, 4

BFS(4) = ~~1~~, ~~2~~, 3, 4, 7, 8, 6, 5

BFS(5) = 1, 4, 6

BFS(3) = 2, 4, 7

BFS(4) = 3, 7, 8 (3 ✗, 7 ✓)



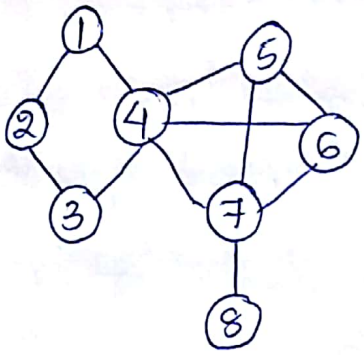
connected components :-

If G is a connective graph, then all vertices of G will get visited on 1st call to BFS. If G is not connected, then atleast two calls to BFS is needed.

Biconnected components :-

A vertex 'v' in a connected graph G is an articulation point iff deletion of vertex 'v' together with all edges incident to 'v' disconnects graph into two or more non-empty components.

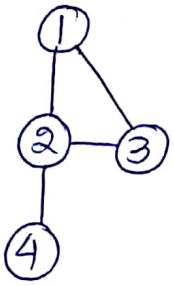
A graph G is biconnected iff it has no articulation points.



Articulation points = 4, 7

How to find articulation points?

(1) Given graph



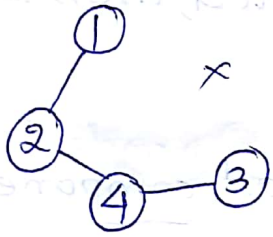
(2) construct depth first spanning tree.

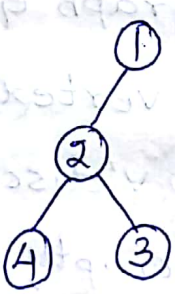
DFS(1) = 2, 3

DFS(2) = 1, 4, 3

DFS(4) = 2

DFS(3) = 2



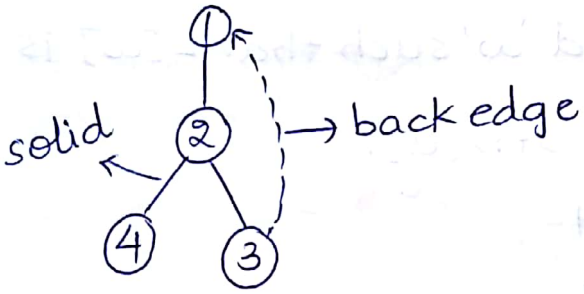


(3) Identify depth first numbers (DFN) corresponding to order in which dfs visits these numbers.

dfn =

1	2	4	3
1	2	3	4

(4) Identify solid edges and back edges.



(5) Now, identify lowest depth first numbers using formula:

$L[u] = \min\{dfn[u], \min\{L[w] / w \text{ is a child of } u\}, \min\{dfsn[w] / (u,w) \text{ is a backedge}\}\}$

$L[4] = \min\{dfn[4], -, -\} = \min\{3, -, -\} = 3$

$L(3) = \min\{dfn[3], -\min\{dfn[1]\}\} =$

$\min\{4, -, 1\} = 1$

$L(2) = \min\{dfn[2], \min\{L[4]\}\min\{L[3]\}, -\}$

$= \min\{2, \min\{3, 1\}, -\}$

$= \min\{2, 1, -\} = 1$

$L[1] = \min\{dfn[1], \min\{L[2]\}, -\}$

$= \min\{1, 1, -\} = 1$

(6) If u is a root then, u is articulation point iff it has 2 children.

ii) If u is not a root, then u is an articulation point iff u has a child 'w' such that $L[w]$ is greater or equals to $dfn[u]$.

	1	2	3	4
dfn	1	2	4	3
l	1	1	1	3

As u=1 is a root, it has only one child, so it is not articulation point.

u=2

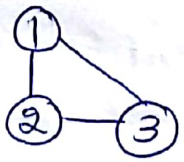
$w=4, L[4] \geq dfn[2]$

$\Rightarrow 3 \geq 2$ ✓

Atleast for one.

As 2 is an articulation point, therefore given graph is not a biconnected graph.

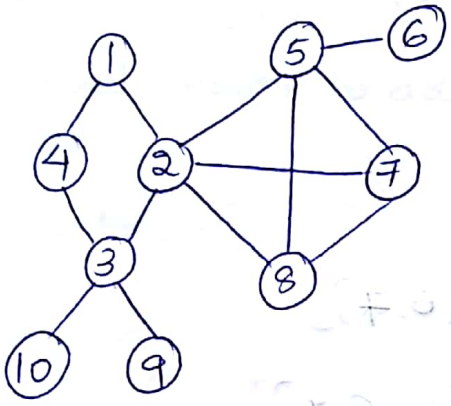
(7) Now identify bi-connected components.



{1, 2, 3}

{2, 4}

→ Identify bi-connected components for follo-wing :-



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

(1) Graph

(2) DFS(1) = 4, 2

DFS(4) = 1, 3

DFS(3) = 4, 2, 10, 9 (4, 10, 9, 2)

DFS(10) = 3

DFS(9) = 3

DFS(2) = 1, 5, 7, 8, 3

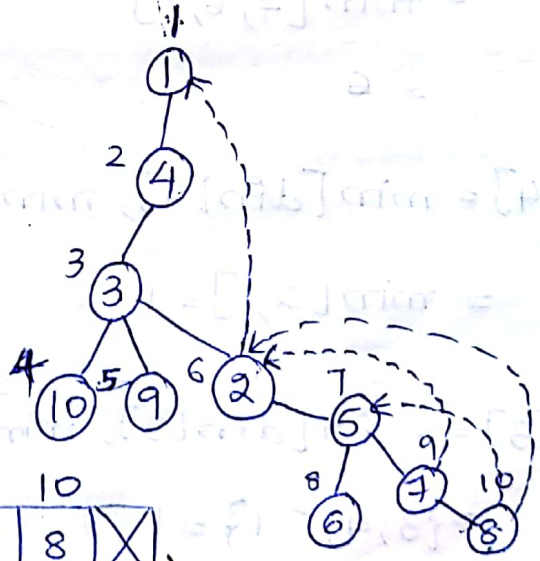
DFS(5) = 2, 6, 7, 8

DFS(6) = 5

DFS(7) = 5, 2, 8

DFS(8) = 2, 5, 7

1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	)	X



1	2	3	4	5	6	7	8	9	10
1	4	3	10	9	2	5	6	7	8	X

backedge :- 1→2, 2→7, 2→8, 5→8

5) $L[10] = \min[dfn\{10\}, -, -]$

$= 4$

$L[9] = \min[dfs\{9\}, -, -]$

$= 5$

$L[8] = \min[dfn\{8\}, \min\{6,7\}]$

~~$\min\{6,5,2\}$~~ $\min\{10,6,7\}$

$= 6$

$L[7] = \min[dfn\{7\}, L[8], \min\{dfn[2]\}]$

$= \min[9,6,6]$

$= 6$

$L[6] = \min[dfn[6], --]$

$= \min[8] = 8$

$L[5] = \min[dfn[5], \min\{L(6), L(7)\} -]$

$= \min[7,6,8]$

$= 6$

$L[4] = \min[dfn[4], \min[L[3]]$

$= \min[2,1] = 1$

$L[3] = \min[dfn[3], \min[L[10], L[9], L[2]], -]$

$= \{3,4,5,1\} = 1$

$L[2] = 1, \; L[1] = 1$

Algorithm Art(V, u) (V → parent, u → child (u, V))

```
{   num:=1;
 dfn[u]=num; L[u]:=num; num:=num+1;
 for each vertex w adjacent from u do
 {
  if(dfn[w]==0) then
  {
     Art(w,u);
    L[u]=min(L(u),L(w)); (solid edge)
  }
  else if(w≠V) then L[u]:=min(L(u), dfn(w)];
 }                          (back edge)
}        * back edge is adjacent vertex. *
```

	1	2	3	4	5	6	7	8	9	10
dfn	1	6	3	2	7	8	9	10	5	4
L	1	1	1	1	6	8	6	6	5	4

As u=1, is not an articulation point because, it has 1 child.

As u=4, child=3

$L[3] \geq dfn[4]$

$= 1 \geq 2$ ×

It is not articulation point.

As u=3, child-10,9,2

$L[10] \geq dfn[3]$

$4 \geq 3$ ✓

$L[9] \geq dfn[3]$

$5 \geq 3$ ✓

$L[2] \geq dfn[3] \Rightarrow 1 \geq 3$ ×

u=3 is an articulation point.

u=2, child=5

$L[5] \geq dfn[2]$

$6 \geq 6$ ✓

It is articulation point

∴ u=5, child=6, 7

$L[6] \geq dfn[5]$ $\quad$ $L[7] \geq dfn[5]$

$8 \geq 7$ ✓ $\quad$ $6 \geq 7$ ×

It is an articulation point.

u=7, child=8

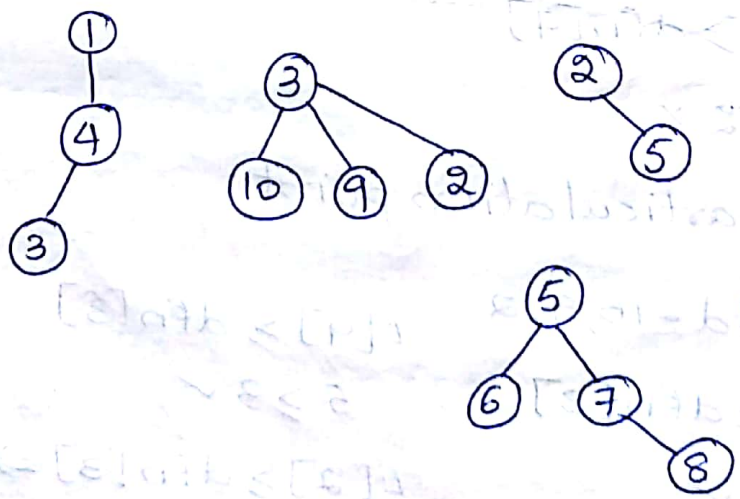
$L[8] \geq dfn[7]$

$6 \geq 9$ ×

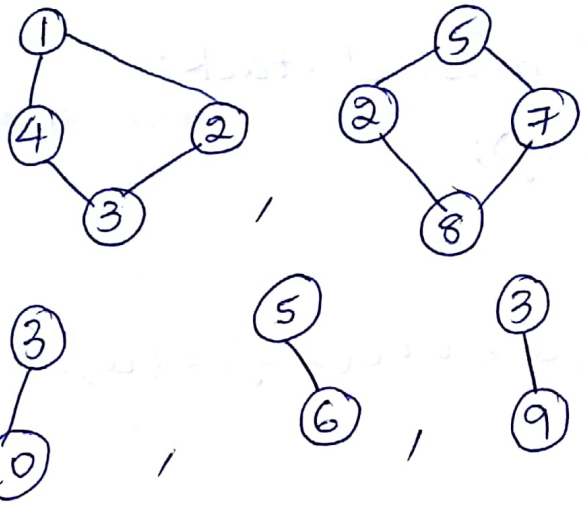
It is not articulation point.

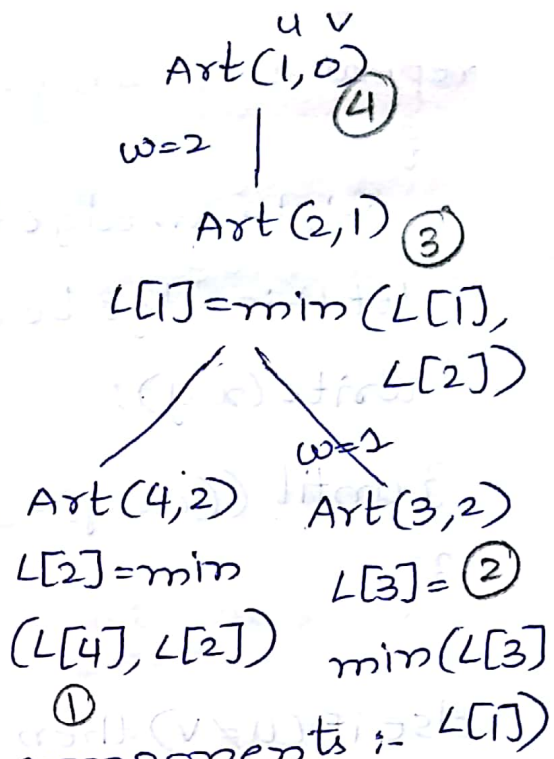
As, 2, 3, 5 are articulation points given point is not biconnected graph.

Bi-connected components :-



Bi-connected components :-





* Algorithm for bi-connected components :-

```
Algorithm Bicomp(u,v)
 {
  dfn[u]:=num; L[u]:=num; num:=num+1;
for each vertex w adjacent from u do
{
 if ((v≠w) and (dfn[w]<dfn(u))) then
      add (u,w) to top of stack s;
 if (dfn[w]==0) then
 {
   Bicomp(w,u);
 L[u]=min (L[u], L[w]);
 if [L[w]≥dfn[u]) then
 {
   write (" New Bicomp");
```

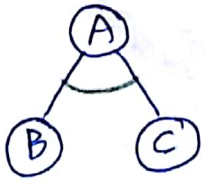
```
repeat
  {
   delete an edge from top of stack;
   let this edge be (x,y);
   write(x,y);
  } until (((x,y)=(u,w)) or ((x,y)=(w,u)));
 }
}
else if (u≠v) then
  L[u] = min[L[u], dfn[w]];
 }
}
```

Time complexity = O(n)+e

* AND-OR Graph :-

Breaking down of a complex problem into several sub-problems can be represented by a directed graph like structure in which nodes represent problems, descending of node represents sub-problems associated with it.

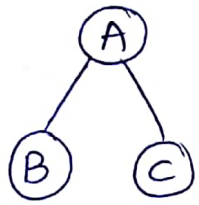
AND Graph



Eg :- Merge sort

problem A is solvable iff both B & C are solva-
-ble.

OR Graph



Eg:- Binary search

Problem A is solvable iff atleast one problem of B, C is solvable.

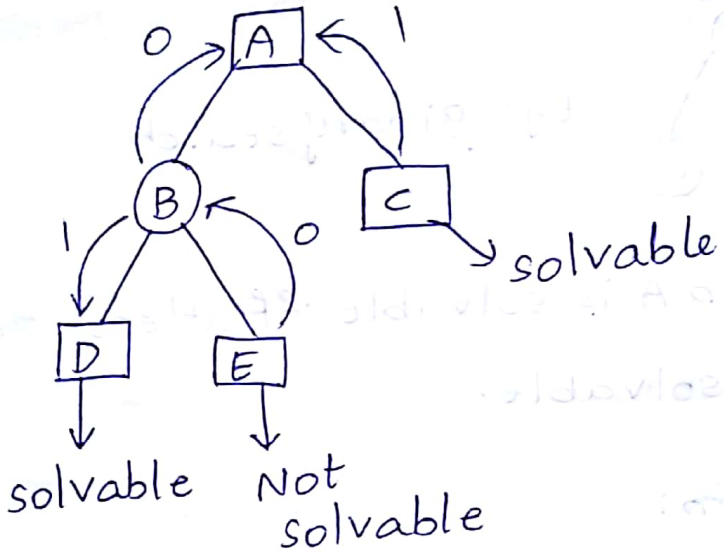
Algorithm:-

```
        procedure solve(T)
case:
    : T is a terminal node. If T is solvable
    then return(1) else return(0).
    : T is AND node. For each child s of T do
    if solve(s)=0 then
        return 0;
    end if
    repeat
    return(1)
else
    for each child s of T do
    if solve(s)=1 then return 1
    end if
    repeat
```

return(0);

end case

end solve



Back-tracking :-

In greedy and dynamic programming we use brute force approach. It means we'll evaluate all possible sol^{n}s among which we select one sol^{n} as optimal sol^{n}.

In back-tracking technique, we'll get same optimal sol^{n} with less no. of steps. In this, we will use bounding function (or) criterial function implicit, explicit constraints for possible sol^{n}s. Major advantage of back-tracking method is If a partial sol^{n} $x_1, x_2 \dots x_i$ can't lead to an optimal sol^{n}, then $x_{i+1} \dots x_n$ can be discarded.

Terminology :-

(1) criterial function :-

It is a function $P(x_1 \dots x_n)$ which needs to be maximized (or) minimized for a problem. (To find optimal soln)

Explicit constraints :-

These are rules that restricts each x_i to take values only from given set.

Ex :- In knapsack problem x_i lies b/w 0 & 1

Implicit constraints :-

These are rules which determine which of pupils in solution space satisfy criterial function.

Ex :- In knapsack problem $m=30$ is an implicit function.

(4) state space tree :-

If we represent soln space in format of tree, it is called as state space tree.

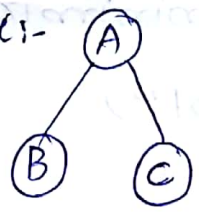
(5) Live node :- A node which have been generated & whose children are not yet generated.

E-node :- Live node whose children are currently being generated.

Deadnode :-

It is generated node that cannot gene-rate children or expend furthur.

Ex:-



(6) Bounding function :-

It is a function that is used to kill a livenode without generating all its children.

control abstraction of general method of back-tracking :-

```
Algorithm Backtract (k)
 {
  for (each x[k] ∈ T (x[1] ---- x(k-1)) do
 {
  if (Bk (x[1], x[2] ---- x[k] ≠ 0) then
  {
    if (x[1], x[2] --- x[k] is a path to an
                answer node)
   then write (x[1:k]);
  if (k<n) then Backtract (k+1);
 }
}
}
```

Applications :-

(1) sum of sub-sets problem :-

Suppose we are given 'n' distinct +ve integ-ers (weights) & we desire to find all possible combinations of these nos whose sum = m, this is called as sum of subsets problem. For a node at level i, left child corresponds to $x_i = 1$ right child corresponding to $x_i = 0$.

The bounding functions are :-

$x_i = 1$

$s + w[k] + w[k+1] \leq m$

$x_i = 0$

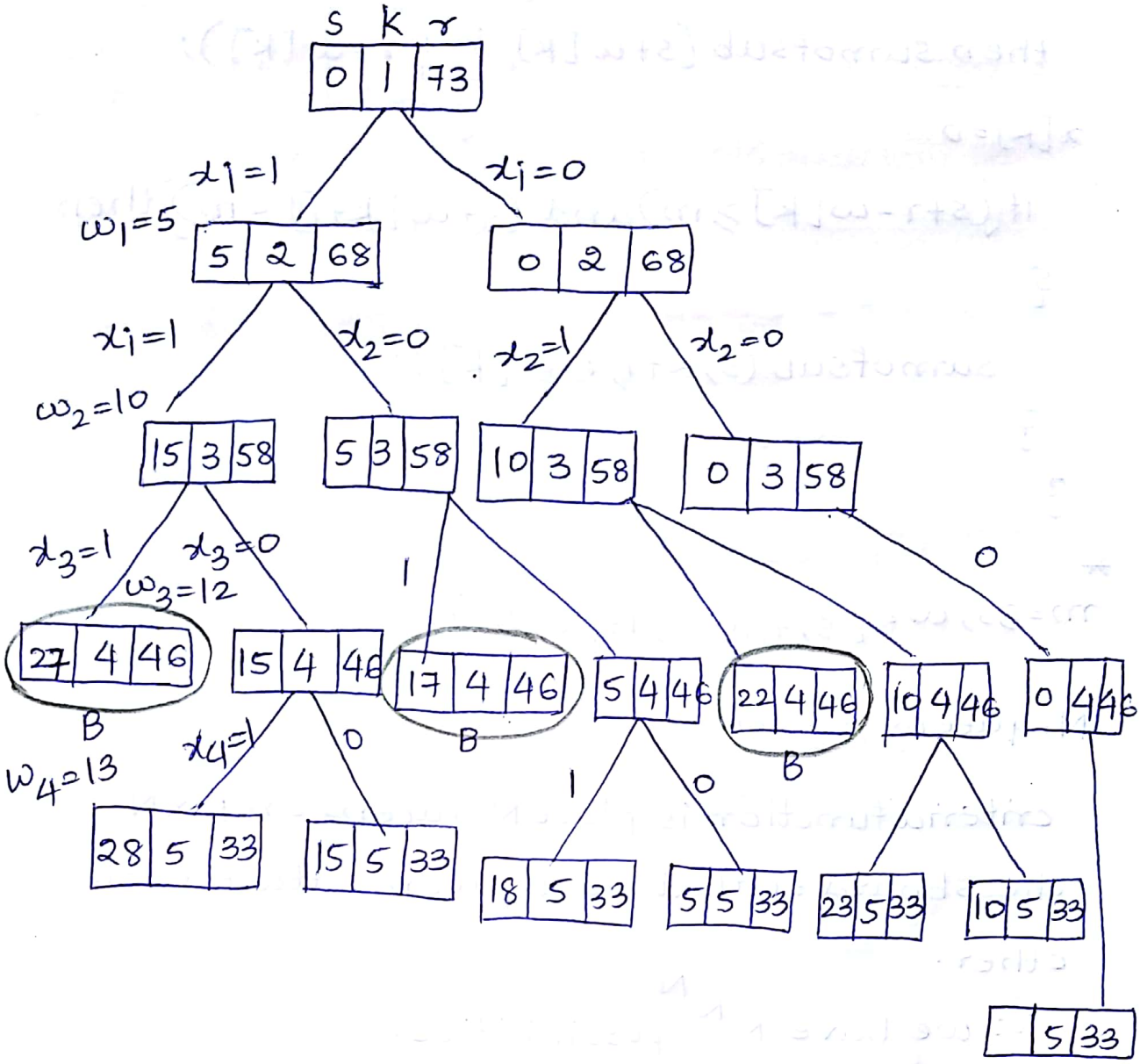
$s + r - w[k] \geq m$ and $s + w[k+1] \leq m$

Ex :- $n = 6, w = m = 30$

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

The no. of possible nodes in state-space tree are $2^n - 1$.

Time-complexity = $O(2^n)$.



The possible soln^s are $x_1, x_2, x_3, x_4, x_5, x_6$ =

(1,1,0,0,1,0), (1,0,1,1,0,0), (0,0,1,0,0,1)

Algorithm sumofsub (s,k,r)

{

x[k] = 1;

if (s+w[k] = m) then write (x[1:k]);

elseif (s+w[k] +w[k+1] ≤ m)

then sumofsub (s+w[k], k+1, r-w[k]);

x[k]=0

if((s+r-w[k] ≥ m) and (s+w[k+1] ≤ m)) then

{

sumofsub (s, k+1, r-w[k]);

}

}

*

m=35, w={5,7,10,12,15,18,20}

N-queens problem:-

criteria function is place N queens on NXN chessboard so that no 2 queens attack each other.

∴ we have N^N possibilities.

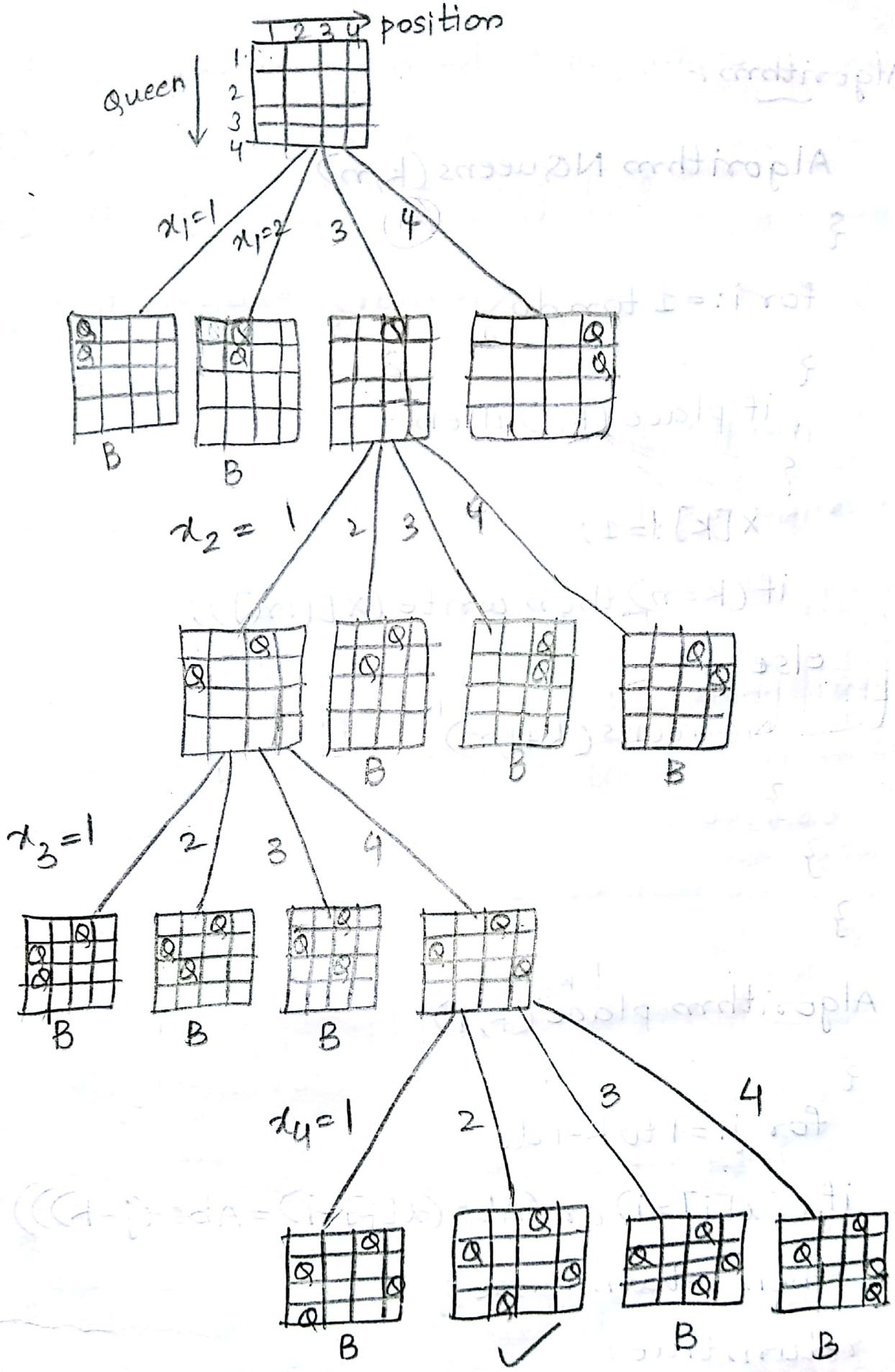
Explicit:-

I^{th} queen has to be placed only in I^{th} row.

possibilities are N^N.

Implicit:-

No two queens must be in column & same diagonal. Therefore, no. of possibilities are reduced to N!.



possible path (or) sol^n is 3142

Algorithm :-

```
Algorithm NQueens(k,n)
{
 for i:=1 to n do
 {
  if place(k,i) then
  {
   x[k]:=1;
  if(k=n) then write(x[1:n]);
 else
  NQueens(k+1,n);
  }
 }
}
Algorithm place(k,i)
{
 for j:=1 to k-1 do
if((x[j]=i) or (Abs(x[j]-i)=Abs(j-k)))
 then return false;
return true;
}
```

Time complexity = O(n!)

suppose 2 queens are placed at positions (i,j) & (k,l), then 2 queens are said to be in same position, when i-j=k-l & i+j=k+l.

```
Algorithm NQueen(row,n)
{
 for col:=1 to n do
 {
   if Place(row,col) then
   { row  col
     x[k]:=n;
     if (row==n) then
       write(x[1:n]);
     else NQueen(row+1,n);
   }
 }
}
```

i j k l
(i, x[i]), (row, col)
↓
position of present queen

i = previous queens
x[i] = position of previous queen

i+j = k+l (or)
i-j = k-l

i-k = l-j
i-k = j-l
i-k = -(j-l)

Abs(i-k) = Abs(j-l)

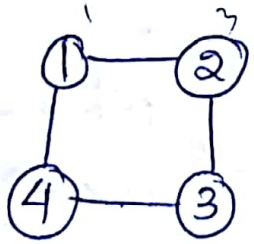
```
Algorithm place(row,col)
{
 for i:=1 to row-1 do
 if (x[i]=col or (Abs(x[i]-col)=(Abs(i-row))
 then return false;
 return true;
}
```

Graph colouring :-

Let G be a graph and 'M' be a given +ve integer, we want to discover whether nodes of G can be coloured in such a way that no 2 adjacent nodes have same colour, yet we use 'M' colours. This is called 'M' colorability decision problem. If 'D' is degree of graph, it can be coloured with (D+1) colours.

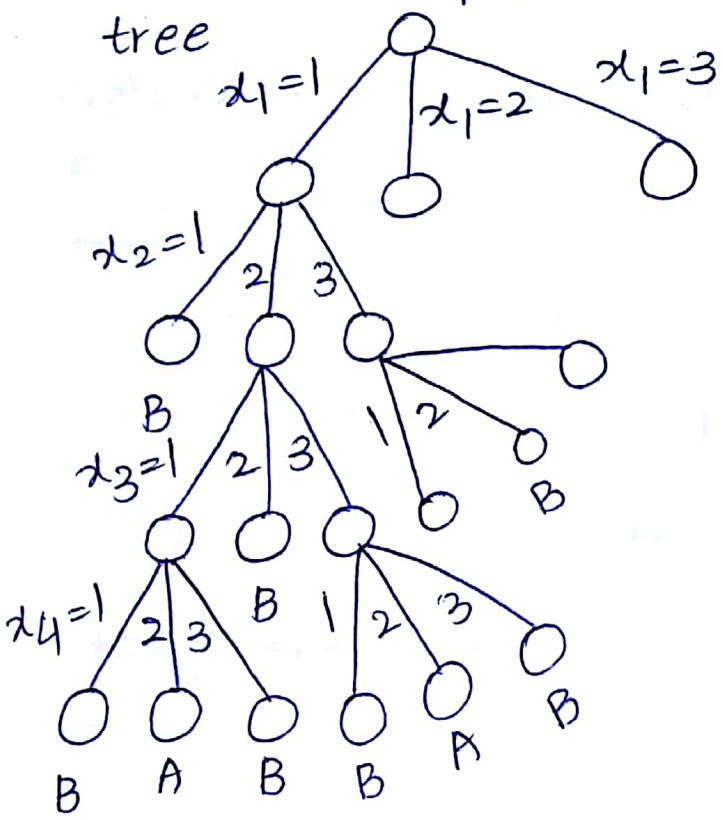
This 'M' colarability optimization problem asks for smallest integer 'M' for which graph can be colored. This integer is referred to as chromatic number of graph.

Degree = number of edges



2+1=3 colorability decision problem

solution space tree



(1,2,1,2)

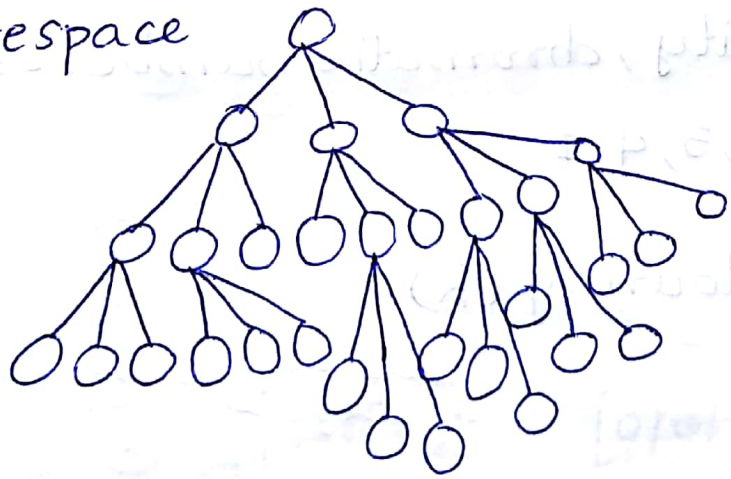
(1,2,3,2)

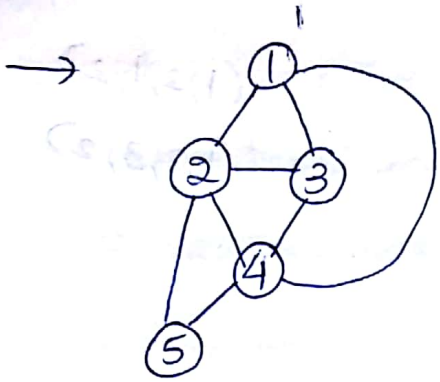
Therefore possible soln^s are (1, 2, 1, 2), (1, 2, 3, 2)

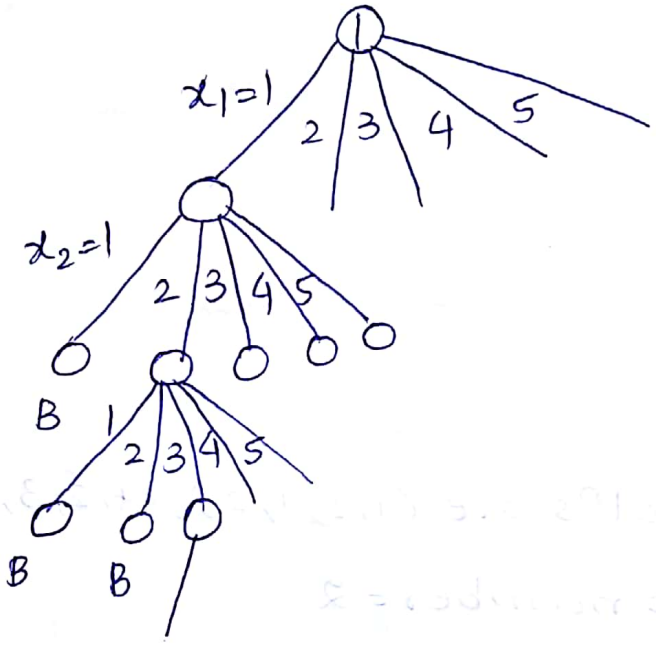
Minimum chromatic number = 2

→ Draw state space tree for n=3, m=3.

statespace







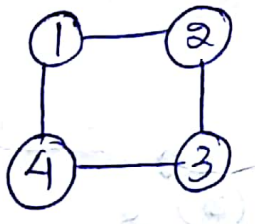
4+1=5 colorability, chromatic number = 3

1, 2, 3, 4, 1

Algorithm mcolouring (k)

```
{
repeat
 {   Nextvalue(k);
  if (x[k]=0) then return;
 if (k=n) then write (x[1:n]);
 else mcolouring (k+1);
} until (false);
```

```
}
Algorithm Nextvalue(k)                    [k=node]
{
repeat
 {
  x[k] := (x[k]+1) mod (m+1);
if (x[k]=0) then return
 for j:=1 to n do
{
  if((G[k,j]≠0 and (x[k]=x[j]))
     then break;
 }
 if (j=n+1) then return;
} until false;
}
```



1	2	3	4
1	2	1	2

1	2	3	4
0	0	0	0

The optimal soln is 1, 2, 1, 2.

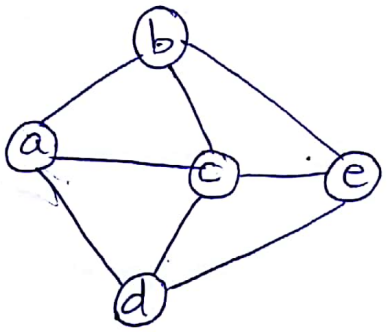
Time complexity is $O(nm^n)$

~~control~~

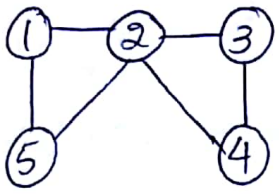
Hamiltonian cycle :-

let $G = (V, E)$ be a connected graph with N vertices. A Hamiltonian cycle is a round trek path along N edges of G that visits every vertex once & returns to its starting position.

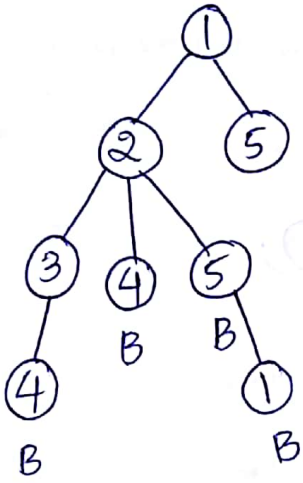
In other words, if a Hamiltonian cycle begin at vertex v_1 and vertices of G are visited in order $v_1, v_2 \ldots v_{n+1}$ then edges (vertices) from v_1 to v_n are distinct & reached atmost once & except $v_1 = v_{n+1}$.

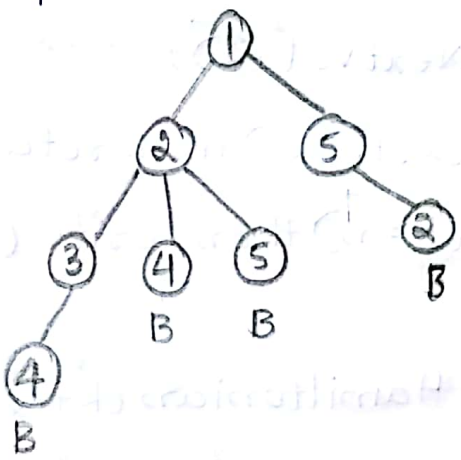


1 2 3 5 4
1 2 5 4 3
1 3 2 5 4
1 4 3 5 2
1 4 5 2 3



No solution possible






```
Algorithm Nextval(k)
{
repeat{
    x[k]=(x[k]+1) mod(n+1);
  if (x[k]=0) then return;
  if (G(x[k-1],x[k]≠0) then
  {
     for j:=1 to k-1 do if (x[j]=x[k]); then
      break;
   if (j=k) then
 if ((k<n) or (k=n) and (G[x[n],x[1]]≠0))
  then return;
 }
} until (false);
```

```
}
Algorithm Hamiltonian (k)
 {
 repeat {
      Nextval (k);
   if (x[k]=0) then return;
   if (k=n) then write (x[1:n]);
  else
      Hamiltonian (k+1);
  } until (false);
 }
```

Time-complexity = $O(n^2 2^n)$

18/18

UNIT-III

GREEDY APPROACH

Divide and conquer approach is applicable only for applications which are divisible.

Greedy method is a straight-forward design technique (Root-force approach) which has problems with 'N' i/ps and required us to obta-in subsets that has to satisfy some constrain-ts.

Any subsets that satisfy these constraints is feasible solution. We need to find a feasible soln that maximizes (or) minimizes given an objective function.

A feasible soln that does this is called optimal soln.

control abstraction:-

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

Applications :-

(1) knapsack problem :-

We are given 'N' objects and knapsack or bag, object i has a weight w_i and knapsack has capacity 'm' and objective is to obtain a filling of knapsack that maximizes total capacity 'M' stated as :-

$$\text{maximize} \sum_{1 \le i \le n} p_i x_i$$

$$\text{subject to} \sum_{1 \le i \le n} w_i x_i \le m$$

$$\text{and } 0 \le x_i \le 1, \quad 1 \le i \le n$$

Q) $N=3, m=20, w=\{18, 15, 10\}, (P_1, P_2, P_3) = (25, 24, 15)$.

$(x_1, x_2, x_3) = \left(\frac{1}{2}, \frac{1}{3}, \frac{1}{4}\right)$

feasible sol^n $= \sum w_i x_i \le m$

$$\Rightarrow \frac{1}{2} \times 18 + \frac{1}{3} \times 15 + \frac{1}{4} \times 10$$

$$= 16.25 \le 20 \checkmark$$

$$\text{Max}(p_i x_i \Sigma) = \frac{1}{2} \times 25 + \frac{1}{3} \times 24 + \frac{1}{4} \times 15$$

$$= 24.25$$

(2)

Increasing weights = $(1, 2/15, 0)$

$\Rightarrow 1\times18 + 2/15 \times 15 + 0\times10 = 20$ ✓

$\max(\sum P_i x_i) = 1\times25 + \frac{2}{15}\times24 + 0\times15$

$= 28.2$

3)

Decreasing

Profits = $(0, 2/3, 1)$

$\Rightarrow 0\times18 + \frac{10}{15}\times15 + 1\times10 = 20$

$\sum P_i x_i \Rightarrow 0\times25 + 2/3\times24 + 1\times15 = 31$

4)

Increasing (P_i/w_i)

$\Rightarrow (1.4, 1.6, 1.5) \Rightarrow (0, 1, 1/2)$

$w_i x_i = 0\times18 + 1\times15 + \frac{1}{2}\times10$

$= 20$

$\max(\sum P_i x_i) = 0\times25 + 1\times24 + \frac{1}{2}\times15$

$= 31.5$

As feasible soln $(0, 1, \frac{1}{2})$ is leading to objecti-ve function, max (profit) = 31.5 is optimal soln

8) $N=4, m=15, w=\{2,4,6,9\}, P=\{10,10,12,18\}$

$(x_1, x_2, x_3, x_4) = (\frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5})$

feasible soln = $\sum w_i x_i \leq m$

$= \frac{1}{2} \times 2 + \frac{1}{3} \times 4 + \frac{1}{4} \times 6 + \frac{1}{5} \times 9$

$= 5.63 \leq 15$

$\leq (p_i x_i) = \frac{1}{2} \times 10 + \frac{1}{3} \times 10 + \frac{1}{4} \times 12 + \frac{1}{5} \times 18$

$= 14.93$

Increasing weights $= (0, 0, 1, 1)$

$w_i x_i = 0 \times 2 + 0 \times 4 + 6 \times 1 + 1 \times 9$

$= 15$

$\max (\leq p_i x_i) = 0 \times 10 + 0 \times 10 + 1 \times 12 + 1 \times 18$

$= 30$

Decreasing profits $= (1, 1, 1\frac{3}{9})$

$\Rightarrow (1 \times 2 + 1 \times 4 + 0 \times 6 + \frac{3}{9} \times 9)$

$= 15$

$(\leq p_i x_i) = 1 \times 10 + 1 \times 10 + 1 \times 12 + \frac{3}{9} \times 18$

$= 38$

Increasing (p_i / w_i)

$\Rightarrow (5, 2.5, 2, 2) = (1, 0, 0)$

~~$1 \times 5 \times$~~ $\Rightarrow 1 \times 2 +$

9) N=7, m=15, P=(10,5,15,7,6,18,3), ω=(2,3,5,7, 1,4,1)

Increasing weights = (0,0,1,1,0,3/4,0)

$7\times1+5\times1+4\times\frac{3}{4}+0+0+0+0$

$\max(\sum P_i x_i) = 0\times10+0\times5+1\times15+7\times1+0\times6+$

$0\times3+\frac{3}{4}\times18$

$= 35.5$

Decreasing profits = (1,1,1/5,1,1,0,1)

$2\times1+3\times1+5\times\frac{1}{5}+7\times1+1\times1+4\times0+1\times1$

$\sum(P_i x_i) = 10\times1+5\times1+15\times\frac{1}{5}+7\times1+6\times1+3\times1$

$= 34$

Increasing (P_i/ω_i) ~~= (5,1, , , , , ,)~~

$= (1, 2/3, 1, 0, 1, 1, 1)$

$\sum(P_i x_i) = 10+3.3+15+6+18+3 = 55.3$

∴ optimal solution is $(1, 2/3, 1, 0, 1, 1, 1)$ with optimal profit = 55.3.

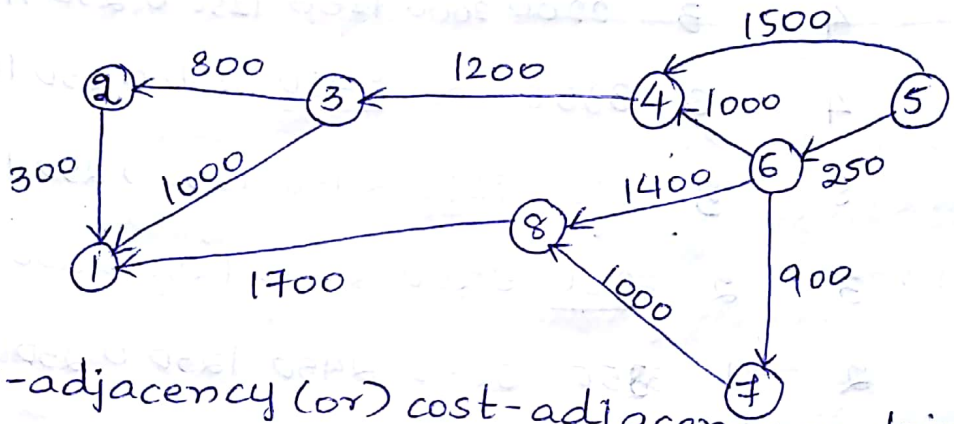
```
Algorithm Greedyknapsack (m,n)
{
 for i:=1 to n do x[i]:=0.0;
   u:=m;
 for i:=1 to n do
 {
   if (w[i] > u) then break;
   x[i]=1.0; u=u-w[i];
 }
if (i≤n) then x[i]:=u/w[i];
}
```

Time-complexity for all feasible soln $= O(2^n)$.

Time-complexity for one soln $= O(n)$.

single-source shortest path (Dijkstra's algorithm)

In this problem, we are given directed graph $G(V, E)$, a waiting function cost for graph G, for edges of G, a source V_0, problem is to determine shortest path from V_0 to all remaining vertices of G.



length-adjacency (or) cost-adjacency matrix :-

	1	2	3	4	5	6	7	8
1	0	∞	∞	∞	∞	∞	∞	∞
2	300	0	∞	∞	∞	∞	∞	∞
3	1000	800	0	∞	∞	∞	∞	∞
4	∞	∞	1200	0	∞	∞	∞	∞
5	∞	∞	∞	1500	0	250	∞	∞
6	∞	∞	∞	1000	∞	0	900	1400
7	∞	∞	∞	∞	∞	∞	0	1000
8	1700	∞	∞	∞	∞	∞	∞	0

The shortest distances from 5 are $5 \to 6 = 250$

$5 \to 7 = 1150$

$5 \to 4 = 1250$ ~~1250~~ 1500

$5 \to 8 = 1650$ (1650)

$5 \to 3 = 2450$ ~~2450~~ 2700 (3500 ~~3700~~ 2550)

$5 \to 2 = 3250$ ~~3250~~, $5 \to 1 = 3950$ ~~3950~~ 3700

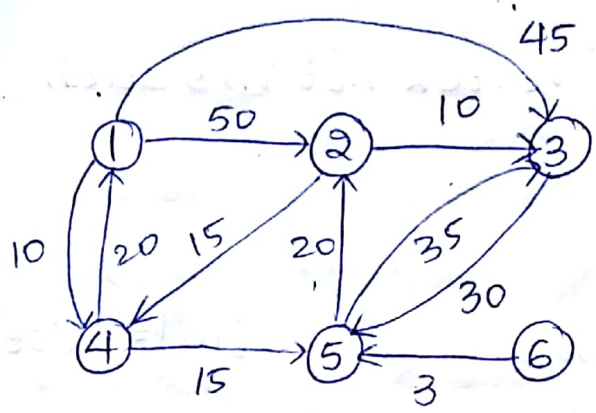
Iteration	s	vertex selected	Distance 1	2	3	4	5	6	7	8
1	–	5 (u)	∞	∞	∞	1500	0	250	∞	∞
2	5	6	∞	∞	∞	1250	0	250	1150	1650
3	{5 6}	7	∞	∞	∞	1250	0	250	1150	1650
4	7	4	∞	∞	2450	1250	0	250	1150	1650
~~5~~	~~4~~	~~3~~	~~2200~~	~~2000~~	~~1200~~	~~1250~~	~~0~~	~~250~~	~~1150~~	~~1650~~
5	4	8	3350	∞	2450	1250	0	250	1150	1650
6	8	3	3350	3250	2450	1250	0	250	1150	1650
7	3	2	3350	3250	2450	1250	0	250	1150	1650
8	2	1	3350	3250	2450	1250	0	250	1150	1650

~~Iteration s vertex selected~~ (u)

+3250

(2)

	1	2	3	4	5	6
1	0	50	45	10	∞	∞
2	∞	0	10	15	∞	∞
3	∞	∞	0	∞	30	∞
4	20	∞	∞	0	15	∞
5	∞	20	35	∞	0	∞
6	∞	∞	∞	∞	3	0



Iteration s vertex selected 1 2 3 4 5 6

1	–	6	(u)	∞	∞	∞	∞	3	0
2	6	5		∞	23	38	∞	3	0
3	{6,5}	2		~~∞~~ ∞	23	33	38	3	0
1450 > 800	4	{6,5,2}		3	∞	23	33	38	3	0
2450 800 3250	5	3		4	58	23	33	38	3	0
	6	4		1	58	23	33	38	3	0

3350 > 3000 + 3250
= 3250
300
3550

shoretest distance from 6 to 5 3
6 to 2 23
6 to 3 33
6 to 4 38
6 to 1 58

```
Algorithm shortpath (v, cost, dist, n)
{
 for i:=1 to n do
 {
   s[i] = false, dist[i] = cost[v,i];
 }   v                v
   s[u]:=true; dist[u]:=0.0;
 for num:=2 to n-1 do
 {
```

```
choose u from among vertex not in s such
    that dist[u] is min;
  s[u] := true;
  for (each w adjacent to u with s[w] = false)
   if (dist[w] > dist[u] + cost[u,w]) then
    dist[w] = dist[u] + cost[u,w];
 }
}
```

Time-complexity = $O((|E|+n)\log n)$

Job-sequencing with deadlines :-

We are given a set of n jobs. Associated with job i is an integer deadline $d_i \geq 0$ and profit $p_i > 0$. For any job i profit p_i is earned iff job is comple--ted by its deadline.

A feasible soln for this problem is a subset J of jobs such that each job in this subset can be completed by its deadline. The value of feasible soln J is sum of profits in J.

$$\sum_{i \in J} p_i$$

An optimal soln is a feasible soln with maximum value.

let $N=4$, $P=\{100,10,15,27\}$, Deadlines $d=\{2,1,2,1\}$

1. Arrange given profits in descending order.

$P_1, P_4, P_3, P_2 = \{100, 27, 15, 10\}$

$d_1, d_4, d_3, d_2 = \{2, 1, 2, 1\}$

since, maximum deadline is 2, feasible soln is can have maximum (or) ≤ 2 jobs.

Iteration	feasible soln	processing sequence	value $\sum_{i \in J} P_i$
1	{1,4}	4,1	127 = 100+27
2	{1,3}	1,3 (or) 3,1	100+15 = 115
3	{1,2}	2,1	110
4	{4,3}	4,3	42
5	{4,2}	4,2 (or) 2,4	37
6	{3,2}	2,3	25
7	{1}	1	100
8	{2}	2	10
9	{3}	3	15
10	{4}	4	27

let N=5, P={20, 15, 10, 5, 1} (1 2 3 4 5) d={2, 2, 1, 3, 3} (1 2 3 4 5)

Iteration	feasible soln	sequence	$\sum_{i \in J} P_i$
1	{1, 2, 3}	3, 2, 1 (or)	20+15+10=45
2	{1, 3, 4}	3, 1, 2	
3	{1, 4, 5}	3, 1, 4	35=10+20+5
		1, 4, 5 (or) 1, 5, 4	
4	{2, 3, 4}	3, 2, 4	
5	{2, 3, 5}	3, 2, 5	
6	{3, 4, 5}	3, 4, 5 (or)	
7	{1, 2} 4}	3, 5, 4	
		1, 2, 4 (or)	
8	{1, 2, 5}	2, 1, 4	
		1, 2, 5 (or)	
9	{2, 4, 5}	2, 1, 5	

N=5, P={1, 5, 20, 15, 10} d={1, 2, 4, 1, 3}

P={20, 15, 10, 5, 1} (3 4 5 2 1)

d={4, 1, 3, 2, 1} (3 4 5 2 1)

Iteration	feasible soln	sequence	$\sum_{i \in J} P_i$
1	{3, 4, 5, 2}	4, 2, 5, 3	50

2. {3, 4, 5, 1}

3. {3, 5, 4, 2}

4. {3, 4, 2, 5}

5. {3, 4, 1, 2}

6. {3, 4, 1, 5}

7. {3, 4, 2, 1}

8. {3, 1, 4, 2}

!

```
Algorithm JS(d, j, n)
{
d[0] := J[0] = 0;
J[1] := 1; k := 1;
for i := 2 to n do {
    r := k;
while ((d[J[r]] > d[i]) and (d[J[r]] ≠ r))
    r := r-1;
if (d[J[r]] < d[i]) and (d[i] > r)) then
{
  for q := k to (r+1) step -1 do J[q+1] := J[q]
    J[r+1] := i; k = k+1;
 }
return k;
}}
```

Time-complexity = $O(n^2)$

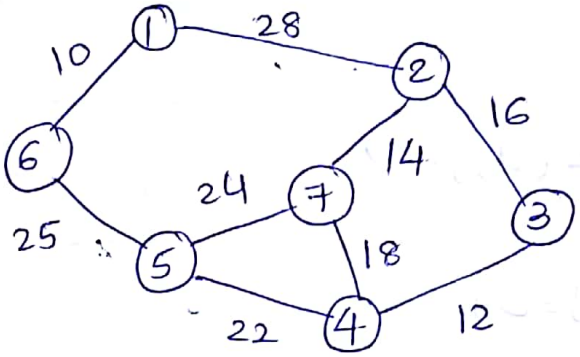
↙ ↘

n profits n deadlines

Minimum cost spanning tree :-

let $G = \{V, E\}$ be an undirected connected graph. A subgraph $t = (V, E')$ of G is spanning tree of G iff it is a tree with n-1 edges.

Prim's algorithm :-

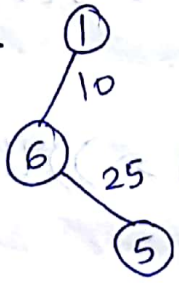


7 nodes

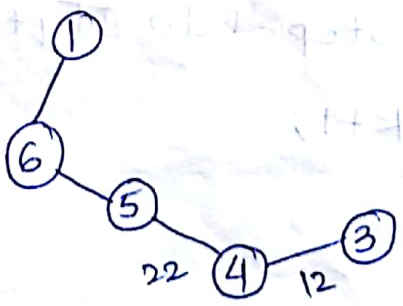
$2^7 - 1$ spanning trees

step-1 :-

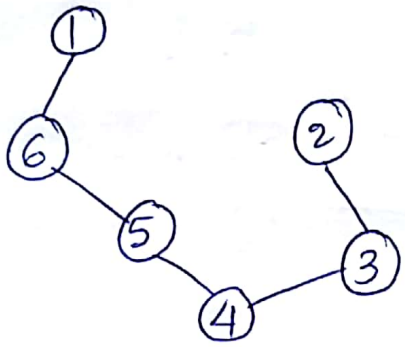
step-2 :-



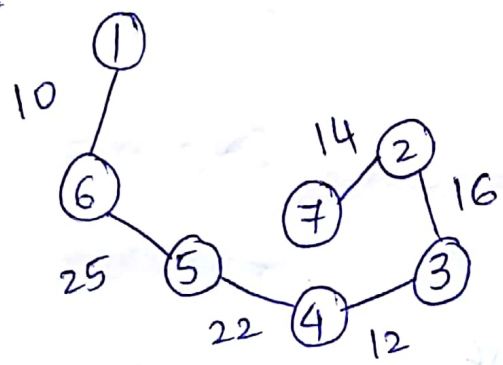
step-3 :-



step-4 :-

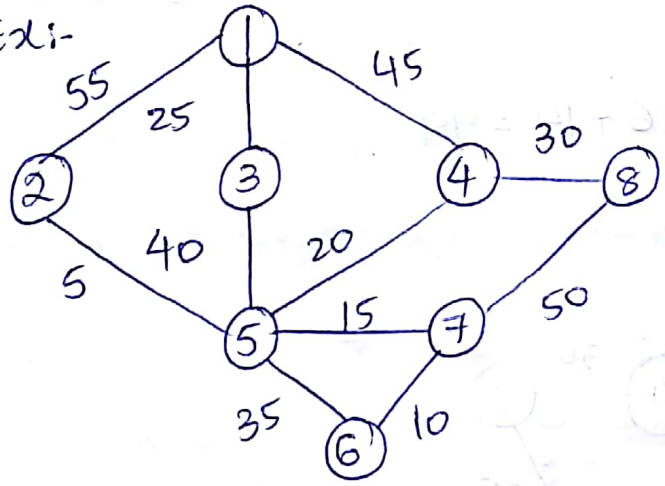


step-5 :-

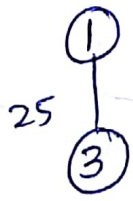


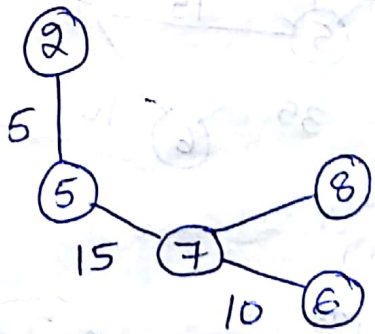
Minimum spanning tree is constructed with a cost of 99.

Ex:-



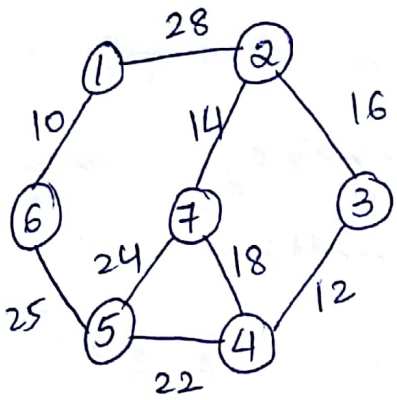
step-1 :-



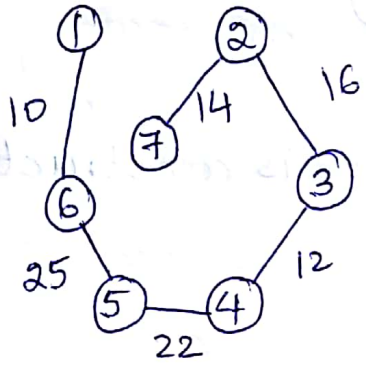


step-1 :-

kruskal's algorithm :-

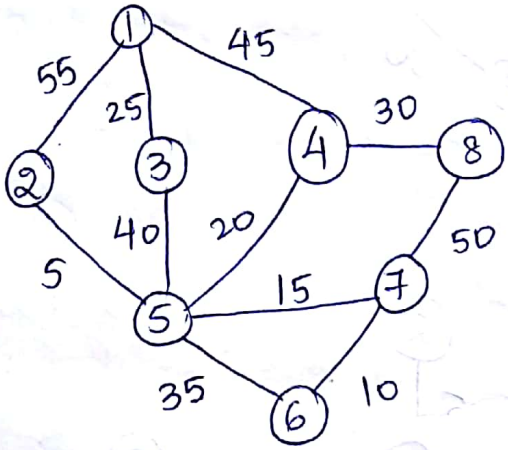


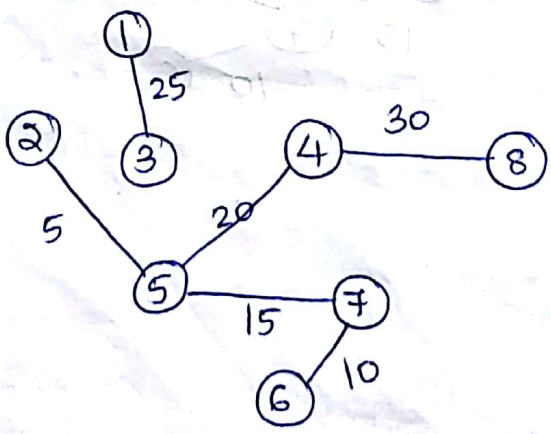
step-1 :-



$10+25+22+12+16+14=99$

Ex :-





Prim's algorithm :-

```
Algorithm Prim (E, cost, n, t)
{
 let (k, l) be an edge of mincost in E;
   mincost := cost [k, l];
 t[1, 1] := k; t[1, 2] := l;
 for i := 1 to n do
 {
  if (cost [i, l] < cost [i, k]) then
        near [i] := l;
   else
        near [i] := k;
  }
  near [k] := near [l] := 0;
 for i := 2 to n-1 do
 {
  let j be an index ∋ near [j] ≠ 0 and
          cost [j, near [j]] is minimum;
 t[i, 1] := j; t [i, 2] = near [j];
 mincost := mincost + cost([j] near [j]);
  near [j] := 0;
 for k := 1 to n do
   if ((near [k] ≠ 0) and (cost [k, neal [k]] >
       then neal [k] = j;  cost [k, j]))
```

return mincost;
}
}
}

$O(n^2)$

Kruskal's algorithm :-

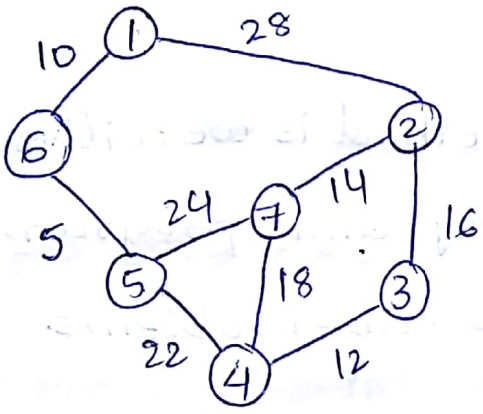
```
Algorithm kruskal (E, cost, n, t)
  {
construct a minimum heap out of edges costs using
   Heapify ;
   for i:=0, mincost:=0.0;
 while ((i<n-1) and heapnotempty) do
   {
  Delete minimum cost edge (u,v) from
heap and reheapify using Adjust,
  j:=Find(u), k:=Find(v);
  if (j≠k) then
   u  v
  {
   i:=i+1;
 t[i,1]:=u, t[i,2]:=v;
 mincost:=mincost+cost[u,v];
   Union(u,v);
 }
}
 if (i≠n-1) then write ("No spanning tree");
```

else return mincost;

}

∴ Time-complexity = O(n+E)(logn)

i	1	2
1	1	6
2	3	4
3	2	7
4	2	3
5		



11/9/18

UNIT-4 DYNAMIC PROGRAM

Dynamic programming is algorithm design technique that can be used when solution to a problem may be viewed as result of sequence of decisions.

Draw back of greedy method is we will make1 decision at a time. In dynamic programming sub-problems share sub-sub-problems.

General method :-

Fundamental dynamic programming method can be written as :-

$$F_n(R) = \max \{P_n(R_n) + P_{n-1}(R-R_n)\}$$
$$F_n(0) = 0 \qquad 0 \le R_n \le R$$

Principle of optimality :-

The principle of optimality states that an optimal sequence of decisions has property that whatever initial state & decision are, the remaining decisions must constitute optimal decision sequence with regards to state resulting from first decision.

0/1 knapsack problem :-

The principle of optimality holds $f_n(m) =$

$$\max\{\underbrace{f_{n-1}(m)}_{0}, \underbrace{f_{n-1}(m-w_n)+P_n}_{1}\}$$

$$s_1^i = \{(P,w) \mid \{(P-P_i, w-w_i) \in s^i\}$$

s^{i+1} can be computed by merging pairs s^i, s_1^i

$$s^{i+1} = s^i + s_1^i$$

If there are 2 pairs (P_j, w_j) and (P_k, w_k) with property that $P_j \le P_k, w_j \ge w_k$ in s^{i+1} then, pair (P_j, w_j) can be discarded which is called as dominance rule (or) discarding rule (or) purging rule.

Ex: $N=3$, $\{w_1, w_2, w_3\} = \{2,3,4\}$, $\{P_1, P_2, P_3\} = \{1,2,5\}$, $m=6$.

$$s^0 = \{(0,0)\} \quad s_1^0 = \{(0+1, 0+2)\}$$

$$= \underbrace{\{1,2\}}_{P_1\ w_1}$$

$$s^1 = s^0 + s_1^0 = \{(0,0), \underline{(1,2)}\} \quad , s_1^1 = \{(2,3), (3,5)\}$$

$$s^2 = s^1 + s_1^1 = \{(0,0), \underline{(1,2)}, (2,3), (3,5)\}$$

$$s_1^2 = \{(0+5, 0+4), (6,6), \underbrace{(7,7), (8,9)}_{\text{discard}}\}$$

According to weighting rule, s_1^i more weight than $m=6$ should be discarded.

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

$S^3 = S^2 + S_1^2 = \{(0,0), (1,2), (2,3), (3,5), (5,4), \underline{(6,6)}\}$

$(x_1, x_2, x_3) = (1,0,1)$

P=

The optimal soln is (1, 0, 1) with $(p, w) = (6, 6)$.

→ N=4, P={10, 10, 12, 18}, w={2, 4, 6, 9} m=15.

$S^0 = \{0,0\}$ $S_1^0 = \{(10,2)\}$

$S^1 = S^0 + S_1^0 = \{(0,0), \underline{(10,2)}\}$
+
(10,4)

$S_1^1 = \{(10,4), \underline{(20,6)}\}$

$S^2 = \{(0,0), (10,2), \underline{(10,4)}, (20,6)\}$

discard

$S^2 = \{(0,0), (10,2), \underline{(20,6)}\}$

$S_1^2 = \{(12,6), (22,8), (32,12)\}$

$S^3 = \{(0,0), (10,2), (20,6), \underline{(12,6)}, (22,8), (32,12)\}$

discard

$S^3 = \{(0,0), (10,2), \underline{(20,6)}, \underline{(12,6)}, (22,8), (32,12)\}$
+ ×
(18,9)

$S_1^3 = \{(18,9), (28,11), (38,15), \underline{(40,17)}, \underline{(50,21)}\}$

discard

$= \{(18,9), (28,11), \underline{(38,15)}\}$

$S^4 = \{(0,0), (10,2), (20,6), (22,8), (32,12), (18,9), (28,11), (38,15)\}$

$S^4 = \{(0,0), (10,2), (20,6), (22,8), (32,12), (38,15)\}$

$sol^m = \{x_1, x_2, x_3, x_4\} = \{1, 1, 0, 1\}$

→

$N=7, m=15, P = \{10,5,15,7,6,18,3\}$ $w = \{2,3,5,7,1,4,1\}$ a / 1 0 0 1 0

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

$S_1^0 = \{(10,2)\}$

$S^1 = S^0 + S_1^0 = \{(0,0), (10,2)\}$

$S_1^1 = \{(5,3), (15,5)\}$

$S^2 = \{(0,0), (10,2), (5,3), (15,5)\}$

$S_1^2 = \{(15,5), (25,7), (20,8), (30,10)\}$

$S^3 = \{(0,0), (10,2), (15,5), (25,7), (30,10)\}$

$S_1^3 = \{(7,7), (17,9), (22,12), (32,14), (37,17)\}$

$S^4 = \{(0,0), (10,2), (15,5), (25,7), (30,10), (7,7), (17,9), (22,12), (32,14), (37,17)\}$

$S^4 = \{(0,0), (10,2), (15,5), (25,7), (30,10), (32,14)\}$

$S_1^4 = \{(6,1), (16,3), (21,6), (31,8), (36,11), (38,15)\}$

$S^5 = \{(0,0), (10,2), (15,5), (25,7), (30,10), (32,14), (6,1), (16,3), (21,6), (31,8), (36,11), (38,15)\}$

$S^5_2 \{(0,0), (10,2), (15,5), (25,7), (30,10), (32,14), (38,15)\}$

$S^5_1 = \{(18,4), (28,6), (33,9), (43,11), (48,14), (50,18), (56,19)\}$

$S^6 = \{(0,0), (10,2), (15,5), (25,7), (30,10), (32,14), (38,15), (18,4), (28,6), (33,9), (43,11), (48,14), (50,18), (56,19)\}$

$S^6 = \{(0,0), (10,2), (15,5), (25,7), (30,10), (32,14), (38,15)\}$

$S^6_1 = \{(3,1), (13,3), (18,6), (28,8), (33,11), (35,15), (41,16)\}$

$S^7 = \{(0,0), (10,2), (15,5), (25,7), (30,10), (32,14), (38,15), (3,1), (13,3), (18,6), (28,8), (33,11), (35,15)\}$

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

→ N=5, P={10,15,6,8,4}, w={4,6,3,4,2}

m=12

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

$S^0_1 = \{(10,4)\}$

$S^1 = \{(0,0), (10,4)\}$

$S^1_1 = \{(15,6), (25,10)\}$

$S^2 = \{(0,0), (10,4), (15,6), (25,10)\}$

$S^2_1 = \{(6,3), (16,7), (21,9), (31,13)\}$

$S^3 = \{(0,0), (10,4), (15,6), (25,10), (6,3), (16,7), (21,9), (31,13)\}$

(6,3), (16,7) ✗ ✗ ; (21,9) ✗

$S^3 = \{(0,0), (10,4), (15,6), (25,10), (31,13)\}$

$S^3_1 = \{(8,4), (18,8), (23,10), (33,14), (39,17)\}$

$S^4 = \{(0,0), (10,4), (15,6), (25,10), (31,13), (8,4), (18,8), (23,10), (33,14), (39,17)\}$

(8,4) ✗ ; (18,8) ✗ ; (23,10) ✗

$S^4 = \{(4,2), (14,6), (19,8), (29,12), (35,15), (37,16), (43,19)\}$

$S^5 = \{(0,0), (10,4), (15,6), (25,10), (31,13), (33,14), (39,17), (4,2), (14,6), (19,8), (29,12), (35,15), (37,16), (43,19)\}$

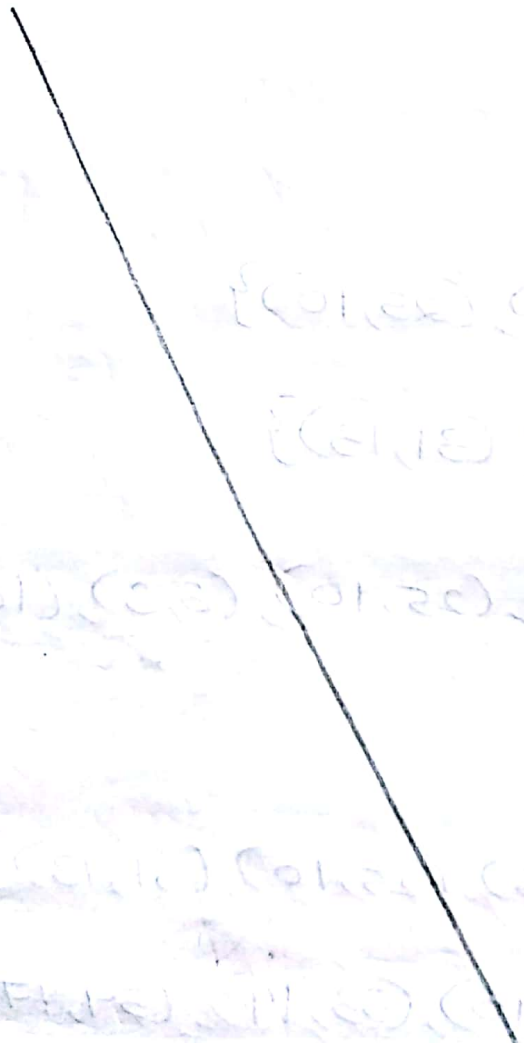
(31,13) ✗ ; (33,14) ✗ ; (39,17) ✗ ; (4,2) ✗ ; (14,6) ✗ ; (19,8) ✗ ; (29,12) ✗ ; (35,15) ✗ ; (37,16) ✗ ; (43,19) ✗

$S^5 = \{0, 0, (10, 2), (15, 6), (25, 10), (29, 12)\}$

$\therefore (x_1, x_2, x_3, x_4, x_5) = (1, 1, 0, 0, 1)$

$= (1, 1, 0, 0, 1)$

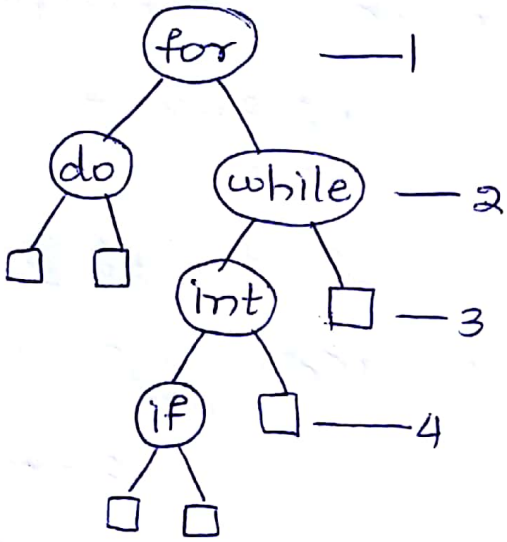
Time-complexity $= O(2^{n/2})$ for 0/1 knapsack problem.



→ optimal Binary search tree :-

The tree which is constructed with a low cost is optimal binary search tree.

Ex:- for, do, while, int, if

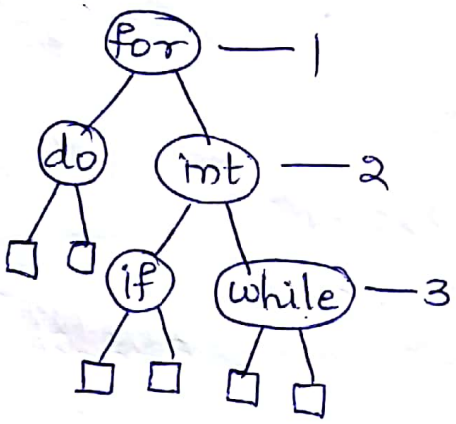


do

$\frac{4+3+2+1+2}{5}$

$= \frac{12}{5}$

q=6, P=5



$\frac{1+2+2+3+3}{5}$

$= \frac{11}{5}$

No. of unsuccessful searches = 6

Successful = 5

P = (1 to 5), q = (0 to 5)

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

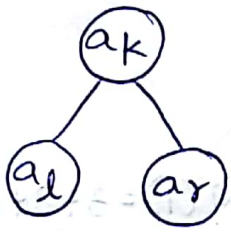
(c(i,k-1): left; c(k,j): right subtree)

$$w(i,j) = P(j) + q(j) + w(i,j-1)$$

$$r_{ij} = k,\ i<k\le j$$

$c(i,i) = 0, r(i,i) = 0$

$w(i,i) = q(i)$



Ex:- $n=4$, $a = \{do, if, int, while\}$

$P(1:4) = (3,3,1,1)$

$q(0:4) = (2,3,1,1,1)$

j-i	i=0	i=1	i=2	i=3	i=4
0	$c_{00}=0$ $w_{00}=2$ $r_{00}=0$	$c_{11}=0$ $w_{11}=3$ $r_{11}=0$	$c_{22}=0$ $w_{22}=1$ $r_{22}=0$	$c_{33}=0$ $w_{33}=1$ $r_{33}=0$	$c_{44}=0$ $w_{44}=1$ $r_{44}=0$
1	$c_{01}=8$ $w_{01}=8$ $r_{01}=1$	$c_{12}=7$ $w_{12}=7$ $r_{12}=2$	$c_{23}=3$ $w_{23}=3$ $r_{23}=3$	$c_{34}=3$ $w_{34}=3$ $r_{34}=4$	
2	$c_{02}=19$ $w_{02}=12$ $r_{02}=1$	$c_{13}=12$ $w_{13}=9$ $r_{13}=2$	$c_{24}=8$ $w_{24}=5$ $r_{24}=3$ (or) 4		
3	$c_{03}=25$ $w_{03}=14$ $r_{03}=2$	$c_{14}=19$ $w_{14}=11$ $r_{14}=2$			
4	$c_{04}=32$ $w_{04}=16$ $r_{04}=2$				

step-1:- $j-i=0$

$c_{00}=0$ $c_{11}=0$ $c_{22}=0$

$r_{00}=0$ $r_{11}=0$ $r_{22}=0$

$w_{00}=q(0)=2$ $w_{11}=q(1)=3$ $w_{22}=1$

$c_{33}=0, r_{33}=0, w_{33}=q(3)=1$ $c_{44}=0, w_{44}=q(4)=1$

step-2 :- $j-i=1$

$c_{01}=8$

$\omega_{01}=P(1)+q(1)=3+3+\omega(0,0)=3+3+2=8$

$r_{01}=1=k$

$c_{01}=\min\{c(0,0)+c(1,1)\}+\omega(0,1)$
$\underbrace{\qquad\qquad}_{k=1}$

$=\min\{0\}+8$

$=8$

$c_{12}=7$

$\omega_{12}=P(2)+q(2)+\omega(1,1)=3+1+3$

$=7$

$r_{12}=2$

$c_{12}=\min\{c(1,1)+c(2,2)\}+\omega(1,2)$
$1<k\leq 2$

$=\{0\}+7$

$=7$

$c_{23}=\min\{c(2,2)+c(3,3)\}+\omega(2,3)$
$2<k\leq 3$

$=3$

$\omega_{23}=P(3)+q(3)+\omega(2,2)$

$=1+1+1$

$=3$

$r_{23}=3$

$c_{34}=\min\{c(3,3)+c(4,4)\}+\omega(3,4)=3$
$3<k\leq 2+2$

$\omega(3,4)=P(4)+q(4)+\omega(3,3)=1+1+1=3$

$r_{34} = 4$

step-3 :- $j - i = 2$

$$c_{02} = \min_{0<k\leq 2} \{\underbrace{c(0,0)+c(1,2)}_{k=1}, \underbrace{c(0,1)+c(2,2)}_{k=2}\} + w(0,2)$$

$$= \min\{0+7, 8+0\} + w(0,2)$$

$$= \min\{7, 8\} + w(0,2)$$

$$= 7 + 12$$

$$= 19$$

$r(0,2) = \min\{1,2\} = 1$ [k values]

$$w(0,2) = P(2) + q(2) + w(0,1)$$

$$= 3 + 1 + 8$$

$$= 12$$

$$c_{13} = \min_{1<k\leq 3} \{\underbrace{c(1,1)+c(2,3)}_{k=2}, \underbrace{c(1,2)+c(3,3)}_{k=3}\}$$

$$= \min\{0+3, 7+0\} + w(1,3)$$

$$= \min\{3, 7\} + w(1,3)$$

$$= 3 + 9 = 12$$

$$w(1,3) = P(3) + q(3) + w(1,2)$$

$$= 1 + 1 + 7$$

$$= 9$$

$r(1,3) = 2$

$$c_{24} = \min_{2<k\leq 4} \{\underbrace{c(2,3)+c(3,4)}_{k=4}, \underbrace{c(2,2)+c(3,4)}_{k=3}\}$$

$$= \min\{3, 3\} + w(2,4)$$

$= 3+5 = 8$

$w(2,4) = P(4) + q(4) + w(2,3)$

$= 1+1+3$

$= 5$

$r(2,4) = 3 \text{ (or) } 4$

Step-4 :- $j - i = 3$

$c_{03} = \min_{0<k\leq 3} \{ \underbrace{c(0,0)+c(1,3)}_{k=1}, \underbrace{c(0,1)+c(2,3)}_{k=2},$

$\underbrace{c(0,2)+c(3,3)}_{k=3} \}$

$8 \quad +$

$= \min\{12, 3, 19\} + w(0,3)$

$= 3+14 = 17+8 = 25$

$w(0,3) = P(3) + q(3) + w(0,2)$

$= 1+1+12$

$= 14$

$r(0,3) = 2$

$c_{14} = \min_{1<k\leq 4} \{ \underbrace{c(1,1)+c(2,4)}_{k=2}, \underbrace{c(1,2)+c(3,4)}_{k=3},$

$\underbrace{c(1,3)+c(4,4)}_{k=4} \} + w(1,4)$

$= \{8, 10, 12\}$

$= 8+11$

$= 18+1 = 19$

$w(1,4) = p(4) + q(4) + w(1,3)$

$= 1+1+9$

$= 11$

$r(1,4) = 1+1 = 2$

step-5 :- $j-i = 4$

$$c_{04} = \min_{0<k\leq 4} \{\underbrace{c(0,0)+c(1,4)}_{k=1}, \underbrace{c(0,1)+c(2,4)}_{k=2}, \underbrace{c(0,2)+c(3,4)}_{k=3}, \underbrace{c(0,3)+c(4,4)}_{k=4}\} + w(0,4)$$

$= \min\{19, 16, 22, 25\} + w(0,4)$

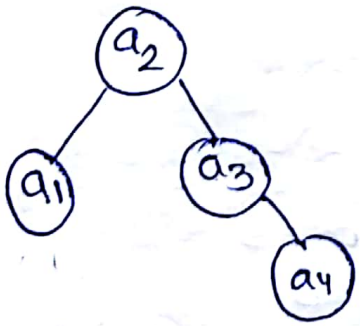
$= 16+16 = 32$

$w(0,4) = p(4) + q(4) + w(0,3)$

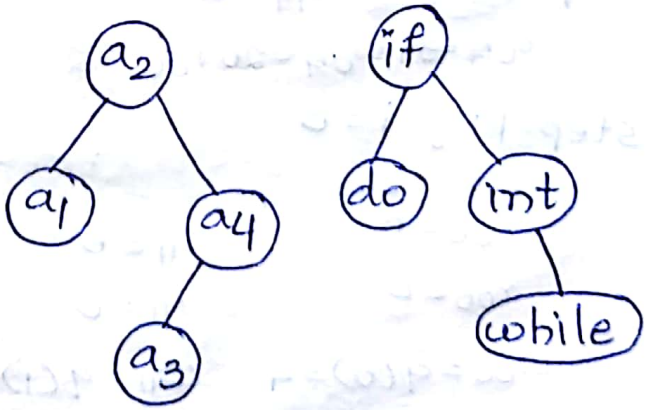
$= 1+1+14$

$= 16$

$r(0,4) = 2$



(or)



Q) a = {cout, float, if, while}, $P(1) = \frac{1}{20}$, $P(2) = \frac{1}{5}$, $P(3) = \frac{1}{10}$, $P(4) = \frac{1}{20}$, $q(0) = \frac{1}{5}$, $q(1) = \frac{1}{10}$, $q(2) = \frac{1}{5}$, $q(3) = \frac{1}{20}$, $q(4) = \frac{1}{20}$.

$P(1:4) = \frac{1}{20} \times 20, \frac{1}{5} \times 20, \frac{1}{10} \times 20, \frac{1}{20} \times 20$

$= (1, 4, 2, 1)$

$q(0:4) = (4, 2, 4, 1, 1)$

j-i \ i	0	1	2	3	4
0	$c_{00}=0$ $w_{00}=4$ $r_{00}=0$	$c_{11}=0$ $w_{11}=2$ $r_{11}=0$	$c_{22}=0$ $w_{22}=4$ $r_{22}=0$	$c_{33}=0$ $w_{33}=1$ $r_{33}=0$	$c_{44}=0$ $w_{44}=1$ $r_{44}=0$
1	$c_{01}=7$ $w_{01}=7$ $r_{01}=1$	$c_{12}=10$ $w_{12}=10$ $r_{12}=2$	$c_{23}=7$ $w_{23}=7$ $r_{23}=3$	$c_{34}=3$ $w_{34}=3$ $r_{34}=4$	
2	$c_{02}=22$ $w_{02}=15$ $r_{02}=2$	$c_{13}=20$ $w_{13}=13$ $r_{13}=2$	$c_{24}=12$ $w_{24}=9$ $r_{24}=3$		
3	$c_{03}=32$ $w_{03}=18$ $r_{03}=2$	$c_{14}=27$ $w_{14}=15$ $r_{14}=2$			
4	$c_{04}=39$ $w_{04}=20$ $r_{04}=2$				

step-1 : j-i = 0

$c_{00}=0$
$r_{00}=0$
$w_{00}=q(0)=4$

$c_{11}=0$
$r_{11}=0$
$w_{11}=q(1)=2$

$c_{22}=0$
$r_{22}=0$
$w_{22}=q(2)=4$

$c_{33}=00$
$r_{33}=0$
$w_{33}=q(3)=1$

$c_{44}=00$
$r_{44}=00$
$w_{44}=q(4)=1$

2:- $j-i=1$

$$c_{01} = \min_{0<k\le 1} \{c(0,0)+c(1,1)\} + w(0,1)$$

$k=1$

$$= 0 + w(0,1)$$

$$= 7$$

$$w_{01} = P(1)+q(1)+w(0,0)$$

$$= 1+2+4$$

$$= 7$$

$$r_{01} = 1$$

$$c_{12} = \min_{1<k\le 2} \{c(1,1)+c(2,2)\} + w(1,2)$$

$$= 0 + w(1,2)$$

$$= 10$$

$w_{01} = P(1)+q(1)+w(0,0)$ $w(1,2) = P(2)+q(2)+$

$= 4+4+2 = 10$ $w(1,1)$

$$r_{12} = 2$$

$$c_{23} = \min_{2<k\le 3} \{c(2,2)+c(3,3)\} + w(2,3)$$

$$= 0 + w(2,3)$$

$$= 7$$

$$w(2,3) = p(3)+q(3)+w(2,2)$$

$$= 2+1+4 = 7$$

$$r_{23} = 3$$

$$c_{34} = \min_{3<k\le 4} \{c(3,3)+c(4,4)\} + w(3,4)$$

$$= 0 + w(3,4) = 3$$

$$w(3,4) = P(4)+q(4)+w(3,3)$$

$$= 1+1+1$$

$$= 3$$

$$r(3,4) = 4$$

step-3 :- $j-i=2$

$$c_{02} = \min_{0<k\le 2} \{c(0,0)+c(1,2),\ c(0,1)+c(2,2)\} + w(0,2)$$

($k=1$ for $c(0,0)+c(1,2)$; $k=2$ for $c(0,1)+c(2,2)$)

$$= \min\{0+10, 7\} + w(0,2)$$

$$= 7+15 = 22$$

$$w(0,2) = P(2)+q(2)+w(0,1)$$

$$= 4+4+7$$

$$= 15$$

$$r_{02} = 2$$

$$c_{13} = \min_{1<k\le 3} \{c(1,1)+c(2,3),\ c(1,2)+c(3,3)\} + w(1,3)$$

($k=2$ for $c(1,1)+c(2,3)$; $k=3$ for $c(1,2)+c(3,3)$)

$$= \min\{0+7, 10+0\} + w(1,3)$$

$$= 7+13 = 20$$

$$w(1,3) = P(3)+q(3)+w(1,2) \Rightarrow 2+1+10 = 13$$

$r_{13} = k = 2$

$$c_{24} = \min_{2<k\le 4} \{\overbrace{c(2,2)+c(3,4)}^{k=3}, \overbrace{c(2,3)+c(4,4)}^{k=4}\} + w(2,4)$$

$= \min\{3,7\} + 9$

$= 12$

$w(2,4) = P(4) + q(4) + w(2,3)$

$= 2+7 = 9$

$r_{24} = k = 3$

step-3 :- $j-i=3$

$$c_{03} = \min_{0<k\le 3} \{\overbrace{c(0,0)+c(1,3)}^{k=1}, \overbrace{c(0,1)+c(2,3)}^{k=2} + c(0,2)+c(3,3)\} + w(0,3)$$

$= \min\{20, 14, 22\} + w(0,3)$

$= 14+18 = 32$

$w(0,3) = P(3) + q(3) + w(0,2)$

$= 2+1+15$

$= 18$

$r_{03} = k = 2$

$$c_{14} = \min_{1<k\le 4} \{\overbrace{c(1,1)+c(2,4)}^{k=2}, \overbrace{c(1,2)+c(3,4)}^{k=3}, c(1,3)+c(4,4)\} + w(1,4)$$

$= \min\{12, 13, 20\} + w(1,4)$

$= 12+15 = 27$

$w(1,4) = p(4) + q(4) + w(1,3)$

$= 1+1+13 = 15$

$r_{14} = 2$

step-5 : $j-i = 4$

$k=1$ $\qquad$ $k=2$

$c_{04} = \min_{0<k\le 4} \{c(0,0)+c(1,4),\ c(0,1)+c(2,4),\ c(0,2)+c(3,4),\ c(0,3)+c(4,4)\} + w(0,4)$

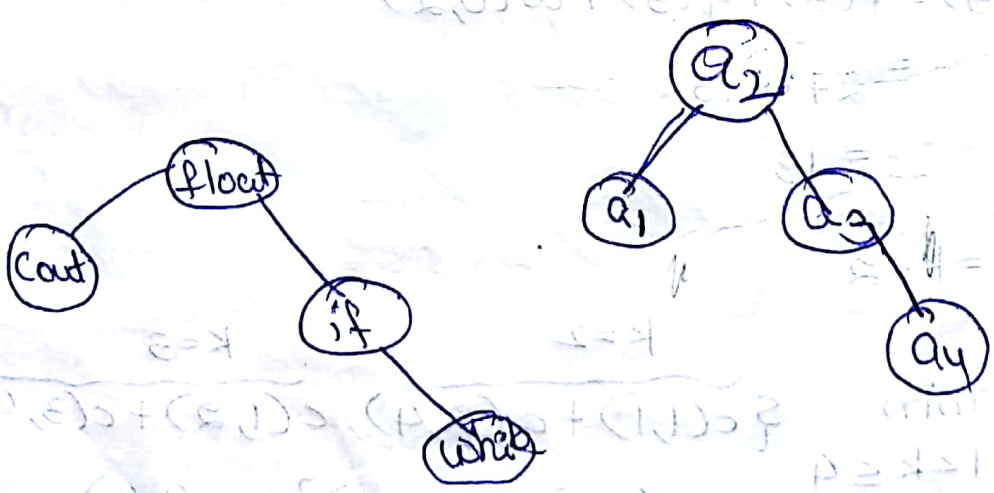
$= \min\{27, 19, 25, 32\} + w(0,4)$

$= 19 + 20 = 39$

$w(0,4) = p(4) + q(4) + w(0,3)$

$= 1+1+18 = 20$

$r_{04} = 2$



Time-complexity is $O(n^3)$.

```
Algorithm OBST (P, q, n)
{
for i := 0 to n-1 do
{
w[i,i] := q[i]; r[i,i] := 0; c[i,i] := 0.0; } j-i=0
 w[i,i+1] := q[i] + q[i+1] + P[i+1];
r[i,i+1] := i+1;                                  } for j-i=1
c[i,i+1] := q[i] + q[i+1] + P[i+1];
}
w[n,n] := q[n]; r[n,n] := 0; c[n,n] := 0.0;
for m := 2 to n do ---> j-i
  for i := 0 to n-m do
{
j := i+m;
w[i,j] = w[i,j-1] + P[j] + q[j];
    k = Find (c, r, i, j);
c[i,j] := w[i,j] + c[i,k-1] + c[k,j];
    r[i,j] := k;
}
write (c[0,n], w[0,n], r[0,n]);
}
Algorithm Find (c, r, i, j)
{
 min := ∞;
```

4th iteration

```
for m := r[i,j-1] to r[i+1,j] do
if (c[i,m-1]+c[m,j]<min) then
{
  min := c[i,m-1] + c[m,j]; l := m;
}
return l;
}
```

for k value

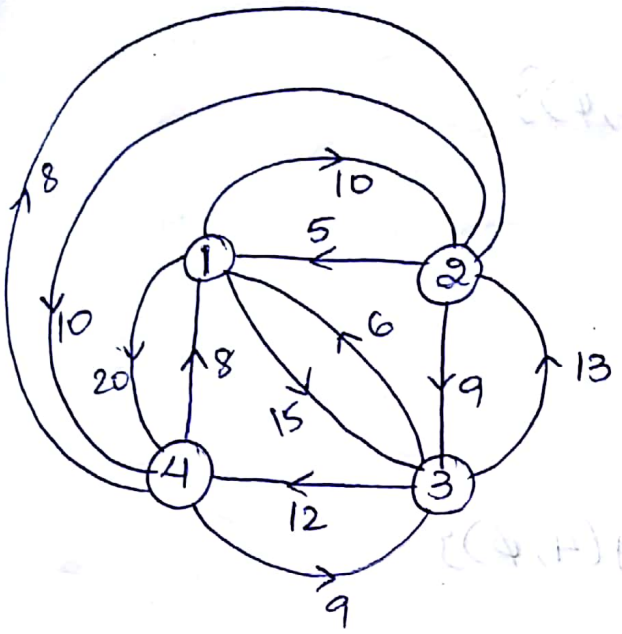
3. Travelling sales person problem :-

A tour of G is the some directed simple cycle that includes every vertex in V. The cost of tour is sum of cost of edges of on tour. The travelling sales person problem is to find a tour of minimum cost.

1) $g(i,\phi) = c_{i,1} \quad 1 \le i \le n$

2) $g(i,s) = \min_{j \in s} \{c_{ij} + g(j, s-\{j\})\}$

3) $g(1, V-\{1\}) = \min_{2 \le k \le n} \{c_{1k} + g(k, V-\{1,k\})\}$



	1	2	3	4
1	0	10	15	20
2	5	0	9	10
3	6	13	0	12
4	8	18	9	0

step-1:- $|s| = \phi$

$g(2,\phi) = c_{21} = 5$

$g(3,\phi) = c_{31} = 6$

$g(4,\phi) = c_{41} = 8$

step-2:- $|s| = 1$

$$g(2,3) = \min_{j \in 3} \{c_{23} + g(3, 3-3)\}$$

$$= \{c_{23} + g(3,\phi)\}$$

$$= 9 + 6 = 15$$

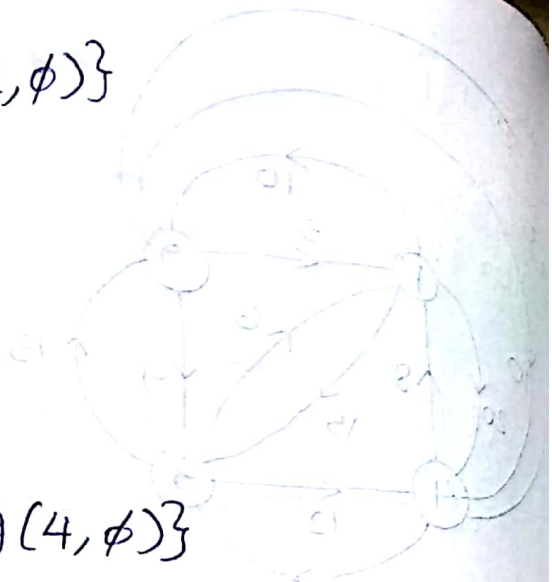
$$g(2,4) = \min_{j \in 4} \{c_{24} + g(4,\phi)\} = 10 + 8 = 18$$

$$g(3,2) = \min_{j \in 2} \{c_{32} + g(2,\phi)\}$$

$$= c_{32} + 5$$

$$= 13 + 5$$

$$= 18$$



$$g(3,4) = \min_{j \in 4} \{c_{34} + g(4,\phi)\}$$

$$= 12 + 8$$

$$= 20$$

$$g(4,2) = \min_{j \in 2} \{c_{42} + g(2,\phi)\}$$

$$= 8 + 5 = 13$$

$$g(4,3) = \min_{j \in 3} \{c_{43} + g(3,\phi)\}$$

$$= 9 + 6 = 15$$

$|S| = 2$

$$g(i,S) = \min_{j \in S} \{c_{ij} + g(j, S - \{j\})\}.$$

$$g(2,\{3,4\}) = \min_{j \in \{3,4\}} \{c_{23} + g(3,4)\},$$

$$\{c_{24} + g\{4,3)\}$$

$$= \{9 + 20, 10 + 15\} \quad j = 4$$

$$= \{29, 25\} \min$$

$$= 25$$

$$g(3,\{2,4\}) = \min_{j \in \{2,4\}} \{c_{32} + g(2, \{2,4\} - \{2,4\})$$

$\Rightarrow \min_{j \in \{2,4\}} \{c_{34} + g\{2,4\} - 4\}$

$\Rightarrow \min\{c_{32} + g(2,4)\}, \{c_{34} + g(4,2)\}\}$

$= \min\{13+18, 12+13\}$

$= \min\{31, 25\}$

$= 25$

$g(4,\{2,3\}) = \min_{j \in \{2,3\}} \{c_{42} + g(2,\{2,3\}-2),$

$c_{43} + g(3,2)\}$

$= \{c_{42} + g(2,3), c_{43} + g(3,2)\}$

$= \min\{8+15, 9+18\}$

$= 23$

step-3 :- $|s| = 3$

$g(1,\{2,3,4\}) = \min_{2 \le k \le 4} \{\underbrace{c_{12} + g(2,\{3,4\})}_{k=2}, c_{13} +$

$\underbrace{g(3,\{2,4\})}_{k=3},$

$\underbrace{c_{14} + g(4,\{2,3\})}_{k=4}$

$= \min\{10+25, 15+25, 20+23\}$

$= \min\{35, 40, 43\}$

$= 35$

The optimal tour obtained is :

1 — 2 — 4 — 3 — 1 is 35

$10 + 10 + 9 + 6 = 35$

Travelling sales person problem :-

2)

	1	2	3	4	5
1	∞	7	3	12	8
2	3	∞	6	14	9
3	5	8	∞	6	18
4	9	3	5	∞	11
5	18	14	9	8	∞

step-1: $|s| = \phi$

$g(2,\phi) = c_{21} = 3$ ✓

$g(3,\phi) = c_{31} = 5$

$g(4,\phi) = c_{41} = 9$

$g(5,\phi) = c_{51} = 18$

step-2:- $|s| = 1$

$g(2,3) = \min_{j \in 3} \{c_{23} + g(2+1, \phi)\}$

$= \min\{c_{23} + 5\}$

$= 11$

$g(2,4) = \min_{j \in 4} \{c_{24} + g(4, \phi)\}$

$= 14 + 9$

$= 23$

$g(2,5) = \min_{j \in 5} \{c_{25} + g(5, \phi)\}$

$= 9 + 18$

$= 27$

$g(3,2) = \min_{j \in 2} \{c_{32} + g(2, \phi)\}$

$= 8 + 3 = 11$

$$g(3,4) = \min_{j \in 4} \{c_{34} + g(4,\phi)\}$$

$$= 6+9 = 15$$

$$g(3,5) = \min_{j \in 5} \{c_{35} + g(5,\phi)\}$$

$$= 18+18 = 36$$

$$g(4,2) = c_{42} + g(2,\phi) \checkmark$$

$$= 3+3 = 6$$

$$g(4,3) = \min_{j \in 3} \{c_{43} + g(3,\phi)\}$$

$$= 5+5 = 10$$

$$g(4,5) = \min\{c_{45} + g(5,\phi)\}$$

$$= 11+18$$

$$= 29$$

$$g(5,2) = \min\{c_{52} + g(2,\phi)\}$$

$$= 14+3 = 17$$

$$g(5,3) = \min\{c_{53} + g(3,\phi)\}$$

$$= 9+5 = 14$$

$$g(5,4) = \min\{c_{54} + g(4,\phi)\}$$

$$= 8+9 = 17$$

$|s| = 2$

$$g\{2,\{3,4\}\} = \min_{j \in \{3,4\}} \{\{c_{23} + g\{3,\{3,4\}-3\}, \{c_{24} + g\{4,\{3,4\}-4\}\}$$

$j=3$

$$\Rightarrow \min\{c_{23} + g(3,4)\}, \{c_{24} + g(4,3)\}\}$$

$\Rightarrow \min\{21, 24\}$

$= 21$

$g\{2,\{3,5\}\} = \min_{j\in\{3,5\}} \{\{c_{23} + g\{3,\{3,5\}-3\}, \{c_{25} + g\{5,\{3,5\}-5\}\}$

$= \min\{c_{23} + g(3,5)\}, \{c_{25} + g(5,3)\}$

$= \min\{6+36, 9+14\}$

$= 23$

$g\{2,\{4,5\}\} = \min_{j\in\{4,5\}} = \{c_{24} + g\{4,\{4,5\}-4\}, \{c_{25} + g\{5,\{4,5\}-5\}$

$= \min\{14+29, 9+17\}$

$= \{43, 26\}$

$= 26$

$g\{3,\{2,4\}\} = \min_{j\in\{2,4\}} \{c_{32} + g(2,4), c_{34} + g(4,2)\}$ ✓

$= \{8+23, 6+6\}$

$= \{31, 12\} = 12$

$g\{3,\{2,5\}\} = \min_{j\in\{2,5\}} \{c_{32} + g(2,5), c_{35} + g(5,2)\}$

$= \{8+27, 18+17\} \Rightarrow \{35, 35\}$

$= 35$

$g\{3,\{4,5\}\} = \min_{j\in\{4,5\}} \{c_{34} + g(4,5), c_{35} + g(5,4)\}$

$= \{6+29, 18+17\} \Rightarrow \{35, 35\} = 35$

$$g\{4,\{2,3\}\} = \min_{j\in\{2,3\}} \{c_{42}+g(2,\{2,3\}-2)\}, \{c_{43}+g(3,\{2,3\}-3)\}$$

$$= \{3+11, 5+11\}$$

$$= 14$$

$$g\{4,\{2,5\}\} = \min\{c_{42}+g(2,5), c_{45}+g(5,2)\}$$

$$= \min\{3+27, 11+17\}$$

$$= \{30, 28\}$$

$$= 28$$

$$g\{4,\{3,5\}\} = \min\{c_{43}+g(3,5), c_{45}+g(5,3)\}$$

$$= \{5+36, 11+14\}$$

$$= \{41, 25\}$$

$$= 25$$

$$g\{5,\{2,3\}\} = \min\{c_{52}+g(2,3)\}, c_{53}+g(3,2)\}$$

$$= \min\{14+11, 9+11\}$$

$$= 20$$

$$g\{5,\{3,4\}\} = \min\{c_{53}+g(3,4), c_{54}+g(4,3)\}$$

$$= \{9+15, 8+10\}$$

$$= 18$$

$$g\{5,\{2,4\}\} = \min\{c_{52}+g(2,4), c_{54}+g(4,2)\}$$

$$= \{14+23, 8+6\}$$

$$= 14$$

$|s|=3$

$g\{2,\{3,4,5\}\} = \min_{j\in\{3,4,5\}} \{c_{23}+g(3,\{4,5\}),\ c_{24}+g(4,\{3,5\}),\ c_{25}+g(5,\{2,3\})\}$

$\Rightarrow \{6+35, 14+25, 9+18\}$

$= 27$

$g\{3,\{2,4,5\}\} = \min_{j\in\{2,4,5\}} \{c_{32}+g(2,\{4,5\}),\ c_{34}+g(4,\{2,5\}),\ c_{35}+g(5,\{2,4\})\}$

$= \{8+26, 6+28, 18+14\}$

$= 32$

$g\{4,\{2,3,5\}\} = 26$

$g\{5,\{2,3,4\}\} = \min_{j\in\{2,3,4\}} \{c_{52}+g(2,\{3,4\}),\ c_{53}+g(3,\{2,4\}),\ c_{54}+g(4,\{2,3\})\}$

$= 21$ for $j=3$

$|s|=4$

$g\{1,\{2,3,4,5\}) = \min_{j\in\{2,3,4,5\}} = c_{12}+g\{2,\{3,4,5\}$

$c_{13}+g(3,\{2,4,5\}),\ c_{14}+g(4,\{2,3,5\}),\ c_{15}+g(5,\{2,3,4\})\}$

$\Rightarrow \{40, 35, 38, 29\}$ $j=5$

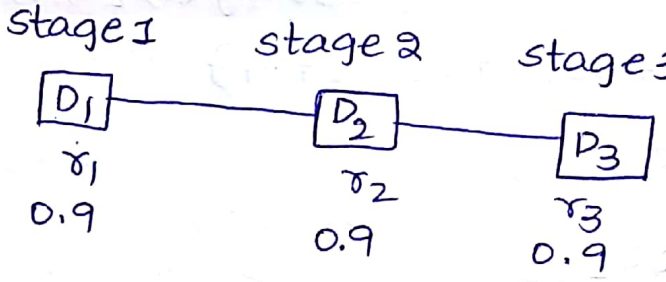
$= 29$

1—5—3—4—2—1

The optimal solution or path is $1 \to 5 \to 3 \to 4 \to 2 \to 1$ with cost of $8+9+6+3+3$ (from matrix) $= 29$

Reliability Design :-

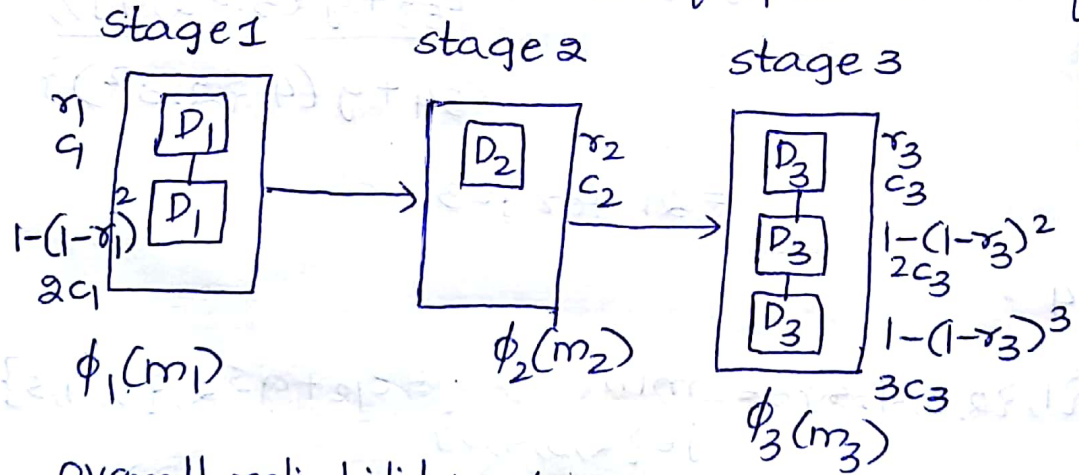
The problem is to design a system that is composed of several devices connected in series. Let R_i be reliability of device D_i, objective function is to maximize reliability. Maximize $\prod_{1 \le i \le n} \phi_i(m_i)$ subject to $\sum_{1 \le i \le n} c_i m_i \le c$



$m_i \ge 1$ integer,

$\prod_{1 \le i \le n} r_i = (0.9)^3 = 0.72$ it is reduced.

No. of copies $\leftarrow M_i = \left[(c + c_i - \sum_{1}^{m} c_j)/c_i \right]$



Overall reliability $= \phi_1(m_1) \cdot \phi_2(m_2) \cdot \phi_3(m_3)$

Total reliability when connected in parallel $= 1-(1-r_i)^{m_i}$

Reliability $= \prod_{1 \le i \le n} \phi_i(m_i)$

→ $c_1 = 30, c_2 = 15, c_3 = 20, C = 105$

$r_1 = 0.9, r_2 = 0.8, r_3 = 0.5$

Now, we have to construct 3 stage system with no. of copies in stage 1 is

$$M_i = \left[(105 + 30 - (30 + 15 + 20))/30 \right] \quad \left[\frac{C + c_i - (\Sigma c_i)}{c_i} \right]$$

$$= [135 - 65/30]$$

$$= \frac{70}{30} = 2.3 = 2$$

$$M_2 = [(105 + 15 - (30 + 15 + 20)/15]$$

$$= [55/15]$$

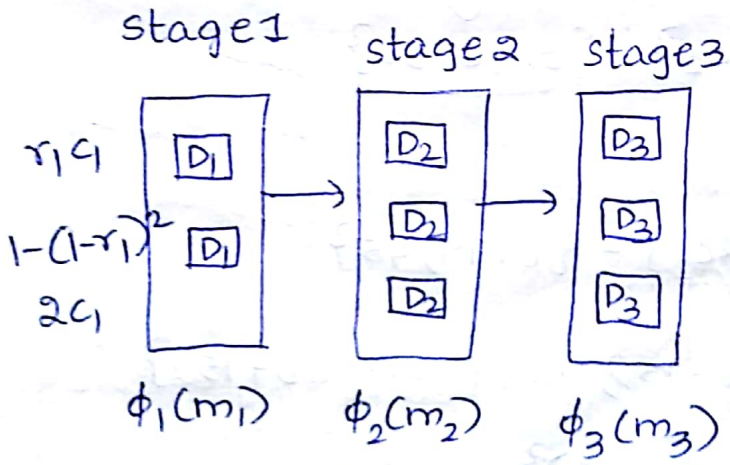
$$= L[3.667] \rightarrow \text{lower boundary}$$

$$= 3$$

$$M_3 = [105 + 20 - (30 + 15 + 20))/20]$$

$$= [60/20]$$

$$= 3$$



stage 0 : $s^0 = (1, 0)$
r, c'

stage 1 : $\phi_1(m_1)$

$m_1 = 1 \rightarrow$ First stage 1st copy

$s_1^1 = (r_1, c_1)$

$= (0.9, 30)$

$m_1 = 2$

$(1-(1-r_1)^2) = (1-(1-0.9)^2)$

$= 0.99$

$2c_1 = 60$

$s_1^1 = \{(1 \times 0.9, 0+30)\} \Rightarrow (0.9, 30)$ ✓

$s_2^1 = \{(1 \times 0.99, 0+60\} = (0.99, 60)$

$\phi_1(m_1) = s_1^1 + s_2^1 = \{(0.9, 30), (0.99, 60)\}$

stage 2 : $\phi_2(m_2)$

$m_2 = 1$

$(r_2, c_2) = (0.8, 15)$

$m_2 = 2$

$\{(1-(1-r_2)^2), 2c_2\} = \{0.96, 30\}$ ✓

$m_2 = 3$

$\{(1-(1-r_2)^3), 3c_2\} = \{0.99, 45\}$

$s_1^2 = \{(0.9 \times 0.8, 30+15), (0.99 \times 0.8, 60+15)\}$

$= \{(0.72, 45), (0.79, 75)\}$

$S_2^2 = \{(0.96 \times 0.9, 30+30), (0.96 \times 0.99, 30+60)\}$

$= \{(0.86, 60), (0.95, 90)\}$

$S_3^2 = \{(0.99 \times 0.9, 45+30), (0.99 \times 0.99, 30+60+45)\}$

$= \{(0.89, 75), (0.98, 105)\}$

$\phi_2(m_2) = S_1^2 + S_2^2 + S_3^2$

$= \{(0.72, 45), (0.79, 75), (0.86, 60), (0.95, 90), (0.89, 75), (0.98, 105)\}$

$= \{(0.72, 45), (0.86, 60), (0.79, 75), (0.89, 75), (0.95, 90), (0.98, 105)\}$

$= \{(0.72, 45), (0.86, 60), (0.89, 75)\}$

Stage 3:

$m_3 = 1$

$(r_3, c_3) = (0.5, 20)$

$m_3 = 2$

$\{(1-(1-r_3)^2), 2c_3\} = (0.75, 40)$

$m_3 = 3$

$\{(1-(1-r_3)^3), 3c_3\} = (0.875, 60)$

$S_1^3 = \{(0.5 \times 0.72, 20+45), (0.5 \times 0.86, 20+60), (0.89 \times 0.5, 20+75)\}$

$= \{(0.36, 65), (0.43, 80), (0.44, 95)\}$

$S_2^3 = \{(0.75 \times 0.72, 40+45), (0.75 \times 0.86, 100), (0.75 \times 0.89, 40+75)\}$

$= \{(0.54, 85), (0.64, 100), (0.66, 115)\}$

$S_3^3 = \{(0.875 \times 0.72, 60+45), (0.875 \times 0.86, 120), (0.875 \times 0.89, 60+75)\}$

$= \{(0.63, 105), (0.75, 120), (0.778, 135)\}$

$\phi_3(m_3) = S_1^3 + S_2^3 + S_3^3$

$= \{(0.36, 65), (0.43, 80), (0.44, 95), (0.54, 85), (0.64, 100), (0.63, 105), (0.75, 120)\}$

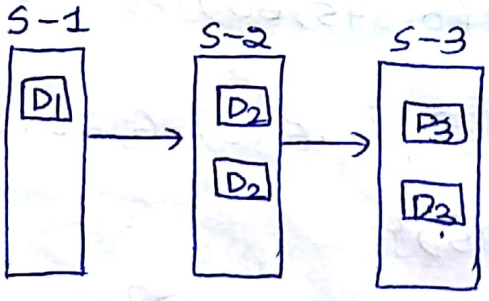
$\phi_3() = \{(0.36, 65), (0.43, 80), (0.54, 85), (0.64, 100)\}$

$m_1 = 1$

$m_2 = 2$

$m_3 = 2$

The optimal system is $m_1=1, m_2=2, m_3=2$.



with reliability of (0.64, cost of 100).

→ $C=90$

$r_1=0.7, r_2=0.6, r_3=0.4$

$c_1=25, c_2=15, c_3=20$

stage 0 : $(0,1) \Rightarrow (1,0)$
$r\,c \qquad r,c$

stage 1 : $\phi_1(m_1)$

$m_1=1$

$(r_1, c_1)=(0.7, 25)$

$m_1=2$

$((1-(1-r_1)^2), 2c_1)=(0.91, 50)$

$S_1^1=\{1\times 0.7, 0+25, 1\times 0.91, 0+50\}$

$=\{0.7, 25\}$

$S_2^1=\{1\times 0.91, 50\}=\{0.91, 50\}$

$\phi_1(m_1)=S_1^1+S_2^1=\{(0.7,25),(0.91,50)\}$

stage 2 :- $\phi_2(m_2)$

$m_2=1$

$(r_2, c_2)=(0.6, 15)$

$m_2=2$

$\{1-(1-r_2)^2, 2c_2\}=(0.84, 30)$

$m_2=3$

$\{1-(1-r_2)^3, 3c_2\}=(0.936, 45)$

$S_1^2 = \{(0.6 \times 0.7, 15+25), (0.84 \times 0.91, 30+50), (0.91 \times 0.836, 50+45)\}$

$= \{(0.42, 40), (0.588, 55), (0.85, 95)\}$

$S_1^2 = \{0.7 \times 0.6, 25+15), (0.91 \times 0.6, 50+15)\}$

$= \{(0.42, 40), (0.546, 65)\}$

$S_2^2 = \{(0.7 \times 0.84, 30+25), (0.91 \times 0.84, 50+30)\}$

$= \{(0.588, 55), (0.76, 80)\}$

$S_3^2 = \{(0.7 \times 0.936, 25+45), (0.91 \times 0.936, 50+45)\}$

$= \{(0.655, 70), (0.85, 95)\}$

$\phi_2(m_2) = \{S_1^2 + S_2^2 + S_3^2\}$

$= \{(0.42, 40), (0.546, 65), (0.588, 55), (0.76, 80), (0.655, 70), (0.85, 95)\}$

$= \{(0.42, 40), (0.546, 65), (0.588, 55), (0.655, 70)\}$

Stage-3 :-

$m_3 = 1$

$(r_3, c_3) = (0.4, 20)$

$m_3 = 2$

$((1-(1-r_3)^2), 2c_3) = (0.64, 60-20) = (0.64, 40)$

$S_1^3 = \{(0.4 \times 0.42, 20+40), (0.4 \times 0.588, 20+55), (0.4 \times 0.655, 70+20)\}$

$= \{(0.168, 60), (0.2352, 75), (0.262, 90)\}$

$S_2^3 = \{(0.64 \times 0.42, 40+40), (0.64 \times 0.588, 40+55), (0.64 \times 0.655, 40+70)\}$

$= \{(0.2688, 80), (0.376, 95), (0.4192, 110)\}$

✓ × ×

$\phi_3(m_3) = S_1^3 + S_2^3$

$= \{(0.168, 60), (0.2352, 75), (0.262, 90), (0.268, 80)\}$

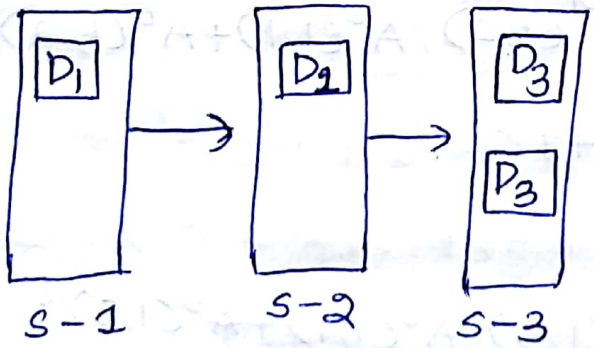
× ✓

$m_1 = 1$

$m_2 = 1$

$m_3 = 2$

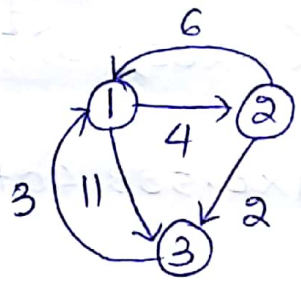
The optimal system is $m_1 = 1, m_2 = 1, m_3 = 2$



with realiability of (0.268 with cost of 80).

→ All pairs shortest path problem :-

$$A^k(i,j) = \min\{A^{k-1}(i,j), A^{k-1}(i,k) + A^{k-1}(k,j)\}, \quad k \geq 1$$



$$w(i,j) = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 4 & 11 \\ 6 & 0 & 2 \\ 3 & \infty & 0 \end{bmatrix} \end{array} = A^0(i,j)$$

step-1 :- Intermediate vertex k=1

$A^1(i,j)$ → calculate

For i=1,

j=1 $A^1(1,1) = \min\{A^{1-1}(1,1), A^0(1,1) + A^0(1,1)\}$

$= \min\{0\}$

$= 0$

2 $A^1(1,2) = \min\{A^{1-1}(1,2), A^0(1,1) + A^0(1,2)\}$

$= \min\{4, 4\}$

$= 4$

3 $A^1(1,3) = \min\{A^0(1,3), A^0(1,1) + A^0(1,3)\}$

$= \min\{11\}$

$= 11$

$i=2$

$j=1$ $A^1(2,1) = \min\{A^0(2,1), A^0(2,1)+A^0(1,1)\}$

$= \min\{6\}$

$= 6$

2 $A^1(2,2) = \min\{A^0(2,2), A^0(2,1)+A^0(1,2)\}$

$= \min\{10\}, 0\}$

$= 0$

3 $A^1(2,3) = \min\{A^0(2,3), A^0(2,1)+A^0(1,3)\}$

$= \min\{2, 6+11\}$

$= \min\{2, 17\} = 2$

$i=3$

$A^1(3,1) = \min\{A^0(3,1), A^0(3,1)+A^0(1,1)\}$

$= \min\{3,3\}$

$= 3$

$A^1(3,2) = \min\{A^0(3,2), A^0(3,1)+A^0(1,2)\}$

$= \min\{\infty, 7\}$

$= 7$

$A^1(3,3) = \min\{A^0(3,3), A^0(3,1)+A^0(1,3)\}$

$= \min\{0, 14\} = 0$

$$A^1(i,j) = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 4 & 11 \\ 6 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix} \end{array}$$

$3 \to 2 = 7$

$3 \to 1 = 3$

$1 \to 2 = 4$

step-2 :- Intermediate vertex k=2

$A^2(i,j)$

For i=1,

$A^2(1,1) = \min(A^{2-1}(1,1), A^{2-1}(1,1+1) + A^1(2,1))$

$= \min\{0, 4+6\}$

$= 0$

$A^2(1,2) = \min\{A^1(1,2), A^1(1,2) + A^1(2,2)\}$

$= \{4, 4\}$

$= 4$

$A^2(1,3) = \min\{A^1(1,3), A^1(1,2) + A^1(2,3)\}$

$= 6$

i=2

$A^2(2,1) = \min\{A^1(2,1), A^1(2,2) + A^1(2,1)\}$

$= 6$

$A^2(2,2) = \min\{A^1(2,2), A^1(2,2) + A^1(2,2)\}$

$= 0$

$A^2(2,3) = \min\{A^1(2,3) + A^1(2,2) + A^1(2,3)\}$

$= 2$

i=3

$A^2(3,1) = \min\{A^1(3,1), A^1(3,2) + A^1(2,1)\}$

$= 3$

$A^2(3,2) = \min\{A^1(3,2), A^1(3,2)+A^1(2,2)\}$

$= 7$

$A^2(3,3) = \min\{A^1(3,3), A^1(3,2)+A^1(2,3)\}$

$= 0$

$$A^1(i,j) = \begin{bmatrix} 0 & 4 & 6 \\ 6 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix} \rightarrow \begin{array}{l} 1 \rightarrow 3 = 6 \\ 1 \rightarrow 2 = 4 \\ 2 \rightarrow 3 = 2 \end{array}$$

Intermediate vertex $k=3$

$A^3(i,j)$

For $i=1$,

$A^3(1,1) = \min\{A^2(1,1), A^2(1,3)+A^2(3,1)\}$

$= 0$

$A^3(1,2) = \min\{A^2(1,2), A^2(1,3)+A^2(3,2)\}$

$= 4$

$A^3(1,3) = \min\{A^2(1,3), A^2(1,3)+A^2(3,3)\}$

$= 6$

For $i=2$,

$A^3(2,1) = \min\{A^2(2,1), A^2(2,3)+A^2(3,1)\}$

$= 5$

$A^3(2,2) = 0$

$A^3(2,3) = \min\{A^2(2,3), A^2(2,3)+A^2(3,3)\}$

$= 2$

For i=3,

$A^3(3,1)=3$

$A^3(3,2)=7$

$A^3(3,3)=0$

$$A^3(i,j)=\begin{bmatrix}0 & 4 & 6\\ 5 & 0 & 2\\ 3 & 7 & 0\end{bmatrix}$$

All pairs are $A^0(i,j)=\begin{bmatrix}0 & 4 & 11\\ 6 & 0 & 2\\ 3 & \infty & 0\end{bmatrix}$

$$A^1(i,j)=\begin{bmatrix}0 & 4 & 11\\ 6 & 0 & 2\\ 3 & (7) & 0\end{bmatrix}$$

$3\to2=7$

$3\to1+1\to2 \Rightarrow 3+4$

$=7$

$$A^2(i,j)=\begin{bmatrix}0 & 4 & (6)\\ 6 & 0 & 2\\ 3 & 7 & 0\end{bmatrix}$$

$1\to3=6$

$1\to2+2\to3=4+2$

$=6$

$$A^3(i,j)=\begin{bmatrix}0 & 4 & 6\\ (5) & 0 & 2\\ 3 & 7 & 0\end{bmatrix}$$

$2\to1=5$

$2\to3+3\to1=2+3$

$=5$

All pairs are $3\to2, 3\to1, 1\to2,$

$1\to3, 1\to2, 2\to3,$

$2\to1, 2\to3, 3\to1$

8)

$$\begin{matrix}1\\2\\3\\4\end{matrix}\begin{bmatrix} 0 & 10 & 15 & \infty \\ 5 & 0 & 9 & 10 \\ 6 & 13 & 0 & 12 \\ 8 & \infty & 9 & 0 \end{bmatrix} = A^0(i,j)$$

Step-1- Intermediate vertex k=1

$A^1(i,j)$

For i=1

$A^1(1,1) = \min(A^0(1,1), A^0(1,1)+A^0(1,1)\}$

$= 0$

$A^1(1,2) = \{\min A^0(1,2), A^0(1,1)+A^0(1,2)\}$

$= 10$

$A^1(1,3) = \{\min A^0(1,3), A^0(1,1)+A^0(1,3)\}$

$= 15$

$A^1(1,4) = \{\min A^0(1,4), A^0(1,1)+A^0(1,4)\}$

$= \{\infty, 0+\infty\} = \infty$

For i=2,

$A^1(2,1) = \min\{A^0(2,1), A^0(2,1)+A^0(1,1)\}$

$= 5$

$A^1(2,2) = 0$

$A^1(2,3) = \{9, 20\} = 9$

$A^1(2,4) = \{10, A^0(2,1)+A^0(1,4)\} = 10$

For i=3,

$A^1(3,1) = \{6,6\} = 6$, $A^1(3,2) = \{13,16\} = 13$

$A^1(3,3)=0$

$A^1(3,4)=\{12, A^0(3,1)+A^0(1,4)\}=12$

for i=4,

$A^1(4,1)=\{8, 8+0\}=8$

$A^1(4,2)=\{\infty, 8+10\}=18$

$A^1(4,3)=\{9, 8+15\}=9$

$A^1(4,4)=\{0\}$

$$A^1(i,j)=\begin{bmatrix} 0 & 10 & 15 & \infty \\ 5 & 0 & 9 & 10 \\ 6 & 13 & 0 & 12 \\ 8 & 18 & 9 & 0 \end{bmatrix}$$

step-2:- Intermediate value k=2, i=1

$A^2(1,1)=0$, $A^2(1,2)=\{10, 10+0\}=10$

$A^2(1,3)=\{15, 10+9\}=15$

$A^2(1,4)=\{\infty, 10+10\}=20$

For i=2,

$A^2(2,1)=\{5, 0+5\}=5$

$A^2(2,2)=0$

$A^2(2,3)=\{9, 0+9\}=9$

$A^2(2,4)=\{10, 0+10\}=10$

For i=3,

$A^2(3,1)=\{6, 13+5\}=6$

$A^2(3,2) = \{13, 13+0\} = 13$

$A^2(3,3) = 0$, $A^2(3,4) = \{12, 13+10\} = 12$

For i = 4,

$A^2(4,1) = \{8, 18+5\} = 8$

$A^2(4,2) = \{18, 18+0\} = 18$

$A^2(4,3) = \{9, 18+9\} = 9$

$A^2(4,4) = 0$

$$A^2(i,j) = \begin{bmatrix} 0 & 10 & 15 & 20 \\ 5 & 0 & 9 & 10 \\ 6 & 13 & 0 & 12 \\ 8 & 18 & 9 & 0 \end{bmatrix}$$

step-3 :- Intermediate value k = 3,

For i = 1

$A^3(1,1) = \{0\}$, $A^3(1,2) = \{10, 15+13\} = 10$

$A^3(1,3) = \{15, 15+0\} = 15$, $A^3(1,4) = \{20, 15+12\} = 20$

For i = 2,

$A^3(2,1) = \{5, 9+6\} = 5$, $A^3(2,2) = 0$

$A^3(2,3) = \{9, 9+0\} = 9$, $A^3(2,4) = \{10, 9+12\} = 10$

For i = 3,

$A^3(3,1) = \{6, 0+6\} = 6$, $A^3(3,2) = \{13, 0+13\} = 13$

$A^3(3,3) = 0$, $A^3(3,4) = \{12, 0+12\} = 12$

For i = 4,

$A^3(4,1) = \{8, 9+6\} = 8$, $A^3(4,2) = \{18, 9+13\} = 18$

$A^3(4,3) = \{9, 9+0\} = 9$, $A^3(4,4) = 0$

$$A^3(i,j) = \begin{bmatrix} 0 & 10 & 15 & 20 \\ 5 & 0 & 9 & 10 \\ 6 & 13 & 0 & 12 \\ 8 & 18 & 9 & 0 \end{bmatrix}$$

step-4:- Intermediate value k=4

For i=1,

$A^4(1,1) = \min\{0\} = 0$

$A^4(1,2) = \{10, 20+18\} = 10$

$A^4(1,3) = \{15, 20+9\} = 15$, $A^4(1,4) = \{20, 20+0\} = 20$

For i=2,

$A^4(2,1) = \{5, 10+8\} = 5$

$A^4(2,2) = 0$, $A^4(2,3) = \{9, 10+9\} = 9$

$A^4(2,4) = \{10, 10+0\} = 10$

For i=3,

$A^4(3,1) = \{6, 12+8\} = 6$

$A^4(3,2) = \{13, 12+18\} = 13$

$A^4(3,3) = 0$, $A^4(3,4) = \{12, 12+0\} = 12$

For i=4,

$A^4(4,1) = \{8, 0+8\} = 8$

$A^4(4,2) = \{18, 0+18\} = 18$

$A^4(4,4) = 0$, $A^4(4,3) = \{9, 0+9\} = 9$

$$A^4[i,j) = \begin{bmatrix} 0 & 10 & 15 & 20 \\ 5 & 0 & 9 & 10 \\ 6 & 13 & 0 & 12 \\ 8 & 18 & 0 & 9 \end{bmatrix}$$

All shortest possible paths are :-

$$A^0(i,j) = \begin{bmatrix} 0 & 10 & 15 & \infty \\ 5 & 0 & 9 & 10 \\ 6 & 13 & 0 & 12 \\ 8 & \infty & 9 & 0 \end{bmatrix}$$

$$A^1(i,j) = \begin{bmatrix} 0 & 10 & 15 & \infty \\ 5 & 0 & 9 & 10 \\ 6 & 13 & 0 & 12 \\ 8 & (18) & 9 & 0 \end{bmatrix}$$

$4 \rightarrow 2 = 18$

$4 \rightarrow 1 + 1 \rightarrow 2 = 8 + 10$

$= 18$

$$A^2(i,j) = \begin{bmatrix} 0 & 10 & 15 & (20) \\ 5 & 0 & 9 & 10 \\ 6 & 13 & 0 & 12 \\ 8 & 18 & 9 & 0 \end{bmatrix}$$

$1 \rightarrow 4 = 20$

$1 \rightarrow 2 + 2 \rightarrow 4 = 10 + 10$

$= 20$

$$A^3(i,j) = \begin{bmatrix} 0 & 10 & 15 & 20 \\ 5 & 0 & 9 & 10 \\ 6 & 13 & 0 & 12 \\ 8 & 18 & 9 & 0 \end{bmatrix}$$

$$A^4(i,j) = \begin{bmatrix} 0 & 10 & 15 & 20 \\ 5 & 0 & 9 & 10 \\ 6 & 13 & 0 & 12 \\ 8 & 18 & 9 & 0 \end{bmatrix}$$

```
Algorithm Allpairs (cost, A, n)
{
 for i:=1 to n do
  for j:=1 to n do                    } for k=0
    A[i,j]:= cost[i,j];
 for k:=1 to n do
  for i:=1 to n do                    } for k=1,2,3 ---
   for j:=1 to n do
     A[i,j]:=min(A[i,j], A[i,k]+A[k,j]);
}
```

Time-complexity $= O(n^3)$

Matrix chain multiplication :-

$M(i,i) = 0$

$$M_{ij} = \min_{i \le k \le j-1} \{ M_{ik} + M_{k+1\,j} + P_i P_{k+1} P_{j+1} \}$$

Given, $A_1 \to 5 \times 4$ $\quad P_1=5, P_2=4, P_3=6, P_4=2,$

$A_2 \to 4 \times 6$ $\quad P_5=7$

$A_3 \to 6 \times 2$

$A_4 \to 2 \times 7$

j-i \ i	1	2	3	4
0	M_{11}	M_{22}	M_{33}	M_{44}
1	M_{12}	M_{23}	M_{34}	X
2	M_{13}	M_{24}	X	X

3 M_{14} =

k=

step-0 :- j-i=0

$M_{11}=0$

$M_{22}=0$

$M_{33}=0$

$M_{44}=0$

step-1 :- j-i=1

$$M_{12} = \min_{1 \le k \le 1} \{M_{11} + M_{22} + P_1 P_2 P_3\}$$

$= \{0+0+120\}$

$= 120$

k=1

$$M_{23} = \min_{2 \le k \le 2} \{M_{22} + M_{33} + P_2 P_3 P_4\}$$

$= \{0+0+48\}$

$= 48$

k=2

$M_{34} = \{P_3 P_4 P_5\}$

$= 84$

k=3

step-2 :- j-i=2

$$M_{13} = \min_{1 \le k \le 2} \{M_{11} + M_{23} + P_1 P_2 P_4\} + \{M_{12} + M_{33} + P_1 P_3 P_4\}$$

$\{\min\{48,40\}, \{120+60\}\}$

$= 40+48 = 88$

$k=1$

$M_{24} = \min_{2\le k\le 3}\{M_{22}+M_{34}+P_2P_3P_5\}, \{M_{23}+M_{44}+P_2P_4P_5\}$

$= \min\{84+168, 48+56\}$

$= 104$

$k=3$

$M_{14} = \min_{1\le k\le 3}\{M_{11}+M_{24}+P_1P_2P_5\}, \{M_{12}+M_{34}+P_1P_3P_5\}, \{M_{13}+M_{44}+P_1P_4P_5\}$

$= \min\{104+140, 120+84+210, 88+70\}$

$= 158$

$k=3$

The optimal chain multiplication is

$(M_{13})M_{44}$

$(M_{11}(M_{23}))M_{44}$

$(A_1(A_2A_3))M_4$

$\Rightarrow (A_1(A_2A_3))A_4$ with minimum no. of multiplications as 158.

2) $A_1 = 3 \times 6$

$A_2 = 6 \times 5$

$A_3 = 5 \times 3$

$A_4 = 3 \times 7$

$A_5 = 7 \times 2$

$A_6 = 2 \times 4$

$P_1 = 3, P_2 = 6, P_3 = 5, P_4 = 3,$

$P_5 = 7, P_6 = 2, P_7 = 4$

step-0 :- $j - i = 0$

	i					
	1	2	3	4	5	6
j-i						
0	M_{11}	M_{22}	M_{33}	M_{44}	M_{55}	M_{66}
1	M_{12}	M_{23}	M_{34}	M_{45}	M_{56}	x
2	M_{13}	M_{24}	M_{46}	x	x	x
3	M_{14}	M_{25}	M_{36}	x	x	x
4	M_{15}	M_{26}	x	x	x	x
5	M_{16}	x	x	x	x	x

step-0 :- $j - i = 0$

$M_{11} = 0, M_{22} = 0, M_{33} = 0, M_{44} = 0, M_{55} = 0,$

$M_{66} = 0$

step-1 :- $j - i = 1$

$$M_{12} = \min_{1 \le k \le 1} \{ M_{11} + M_{22} + P_1 P_2 P_3 \}$$

$$= 90$$

$M_{23} = P_2 P_3 P_4$

$= 90$

$M_{34} = P_3 P_4 P_5$

$= 105$

$M_{45} = P_4 P_5 P_6$

$= 42$

$M_{56} = P_5 P_6 P_7$

$= 56$

step-2:- $j - i = 2$

$M_{13} = \min_{1 \le k \le 2} \{M_{11} + M_{23} + P_1 P_2 P_4, M_{12} + M_{33} + P_1 P_3 P_4\}$

$= \min\{90+54, 90+45\}$

$= 135$

$k = 2$

$M_{24} = \min_{2 \le k \le 3} \{M_{22} + M_{34} + P_2 P_3 P_5, M_{23} + M_{44} + P_2 P_4 P_5\}$

$= \min\{105+210, 90+126\}$

$= 216$

$k = 3$

$M_{35} = \min_{3 \le k \le 4} \{M_{33} + M_{45} + P_3 P_4 P_6, M_{34} + M_{55} + P_3 P_5 P_6\}$

$= \min\{42+30, 105+70\}$

$= 72$

$k=3$

$M_{46} = \min_{4 \le k \le 5} \{M_{44} + M_{56} + P_4 P_5 P_7, M_{45} + M_{66} + P_4 P_6 P_7\}$

$= \min\{56+84, 42+24\}$

$= 66$

$k=5$

step-3:- $j-i=3$

$M_{14} = \min_{1 \le k \le 3} \{M_{11} + M_{24} + P_1 P_2 P_5, M_{12} + M_{34} + P_1 P_3 P_5, M_{13} + M_{44} + P_1 P_5 P_4\}$

$= \min\{342, 210, 198\}$

$= 198$

$k=3$

$M_{25} = \min_{2 \le k \le 4} \{M_{22} + M_{35} + P_2 P_3 P_6, M_{23} + M_{45} + P_2 P_4 P_6, M_{24} + M_{55} + P_2 P_5 P_6\}$

$= \min\{132, 126, 300\}$

$= 126$

$\boxed{k=3}$

$$M_{36} = \min_{3 \le k \le 5} \{ M_{33} + M_{46} + P_3 P_4 P_7, M_{34} + M_{56} + P_3 P_5 P_7, M_{35} + M_{66} + P_3 P_6 P_7 \}$$

$= \min\{126, 301, 112\}$

$= 112$

$k = 5$

step-4 :- $j - i = 4$

$$M_{15} = \min_{1 \le k \le 4} \{ M_{11} + M_2 M_5 + P_1 P_2 P_6, M_{12} + M_{35} + P_1 P_3 P_6, M_{13} + M_{45} + P_1 P_4 P_6, M_{14} + M_{55} + P_1 P_5 P_6 \}$$

$= \min\{162, 192, 195, 240\}$

$= 162$

$k = 1$

$$M_{26} = \min_{2 \le k \le 5} \{ M_{22} + M_{36} + P_2 P_3 P_7, M_{23} + M_{46} + P_2 P_4 P_7, M_{24} + M_{56} + P_2 P_5 P_7, M_{25} + M_{66} + P_2 P_6 P_7 \}$$

$= \min\{232, 228, 440, 174\} = 174$

$k = 5$

step-5 :- $M_{16} = \min_{1 \le k \le 5} \{ M_{11} + M_{26} + P_1 P_2 P_7,$
$M_{12} + M_{36} + P_1 P_3 P_7,$
$M_{13} + M_{46} + P_1 P_4 P_7,$
$M_{14} + M_{56} + P_1 P_5 P_7,$
$M_{15} + M_{66} + P_1 P_6 P_7 \}$

$= \min\{246, 262, 237, 338, 186\}$

$= 186$

$\boxed{k=5}$

The optimal chain multiplication is

$(M_{15}) M_{66}$

$(M_{11} (M_{25})) M_{66}$

$(M_{11} (M_{23} (M_{45}))) M_{66}$

$\Rightarrow (A_1 (A_2 A_3 (A_4 A_5))) A_6$ with minimum number of multiplication as 186.

3/10/18

UNIT-V

Branch & Bound

The backtracking algorithm is effective for decision problems, but it is not designed for optimization problems.

Branch & bound technique is applicable only for minimization problem.

A bound value of objective function should be lower for minimization problem & upper bound for maximization process, for every node of state-space tree.

There are 3 types of such strategies in branch-bound :-

(1) FIFO search

(2) LIFO search

(3) LC (least cost) search

Control abstraction :-

```
Algorithm LCSearch (t)
{
 if *t is an answer node then output *t and
 return;
 E:=t;
repeat
 {
```

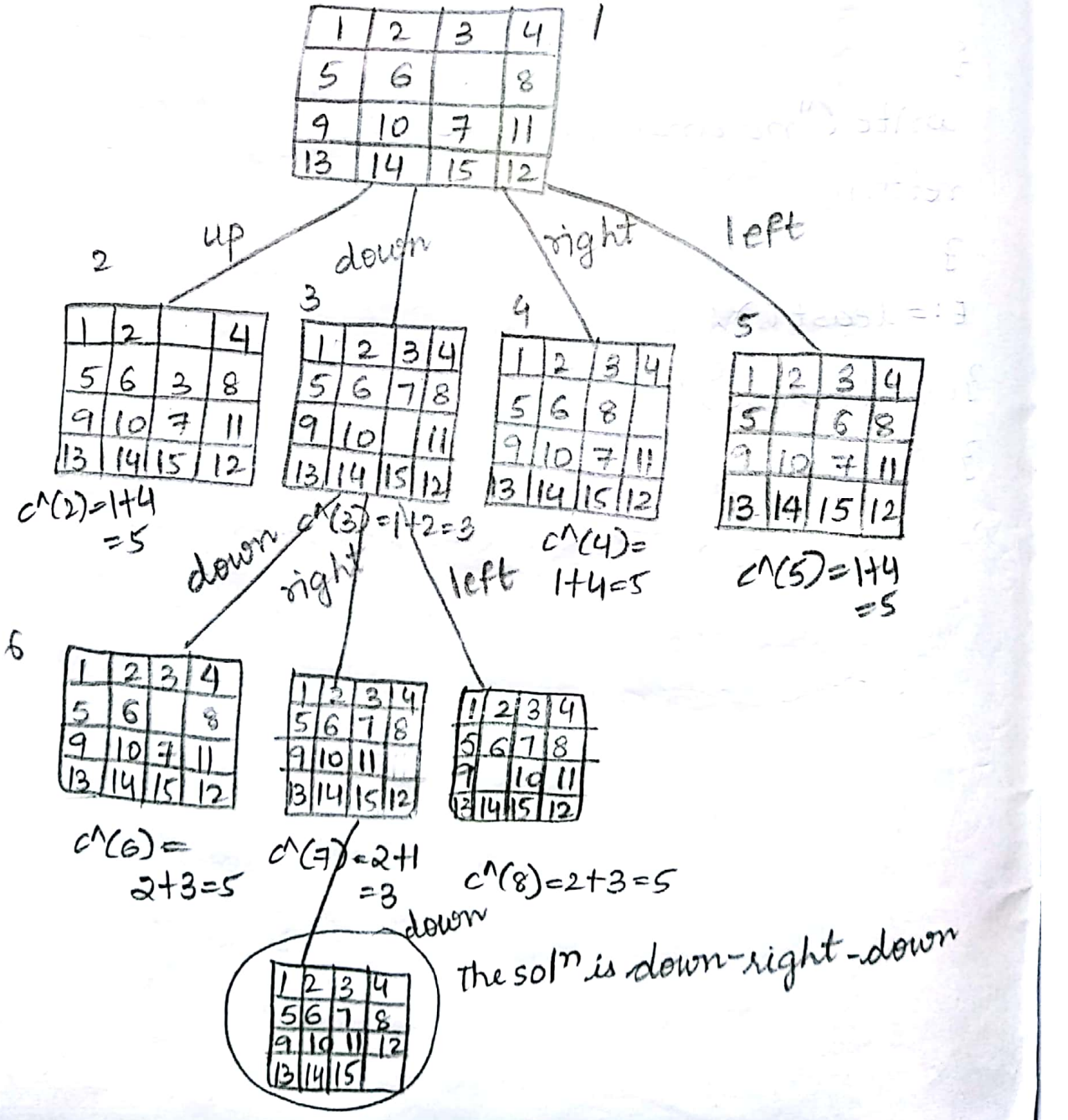
```
for each child x of E do
{
  if x is an answer node then output path from
  x to t and return;
  Add (x);
 (x → parent) := E;
}
if there are no more live nodes then
 {
  write ("no answer node");
  return;
 }
 E := least ();
} until (false);
}
```

Gametree (or) 15-puzzle problem using LC search

1	2	3	4
5	6		8
9	10	7	11
13	14	15	12

$c^{\wedge}(x) = f(x) + g^{\wedge}(x)$

$f(x)$ = length of path from root to node x, $g^{\wedge}(x)$ is number of non-blank tiles not in their goal position.



Applications :-

*** Travelling sales person problem :-

Let A be the reduced cost matrix for node R. Let s be child of R such that (R,s) Edge corresp--onds to including edge (i,j) in tour. If s is not a leafnode then reduced cost matrix for node s can be obtained as follows :-

(1) change all entries in row i, column j of A to ∞.

(2) set A(j,1) position to ∞.

(3) Apply row reduction and column reduction except for rows & columns containing ∞.

(4) Total cost for node s can be calculated as

$$c\hat{}(s) =$$

where, r = total amount obtained in step-3.

Ex:- solve following instance of travelling salesperson using LCBB (LC branch & bound)

	1	2	3	4	5
1	∞	7	3	12	8
2	3	∞	6	14	9
3	5	8	∞	6	18
4	9	3	5	∞	11
5	18	14	9	8	∞

(1) calculate $c\hat{}(1) = r = 27$

$c\hat{}(1) = 27$

①

Row reduction :-

$$\begin{bmatrix} \infty & 7 & 3 & 12 & 8 \\ 3 & \infty & 6 & 14 & 9 \\ 5 & 8 & \infty & 6 & 18 \\ 9 & 3 & 5 & \infty & 11 \\ 18 & 14 & 9 & 8 & \infty \end{bmatrix} \longrightarrow \begin{bmatrix} \infty & 4 & 0 & 9 & 5 \\ 0 & \infty & 3 & 11 & 6 \\ 0 & 3 & \infty & 1 & 13 \\ 4 & 0 & 2 & \infty & 8 \\ 10 & 6 & 1 & 0 & \infty \end{bmatrix} \begin{matrix} \rightarrow 3 \\ \rightarrow 3 \\ \rightarrow 5 \\ \rightarrow 3 \\ \rightarrow 8 \end{matrix}$$

22

column reduction :-

$$A = \begin{bmatrix} \infty & 4 & 0 & 9 & 5 \\ 0 & \infty & 3 & 11 & 6 \\ 0 & 3 & \infty & 1 & 13 \\ 6 & 0 & 2 & \infty & 8 \\ 10 & 6 & 1 & 0 & \infty \end{bmatrix} \longrightarrow \begin{bmatrix} \infty & 4 & 0 & 9 & 0 \\ 0 & \infty & 3 & 11 & 1 \\ 0 & 3 & \infty & 1 & 8 \\ 6 & 0 & 2 & \infty & 3 \\ 10 & 6 & 1 & 0 & \infty \end{bmatrix} \} \downarrow 5$$

$\therefore r$ = row reduction + column reduction

= 22 + 5

= 27

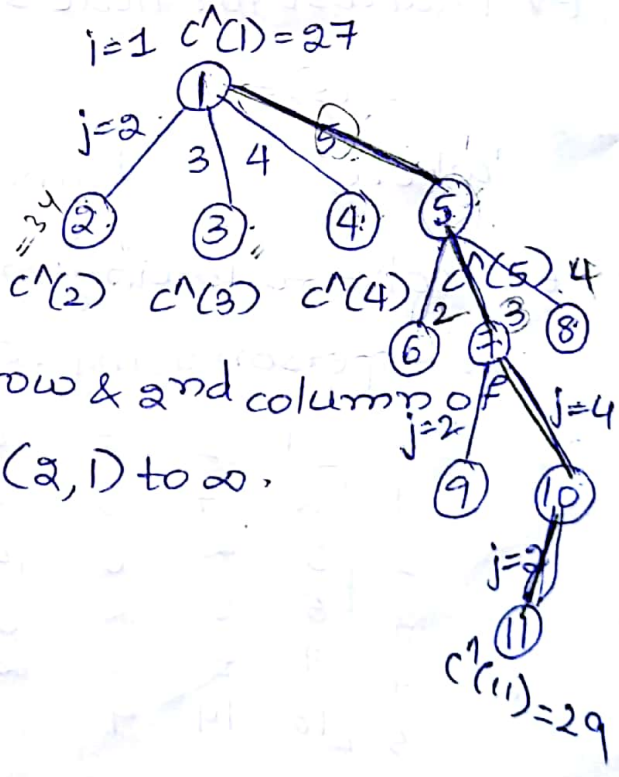
step-2 :-

calculate $c^\wedge(2)$

2.1) consider path (1,2)

change all entries of 1st row & and column of reduced matrix to ∞. set A(2,1) to ∞.

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 3 & 11 & 1 \\ 0 & \infty & \infty & 1 & 8 \\ 6 & \infty & 2 & \infty & 3 \\ 10 & \infty & 1 & 0 & \infty \end{bmatrix}$$



row-reduction :-

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 2 & 10 & 0 \\ 0 & \infty & \infty & 1 & 8 \\ 4 & \infty & 0 & \infty & 1 \\ 9 & \infty & 01 & 0 & \infty \end{bmatrix} \begin{matrix} \\ \rightarrow 1 \\ \rightarrow 0 \\ \rightarrow 2 \\ \\ \end{matrix}$$

10

3

column reduction :-

$$A = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 2 & 10 & 0 \\ 0 & \infty & \infty & 0 & 7 \\ 6 & \infty & 01 & \infty & 2 \\ 10 & \infty & 0 & 0 & \infty \end{bmatrix} \quad \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 2 & 10 & 0 \\ 0 & \infty & \infty & 1 & 8 \\ 4 & \infty & 0 & \infty & 1 \\ 10 & \infty & 1 & 0 & \infty \end{bmatrix}$$

$$\downarrow \quad \downarrow \quad \downarrow \qquad \downarrow$$

$$1 \quad 1 \quad 1 \qquad 0$$

3

$c^{\wedge}(2) = 3$

2.4 :- $c^{\wedge}(2) = c^{\wedge}(1) + r + A(1,2)$

$27 + 3 + 4$

$= 34$

2.5 :- consider path (1,3)

change all entries of 1st row & 3rd column to ∞

set (3,1) to ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 11 & 1 \\ \infty & 3 & \infty & 1 & 8 \\ 6 & 0 & \infty & \infty & 3 \\ 10 & 6 & \infty & 0 & \infty \end{bmatrix}$$

row-reduction :-

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 11 & 1 \\ \infty & 2 & \infty & 0 & 7 \\ 6 & 0 & \infty & \infty & 3 \\ 10 & 6 & \infty & 0 & \infty \end{bmatrix} \longrightarrow 1$$

column reduction :-

$$A = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 11 & 0 \\ \infty & 2 & \infty & 0 & 6 \\ 6 & 0 & \infty & \infty & 2 \\ 10 & 6 & \infty & 0 & \infty \end{bmatrix} \downarrow 1$$

r = 2

$$c^{\wedge}(3) = c^{\wedge}(1) + A(1,3) + r$$

$$= 27 + 9 + (-9) + 2$$

$$= 29$$

2.6 :- consider path (1,4)

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 3 & \infty & 1 \\ 0 & 3 & \infty & \infty & 8 \\ \infty & 0 & 2 & \infty & 3 \\ 10 & 6 & 1 & \infty & \infty \end{bmatrix}$$

row-reduction =

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 3 & \infty & 1 \\ 0 & 3 & \infty & \infty & 8 \\ \infty & 0 & 2 & \infty & 3 \\ 9 & 5 & 0 & \infty & \infty \end{bmatrix} \longrightarrow 1$$

column-reduction :-

$$A = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 3 & \infty & 0 \\ 0 & 3 & \infty & \infty & 7 \\ \infty & 0 & 2 & \infty & 2 \\ 9 & 6-1 & 0 & \infty & \infty \end{bmatrix} \downarrow 1$$

$r = 1+1 = 2$

$$c\hat{}(4) = c\hat{}(1) + A(1,4) + r$$
$$= 27 + 9 + 2$$
$$= 38$$

2.7 :- consider path (1,5) make $A(5,1) \to \infty$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 3 & 11 & \infty \\ 0 & 3 & \infty & 1 & \infty \\ 6 & 0 & 2 & \infty & \infty \\ 10 & 6 & 1 & 0 & \infty \end{bmatrix}$$

row-reduction

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 3 & 11 & \infty \\ 0 & 3 & \infty & 1 & \infty \\ 6 & 0 & 2 & \infty & \infty \\ 100 & 6 & 1 & 0 & \infty \end{bmatrix} \to 0$$

column reduction

$$A = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 2 & 11 & \infty \\ 0 & 3 & \infty & 1 & \infty \\ 6 & 0 & 1 & \infty & \infty \\ 100 & 6 & 0 & 0 & \infty \end{bmatrix} \downarrow 1$$

$r = 1$

$$c\hat{}(5) = c\hat{}(1) + r + A(1,5) = 27 + 1 + 0 = 28$$

Rc

step-3 :- j=2, i=5 (5,2)

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 2 & 11 & \infty \\ 0 & 3 & \infty & 1 & \infty \\ 6 & 0 & \infty & \infty & \infty \\ \infty & 6 & 0 & 0 & \infty \end{bmatrix}$$

consider path A(2,1), 5 row, 2nd column to ∞.

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 2 & 11 & \infty \\ 0 & \infty & \infty & 1 & \infty \\ 6 & \infty & 1 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

row-reduction =

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & 9 & \infty \\ 0 & \infty & \infty & 1 & \infty \\ 5 & \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix} \begin{matrix} \\ \rightarrow 2 \\ \\ \rightarrow 1 \\ \\ \hline 3 \end{matrix}$$

column-reduction :-

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & 8 & \infty \\ 0 & \infty & \infty & 0 & \infty \\ 5 & \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

$$\downarrow$$

$$1$$

$r = 3+1 = 4$

$c^\wedge(6) = c^\wedge(5) + r + A(5,2)$

$= 28 + 4 + 6$

$= 38$

consider path (3,1), 5th row 3rd column to ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 2 & 11 & \infty \\ 0 & 3 & \infty & 1 & \infty \\ 6 & 0 & 1 & \infty & \infty \\ \infty & 6 & 0 & 0 & \infty \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 11 & \infty \\ \infty & 3 & \infty & 1 & \infty \\ 6 & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

row-reduction = $\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 11 & \infty \\ \infty & 2 & \infty & 0 & \infty \\ 6 & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix} \rightarrow 1$

column-reduction :- $\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 11 & \infty \\ \infty & 2 & \infty & 0 & \infty \\ 6 & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$

$r = 1$

$c^{\wedge}(7) = c^{\wedge}(5) + r + A(5,3)$

$= 28 + 1 + 0$

$= 29$

consider path A(4, 1) 5th row, 4th column to ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 2 & \infty & \infty \\ 0 & 3 & \infty & \infty & \infty \\ \infty & \infty - \infty & 1 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

Row-reduction = $\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 1 & \infty & \infty \\ 0 & 03 & \infty & \infty & \infty \\ \infty & 0 & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix} \rightarrow 1$

column 0 ↓

$r = 1$

Rc

$$c\hat{}(8) = A(5,4) + r + c\hat{}(5)$$

$$= 28 + 1 + 0$$

$$= 29$$

step-4 consider $c\hat{}(7)$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 11 & \infty \\ \infty & 2 & \infty & 0 & \infty \\ 6 & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

$(2,1) \rightarrow \infty$

consider path (3, 2), 3rd row, 2th column to ∞.

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 11 & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 6 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

row-reduction =

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 11 & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix} \rightarrow 6$$

column-reduction =

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 6 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

$\downarrow$

11

$r = 11 + 6 = 17$

$$c\hat{}(9) = r + A(3,2) + c\hat{}(7)$$

$$= 17 + 2 + 29$$

$$= 48$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 11 & \infty \\ \infty & 2 & \infty & 0 & \infty \\ 6 & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

$(4,1) \rightarrow \infty$

consider path A(3,4), 3rd row, 4th column to ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

Row-reduction $= \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$

column-reduction $= \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$

$c\hat{}(10) = r + A(3,4) + c\hat{}(7)$

$= 0 + 0 + 29$

$= 29$

5) consider $c\hat{}(10) = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 6 & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$

consider path A(4,2), make $(2,1) \rightarrow \infty$

4th row, 2nd column $\rightarrow \infty$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

$r=0$

No row reduction, column reduction

$c\hat{}(11) = r + A(4,2) + c\hat{}(10)$

$= 0 + 0 + 29$

$\boxed{c\hat{}(11) = 29}$

Optimal path is $1 \to 4 \to 2 \to 5 \to 3 \to 1$ with cost = 28

$$\begin{bmatrix} \infty & 20 & 30 & 10 & 11 \\ 15 & \infty & 16 & 4 & 2 \\ 3 & 5 & \infty & 2 & 4 \\ 19 & 6 & 18 & \infty & 3 \\ 16 & 4 & 7 & 16 & \infty \end{bmatrix}$$

(1) $c\hat{}(1) = 25$

row-reduction =

$$\begin{bmatrix} \infty & 10 & 20 & 0 & 1 \to 10 \\ 13 & \infty & 14 & 2 & 0 \to 2 \\ 1 & 3 & \infty & 0 & 2 \to 2 \\ 16 & 3 & 15 & \infty & 0 \to 3 \\ 12 & 0 & 3 & 12 & \infty \to 4 \end{bmatrix}$$

21

^{x}(column-reduction =

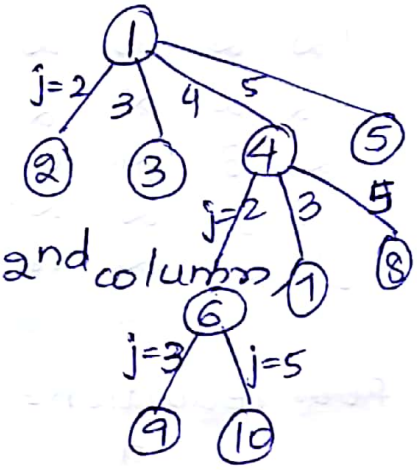
$$\begin{bmatrix} \infty & 16 & 23 & 8 & 9 \\ 12 & \infty & 9 & 2 & 0 \\ 0 & 01 & \infty & 0 & 2 \\ 16 & 2 & 11 & \infty & 1 \\ 13 & 0 & 0 & 14 & \infty \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 3 & 4 & 7 & 2 & 2 \end{bmatrix})x$$

column-reduction:-

$$\begin{bmatrix} \infty & 10 & 17 & 0 & 1 \\ 12 & \infty & 11 & 2 & 0 \\ 0 & 3 & \infty & 0 & 2 \\ 15 & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & 12 & \infty \end{bmatrix}$$

$\downarrow$ 1 $\quad$ $\downarrow$ 3 $\quad$ 4

$c\hat{}(1) = r = 25$



step-1:- consider path A(1,2)

change all entries of 1st row & 2nd column A(2,1) to ∞

$$\begin{bmatrix} \infty & 10 & 17 & 0 & 1 \\ 12 & \infty & 11 & 2 & 0 \\ 0 & 3 & \infty & 0 & 2 \\ 15 & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & 12 & \infty \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & 2 & 0 \\ 0 & \infty & \infty & 0 & 2 \\ 15 & \infty & 12 & \infty & 0 \\ 11 & \infty & 0 & 12 & \infty \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 11 & 2 & 0 \\ 0 & \infty & \infty & 0 & 2 \\ 15 & 0 & 12 & \infty & 0 \\ 11 & \infty & 0 & 12 & \infty \end{bmatrix}$$

$c\hat{}(2) = A(1,2) + r + c\hat{}(1)$

$= 25 + 10 + 0$

$= 35$

consider path A(1,3) ; A(3,1) to ∞

$$\begin{bmatrix} \infty & 10 & 17 & 0 & 1 \\ 12 & \infty & 11 & 2 & 0 \\ 0 & 3 & \infty & 0 & 2 \\ 15 & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & 12 & \infty \end{bmatrix}$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & \infty & 2 & 0 \\ \infty & 3 & \infty & 0 & 2 \\ 15 & 3 & \infty & \infty & 0 \\ 11 & 0 & \infty & 12 & \infty \end{bmatrix}$$

~~Row~~-reduction = column ↗

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 1 & \infty & \infty & 2 & 0 \\ \infty & 3 & \infty & 0 & 2 \\ 4 & 3 & \infty & \infty & 0 \\ 0 & 0 & \infty & 12 & \infty \end{bmatrix}$$

↓

11

$r = 11$

$c\hat{}(3) = c\hat{}(1) + A(1,3) + r$

$= 25 + 11 + 17$

$= 53$

1
46
17
3

consider path A(1,4)

$$\begin{bmatrix} \infty & 10 & 17 & 0 & 1 \\ 12 & \infty & 11 & 2 & 0 \\ 0 & 3 & \infty & 0 & 2 \\ 15 & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & 12 & \infty \end{bmatrix}$$

A(4,1) to ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & \infty & 0 \\ 0 & 3 & \infty & \infty & 2 \\ \infty & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & \infty & \infty \end{bmatrix}$$

$r=0$

$c\hat{}(4) = c\hat{}(1) + r + A(1,4)$

$= 25+0+0$

$= 25$

consider path A(1,5)

$$\begin{bmatrix} \infty & 10 & 17 & 0 & 1 \\ 12 & \infty & 11 & 2 & 0 \\ 0 & 3 & \infty & 0 & 2 \\ 15 & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & 12 & \infty \end{bmatrix}$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & 2 & \infty \\ 0 & 3 & \infty & 0 & \infty \\ 15 & 3 & 12 & \infty & \infty \\ \infty & 0 & 0 & 12 & \infty \end{bmatrix} \begin{matrix} \\ \\ \rightarrow 2 \\ \\ \rightarrow 3 \\ \underline{5} \end{matrix} \Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 10 & \infty & 9 & 0 & \infty \\ 0 & 3 & \infty & 0 & \infty \\ 12 & 0 & 9 & \infty & \infty \\ \infty & 0 & 0 & 12 & \infty \end{bmatrix}$$

$r=5$

$c\hat{}(5) = 25 + A(1,5) + r$

$= 25+1+5$

$= 31$

step-3 :- $i=4, j=2$

A(4,2); A(2,1) → ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & \infty & \infty \\ 0 & 3 & \infty & \infty & 2 \\ \infty & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & \infty & \infty \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 11 & \infty & 0 \\ 0 & \infty & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty \\ 11 & \infty & 0 & \infty & \infty \end{bmatrix}$$

$r=0$

$$c\hat{}(6) = c\hat{}(4) + A(4,2) + r$$

$$= 25+3+0$$

$$= 28$$

$i=4, j=3$

$A(3,1) \rightarrow \infty$, consider path $A(4,3)$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & \infty & 0 \\ 0 & 3 & \infty & \infty & 2 \\ \infty & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & \infty & \infty \end{bmatrix}$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & \infty & \infty & 0 \\ \infty & 3 & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty \\ 11 & 0 & \infty & \infty & \infty \end{bmatrix}$$

Row and column reduction $= \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & \infty & \infty & 0 \\ \infty & 1 & \infty & \infty & 0 \\ \infty & \infty & \infty & \infty & \infty \\ 11 & 0 & \infty & \infty & \infty \end{bmatrix} \rightarrow 2$

$$CR=\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & \infty & \infty & 0 \\ \infty & 1 & \infty & \infty & 0 \\ \infty & \infty & \infty & \infty & \infty \\ 11 & 0 & \infty & \infty & \infty \end{bmatrix} \longrightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 1 & \infty & \infty & \infty & 0 \\ \infty & 1 & \infty & \infty & 0 \\ \infty & \infty & \infty & \infty & \infty \\ 0 & 0 & \infty & \infty & \infty \end{bmatrix}$$

$\downarrow$

11

$r=13$

$c\hat{}(7) = 25+13+A(4,3)$

$= 25+13+12$

$=50$

$i=4, j=5$

$A(5,1) \rightarrow \infty$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 11 & \infty & 0 \\ \infty & 3 & \infty & \infty & 2 \\ \infty & 3 & 12 & \infty & 0 \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix} \Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 11 & \infty & 0 \\ \infty & 1 & \infty & \infty & 0 \\ \infty & 3 & 12 & \infty & 0 \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix} \rightarrow 2$$

Row-reduction $= \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & 0 \\ \infty & 0 & \infty & \infty & 0 \\ \infty & 2 & 1 & \infty & 0 \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix} \rightarrow 1$

3 11

14

$r=11$

~~$r=15$~~

$c\hat{}(8) = 15+25+A(4,5)$

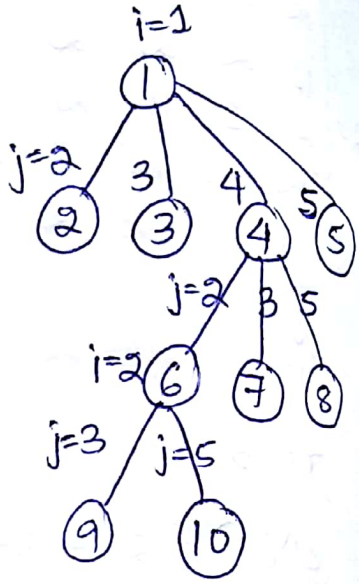
$= 40+A(4,5)$

$= 40-4=36$

$c\hat{}(8)=36$

step-4 :- i=2, j=3, A(3,1) → ∞

$$c^{\wedge}(9) = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty \\ 11 & \infty & \infty & \infty & \infty \end{bmatrix}$$



row-reduction :-

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 0 \\ \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \end{bmatrix} \begin{matrix} \\ \\ \rightarrow 2 \\ \\ \rightarrow 11 \end{matrix}$$

13

column-reduction :-

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 0 \\ \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \end{bmatrix}$$

r = 13

$$c^{\wedge}(9) = c^{\wedge}(6) + A(2,3) + r$$

$$= 28 + 11 + 13$$

$$= 52$$

step-5 :- i=2, j=5 → ∞

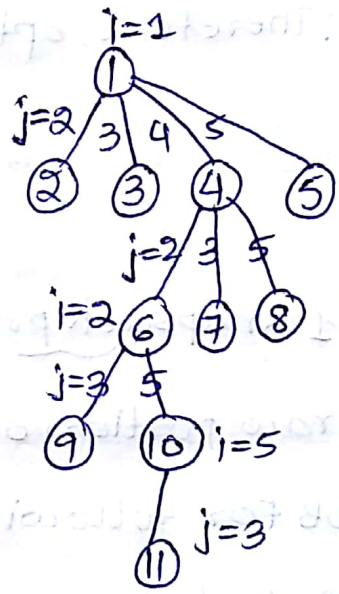
A(5,1) → ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty \end{bmatrix}$$

row-reduction :-

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty \end{bmatrix} \quad r=0$$

$$c\hat{}(10) = c\hat{}(6) + A(2,5) + r$$
$$= 28 + 0 + 0$$
$$= 28$$



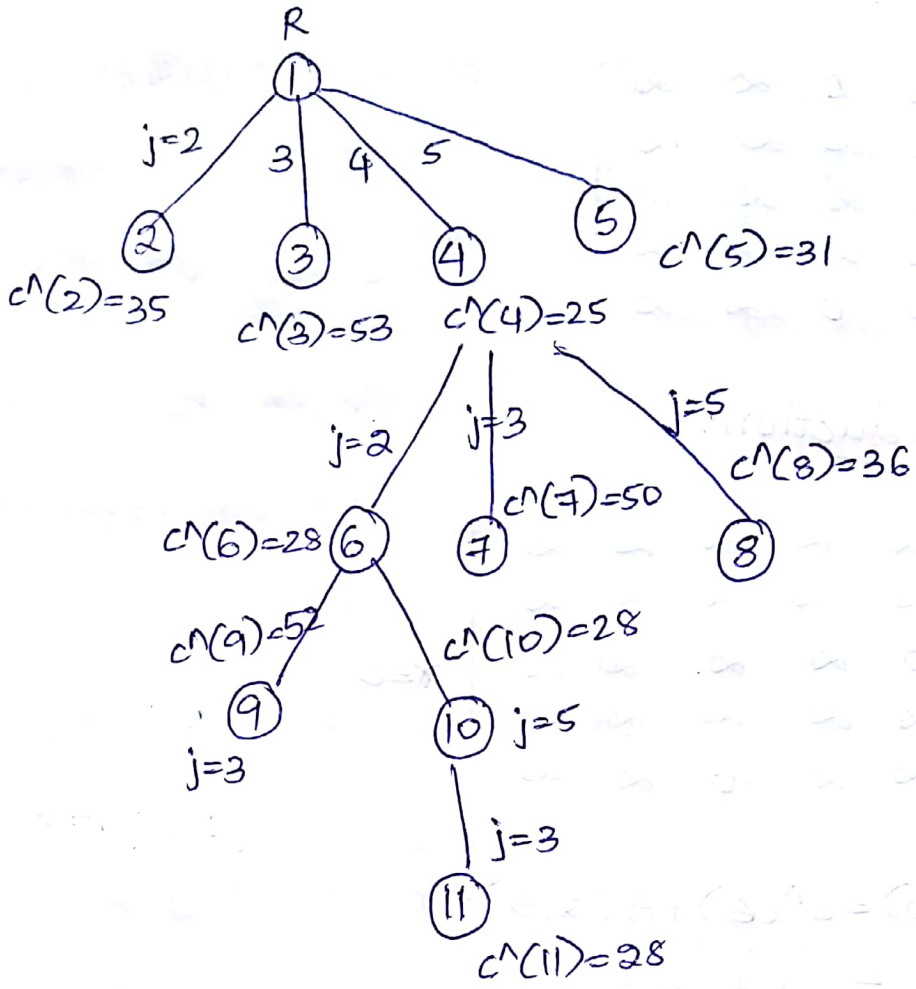
step-5 :- 5th row & 3rd col $\rightarrow \infty$

$A(3,1) \rightarrow \infty$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

$$c\hat{}(11) = c\hat{}(10) + A(5,3) + r$$
$$= 28 + 0 + 0$$
$$= 28$$

$$\boxed{c\hat{}(11) = 28}$$



∴ Therefore, optimal tour is 1, 4, 2, 5, 3, 1
10 6 2 7 3

$= 10+6+2+7+3 = 28$

0/1 knapsack problem :-

Draw portion of state space tree generated by LCBB for following knapsack instance :-

$n=4,\ t=\{10,10,12,18\}\ w=\{2,4,6,9\},\ m=15$

Knapsack is a maximization problem but branch & bound is applicable for minimization problem only. To convert problem into minimization, take -ve signs for profits.

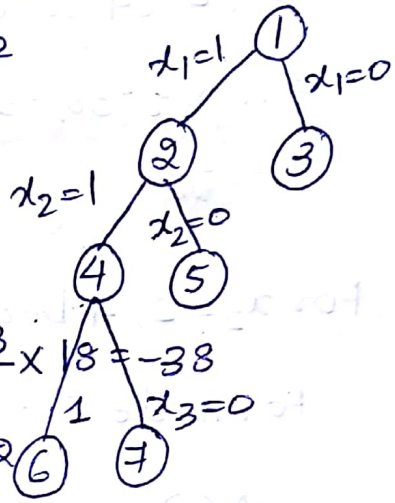
$t=\{-10,-10,-12,-18\}$

Now we calculate lower bound $c\hat{}()$ & upper bound $u()$ for each node.

1. for node 1 :-

$$c\hat{}(1) = -1\times10 - 1\times10 - 1\times12 - \frac{3}{9}\times18 = -38$$

$$u(1) = -1\times10 - 1\times10 - 1\times12 = -32$$



Step-2 :-

For node 2 :- $x_1=1$

$$c\hat{}(2) = -1\times10 - 1\times10 - 1\times12 - \frac{3}{9}\times18 = -38$$

$$u(2) = -1\times10 - 1\times10 - 1\times12 = -32$$

For node 3 :- $x_1=0$

$$c\hat{}(3) = -0\times10 - 1\times10 - 1\times12 - \frac{5}{9}\times18 = -32$$

$$u(3) = -0\times10 - 1\times10 - 1\times12 = -22$$

select lower bound minimum value

$$\min\{c\hat{}(2), c\hat{}(3)\}$$

$$= \min\{-32, -38\}$$

$$= -38$$

For $x_1=1$, now 2 becomes e-node

For node 4 :- $x_2=1, x_1=1$

$$c\hat{}(4) = -1\times10 - 1\times10 - 1\times12 - \frac{3}{9}\times18 = -38$$

$$u(4) = -1\times10 - 1\times10 - 1\times12 = -32$$

For node 5 :- $x_1=1, x_2=0$

$c\hat{}(5) = -1\times10 - 0\times10 - 1\times12 - \frac{7}{9}\times18 = -36$

$u(5) = -1\times10 - 1\times12 = -22$

$\min\{c\hat{}(4), c\hat{}(5)\}$

$= \min\{-38, -36\}$

$= -38$

For $x_2=1$, 4 becomes e-node.

For node 6 :- $x_1=1, x_2=1, x_3=1$

$c\hat{}(6) = -1\times10 - 1\times10 - 1\times12 - \frac{3}{9}\times18 = -38$

$u(6) = -32$

For node 7 :- $x_1=1, x_2=1, x_3=0$

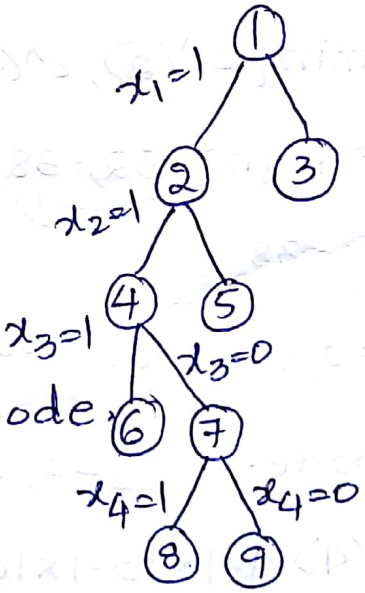
$c\hat{}(7) = -1\times10 - 1\times10 - 0\times12 - 1\times18 = -38$

$u(7) = -38$

$\min\{u(6), u(7)\}$

$= \min\{-32, -38\}$

$= u(7)$



For $x_3=1$, 7 becomes e-node.

For node 8 :-

$x_1=1, x_2=1, x_3=0, x_4=1$

$c\hat{}(8) = -1\times10 - 1\times10 - 0\times12 - \frac{-1}{9}\times18 = -38$

$u(8) = -1\times10 - 1\times10 - 1\times12 = -32$

$x_1 = 1, x_2 = 1, x_3 = 1, x_4 = 0$

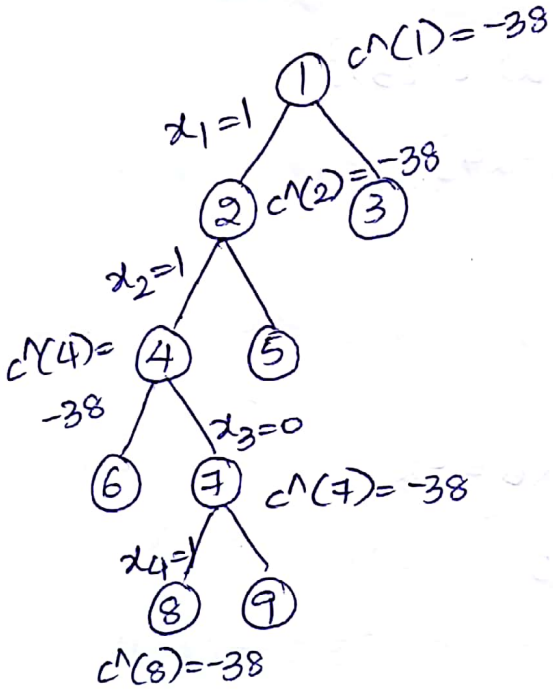
$c\hat{}(9) = -1\times10 - 1\times10 - 0\times12 - 0\times18 = -20$

$u(9) = -20$

select $\min\{c\hat{}(8), c\hat{}(9)\}$

$= \{-38, -20\}$

$= 8$



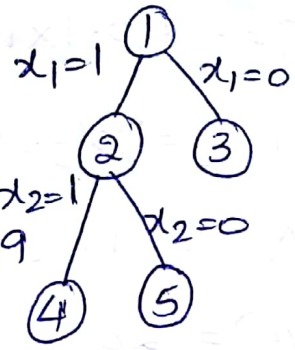
solution for given problem is (1, 1, 0, 1) with maximum profit of $1\times10 + 1\times10 + 0\times12 + 1\times18 = 38$

→ N=5, P={10, 15, 6, 8, 4}, w={4, 6, 3, 4, 2} m=12

1. For node 1 :-

$c\hat{}(1) = -1\times10 - 1\times15 - \frac{2}{3}\times6 = -29$

$u(1) = -1\times10 - 1\times15 = -25$



2:- For node 2,

$x_1 = 1$

$c\hat{}(2) = -1\times10 - 1\times15 - \frac{2}{3}\times6 = -29$

$u(2) = -1\times10 - 1\times15 = -25$

$x_1 = 0$, node 3 :-

$c\hat{}(3) = -0\times10 - 1\times15 - 1\times6 - \frac{3}{4}\times8 = -27$

$u(3) = -0\times10 - 1\times15 - 1\times6 = -21$

3. For node 3, find $\min\{c\hat{}(2), c\hat{}(3)\}$

$= \min\{-29, -27\}$

$= c\hat{}(2)$

· For node 4, $x_1 = 1, x_2 = 1$

$c\hat{}(4) = -1\times10 - 1\times15 - \frac{2}{3}\times6 = -29$

$u(4) = -1\times10 - 1\times15 = -25$

For node 5, $x_1 = 1, x_2 = 0$

$c\hat{}(5) = -1\times10 - 0\times15 - 1\times6\times1\times8 - \frac{1}{2}\times4$

$= -10 - 6 - 8 - 2$

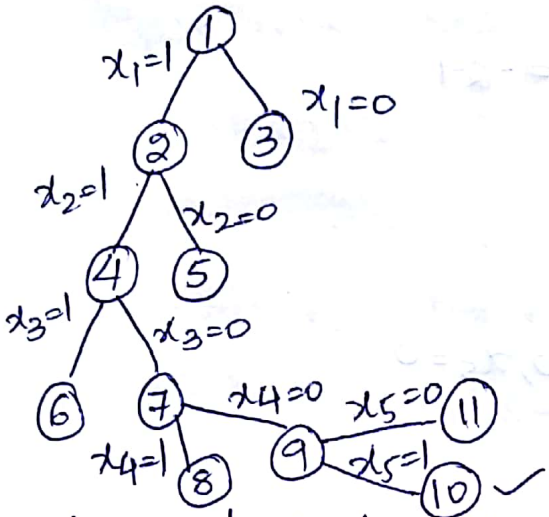
$= -26$

$u(5) = -10 - 6 - 8$

$= -24$

$\min\{c\hat{}(4), c\hat{}(5)\} = \{-26, -29\}$

$= 4$



For node 6, $x_1=1, x_2=1, x_3=1$

$c\hat{}(6) = -1\times10-1\times15-1\times6$ exceeds 15 $\times$

$u(6) = \times$

For node 7, $x_1=1, x_2=1, x_3=0$

$c\hat{}(7) = -1\times10-1\times15-0\times6-\frac{2}{4}\times8 = -29$

$u(7) = -25$

For node 8, $x_1=1, x_2=1, x_3=0, x_4=1$

$c\hat{}(8) = -1\times10-1\times15-\frac{2}{4}\times8 = -29$

$u(8) = -25$

For node 9, $x_1=1, x_2=1, x_3=0, x_4=0$

$c\hat{}(9) = -1\times10-1\times15-1\times\underset{\times}{2}\underset{2}{} = -27-2 = -29$

$u(9) = -27-2 = -29$

$\min\{c\hat{}(8), c\hat{}(9)\} = \{-29, -29\}$

$= 9$ (comparing $u(8), u(9)$)

For node 10, $x_1=1, x_2=1, x_3=0, x_4=0, x_5=1$

$c\hat{}(10) = -1\times10 - 1\times15 - 1\times4 = -29$

$u(10) = -29$

For node 11,

$x_1=1, x_2=1, x_3=0, x_4=0, x_5=0$

$c\hat{}(11) = -1\times10 - 1\times15 \times$

$u(11) = \times$ can't be expanded

The soln is $(1,1,0,0,1)$ with a profit of $1\times10+1\times15 +0\times6+0\times8+1\times4 = 29.$

→ $n=7, m=15, P=\{10,5,15,7,6,18,3\}$

$w=\{2,3,5,7,1,4,1\}$

For node I :-

$c\hat{}(1) = -1\times10 - 1\times5 - 1\times15 - \frac{5}{7}\times7 = -35$

$u(1) = -1\times10 - 1\times5 - 1\times15 = -30$

For node 2,

$x_1=1$

$c\hat{}(2) = -1\times10 - 1\times5 - 1\times15 - \frac{5}{7}\times7 = -35$

$u(2) = -30$

For node 3,

$x_1=0$

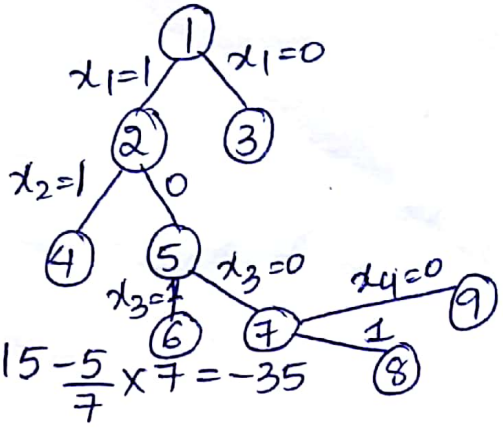
$c\hat{}(3) = -0\times10 - 1\times5 - 1\times15 - 1\times7 = -27$

$u(3) = -27$

$\min\{c\hat{}(2), c\hat{}(3)\}$

$= \{-35, -27\}$

$= 2$



For node 4, $x_1=1, x_2=1$

$c\hat{}(4) = -1\times10 - 1\times5 - 1\times15 - \frac{5}{7}\times7 = -35$

$u(4) = -30$

For node 5, $x_1=1, x_2=0$

$c\hat{}(5) = -1\times10 - 0\times5 - 1\times15 - 1\times7 - 1\times6 = -38$

$u(5) = -38$

$\min\{c\hat{}(4), c\hat{}(5)\}$

$= \min\{-35, -38\}$

$= 5$

For node 6, $x_1=1, x_2=0, x_3=1$

$c\hat{}(6) = -1\times10 - 0\times5 - 1\times15 - 1\times7 - 1\times6 = -38$

$u(6) = -38$

For node 7, $x_1=1, x_2=0, x_3=0$

$c\hat{}(7) = -1\times10 - 0\times5 - 0\times15 - 1\times7 - 1\times6 - 1\times18 + -1\times3$

$= -44$

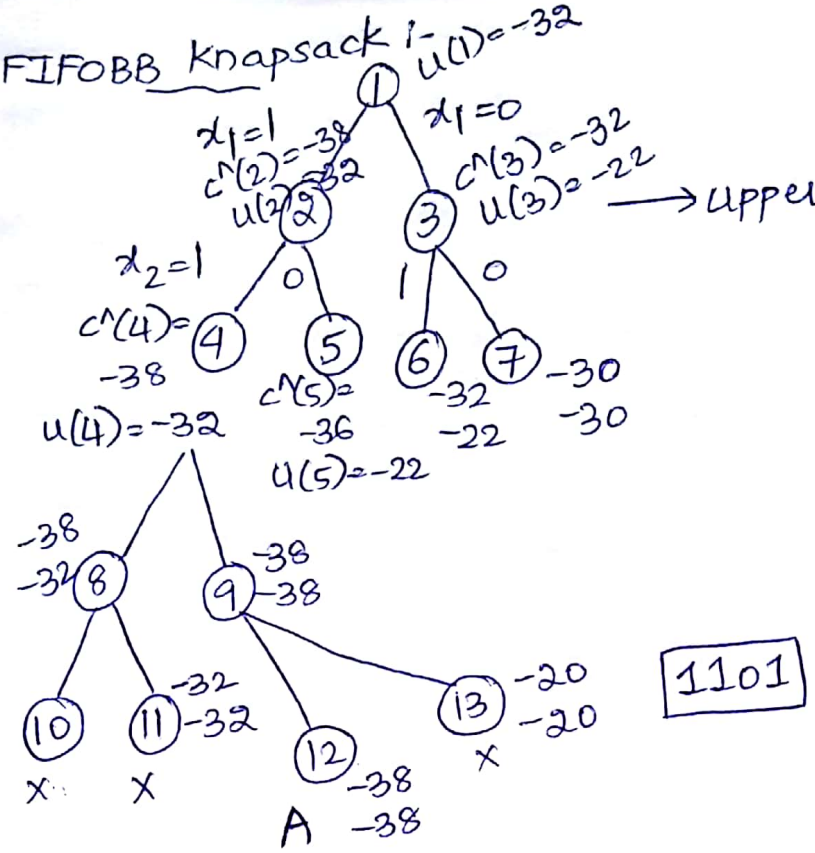
$u(7) = -44$

$\min\{c\hat{}(6), c\hat{}(7)\} = 7$

For node 8, $x_1=1, x_2=0, x_3=0, x_4=1$

$c\hat{}(8) = -1\times10 - 0\times5 - 0\times15 - 1\times7 - 1\times6 - 1\times18 - 1\times3$

FIFOBB knapsack :-



optimal soln is $x_1=1, x_2=1, x_3=0, x_4=1$ where profit = 38.

$c^(6) = x_1=0, x_2=1$

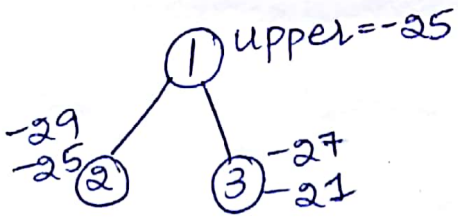
$c^(6) = -0\times10-1\times10-1\times12$

→ $m=12, N=5, w=\{4,6,3,4,2\}, P=\{10,15,6,8,4\}$

For node 1:

$c^(1) = -1\times10-1\times15-\frac{2}{3}\times6 = -29$

$u(1) = -25$



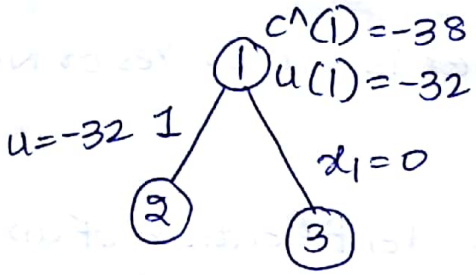
X	2	3	

→ n=4,

$P_1 - - P_4 = 10, 10, 12, 18$

$w_1 - - w_4 = 2, 4, 6, 9$, m=15

$(P_1, P_2, P_3, P_4) = (-10, -10, -12, -18)$



c^(2) < -32 can be expanded

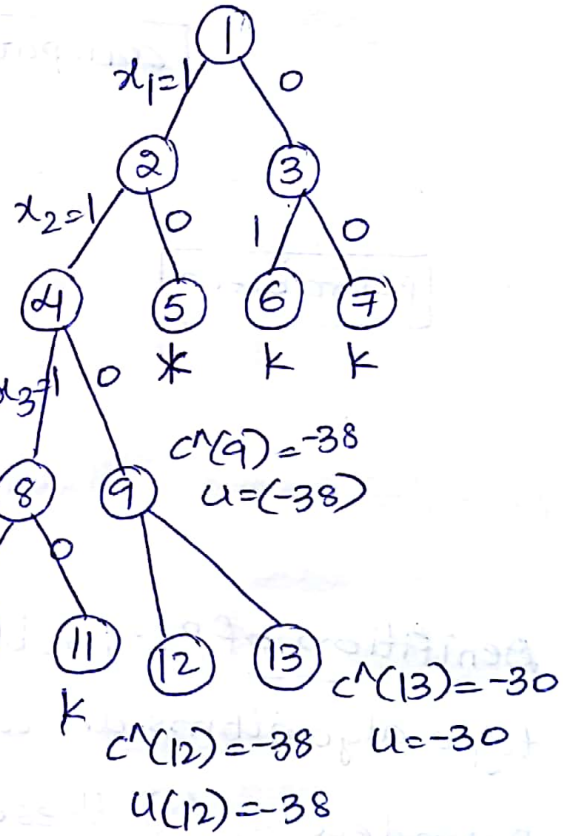
upper = -32 = -32 (not change)

$c\hat{}(3) = -0\times10 - 1\times10 - 1\times12$

$\frac{-5}{9}\times18$

$= -32$

$u(3) = -32$



c^(6) = -32 u(6) = -22

c^(7) = -30 u(7) = -30

c^(12) = -38 u(12) = -38

c^(13) = -30 u(13) = -30

The solution for given problem is

$\{x_1, x_2, x_3, x_4\} = \{1, 1, 0, 1\}$ with a maximum profit of $1\times10 + 1\times10 + 0\times12 + 1\times18 + 0\times3$

$= 38$

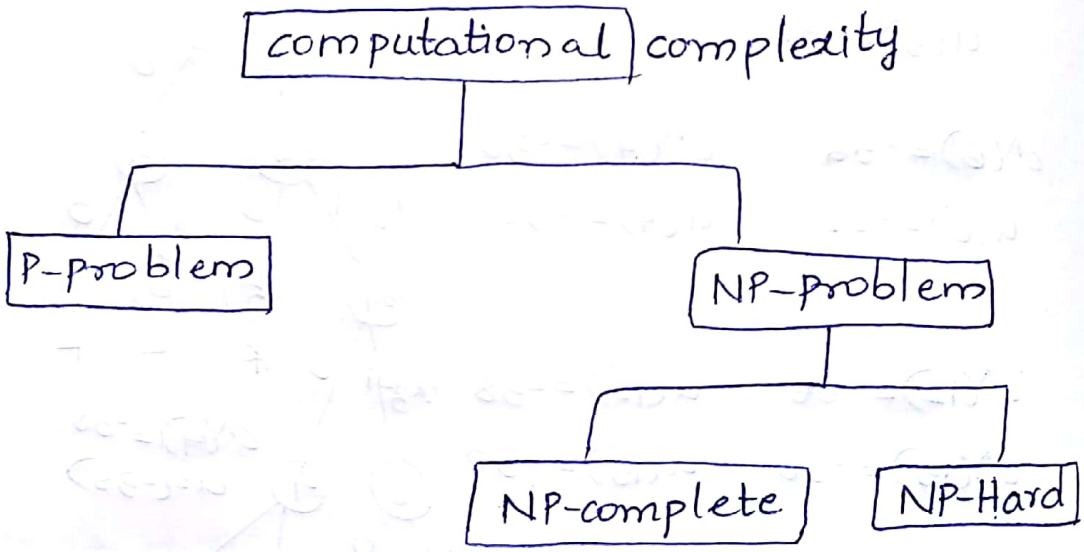
UNIT-V

Q) Different types of NP Problem?

Any problem for which answer is either Yes or No is called decision problem.

→ Any problem that includes identification of an optimal cost (min (or) max) is called optimization problem.

There are 3 different types of problems.



Definition of P:- problems that have polynomial type algorithms are called p-problems.

Bounded by (1) these problems are with sol^{n}s are bounded by polynomials of small degree.

Eg:- T.C $n, \log n, n^2, 2^n$ these are small degree

linear search : T.C = O(n)

Binary " T.C = O(log n)

Bubble short : T.C = O(n^2)

Definition of NP :- (NP stands for Non-deterministic polynomial function)

↓ cannot say T.C

These are problems solved in NP type :-

NP-complete problem :-

A problem is said to be NP-complete iff following conditions are satisfied :-

(1) problem should be present in class NP.

(2) Each problem should also be solved in polynomial type ($n \log n$, n^2 etc)

Eg :- Travelling sales person

↓

$T.C = (O(n^2 2^n))$

Eg :- NP-complete is (mergesort) $T.C = O(n \log n)$

$\therefore P \subseteq NP$

NP-Hard :- A problem is said to be NP-Hard iff it is solved in polynomial type (or) reduced to an NP-Hard problem.

Eg :- knapsack $T.C = O(2^{n/2})$

Any NP-complete problem can be considered as NP-Hard problem, but NP-hard problems cannot

be considered as NP-complete problem.
Generally NP-Hard problem is not solvable.

2) Diff b/w NP-Hard & NP-complete.

NP-Hard	NP-complete
1) Problem P_i is harder than any other proble-ms in Np.	1) problem P_i is most compled of all prob-lems.
2) symbiolically, it is represented as $P_j \rightarrow NP$, Pj is polynomial type not equal to P_i	2) symbollically it is represented as $P_i \rightarrow NP$
3) A decision problem is said to be NP-Hard if every problem in NP -Hard is reduced to Pj in polynomial type.	3) A decision problem P_i is said to be NP-complete if it is NP-complete & solvable in polynomial type P_i.

3) ★★★ write a short notes on Non-deterministic algo-rithm?

→ Non-deterministic algorithm is algorithm in which each operation generates 1 (or) more results i.e, algorithm donot produce unique results.

These algorithms contain 2 stages

stage 1 :-

Non-deterministic stage (Guessing stage)

stage 2 :-

Deterministic stage (verification stage)

Non-deterministic algorithm can be specified using following 3 functions :

(i) choice (s) :- This function is used to select an element randomly from given sets.

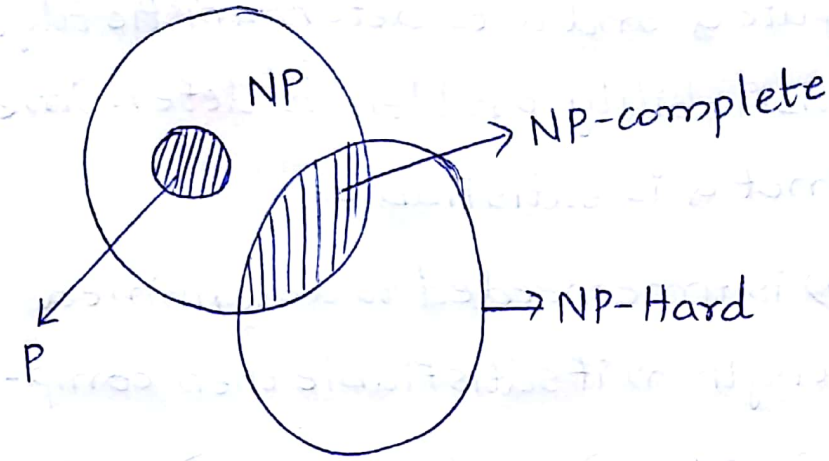
ii) failure () :- This function is used to denote an unsuccessful completion.

iii) success() :- It is used to denote a successful completion.

Non-deterministic searching algorithm :-

```
Algorithm search (e)
{
 j := choice (1,*);
 If (M[j] = e)
then
 {
  output (j); # success ();
 }
 else {
```

```
output(o); #failure();
}
}
Algorithm sortascend (N,X)
{
//'X' is no.of elements in array 'N'.
 for i:=1 to x
  M[i]=0;
 {
   j:=choice(1,x);
 if (M[j]!=0)
 then failure();
 else
 M[j]=N[i];
 }
 for i:=1 to x-1
 {
  if (M[i] > M[i+1])
  then failure();
 else
   write (M[1:x]);
   success();
 }
}
```



Cook's theorm :-

→ cook's theorm states that satisfability is in P if P=NP.

→ we already know that satisfiability is in NP, hence if P=NP, then satisfiability is in P.

i.e, to show that if satisfiability is in P then P=NP.

→ In order to prove that statement, we show how to obtain any polynomial time non-deterministic algorithm A & Input I, a formula $Q(A,I) \Rightarrow Q$ ($Q(A,I) \ni Q$) is satisfiable, if i/p I has successful termination with 'A'.

→ If length of I is n & time complexity of A is $P(n)$ for some polynomial P, then length of 'Q' will be $O(P^3(n)\log n) = O(P^4(n))$.

→ Time needed to construct 'Q' will also be same.

→ The deterministic algorithm to determine outcome of A on any i/p I may be obtained :-

→ If we compute is' and uses deterministic algori-thm for a satisfiability problem to determine whether (or) not is satisfiable.

→ If $O(g(m))$ is time needed to determine a formula of length m if satisfiable then comp-lexity is $O(p^3(n)\log n) + O(p^3(n)\log n)$.

1) write difference between divide & conquer and greedy approach.

Divide & conquer	Greedy approach
→ It is result-oriented.	In greedy method, there are some chances of getting optimal soln to specific problem.
This approach donot depend on constraints, to solve a specific pro-blem.	This approach cannot make furthur move if subset user donot satis-fy specific subset. constraint.
This approach is not effective for larger pro-blems.	This approach is applicable as well as efficient for a wide variety of problems.

This approach is not applicable to problems which are not divisible.

Time taken by this algo-rithm is efficient compa-red to greedy approach.

As problems divided into a large number of subpro-blems space req's is very high.

Applications :-

Merge sort, binary search, quick sort

This problem is rectified in greedy method.

Time taken by this algo-rithm is not that much efficient when compared to divide and conquer approa-ch

→ Space req's is less when compared to divide & conquer.

Applications :- Job sequence

* Minimum cost spanning tree
* Dijkstra's shortest path

→ Difference b/w Dynamic programming and greedy approach.

→ In greedy method only one decision sequence is generated, whereas in dyanamic programming many decision sequences are generated.

→ Greedy method is straight-forward method for choosing optimal s/n, it selects optimal s/n without revising previous false solutions.

whereas dynamic program considers all possible sequences in order to obtain optimal soln.

→ Greedy approach is top-down & dynamic progr-amming is bottom-up approach.

→ Both soln's lead to optimal soln.

Difference b/w Backtracking & brute approach?

→ In case of brute force approach we'll evaluate all possible soln's among which one soln is selected as optimal soln.

→ In Backtracking technique we'll get same soln with less no. of steps in efficient way.

→ In this method, we use bounding function, implicit & Explicit constraints.

→ Major advantage of back-tracking is if a partial soln $x_1, x_2, x_3 --- x_i$ can't lead to optimal soln $x_{i+1} + x_{i+2} --- x_n$ can be discarded.

Back-tracking	Branch & Bound
In back-tracking soln is obtained using bfs, dfs (d-search).	In this technique any of search methods FIFO, LIFO, LC search can be used to obtain soln.
This approach provides soln to decision problem.	This technique is used to solve optimization problem.
A statespacetree is not generated completely, instead process of searc-hing terminates as soon as soln is obtained.	statespace tree is gene--rated using branch & bound method is expand--ed completely, there is a possibility of obtaining optimal soln at any point in statespace tree.
Back-tracking is applica--ble for subsets, hamilto--nian cycle, N-Queens, Graph colouring.	Branch & bound is appli--cable for optimization problems like travelling sales person, 0/1 knapsa--ck problem.

Matrix chain multiplication :-

```
Algorithm Matrix_chain_mul (array P[1:n])
{
for i:=1 to n do
  m[i,i]:=0;
for len:=1 to n-1 do
 {
 for i:=1 to (n-len) do
 {
  j:=i+len;
 m[i,j]:=∞;
for k:=i to j-1 do
{
 q:= m[i,k]+m[k+1,j]+p[i]*p[k+1]*p[j+1];
if (q<m[i,j])
  {
  m[i,j]:=q;
  s[i,j]:=k;
 }
}
}
}}
return m[1,n];
}
Algorithm mul (i,j)*
{
if (i==j) then
```

```
return A[i];
}
else{
        k:=s[i,j];
      p:=mul(i,k);
      q:=mul(k+1,j);
   return p*q;
     }
   }
```