DATE : / /

PAGE NO. :

Branch & Bound :—

Travelling Salesman Problem :-

P.D. ⇒ Given a graph G with cost function $c[i,j]$ defined for each edge (i,j) in G, we want to find the minimum cost tour that visit every vertex in G and return to the starting vertex.

Algo ⇒

1. At any point we have a partial path P and starting at vertex i, we consider only those vertex which are not in P for argumenting the path. This is the branching step.
2. A path has 'dead end' if the vertices not in P are disconnected in $G - \{e\}$ this is used to stop branching.
3. A lower bound

$$l(j) = l(i) + R\,c[i,j] + r$$

Where $l(i)$ is lower bound of parent ~~mode~~ node i.

$l(j)$ = is lower bound of current node

$R\,c[i,j]$ = cost matrix

r = factor

Q.

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & \infty & 12 & 5 & 7 \\ B & 11 & \infty & 13 & 6 \\ C & 4 & 9 & \infty & 18 \\ D & 10 & 3 & 2 & \infty \end{array}$$

Ans –

$$\begin{array}{c|cccc|c} & A & B & C & D & \\ \hline A & \infty & 7 & 0 & 2 & 5 \\ B & 5 & \infty & 7 & 0 & 6 \\ C & 0 & 5 & \infty & 14 & 4 \\ D & 8 & 1 & 0 & \infty & 2 \end{array}$$

[rows deletion]

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & \infty & 6 & 0 & 2 \\ B & 5 & \infty & 7 & 0 \\ C & 0 & 4 & \infty & 14 \\ D & 8 & 0 & 0 & \infty \\ & & 1 & & \end{array} \quad 18$$

[Column deletion]

This is Reduced Cost matrix

To find RCM this is done by selecting the smallest element from each row & column & substracting it from all other element.

Step-3 A be the root node of tree
If it is not a leaf node then find reduced cost matrix

(i) If A is the root node/parent node and B is child node then edge b/w the two node as (A, B) [change all entries in row A and column B of the reduced cost matrix to ∞].

(ii) Set (row B, column 1) to ∞

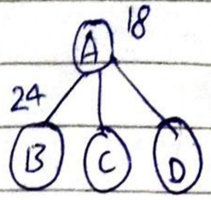
(iii) Reduced all rows and column in the resulting matrix except the rows and columns containing only ∞.

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & \infty & \infty & \infty & \infty \\ B & \infty & \infty & 7 & 0 \\ C & 0 & \infty & \infty & 14 \\ D & 8 & \infty & 0 & \infty \end{array}$$

$l(i) = 18$, $R(A,B) = 6$, $r = 0$

$B = l(i) + R(A,B) + r$

$B = 18 + 6 + 0 = 24$



Now $A \to C$

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & \infty & \infty & \infty & \infty \\ B & 5 & \infty & \infty & 0 \\ C & \infty & 4 & \infty & 14 \\ D & 8 & 0 & \infty & \infty \end{array}$$

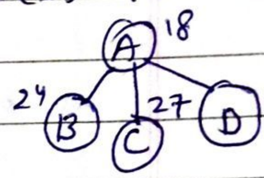
$(A \to C) = l(i) + R_c(A,C) + r$

$$\begin{array}{c|cccc|c} A & \infty & \infty & \infty & \infty & \\ B & 0 & \infty & \infty & 0 & \\ C & \infty & 0 & \infty & 10 & 4 \\ D & 3 & 0 & \infty & \infty & \\ \hline & 5 & & & & \end{array}$$

$r = 5 + 4 = 9$, $l(i) = 18$

$R_c(A,C) = 0$

$l(j) = 18 + 9 + 0 = 27$



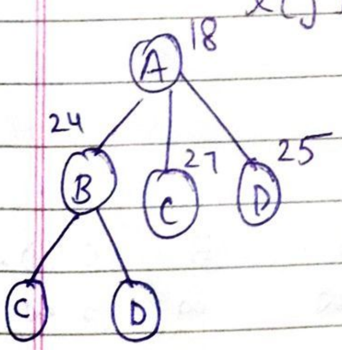
Now for $A \to D$

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & \infty & \infty & \infty & \infty \\ B & 5 & \infty & 7 & \infty \\ C & 0 & 4 & \infty & \infty \\ D & \infty & 0 & 0 & \infty \end{array} \Rightarrow \begin{array}{c|cccc|c} & A & B & C & D & \\ \hline A & \infty & \infty & \infty & \infty & \\ B & 0 & \infty & 2 & \infty & 5 \\ C & 0 & 4 & \infty & \infty & \\ D & \infty & 0 & 0 & \infty & \end{array}$$

$r = 5$

$R_c(A,D) = 2$

$l(j) = 18 + 2 + 5 = 25$



Now $A \to B \to C$

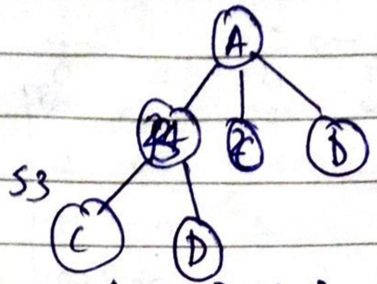
$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & \infty & \infty & \infty & \infty \\ B & \infty & \infty & \infty & \infty \\ C & \infty & 4 & \infty & 14 \\ D & 8 & 0 & \infty & \infty \end{array}$$

$$\begin{array}{c|cccc|c} & A & B & C & D & \\ \hline A & \infty & \infty & \infty & \infty & \\ B & \infty & \infty & \infty & \infty & \\ C & \infty & 0 & \infty & 0 & 4 \\ D & 0 & 0 & \infty & \infty & \\ \hline & 8 & & & 10 & \end{array}$$

$r = 4 + 8 + 10 = 22$

$R_c(B,C) = 7$

$l(j) = 24 + 22 + 7$
$= 53$



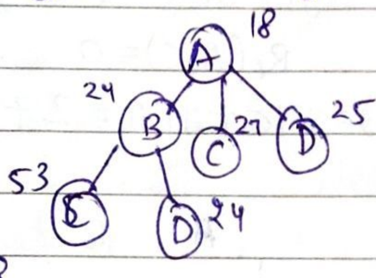
Now for $A \to B \to D$

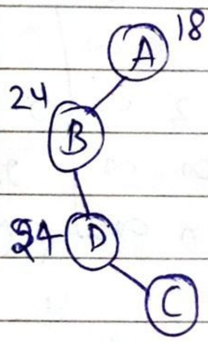
$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & \infty & \infty & \infty & \infty \\ B & \infty & \infty & \infty & \infty \\ C & 0 & 4 & \infty & \infty \\ D & \infty & 0 & 0 & \infty \end{array}$$

$r = 0$

$R_c(B, D) = 0$

$l(j) = 24 + 0 + 0 = 24$





$A \to B \to D \to C$

A rows ∞

B rows ∞

D rows ∞

C columns ∞

$B, 1 \to \infty$

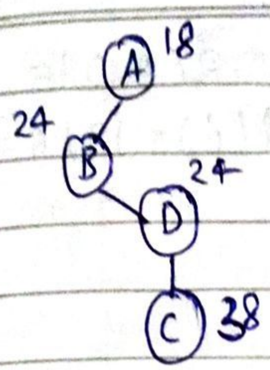
$D, 1 \to \infty$

$C, 1 \to \infty$

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & \infty & \infty & \infty & \infty \\ B & \infty & \infty & \infty & \infty \\ C & \infty & 4 & \infty & 14 \\ D & \infty & \infty & \infty & \infty \end{array} \Rightarrow \begin{array}{c|cccc|c} & A & B & C & D & \\ \hline A & \infty & \infty & \infty & \infty & \\ B & \infty & \infty & \infty & \infty & \\ C & \infty & 0 & \infty & 0 & 4 \\ D & \infty & \infty & \infty & \infty & \\ \hline & & & & 10 & \end{array}$$

$r = 14$, $R_c(D, C) = 0$, $l(i) = 24$

$l(j) = 24 + 0 + 14 = 38$



$A \to B \to D \to C \to A$

~~cost = 18+24+24+38~~

$\text{cost} = 12 + 6 + 2 + 4$

$= 24$ Ans

Q.

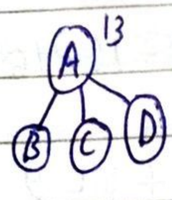
$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & \infty & 4 & 7 & 11 \\ B & 2 & \infty & 5 & 3 \\ C & 9 & 2 & \infty & 6 \\ D & 13 & 1 & 5 & \infty \end{array}$$

$\Rightarrow$

$$\begin{array}{c|cccc|c} & A & B & C & D & \\ \hline A & \infty & 0 & 3 & 7 & 4 \\ B & 0 & \infty & 3 & 1 & 2 \\ C & 7 & 0 & \infty & 4 & 2 \\ D & 12 & 0 & 4 & \infty & 1 \end{array}$$

$$\begin{array}{c|cccc|c} & A & B & C & D & \\ \hline A & \infty & 0 & 0 & 6 & 4 \\ B & 0 & \infty & 0 & 0 & 2 \\ C & 7 & 0 & \infty & 3 & 2 \\ D & 12 & 0 & 1 & \infty & 1 \\ \hline & & & 3 & 1 & 13 \end{array}$$

RCM



$A \to B$

$$\begin{array}{c|cccc|c} & A & B & C & D & \\ \hline A & \infty & \infty & \infty & \infty & \\ B & \infty & \infty & 0 & 0 & \\ C & 0 & \infty & \infty & 0 & 3 \\ D & 7 & \infty & 0 & \infty & 1 \\ \hline & 4 & & & & \end{array}$$

$l(i) = 13$

$r = 8$

$R(A,B) = 0$

$l(j) = 13 + 8 + 0 = 21$

Now $A \to C$

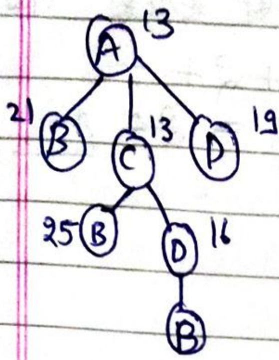
$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & \infty & \infty & \infty & \infty \\ B & 0 & \infty & \infty & 0 \\ C & \infty & 0 & \infty & 3 \\ D & 12 & 0 & \infty & \infty \end{array}$$

$r = 0$ $\quad l(j) = 13$

DATE : / /

PAGE NO. :

$A \to D$



$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & \infty & \infty & \infty & \infty \\ B & 0 & \infty & 0 & \infty \\ C & 7 & 0 & \infty & \infty \\ D & \infty & 0 & 1 & \infty \end{array}$$

$r = 0$ $R_c(A,D) = 6$

$l(j) = 13 + 6 = 19$

$A \to C \to B$

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & \infty & \infty & \infty & \infty \\ B & \infty & \infty & 0 & 0 \\ C & \infty & \infty & \infty & \infty \\ D & 0 & \infty & 0 & \infty \end{array} \; 1$$

11

$r = 12$

$R_c(C,B) = 0$

$l(j) = 13 + 12 + 0 = 25$

$A \to C \to D$

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & \infty & \infty & \infty & \infty \\ B & 0 & \infty & 0 & \infty \\ C & \infty & \infty & \infty & \infty \\ D & \infty & 0 & 1 & \infty \end{array}$$

$r = 0$ $l(j) = 13 + 3 = 16$

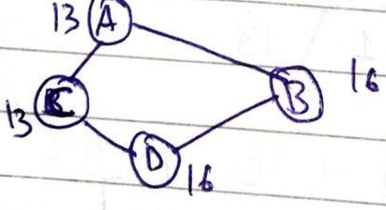
$R_c(C,D) = 3$

$A \to C \to D \to B$

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & \infty & \infty & \infty & \infty \\ B & \infty & \infty & 0 & 0 \\ C & \infty & \infty & \infty & \infty \\ D & \infty & \infty & \infty & \infty \end{array}$$

$r = 0$ $l(j) = 16$

$R_c(D,B) = 0$



~~cost = 13+13+16+16 = 58 Ay~~

$A \to C \to D \to B \to A$ Cost $= 7 + 6 + 1 + 2$

$= 16$ Ans

32
26

DATE : / /

PAGE NO. :

Back Tracking [1950] :— There are problem composed of stages and at every stage we make a decision.

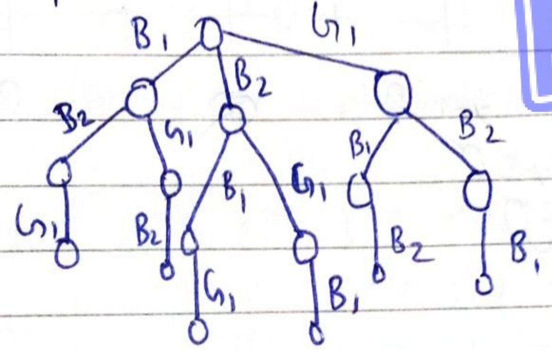
We need to back track and change our previous decision.

Brute Force Approach

B_1 B_2 G_1

$n = 3$ $3! = 6$

state space Tree



~~Branch & Bound~~

N - Queen Problem ⇒

	1	2	3	4
1		Q_1		
2				Q_2
3	Q_3			
4			Q_4	

mirror image
3142

Same
→ row
→ column
→ Diagonal

Q_1 Q_2 Q_3 Q_4

No two Queens are under attack.

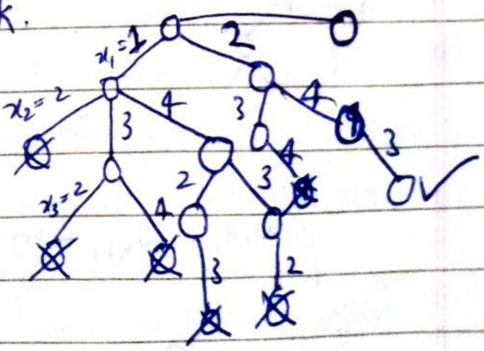
These Queen moves horizontal, vertical & Diagonal.

$^{16}C_4$

So we have to avoid and arrange them as they are not under attack.

x | 2 | 4 | 1 | 3 |
1 2 3 4

column no.



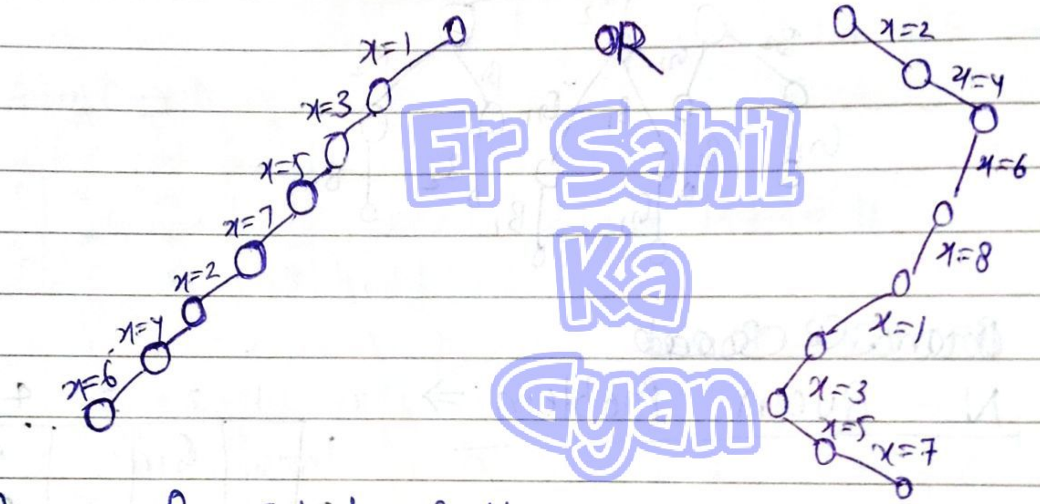
Bound function = row, column, diagonal

$T.C. = O(n!)$

DATE : / /

PAGE NO. :

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



Sum of Subsets Problem:-

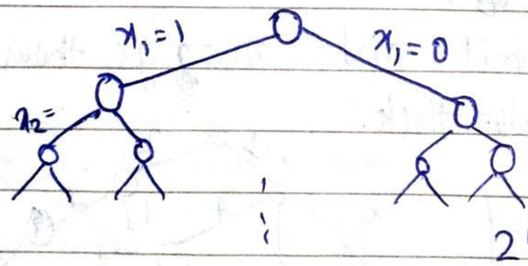
$w[1:6] = \{5, 10, 12, 13, 15, 18\}$

$n=6, \quad m=30$

x | 1 | 1 | 0 | 0 | 1 | 0 |

1 2 3 4 5 6

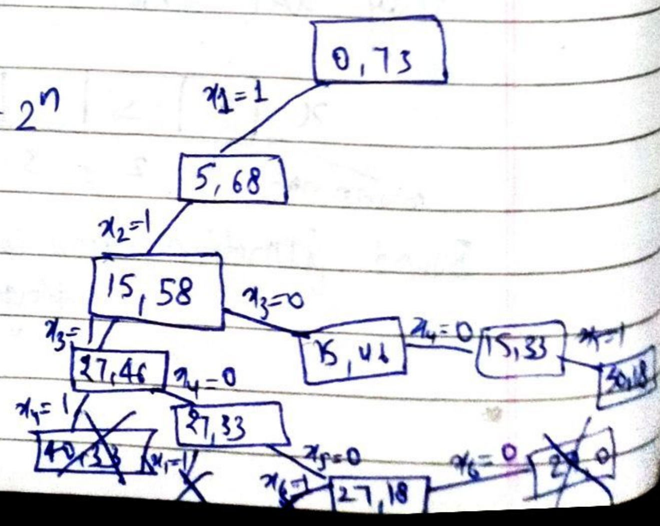
$x_i = 0/1$



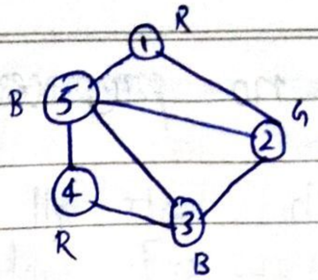
$2^6 = 2^n$

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

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

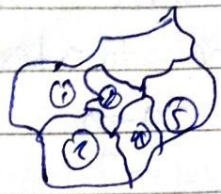


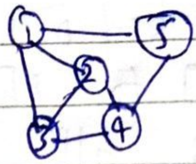
Graph Coloring :—



m=3
R, G, B

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





Pattern Matching Algorithms :— It is the way of comparing two or more text or array of characters. It is also called String Matching. This is a vital class of string algo. is declared as " this is the method to find a place where one is several strings are found within the larger string."

Algo. used for String Matching :— There are diff type of method is used to finding string

(i) Naive String Matching Algo.
(ii) Rabin karp Algo.
(iii) Finite Automata Algo.
(iv) knuth - Morris - Pratt (KMP) Algo.
(v) Boyer - Moore Algo.

(1) Naive String Matching Algorithm :— It generally following two step

(i) Pre - processing of the string
(ii) Then comparison with shifting of pattern

Naive algo. , no preprocessing done

- Naive approach tests all possible placement of Pattern $P[1 ... m]$ relative to text $T[1 ... n]$. We try shift $s = 0, 1, ... n-m$. Compare $T[s+1 ... s+m]$ to $P[1 ... m]$.
- Naive algo finds all valid shift using a loop that checks condition $P[1 ... m] = T[s+1 ... s+m]$ for each of $n-m+1$ possible value of s.

Complexity :-
A & W $\boxed{O((n-m+1)m)}$
B $O(m)$

Algo :-

```
NAIVE-STRING_MATCHER (T, P)
n ← length[T]
m ← length[P]
For s ← 0 to n-m
do if P[1...m] = T[s+1 -- s+m]
then print " Pattern occurs with shift " s.
```

Complexity :- $O(n-m+1)$

(2) **Rabin Karp Algorithm :—** It is string searching algo. created by Michel O. Rabin & Richard M. Karp.

- The idea is to reduce no. of comparisons, by following rules:

(i) If a substring of pattern is of length less than the length of pattern it will never match pattern.

(n) (m)

(ii) The no. value of a substring should be same as of pattern before we actually compare them

26 = 92

This string searching algorithm calculates a hash value for the pattern and for each character subsequence of text of length m to be compared.
If the hash values are unequal the algorithm will calculate the hash value for next m-character sequence.

Algo: –

RABIN-KARP_MATCHER (T, P, d, q) $O((n-m+1)m)$

1. $n \leftarrow length[T]$
2. $m \leftarrow length[P]$
3. $h \leftarrow d^{m-1} \bmod q$
4. $p \leftarrow 0$
5. $t_0 \leftarrow 0$
6. for $i \leftarrow 1$ to m
7. do $p \leftarrow (dp + P[i]) \bmod q$
8. $t_0 \leftarrow (dt_0 + T[i]) \bmod q$
9. For $s \leftarrow 0$ to $n-m$
10. do if $p = t_s$
11. then if $P[1 \dots m] = T[s+1 \dots s+m]$
12. then "Pattern occurs with shift" s
13. if $s < n-m$
14. then $t_{s+1} \leftarrow (d(t_s - T[s+1]h) + T[s+m+1]) \bmod q$
15. Exit

Complexity: –
B&A: – $O(n+m)$
W: – $O(nm)$

Ex – $T = <4 5 7 5 9 1 6 2 3 4 5 6 9 8 7 6 5 5 3 1>$
$P = <2 3 4>$, $q = 11$

1. $n = length[T] = 20$
2. $m = length[P] = 3$

radix $d = 10$, $q = 11$

3. $h = d^{m-1} \bmod q = (10)^{3-1} \bmod 11 = 10^2 \bmod 11$
$= 100 \bmod 11 = 1$, $h = 1$

DATE : / /

PAGE NO. :

Er Sahil Ka Gyan

4. $p = 0$ 5. $t_0 = 0$

6. for $i = 1$ to 3

7. $p = (dp + P[i]) \bmod q$

$p = (d_0 + P[i]) \bmod q$

$p = (234) \bmod 11$

$p = 3$

8. $t_0 = (d_{t_0} + T[i]) \bmod q$

$= (d_0 + T[i]) \bmod q$ similar 2, & 3

$t_0 = (457) \bmod 11$

$t_0 = 6$

$t_0 = 6$ & $p = 3$, $p \neq t_0$

As Pattern is unmatched so

$s < n - m$, yes

$t_{s+1} = (d[t_s - T[s+1]h) + T[s+m+1]) \bmod q$

$t_1 = (10(t_0 - T[1]1) + T[4]) \bmod q$

$t_1 = (10(6-4) + 5) \bmod q$

$t_1 = 25 \bmod 11$

$t_1 = 3$

Go back on step-10

$p = t_s$

$3 = 3$

Match value $234 \neq 575$, which are not equal so pattern is unmatched and shift is applied.

$t_2 = (10[t_1 - T[2]1) + T[5]) \bmod 11$

$(10(3-5) + 9) \bmod 11 = -11 \bmod 11$

$t_2 = 0$, $p = 3$ Not matched

At s=2 Again, $t_3 = 8$, $p = 3$ not matched

At s=3 , $t_4 = 3$, $p = 3$

Now for value $234 \neq 916$, No

At $s=4$, $t_5 = 8$, $p = 3$ not matched
At $s=5$, $t_6 = 7$, $p = 3$ not matched
At $s=6$, $t_7 = 3$, $p = 3$

$p = t_7$

$234 = 234$, which are matched

$\boxed{\text{Shift} = 6}$

Eg - $T = \langle 3\,1\,4\,1\,5\,9\,2\,6\,5\,3\,5\,8\,9\,7\,9\,3 \rangle$

$P = \langle 2\,6 \rangle$, $q = 11$

Ay - 1. $n = \text{length}[T] = 16$

2. $m = \text{length}[P] = 2$

radix $d = 10$, $q = 11$

3. $h = d^{m-1} \bmod q = (10)^{2-1} \bmod 11 = 10$

4. $p = 0$, 5. $t_0 = 0$ 6. for $i = 1$ to 2

7. $p = (dp + P[i]) \bmod q$

$p = (d0 + P[i]) \bmod q$

$p = 26 \bmod 11 = 4 \Rightarrow p = 4$

8. $t_0 = (dt_0 + T[i]) \bmod q$

$t_0 = 31 \bmod 11 = 9$

$p = 4$ & $t_0 = 9$, $p \neq t_0$

so $s < n-m$, yes

$t_{s+1} = (d[t_s - T[s+1]h) + T[s+m+1]) \bmod q$

$t_1 = (10((9-3)\times 10) + 4) \bmod 11$

$t_1 = (640) \bmod 11 = 2$

$t_1 \neq p$

At $s=1$, $t_2 = (10((2-1)10+4) \bmod 11$

$t_2 = 140 \bmod 11 = 8$

$t_2 \neq p$

At $s=2$, $t_3 = (10((8-4)10) + 1) \bmod 11$

$t_3 = 410 \bmod 11 = 3$

$t_3 \neq p$

$t_1 = (10(9 - 3\times10) + 4) \bmod 11$

$(-210 + 4) \bmod 11$

$209 = 11\times19$

$t_1 = -206 \bmod 11$

$t_1 = 3$

$t_1 \neq P$

At $s=1$, $t_2 = (10(3 - 1\times10) + 9) \bmod 11$

$t_2 = -61 \bmod 11 = 8$

$t_2 \neq P$

At $s=2$, $t_3 = (10(8 - 4\times10) + 5) \bmod 11$

$-315 \bmod 11$

$330 = 30\times11$

$330 - 315 = 15$

$t_3 = 15 \bmod 11 = 4$

$t_3 = P$

in value $15 = 26$ Not match

-460

At $s=3$, $t_4 = (10(4 - 1\times10) + 9) \bmod 11$

$t_4 = -51 \bmod 11 = 4$

in Value $59 \neq 26$

42

At $s=4$, $t_5 = (10(4 - 5\times10) + 2) \bmod 11$

$-458 \bmod 11 = 462 - 458$

$= 4 \bmod 11 = 4$

in 92 ≠ 26

At $s=5$, $t_6 = (10(4 - 9\times 10) + 6) \bmod 11$

$t_6 = (-860 + 6) \bmod 11$

$t_6 = -854 \bmod 11$ (78x11)

(858-854) $t_6 = 4 \bmod 11 = 4$

$t_6 = P$

value $26 = 26$ ✓

Shift = 5

19 19 869

Knuth - Morris - Pratt (KMP) Algorithm:— KMP was the first linear time complexity algorithm for string matching. This algo. compares character by character from L to R. It uses Prefix table to skip characters comparison.

48 78 858

Compute-Prefix Algorithm :—

```
Compute-Prefix(p)
m = length[p]
π[1] = 0, k = 0
for q = 2 to m
do while (k>0 & p[k+1] != p[q])
    k = π[k]
if (p[k+1] = p[q])
    k = k+1
    π[q] = k
End for
Return π
```

Eg- P = <aabab>

q	1	2	3	4	5
P[q]	a	a	b	a	b
π[q]	0	1	0	1	0

k: 0 1 2 3 4

q	1	2	3	4	5
P(q)	a	a	b	a	b
π(q)	0	1	0	1	0

DATE : / /

PAGE NO. :

KMP_MATCHER ALGORITHM:—

Complexity:— $O(m+n)$

KMP_Matcher (T, P)

1. n = length[T]
2. m = length[P]
3. π = Compute_Prefix(P)
4. q = 0
5. For i = 1 to n
6. do while (q > 0 and P[q+1] != T[i])
7. q = π(q)
8. if P[q+1] = T[i]
9. q = q + 1
10. if q = m
11. print "pattern occurs with shift" i−m
12. Exit
13. End for

Eg— P = <a a b a b>

T = <a a a b a b a a b a a b a b a a b>

Sol

1. n = length[T] = 17
2. m = length[P] = 5

~~q = 0~~

3. π = Compute_Prefix(P)

q	1	2	3	4	5
p[q]	a	a	b	a	b
π[q]	0	1	0	1	0

4. q = 0
5. for i = 1, q = 0

P[q+1] = P[1] = a

DATE : / /

PAGE NO. :

$T[i] = T[1] = a$

$P[q+1] = T[i]$

$a = a$

then $q = q+1$, $q = 1$

$m = 5$, $q \neq m$

$1 \neq 5$

For $i = 2$, $q = 1$

$P[q+1] = P[2] = a$

$T[i] = T[2] = a$

$q = q+1$, $q = 2$

$m = 5$

$q \neq m$

For $i = 3$,

$P[q+1] = P[3] = b$

$T[i] = T[3] = a$

$P[q+1] \neq T[i]$ and $q > 0$

then $q = \pi(q)$

$q = \pi(2) = 1$ but still $q > 0$

$P[q+1] = P[2] = a$

$T[i] = T[3] = a$

$P[q+1] = T[i]$

then $q = q+1 \Rightarrow 2$

$q \neq m$

For $i = 4$,

$P[q+1] = P[3] = b$

$T[i] = T[4] = b$

$P[q+1] = T[i]$

then $q = q+1 \Rightarrow 3$

$q \neq m$

For $i = 5$, $P[q+1] = P[4] = a$

$T[i] = T[5] = a$

DATE : / /

PAGE NO. :

then $P[q+1] \overset{=}{\cancel{\neq}} T[i]$ ~~& q > 0~~

~~$q = \pi(4) = 1$ still $q > 0$~~

~~$P[q+1] = P[2] = a$~~ $q = q+1 = 4$

$q \neq m$

for $i = 6$

$P[q+1] = P(5) = b$

$T[i] = T[6] = b$

$q = q+1 = 5$, $m = 5$

$q = m$

so pattern occurs with shift 1

~~for i = 7~~ ~~q=0~~

$q = \pi[q] = \pi[5] = 0$

$q = 0$

For $i = 7$, we have to search for another pattern match in the rest of text, $q = 0$

$P[q+1] = P[1] = a$

$T[i] = T[7] = a$

$P[q+1] = T[i]$

$q = q+1 = 1$

$q \neq m$

For $i = 8$, $q = 1$

$P[2] = a$

$T[8] = a$ $P[q+1] = T[i]$

$q = 2$ $q \neq m$

for $i = 9$, $q = 2$

$P[3] = b$

$T[9] = b$

For $i = 17$, —— $q = 3$ $q \neq m$ - -

Q. $T = \langle bbbaababbaabaab \rangle$

$P = \langle baab \rangle$

Sol^n – $n = length[T] = 15$

$m = length[P] = 4$

π = Compute-Prefix (P)

q	1	2	3	4
P(q)	b	a	a	b
π(q)	0	0	0	1

b a a b
0 0 0 1
0 1 2 3

4. $q = 0$

for $i = 1$, $q = 0$

$P[q+1] = P[1] = b$

$T[i] = T[1] = b$

$P[q+1] = T[i]$

$q = q+1 \Rightarrow q = 1$

$q \neq m$

for $i = 2$, $q = 1$

$P[q+1] = a$

$T[2] = b$

$P[q+1] \neq T[i]$

$q = \pi(q)$

$q = 0$

for $i=3$, $q=0$

$P[1] = b$

$T[3] = b$

$q = q+1 = 1$

$q \neq m$

for $i=4$, $q=1$

$P[2] = a$

$T[4] = a$

$q = 2$

$q \neq m$

for $i=5$, $q=2$

$P[3] = a$

$T[5] = a$

$q = 3$

$q \neq m$

for $i=6$, $q=3$

$P[4] = b$

$T[6] = b$

$q=4$ $q=m$ so pattern occurs with shift 2

$q = \pi(q) = 1$

For $i=7$, $q=1$
$P[2] = a$
$T[7] = a$
$q = 2$
$q \neq m$

for $i=8$, $q=2$
$P[3] = a$
$T[8] = b$
$T[i] \neq P[q+1]$
$q = \pi(q) = 0$

For $i=9$, $q=0$
$P[1] = b$
$T[9] = b$
$q = 1$
$q \neq m$

for $i=10$, $q=1$
$P[2] = a$
$T[10] = a$
$q = 2$
$q \neq m$

for $i=11$, $q=2$
$P[3] = a$
$T[11] = a$
$q = 3$
$q \neq m$

for $i=12$, $q=3$
$P[4] = b$
$T[12] = b$
$q = 4$
$q = m$ so shift 8
$q = \pi(4) = 1$

for $i=13$, $q=1$
$P[2] = a$
$T[13] = a$
$q = 2$
$q \neq m$

for $i=14$, $q=2$
$P[3] = a$
$T[14] = a$, $q=3$
$q \neq m$

for $i=15$, $q=3$
$P[4] = b$
$T[15] = b$
$q = 4$
$q = m$
So Shift $= 11$

Ay $\boxed{\text{On Shift} - 2, 8, 11}$

Boyer - Moore Algorithm :- It takes a 'backward' approach : the pattern string (P) is aligned with start of text T and compares the characters of pattern from right to left beginning with righmost character.

- If a character is compared that is not within pattern, no match can be found by analyzing any further aspects at this position so pattern can be changed entirely past the mismatching character.

Two strategies are called heuristics of B-M

(i) Bad Character Heuristics
(ii) Good Suffix Heuristics

(i) **Bad Character Heuristics :-**

(a) There is a character in text in which does not occur in a pattern at all. When mismatch happens at this character, the whole pattern can be changed, begin matching form substring next to this 'bad chara'.

(b) It might be that a bad character is present in the pattern in this case, align nature of pattern with a bad chara. in the text.

```
COMPUTE_LAST_OCCURRENCE_FUNCTION (P, m, Σ)
For each character a ∈ Σ
  do λ [a] = 0
    for j ← 1 to m
    do λ [P[j]] ← j
    Return λ
```

DATE : / /

PAGE NO. :

Good Suffix Heuristics:— A suffix that has matched successfully. After mismatch which has a negative shift in bad character heuristics, look if a substring of pattern matched till bad character has a good suffix in it, if it is so then we have a onward

COMPUTE_GOOD_SUFFIX_FUNCTION (P, m)

```
π ← C_P_F (P)
  P' ← reverse (P)
  π' ← C_P_F (P')
  For j ← 0 to m
    do γ[j] ← m − π[m]
For l ← 1 to m
do j ← m − π'[l]
  If γ[j] ← l − π'[l]
then γ[j] ← l − π'[l].
Return γ
```


BOYER_MOORE_MATCHER Algorithm

BOYER_MOORE_MATCHER (T, P, Σ)

```
n ← length[T]
m ← length[P]
λ ← C_L_O_F (P, m, Σ)
γ ← C_G_S_F (P, m)
  s ← 0
While s ≤ n−m
```

DATE : / /

PAGE NO. :

do $j \leftarrow m$

while $(j > 0$ & $p[j] = T[s+j])$

do $j \leftarrow j-1$

if $j = 0$

then print "Pattern occurs at shift" s.

$s \leftarrow s + \gamma[0]$

Else

$s \leftarrow s + \max(\gamma[j],\ j - \lambda[T[s+j]])$

<u>Complexity:-</u> (i) Last occurrence $O(m + |\Sigma|)$

~~Good~~

BM :- $O(mn)$

Eg- T = WELCOME TO SAHIL EMPIRE

P = SAHIL (0 1 2 3 4)

$n = 20$, $m = 5$

Bad Match Table

S	A	H	I	L	*
4	3	2	1	5	5

value = length - index - 1

S = 5 - 0 - 1 = 5

A = 5 - 1 - 1 = 3

H = 5 - 2 - 1 = 2

I = 5 - 3 - 1 = 1

Step 1 WELCOME TO SAHIL → WELCOME TO SAHIL
SAHIL SAHIL

P present in T at 9^{th} position,

L-4

DATE : / /

PAGE NO. :

Assignment Problem :-

It is a special type of linear programming problem which deals with the allocation of the various resources to the various activities on one to one basis. It does it in such a way that the cost or time involved in process is minimum & profit or sale is maximum. Though there problems can be solved by simplex method or by transportation method.

Quadratic Assignment Problem (QAP) :- It is to assign n facilities to n locations in such a way as to minimize the assignment cost.

Job Assignment Problem (JAP) ⇒

Given n job <J1---JN> and n person $<P_1, P_2 --- P_n>$.
It is required to assign all n jobs to all n person with the ~~constrain~~ constraint that one job has to be assigned to one ~~and~~ person and the cost involved.
Job should be minimized.

Eg-

$$\begin{array}{c|cccc|c} & J_1 & J_2 & J_3 & J_4 & \\ \hline a & 9 & 2 & 7 & 8 & 2 \\ b & 6 & 4 & 3 & 7 & 3 \\ c & 5 & 8 & 1 & 8 & 1 \\ d & 7 & 6 & 9 & 4 & 4 \end{array}$$

Solution :- (LB)

$2+3+1+4 = 10$

Consider person 'a'

Let $a \rightarrow J_1$ where cost is 9

Now $a \rightarrow J_1$, leave row a & column 1

$$\begin{array}{c|cccc|c} & J_1 & J_2 & J_3 & J_4 & \\ \hline a & 9 & \cancel{2} & \cancel{7} & \cancel{8} & 9 \\ b & 6 & 4 & 3 & 7 & 3 \\ c & 5 & 8 & 1 & 8 & 1 \\ d & 7 & 6 & 9 & 4 & 4 \end{array}$$

$LB = 9+3+1+4 = 17$

~~Consider~~

Let $a \rightarrow 2$ where cost $\rightarrow 2$

Now $a \rightarrow 2$, leave row a & column 2

$$\begin{array}{c|cccc|c} & J_1 & J_2 & J_3 & J_4 & \\ \hline a & \cancel{9} & 2 & \cancel{7} & \cancel{8} & 2 \\ b & 6 & 4 & 3 & 7 & 3 \\ c & 5 & 8 & 1 & 8 & 1 \\ d & 7 & 6 & 9 & 4 & 4 \end{array}$$

$LB = 10$

$a \rightarrow 3$, cost $\rightarrow 7$

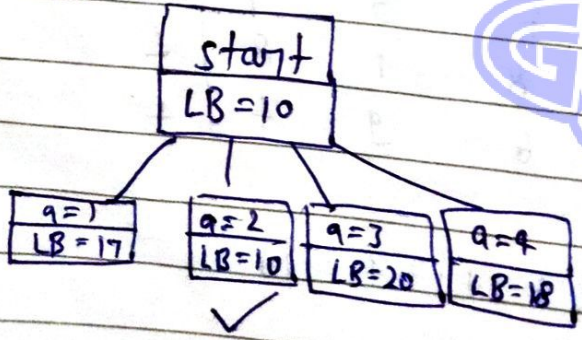
$$\begin{array}{c|cccc|c} a & \cancel{9} & \cancel{2} & 7 & \cancel{8} & 7 \\ b & 6 & 4 & 3 & 7 & 4 \\ c & 5 & 8 & 1 & 8 & 5 \\ d & 7 & 6 & 9 & 4 & 4 \end{array}$$

$LB = 20$

$a \rightarrow 4$, cost $\rightarrow 8$

$$\begin{array}{c|cccc|c} a & \cancel{9} & \cancel{2} & \cancel{7} & 8 & 8 \\ b & 6 & 4 & 3 & 7 & 3 \\ c & 5 & 8 & 1 & 8 & 1 \\ d & 7 & 6 & 9 & 4 & 6 \end{array}$$

$LB = 18$



Consider person 'b'

Let $b \to 1$, cost $\to 6$

$$\begin{array}{c|cccc|c} & J1 & J2 & J3 & J4 & \\ a & 9 & 2 & 7 & 8 & 2 \\ b & 6 & 4 & 3 & 7 & 6 \\ c & 5 & 8 & 1 & 8 & 1 \\ d & 7 & 6 & 9 & 4 & 4 \end{array}$$

LB = 13

Let $b = 3$, cost $\to 3$

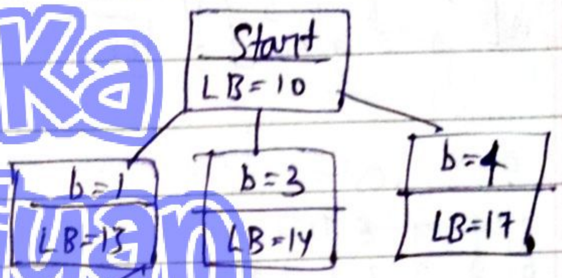
$$\begin{array}{c|cccc|c} & J1 & J2 & J3 & J4 & \\ a & 9 & 2 & 7 & 8 & 2 \\ b & 6 & 4 & 3 & 7 & 3 \\ c & 5 & 8 & 1 & 8 & 5 \\ d & 7 & 6 & 9 & 4 & 4 \end{array}$$

LB = 14

Let $b = 4$, cost $\to 7$

$$\begin{array}{c|cccc|c} & J1 & J2 & J3 & J4 & \\ a & 9 & 2 & 7 & 8 & 2 \\ b & 6 & 4 & 3 & 7 & 7 \\ c & 5 & 8 & 1 & 8 & 1 \\ d & 7 & 6 & 9 & 4 & 7 \end{array}$$

LB = 17



Consider person 'c'

$c = 3$, cost $= 1$

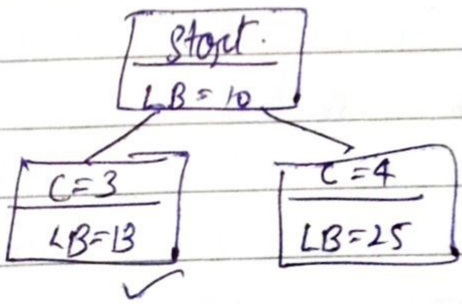
$$\begin{array}{c|cccc|c} & J1 & J2 & J3 & J4 & \\ a & 9 & 2 & 7 & 8 & 2 \\ b & 6 & 4 & 3 & 7 & 6 \\ c & 5 & 8 & 1 & 8 & 1 \\ d & 7 & 6 & 9 & 4 & 4 \end{array}$$

LB = 13

$c = 4$, cost $= 8$

$$\begin{array}{c|cccc|c} & J1 & J2 & J3 & J4 & \\ a & 9 & 2 & 7 & 8 & 2 \\ b & 6 & 4 & 3 & 7 & 6 \\ c & 5 & 8 & 1 & 8 & 8 \\ d & 7 & 6 & 9 & 4 & 9 \end{array}$$

LB = 25



Consider person 'd'

$c = 4$, cost $= 4$

$$\begin{array}{c|cccc|c} a & 9 & 2 & 7 & 8 & 2 \\ b & 6 & 4 & 3 & 7 & 6 \\ c & 5 & 8 & 1 & 8 & 1 \\ d & 7 & 6 & 9 & 4 & 4 \end{array}$$

$LB = 2 + 6 + 1 + 4 = 13$ UB

10-13

DATE : / /

PAGE NO. :

QAP :— It is to assign n facilities to n locations in such a way as to minimize the assignment cost.

→ QAP can thus be written as $A = (a_{ij})$ and the distance matrix $B = (b_{ij})$. The QAP can thus be written as

$$\Pi \in S_n <= \sum_{i=1}^{n} \sum_{j=1}^{n} a_{\Pi}(i)\Pi(j) b_{ij}$$

where

S_n is the set of permutation $a_{\Pi(i)\Pi(j)} b_{ij}$ = individual product cost caused by assigning facility $\Pi(i)$ to locate i and facility $\Pi(j)$ to location j

$$\xrightarrow{} C_{ij} = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

cost

$$x_{ij} = \begin{cases} 1 & , \text{job assigned to} \\ 0 & , \text{otherwise} \end{cases}$$

Randomized Algorithms:— It is defined as an algorithm that is allowed to access a source of independent, unbiased random bit, and it is then allowed to use random bit to influence its computation.

It belong to class of probabilistic algo.

• Need of Randomization ⇒

It uses uniformly random bits an input to achieve good performance in the average case over all possible choice of random bit.

RA can be categorized into two classes

(i) Las Vegas

(ii) Monte Carlo

DATE : / /

PAGE NO. :

Las Vegas Algorithms:–

If it always returns the correct answer, but its runtime bounds hold only in expectation call Las Vegas Algorithm. or inform about a failure

Randomized Quick Sort ⇒

```
Random_Qsort (A, lb, ub)
    if lb < ub
    Pivot = Random (lb, ub)
    p = partition (pivot, lb, ub)
    Random_Qsort (A, lb, p-1)
    Random_Qsort (A, p+1, ub)
    endif
    return
```


→ It uses an inbuild function rand(), which gives a real value b/w 0 & 1, generated by pseudorandom no. generator.

```
RANDOM(x, y)
if x < y
return (x + rand() * (x-y))
```

Eg– we need random no. 2 to 8

$x = 2 + rand() * (8-2)$

Maximum value of rand() is 1

$x = 2 + 6 = 8$

Minimum value of rand() is 0

$x = 2$

DATE : / /

PAGE NO. :

Monte Carlo Algorithm:— It is one whose running time is deterministic, but whose output may be correct only with a certain probability.

Monte carlo methods are a class of computational algorithm that rely on repeated random sampling to computer their result.

It follows particular pattern :—

→ Computations into final result.
→ Generate input randomly from domain.
→ Perform a deterministic computation using inputs.
→ Aggregate the results of individual computations into final result.

⇓

Step-1 Define input randomly from the domain using a certain specific.
Step-2 Generate input randomly from domain
Step-3 Deterministic computation
Step-4 Aggregate the result.

A monte carlo Algorithm for a decision problem has 2 subtype:

(i) One sided Error

↓

Eg- Someone failed? no – 100% Correct
yes – it may not be correct every time

(ii) Two Sided Error

↓

who will do top?

yes Everybody but in mind someone no

DATE : / /

PAGE NO. :

Monte Carlo	Las Vegas
(i) Result of algo. has a probability of being correct.	Result is 100%. correct.
(ii) Can be bounded by resources used. That is, running time can be found.	• Running time is random variable. Hence, resources usage can't be found.
(iii) Error probability is decreased by repetitive time application that is at cost of runtime	• Runtime can't be improved rather randomized can be made efficient

Application of Monte Carlo:—

→ For simulating physical & mathematical systems.

→ Used when it's infeasible or impossible to compute an exact result with deterministic algo.

→ Used for modeling phenomena with significant uncertainity in inputs.

Monte Carlo Algo. components:—

→ Error Estimation

→ Variance Reduction Technique

→ Parallelization & Vectorization

Randomized Algorithm for Min-cut Concept:— It is used to select the edge with mini. weight & according this min. weight edge next move is done in a network graph.

For its better working followed points must be taken into consideration:

A cut of connected graph is obtained by dividing vertex set V of graph G into 2 sets V_1 & V_2.

(i) There are no common vertices in V_1 & V_2, that is, two sets are disjoint.

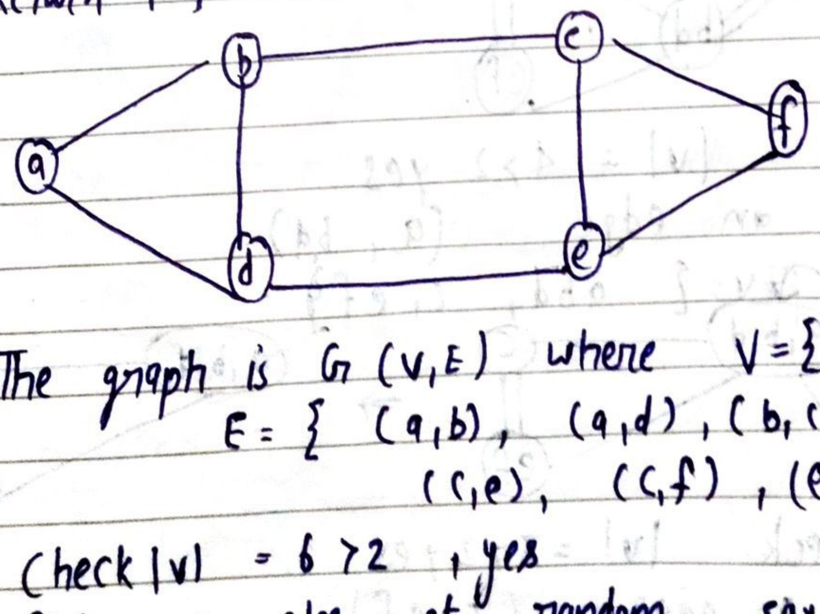
(ii) $V_1 \cup V_2 = V$

Randomized Algorithm for Min-cut Concept:—

Random-Min-Cut (G)

1. Repeat step 2 to 4 until only two vertices are left
2. Pick an edge $e(u, v)$ at random
3. Merge u & v.
4. Remove self loops from E
5. Return $|E|$

Q.



Soln- The graph is $G(V,E)$ where $V = \{a, b, c, d, e, f\}$
$E = \{(a,b), (a,d), (b,c), (b,d), (d,e), (c,e), (c,f), (e,f)\}$

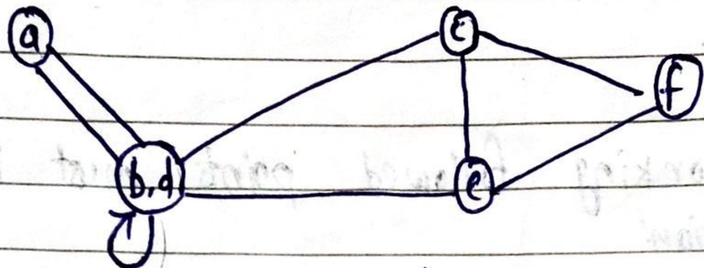
Step-1 Check $|V| = 6 > 2$, yes

Step-2 Pick an edge at random say (b,d)

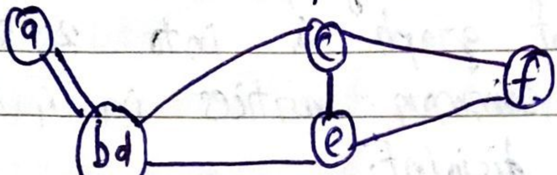
Step-3 Merge b & d. Now we have a new vertex (b,d)

all edge incident on b are incident on this vertex

v = { a, bd, c, e, f }



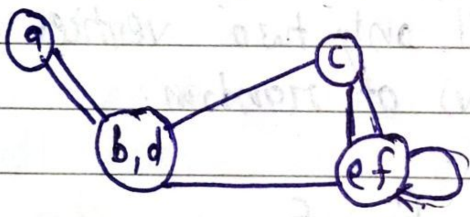
Remove self loop then



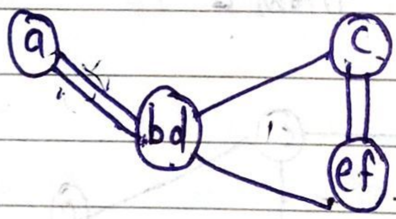
Check |v| = 5 > 2 yes

Pick an edge at random (e, f)

V = { a, bd, c, ef }



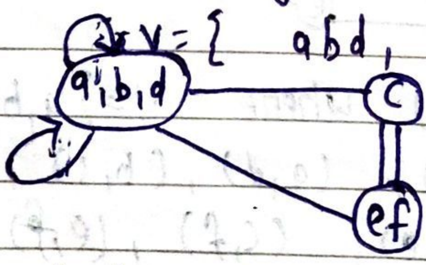
Remove self loop then



check |v| = 4 > 2 yes

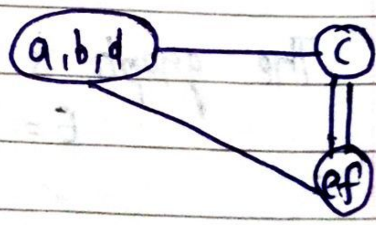
Pick an edge (a, bd)

v = { abd, c, ef }



Remove self loop

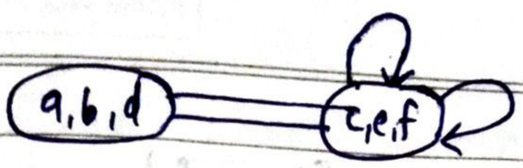
→



check |v| = 3 > 2 yes

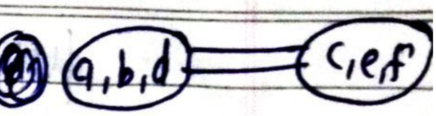
pick an edge (c, ef)

v = { abd, cef }



Remove selfloop

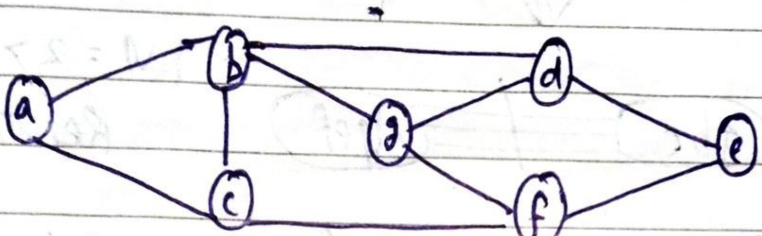
$\Rightarrow$



check $|V|$ $2 > 2$, No

Return $|E|$

Q.

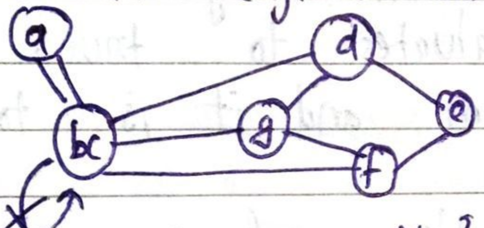


$V = \{a, b, c, d, e, f, g\}$

$E = \{(a,b)\ (a,c)\ (b,c)\ (b,d)\ (b,g)\ (c,f)\ (d,e)\ (d,g)\ (d,f)\ (f,e)\}$

Check $|V| = 7 > 2$ yes

Pick an edge at random (b,c)



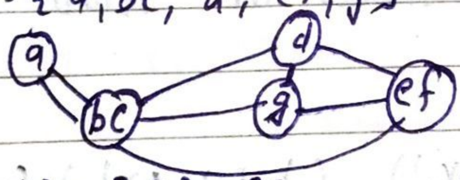
remove

$V = \{a, bc, d, e, f, g\}$

$V = 6 > 2$ yes

Pick an edge (e,f)

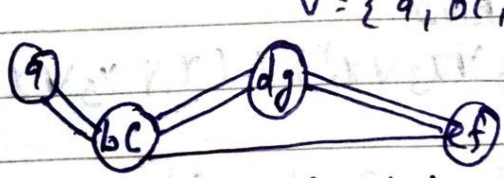
$V = \{a, bc, d, ef, g\}$



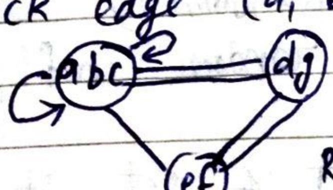
$V = 5 > 2$ yes

Pick edge (d,g)

$V = \{a, bc, ef, dg\}$

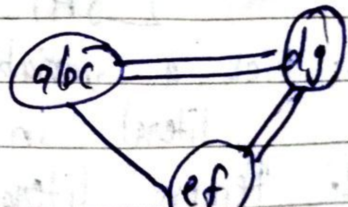


Pick edge (a, bc)



$\Rightarrow$

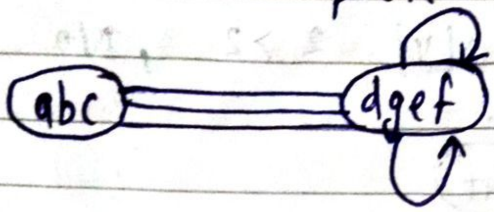
Remove



$V = \{abc, ef, dg\}$

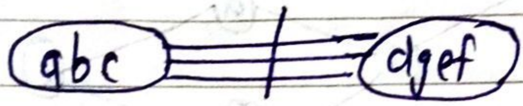
$|V| = 3 > 2$ yes

pick an edge (dg, ef)



Remove selfloop

⇓



$|V| = 2 > 2$, No

Return E

Randomized Algorithm for 2-sat Problem:-

It is the problem of deciding whether a boolean formula in propositional logic has an assignment that evaluates to true. SAT occurs as a problem and it is tool in applications.

SAT problem is problem of answering whether a boolean is satisfiable or not and boolean formula is in Conjunction Normal form (CNF).

It is collection of clauses in AND, OR of several literals.

Eg- $(x_1' + x_2)\ (x_5 + x_3 + x_2')\ (x_1 + x_5 + x_2')\ (x_3 + x_4')$

$\vee \rightarrow$ OR, $\wedge \rightarrow$ AND

$(x_1' \vee x_2) \wedge (x_5 \vee x_3 \vee x_2') \wedge (x_1 \vee x_5 \vee x_2') \wedge (x_3 \vee x_4')$

Algo:-

2_SAT_Problem_Random(f)

1. Pick a variable v at random in f, put its positive literal in list T.
2. Remove literal v from T & set it to True.
3. Remove all clauses from containing positive v,

DATE : / /

PAGE NO. :

4. For each clause containing negative literal 'u' put other variable in T.
5. Until T is empty or contradiction is found, for each variable in T repeat from step-②
   (i) If T is empty and f still contains some clauses goto step ①
   (ii) If a contradiction is found, report that f is not satisfiable.

Eg- $(a \vee b) \wedge (a \vee \bar{b}) \wedge (\bar{a} \vee c) \wedge (\bar{c} \vee b)$

Soln:- If $a = b = c = 1$

$(1 \vee 1) \wedge (1 \vee 0) \wedge (0 \vee 1) \wedge (0 \vee 1)$

$1 \wedge 1 \wedge 1 \wedge 1 = 1$

Eg- $(x_1' + x_2)\ (x_5 + x_3)\ (x_1 + x_4)\ (x_3' + x_1)\ (x_4 + x_5)$

Soln Pick x_1 at random so, $T = \{x_1\}$

Put x_1 = TRUE

Remove all clauses containing x_1 which is $(x_3' + x_4)$ Now

$f = (x_1' + x_2)\ (x_5 + x_3)\ (x_1' + x_4)\ (x_4 + x_5)$

Due to x_1' the forced variable x_2 & x_4, $T = \{x_2, x_4\}$

Put x_2 = TRUE

Remove all clauses containing x_2 which is $(x_1' + x_2)$

Now

$f = (x_5' + x_3)\ (x_1' + x_4)\ (x_4 + x_5)$

There are no forced variable due to x_2'

$T = \{x_4\}$ put x_4 = TRUE

Remove all clauses containing x_4 which is $(x_1' + x_4)$ and $(x_4 + x_5)$. Now $f = (x_5' + x_3)$

There are no ——— x_4, T = Empty

Again pick random x_3

PUT x_3 = TRUE

Remove clause containing x_3, which is $(x_5' + x_3)$
f = TRUE so stop {T, T, T, T, F} which represents
{x_1, x_2 x_3 x_4 x_5}.

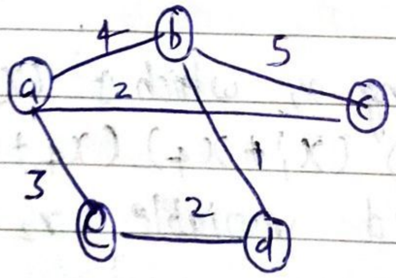
1-sat :- $O(n)$
2-sat :- Polynomial time
3-sat :- NP complete problem

Maximum Flow :- Given a flow network $G(V,E)$ with source s & sink t, MF is max. quantity of any commodity that can reach sink from source. There are several methods:-

(i) Linear Programming (ii) Ford-Fulkerson Method
(iii) Edmond's karp Algo. (iv) Dinitz Blocking Flow Algo.
(v) Push-Relabel Maximum Flow Algo.

Max-Flow-Min-cut-Theorem :- If f is flow graph then max. Flow from source to sink is equal to its max. s-t cut.

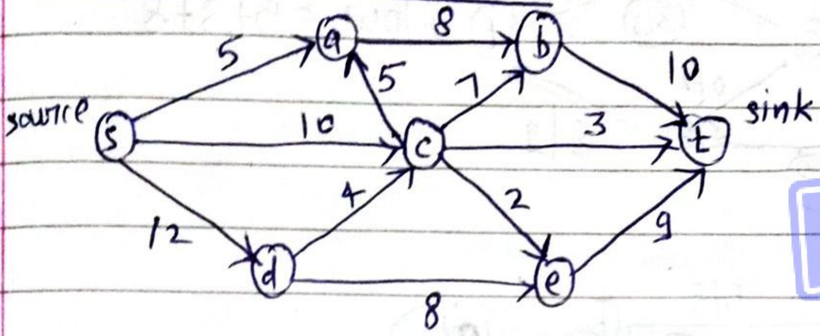
Eg -



By using hit & trial method cut of min value
$V = \{a, b, c, d, e\}$ Let's partition
$V_1 = \{a, b, e\}$, $V_2 = \{c, d\}$

cut-set = {(a,c), (b,d)} → value of cut = 3+1 = 4
{(a,b) (a,e) (b,d)} → 4+2+1 = 7 so min. is 4.

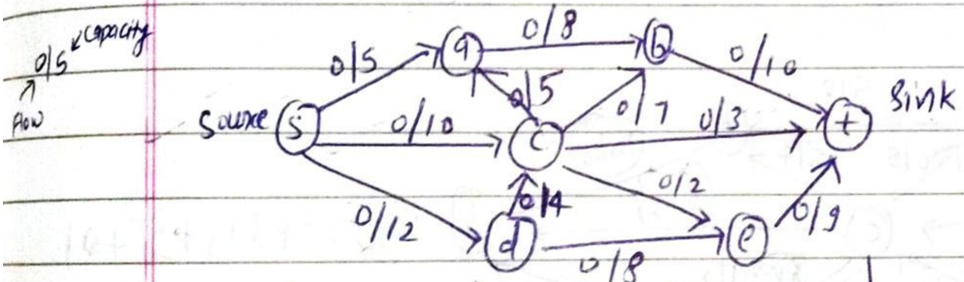
Ford - Fulkerson Method :- To find maximum flow.



(s) → (a) → (b) → (t) Bottle neck capacity = 5

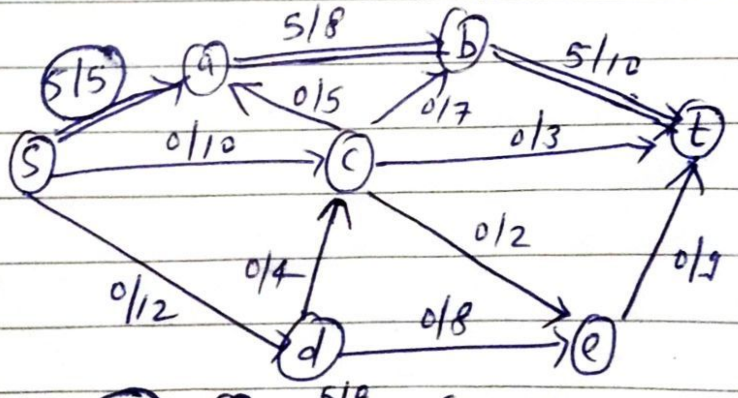
(s) → (d) → (e) → (t) ——— = 9

(s) → (c) → (t) ——— = 3



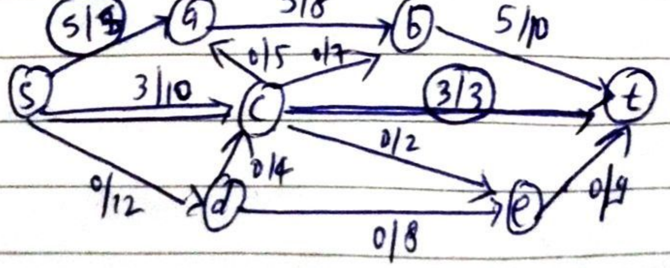
See residual capacity = capacity - Flow

Augumenting path	Bottleneck capacity
S → A — B → t	5
S → c — t	3
S → d → e → t	8
S → c → b → t	~~4~~ 5
S → d → c → e - t	~~2~~ 1

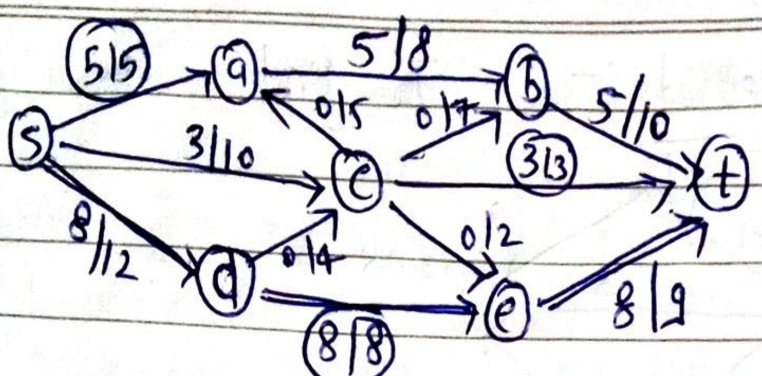


Flow = 5

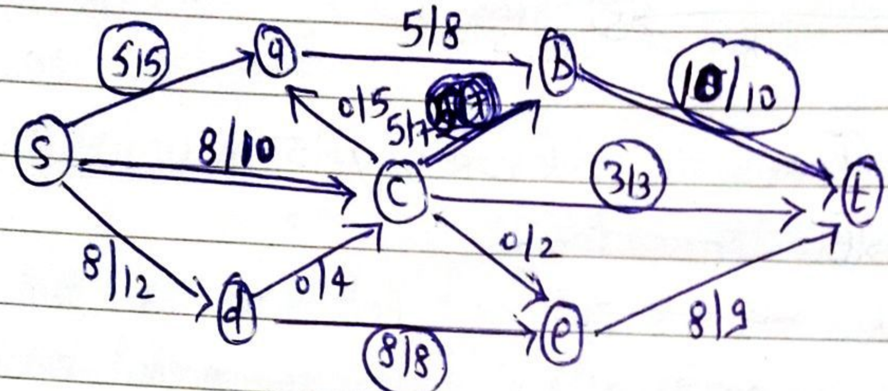
Again



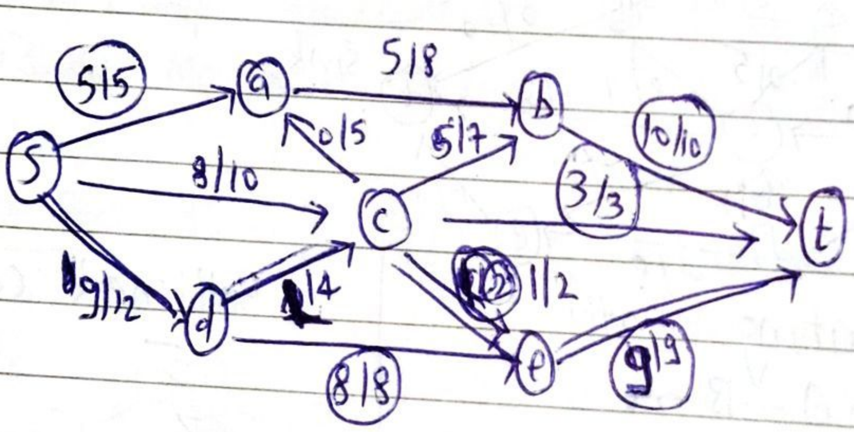
Flow 5+3



Flow = 5+3+8



→ Flow = 5+3+8+5



Flow = 5+3+8+5+1

Maximum Flow = 5+3+8+5+1 = 22