Syllabus:-

Branch and Bound: General method, applications - 0/1 knapsack problem, LC Branch and Bound solution, FIFO Branch and Bound solution, Non-deterministic algorithms, NP-Hard, NP-complete classes, cooks theorem. Travelling sales person problem.

NP Hard NP complete Problems:

Basic concepts. Non-deterministic algorithms, NP-hard and NP-complete classes, cook's theorem.

The branch and bound is a systematic method for handling optimization problems. This method is applied where greedy method and dynamic method fails.

The functionality of branch and bound is like BFS for searching optimal solution and it uses bounding function to determine which node and when to expand and there by provides solution.

collection of every possible path from root to the each node is called the statespace.

The problem states from which the tuple is defined the solution space are called solution states. A live node is one which is generated and all children of 'n' are not completly generated.

A dead node is generated one which cannot be expanded further or all all its children are generated. The live node whose children are currently generated called e-node.

Breadth first node generation with bounding function is called branch and bound.

Difference b/w backtracking and branch and bound

back tracking	branch and bound
(1) It is used to find all possible solutions available to the problem	1. It is used to solve optimization problem.
(2) It traverse tree by DFS	2. It may traverse tree in any manner. DFS or BFS.
(3) It realizes that it has made a bad choice and undoes the last choice by backup	3. It realizes that it is already has a better optimal solution and the pre-solution lead to so it uses that solution.
(4) It searches the statespace tree until it found a solution	4. It completely searches the state space tree to get optimal solution
(5) It involves feasibility function	(5) It involves bounding function.

These are 3 common search strategies of branch and bound technique they are.

① FIFO Branch and Bound
② LIFO Branch and Bound
③ Least cost Branch and Bound.

BFS will be called FIFO (uses queue)
DFS will be called LIFO (uses stack).

FIFO Branch and Bound:

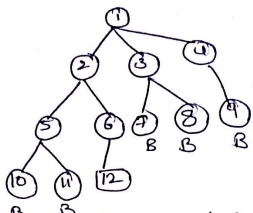
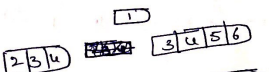


fig: state space tree.

Assume the node 12 is an answer node To begin with node 1 is E-node, next children node, are generated and these are placed in queue.



Queue storing live nodes.

Now we will delete an element from queue node 2, node 2 becomes E-node generate childrens of node 2

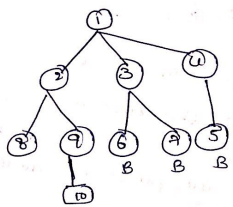
3 is deleted from queue and it becomes E-node. childrens of node 3 are generated. and these live nodes are killed by bounding function. so we will not include in the queue.

4 is deleted from queue child of node 4 are generated. and these lives nodes are killed by

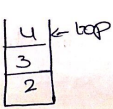
bounding function.

Next delete 5 and 10 & 11 are added. and killed by bounding function. The child node of 6 is 12 satisfies condition. which is answer node so terminates.

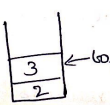
LIFO BRANCH AND BOUND



Initially stack is empty. The operations of stack are



Children of 4 are killed by bounding function



Children of node 3 are 6 & 7 those are killed by bounding function

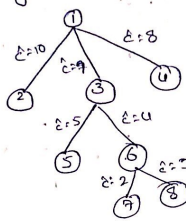
children of node 2 are added

children of node 9 is 10

Answer node find and all conditions are satisfied so terminates

Least cost branch and bound :-

Here the node with minimum cost should explore further. Here every node has some cost.



Note:- If any two nodes have some cost then left most node is always choosen.

To solve the least cost we choose method called LC-search. It is a kind of search in which a least cost bound for reaching the answer node.

LC is find out by ranking function denoted by

$$\hat{c}(x) = f(h(x)) + \hat{g}(x)$$

where $f(h(x))$ denotes level of node in BFS

$g(x)$ = estimate additional effort needed to reach a node.

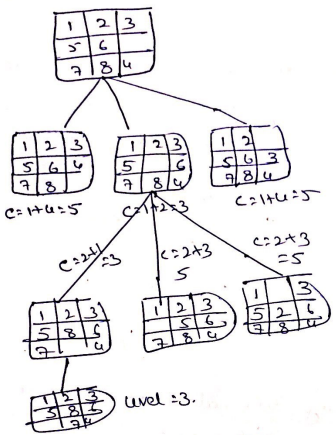
Ex:- 8-puzzle

Initial state

1	2	3
5	6	
7	8	

Final state

1	2	3
5	8	6
	7	4



Algorithm for BB:

```
1. Algorithm Branch-Bound [ ] {
2. // E-node is node pointer.
3. E ← new nodes //
4. while (true)
5. {
6. If (E is a final leaf) then
7. {
8. write (path from E to the root)
9. return;
10. expand (E);
11. if (heap is empty.) then {
12. write ("there no solution")
13. return; }.
14. E ← delete - top (H); }.
```

⇒ 0/1 knapsack Problem:-

Problem Statement:-

The 0/1 knapsack problem states that, there are n objects given and capacity of knapsack is 'm'. Then select some objects to fill the knapsack in such a way that it should not exceed the capacity of knapsack and maximum profit can be earned.

Branch and bound technique is used to find solution to knapsack problem.

Branch and bound technique cannot be directly applied to the knapsack problem, because the branch and bound deals with minimization problems.

The knapsack problem is modified and converted into the minimization problem.

The modified problem is

minimize profit $= -\sum_{i=1}^{n} P_i x_i$

subject to $\sum_{i=1}^{n} w_i x_i$

such that $\sum w_i x_i \leq m$ and

$x_i = 0$ or 1, where $1 \leq i \leq n$

LC Branch and Bound solution:

The LC Branch and Bound solution can be obtained using fixed tuple size formulation.

The steps to be followed for LC BB solution are

(1) Draw state space tree

(2) compute $\hat{c}(\cdot)$ and $u(\cdot)$ for each node.

(3) if $\hat{c}(x) >$ upper kill node x

(4) otherwise the minimum cost $\hat{c}(x)$ becomes E-node. Generate children for E-node.

(5) Repeat step 3 and step 4 until all the nodes get covered.

(6) The minimum cost $\hat{c}(x)$ becomes the answer node. trace the path in backward direction from x to root for solution subset.

example: consider knapsack instance $n=4$ with capacity $m=15$ such that:

object i	P_i	w_i
1	10	2
2	10	4
3	12	6
4	18	9

solution: Let us design state space tree using fixed tuple size formulation. The computation of $\hat{c}(x)$ and $u(x)$ for each node x is done.

Step:-1 $U(i)$ can be computed as

for $i = 1, 2$ and 3

$-\sum P_i = -(10+10+12)$

$U(i) = -32$

$u = \sum_{i=1}^{2} P_i x_i$ without fraction

$c = \sum_{i=1}^{2} P_i x_i$ with fraction

* If we select $i = u$, then it will exceed capacity of knapsack.

The computation of $\hat{c}(i)$ can be done as

$$\hat{c}(i) = U(i) - \left[\frac{m - \text{current total weight}}{\text{actual weight of remaining object}}\right] * \left[\text{Actual profit of remaining object}\right]$$

$$\hat{c}(i) = \left[-32 - \left[15 - \left[\frac{(2+4+6)}{9}\right] * 18\right]\right]$$

$$= \left[-32 - \left[\frac{3}{9} * 18\right]\right]$$

$$= \hat{c}(i) = -38$$

$U(i) = -32$

$\hat{c}(i) = -38$

with x_1

$U = -32$

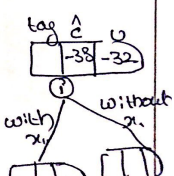
$\hat{c} = -38$

without x_1

we can place objects x_2 and x_3 into knapsack then

$U = -[\text{Profit of } x_2 + \text{Profit of } x_3]$

$U = -(10+12) = -22$



$$\hat{c} : U(i) - \left[\frac{m - \text{current total weight}}{\text{Actual weight of remaining object}}\right] * \left[\text{actual profit of remaining object}\right]$$

$$\hat{c} = -22 - \left[\frac{15-[4+6]}{9}\right] * 18$$

$$= -22 - \left[\frac{15-10}{9}\right] * 18$$

$$= -22 - \left(\frac{5}{9} * 18\right)$$

$$= -22 - 10 = -32$$

without x_1

$U = -22$

$\hat{c} = -32$

From with x_1 and without x_1, with x_1 is min

cost $= -38$ [-38 is less than -32]

↳ $\hat{c} \leq u \times$

Branches are expanded from node with LC "-38"

Step 3

with x_2

$U = -32$

$C(x) = -38$

without x_2

We can place objects x_1 and x_3 into knapsack then

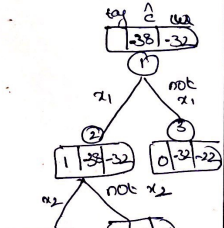
$$U = \left[\text{profit}(x_1) + \text{profit}(x_3)\right]$$

$$= U = -[10+12]$$

$$= -22$$

$$\hat{c} = U(2) - \left[\frac{m - \text{current total weight}}{\text{Actual weight of remaining object}}\right] * \text{actual profit of remaining object}$$

$$= -22 - \left[\frac{15 - [2+6]}{9}\right] * 18$$



$= -22 - \left[\frac{15-8}{9}\right] * 18$

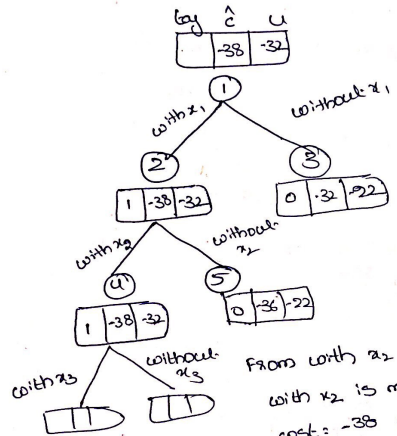
$= -22 - \left(\frac{7}{9}\right) * 18^{2}$

$= -22 - 14 = -36$

without x_2

$u = -22$

$\hat{c} = -36$



From with x_2 and without x_2

with x_2 is min $(u > \hat{c})$

cost = -38 (-38 less than -36)

expand from node LC = -38

Step 4

with x_3

here $u = -32$

$c(x) = -38$

without x_3

we can place object x_1, x_2, x_4 into the knapsack bag. then

$u = -\left[p(x_1 + x_2 + x_4)\right]$

$= -\left[10 + 10 + 18\right] = -38$

$\hat{c} = u(3) - \left[\frac{m - \text{current total weight}}{\text{Actual weight of remaining object}}\right] * \text{actual profit of remaining object}$

$$= -38\left[\frac{15-[15]}{0}\right]*0$$

$$= -38$$

without x_3

$U = -38$

$\hat{c} = -38$

from with x_3 and without x_3

without x_3 is minimum

$U = -38$ is less than -32

so expand from without x_3

with x_4

$U = -38$

$C = -38$

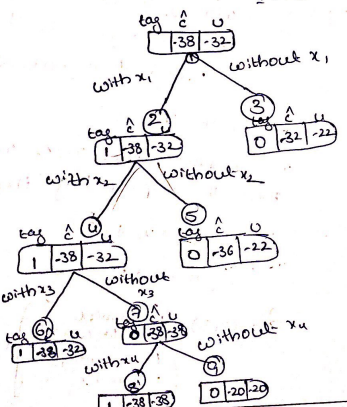
without x_4

$U = -[P_1(x_1 + x_2)]$

$= -20$

$\hat{c}(x) = -20 + [0]$

$= -20$



(7)

∴ Therefore the path is

$1 \to 2 \to 4 \to 7 \to 8$

(x_1, x_2, x_3, x_4)

$(1 \quad 1 \quad 0 \quad 1)$

maximum profit =

$10 + 10 + 0 + 18$

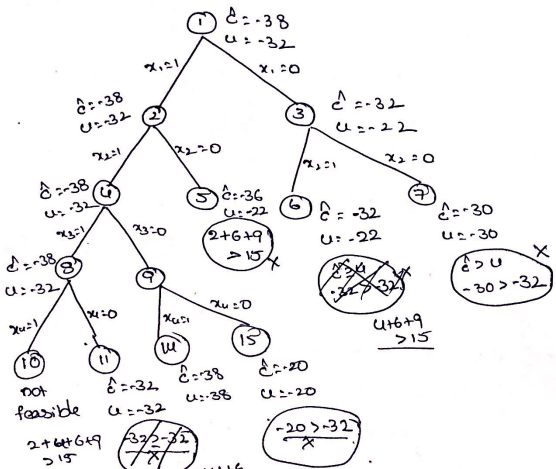
$= \boxed{38}$

FIFO BRANCH AND BOUND SOLUTION

The state space tree with variable tuple size formulation can be drawn and $\hat{c}(\cdot)$ and $u(\cdot)$ is computed

$n = 4$ $(P_1, P_2, P_3, P_4) = (10, 10, 12, 18)$ $m = 15$

$(w_1, w_2, w_3, w_4) = (2, 4, 6, 9)$



Initially upper $u(1) = -32$ then child of node 1 are generated. node 2 becomes E-node and hence 4, 5 are generated. node 4 are added in the list of the nodes. node 4 and 5 are added in the list of the nodes. next node 3 becomes enode and children 6, 7 are generated. As $\hat{c}(7) >$ upper we will kill node 7

→ Hence node 6 will be added in the list of live nodes. Node 4 is e-node and children 8, 9 are generated.

→ The upper is updated and it is now $u(9) = -38$ nodes 8 and 9 are added in the list of live nodes.

→ Nodes 5 & 6 becomes the next e-node but those values are exceed than m

node 5 : $2+6+9 > 15$ } so kill nodes 5 & 6
node 6 : $4+6+9 > 15$

→ Node 8 becomes next e node and children 10 & 11 are generated. As node 10 is infeasible do not consider.

→ As $\hat{c}(11) \geq$ upper Hence kill node.

→ Node 9 becomes next e-node and upper $= -38$ children 12 and 13 are generated.

→ But $\hat{c}(13) >$ upper so kill node 13

→ Finally node 12 becomes an answer node.

solution is $x_1 = 1 \quad x_2 = 1 \quad x_3 = 0 \quad x_4 = 1$

Object i	p_i	w_i
1	10	2
2	10	4
3	12	6
4	18	9

Solution :- $u(i) : i = 1,2,3$

$u(x) = -(10+10+12)$

$= -32$

$\hat{c}(x) = u - \left[\frac{m - \text{current total weight}}{\text{weight of remaining object}}\right] * \text{Profit of actual remaining object}$

$\hat{c}[x] = -32\left[\frac{-15-12}{9}\right] * 18$

$= -32-6 = -38$

Step 1

with x_1

$\hat{c} = -38$

$u = -32$

without x_1

we can place x_2 & x_3

$u = -(x_2+x_3)$

$= (-10+12) - 22$

$\hat{c} = -22\left[\frac{15-(0+6)}{9}\right] * 18$

$= -22-\left[\frac{9}{9} \times 18\right] = -32$

$u = -22$

$\hat{c} = -32$

Step 2

with x_2

$u = -32$

$\hat{c} = -38$

without x_2

same as above LCBB

$u = -22$

$c = -36$

Step 3

with x_2

$u = (P(x_2+x_3))$

$= -(10+12)$

$u = -22$

$c = -22\left[\frac{15-10}{9}\right] * 18$

$c = -32$

without x_2

$u = P(x_3+x_4)$

$u = -[12+18] = -30$

$c = -30\left[\frac{-15-15}{0}\right]$

$c = -30$

Step 3:- from node 4

with x_3

$C = -38$
$U = -32$ — same as above LCBB

without x_3

$C = -38$
$U = -38$

from node 5

with x_3

$= C = -36$
$U = -22$

without x_3

$U = 10+18 = 28$
$\hat{C} = -28$

Step 4:- from node 8

with x_4
not feasible.

without x_4

$U = -(x_1+x_2+x_3)$

$U = -[10+10+12]$

$U = -32$

$\hat{C} = -32\ [15-15]$

$\hat{C} = -32$

Step 5: from node 9

with x_4

U = with x_1, with x_2, with x_3, with x_4

$U = -[10+10+18]$

$U = -38$

$C = -38\ [15-(2+4+9)]$

$= -38$

$\hat{C} = -38$

without x_4

$U = -[10+10]$

$= -20$

$C = -20-0$

$C = 20$

$2+4+0+9$
$10+10+0+18$

x_1	x_2	x_3	x_4
1	1	0	1

Algorithm:. For computing $\hat{c}(x)$

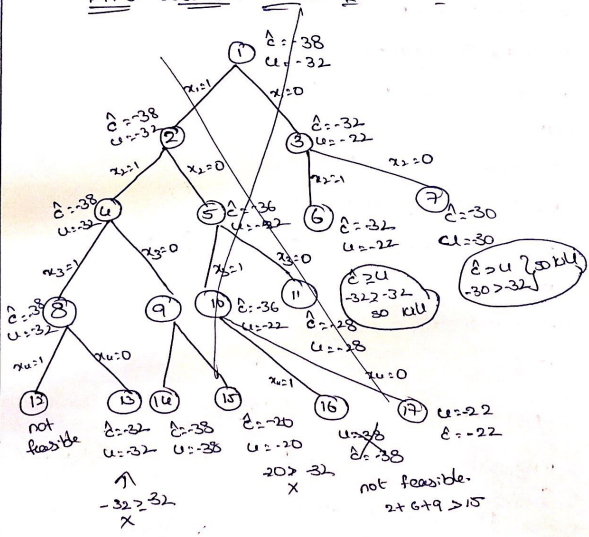
```
1. Algorithm Bound( total profit, total wt, k)
2. // Total profits denotes current total profit
3. // Total weight denotes current total weight
4. // k is index of last removed object
5// wt[] represents weight of object;
6 // p[] represents profit of object;
7. // m is weight capacity of knapsack.
8. {
9. pt ← total - profit;
10. wt ← total - wt;
11. for (i ← k+1 to n) do
12. {
13. wt ← wt + wt[i];
14. if (wt < m) then pt ← pt + p(i);
15. else return ( pt + (1-wt-m)/wi * P(i));
16. }
17. return pt;
18. }
```

Algorithm for compute $\hat{u}(x)$

```
1. Algorithm U-Bound (total profit, total wt, k, m)
{
    pt ← total - Profit;
    wt ← total - wt;
```

```
for ( i ← k+1 to n) do
{
    if (wt + w(i) <= m) then
    {
        pt ← pt - p[i];
        wt ← wt + w[i];
    }
    return pt;
}
```

FIFO solution for the above problem.



TRAVELLING SALES PERSON PROBLEM:

If there are 'n' cities and cost of travelling from one city to other city is given. A salesman start from one city and has to visit all the cities exactly once and has to return to the starting place with shortest distance or minimum cost.

Let $G = (V, E)$ be a directed graph defining an instance of the travelling sales person problem. Let c_{ij} be the cost of edge (i, j) and $c_{ij} = \infty$ if $(i, j) \notin E(G)$ and let $|V| = n$.

The following fig shows the state space tree organization for the case of a complete graph with $n = 4$. Each leaf node is a solution node.

The tree is defined by the path from root to the leaf node L.

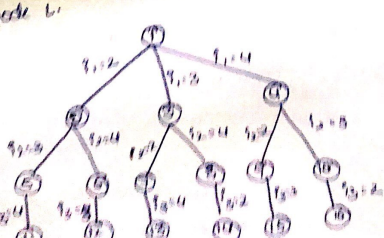


Fig: state space tree for the travelling sales person problem with $n = 4$ and i_0 to i_4.

Robust cost matrix:

A row or column is said to be reduced if it contains at least one zero and all remaining entries are

non-negative. A matrix is reduced if every row and column is reduced. If a constant 't' is choosen to be minimum entry in row 'i' or column 'j' then subtract it from all entries in row 'i' will introduce a zero into a row 'i'.

The total amount subtracted from the columns and rows is lower bound on the length of a minimum cost tour and can be used as the $\hat{c}(x)$ value for the root of state space tree.

With every node in state space tree, we associate a reduced cost matrix. To find a reduced cost matrix we have to following these steps:

(1) Change all entries in row i column j of 'A' to ∞

(2) Set A(j,i) to ∞

(3) Apply row reduction and column reduction except for rows and columns containing ∞

(4) Total cost for node 'S' can be calculated as

$$\hat{c}(s) = \hat{c}(R) + A(i,j) + x$$

where 'x' is the total amount subtracted in step 3.

Problem:- solve the following instance of travelling sales person cost matrix as shown below using LCBB

$$\begin{bmatrix} \infty & 20 & 30 & 10 & 11 \\ 15 & \infty & 16 & 4 & 2 \\ 3 & 5 & \infty & 2 & 4 \\ 19 & 6 & 18 & \infty & 3 \\ 16 & 4 & 7 & 16 & \infty \end{bmatrix}$$

fig: cost matrix

$$\begin{bmatrix} \infty & 20 & 30 & 10 & 11 \\ 15 & \infty & 16 & 4 & 2 \\ 3 & 5 & \infty & 2 & 4 \\ 19 & 6 & 18 & \infty & 3 \\ 16 & 4 & 7 & 16 & \infty \end{bmatrix} \begin{matrix} 10 \\ 2 \\ 2 \\ 3 \\ 4 \end{matrix} \quad \overline{21}$$

Reduce first row by 10
Reduce second row by 2
Reduce third row by 2
Reduce Fourth row by 3
Reduce Fifth row by 4

$$\begin{matrix} R_1-10 \\ R_2-2 \\ R_3-2 \\ R_4-3 \\ R_5-4 \end{matrix} \begin{bmatrix} \infty & 10 & 20 & 0 & 1 \\ 13 & \infty & 14 & 2 & 0 \\ 1 & 3 & \infty & 0 & 2 \\ 16 & 3 & 15 & \infty & 0 \\ 12 & 0 & 3 & 12 & \infty \end{bmatrix} \begin{matrix} & & & & \\ & & & & \\ & & & & \\ & & & & \\ \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\ 1 & 0 & 3 & \text{ignore} & \text{ignore} \end{matrix} = \begin{matrix} C_1-1 \\ C_3-3 \end{matrix} \begin{bmatrix} \infty & 10 & 20 & 0 & 1 \\ 12 & \infty & 11 & 2 & 0 \\ 0 & 3 & \infty & 0 & 2 \\ 15 & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & 12 & \infty \end{bmatrix}$$

ignore

$1+3=4$

Finally, the reduced cost matrix is

$$\begin{bmatrix} \infty & 10 & 19 & 0 & 1 \\ 12 & \infty & 11 & 2 & 0 \\ 0 & 3 & \infty & 0 & 2 \\ 15 & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & 12 & \infty \end{bmatrix} \quad 21+4$$

Reduced cost matrix $L=25$

Consider the path $(1,2)$: The first row and second column of reduced matrix are made infinity (∞) and element $(2,1)$ is also made ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 11 & 2 & 0 \\ 0 & \infty & \infty & 0 & 2 \\ 15 & \infty & 12 & \infty & 0 \\ 11 & \infty & 0 & 12 & \infty \end{bmatrix} \begin{matrix} \leftarrow \text{ignore} \\ \leftarrow \text{ignore} \\ \leftarrow \text{ignore} \\ \leftarrow \text{ignore} \\ \swarrow \text{path } 1,2 : \text{node } 2 \end{matrix}$$

Apply row and column reduction $\alpha = 0$

$$\hat{c}(2) = \hat{c}(1) + A(1,2) + \alpha$$

node $= 25 + 10 + 0 = 35$

consider path $(1,3)$ The first row third column of reduced matrix are made (∞) and element 3 is also made ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & \infty & 2 & 0 \\ \infty & 3 & \infty & 0 & 2 \\ 15 & 3 & \infty & \infty & 0 \\ 11 & 0 & \infty & 12 & \infty \end{bmatrix} \begin{matrix} \leftarrow \text{ignore} \\ \leftarrow \text{ignore} \\ \leftarrow \text{ignore} \\ \leftarrow \text{ignore} \\ \leftarrow \text{ignore} \end{matrix}$$

$\downarrow$ $\downarrow$ $\downarrow$
11 ignore ignore

Path $(1,3)$ node 3.

Above matrix is not reduced. Subtract 11 from 1st column then matrix is reduced.

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 1 & \infty & \infty & 2 & 0 \\ \infty & 3 & \infty & 0 & 2 \\ 4 & 3 & \infty & \infty & 0 \\ 0 & 0 & \infty & 12 & \infty \end{bmatrix}$$

path 1,3 node 3

Apply row and column reduction $\alpha = 11$

$$\hat{c}(3) = \hat{c}(1) + A(1,3) + \alpha$$

node $= 25 + 17 + 11 = 53$

consider the path $(1,4)$: The first row and fourth column of reduced matrix are made ∞ and element 4,1 also made ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & \infty & 0 \\ 0 & 3 & \infty & \infty & 2 \\ \infty & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & \infty & \infty \end{bmatrix}$$

fig: path 1,4 ; node 4

Apply row and column reduction $\alpha = 0$

$$\hat{c}(4) = \hat{c}(1) + A(1,4) + \alpha$$
$$= 25 + 0 + 0 = 25$$

Consider the path (1,5) : The first row and fifth column of reduced matrix are made ∞ and element (5,1) is also made ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & 2 & \infty \\ 0 & 3 & \infty & 0 & \infty \\ 15 & 3 & 12 & \infty & \infty \\ \infty & 0 & 0 & 12 & \infty \end{bmatrix} \leftarrow 9,\ \leftarrow 2,\ \leftarrow 9,\ \leftarrow 3,\ \leftarrow 9$$

Path(1,5) node 5

This matrix is not reduced. subtract 2 from 2nd row 3 from 4th row.

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 10 & \infty & 9 & 0 & \infty \\ 0 & 3 & \infty & 0 & \infty \\ 12 & 0 & 9 & \infty & \infty \\ \infty & 0 & 0 & 12 & \infty \end{bmatrix}$$

Apply row reduction and column reduction $r = 5+0 = 5$

$\hat{c}(u) = \hat{c}(i) + A(1,5) + r$

$= 25 + 1 + 5 = 31$

fig : path(1,5) : node 5

Since the minimum cost is 25. so select nodes. The matrix obtained for path (1,4) is considered as reduced cost matrix. and apart of space tree generated by procedure LCBB

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & \infty & 0 \\ 0 & \infty & \infty & \infty & 2 \\ \infty & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & \infty & \infty \end{bmatrix}$$

Reduced cost matrix

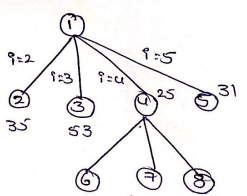


fig: Part of state space tree generated by procedure LCBB

Consider the path (1, 4, 2): The fourth row and second column of reduced matrix are made ∞ and element (2,1) also made ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 10 & \infty & 0 \\ 0 & \infty & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty \\ 11 & \infty & 0 & \infty & \infty \end{bmatrix} \begin{matrix} \leftarrow 9 \\ \leftarrow 9 \\ \leftarrow 9 \\ \leftarrow 9 \\ \leftarrow 9 \end{matrix}$$

$\uparrow \quad \uparrow \quad \uparrow \quad \uparrow \quad \uparrow$

Path 1,4,2 : node 6

Apply row and column reduction $r = 0$

$\hat{c}(6) = \hat{c}(4) + A(4,2) + r$

$= 25 + 3 + 0$

$= 28$

Consider the path (1, 4, 3): The fourth row and third column of reduced matrix are ∞ and element (3,1) is also made ∞.

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & \infty & \infty & 0 \\ \infty & 3 & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty \\ 11 & 0 & \infty & \infty & \infty \end{bmatrix} \begin{matrix} \leftarrow 9 \\ \leftarrow 9 \\ \leftarrow 2 \\ \leftarrow 9 \\ \leftarrow 9 \end{matrix}$$

$\uparrow \quad \uparrow \quad \uparrow \quad \uparrow \quad \uparrow$

Path 1,4,3 : node 7.

This matrix is not reduced, substract 11 from 1st column 2 from 3rd row.

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 1 & \infty & \infty & \infty & 0 \\ \infty & 1 & \infty & \infty & 0 \\ \infty & \infty & \infty & \infty & \infty \\ 0 & 0 & \infty & \infty & \infty \end{bmatrix}$$

Path 1,2,3 : node 7

Apply row and column reduction $r = 2 + 11 = 13$

$\hat{c}(7) = \hat{c}(4) + A(4,3) + r$

$= 25 + 12 + 13 = 50$

Consider the path (1,4,3) :- The fourth row and fifth column of reduced matrix and made ∞ and element 5,1 is also made ∞.

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & \infty & 0 \\ 0 & 3 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & 0 & \infty & \infty \end{bmatrix} \begin{matrix} \leftarrow 9 \\ \leftarrow 11 \\ \leftarrow 9 \\ \leftarrow 9 \\ \leftarrow 9 \end{matrix}$$

$\downarrow 9 \quad \downarrow 0 \quad \downarrow 0 \quad \downarrow \infty \quad \downarrow \infty$

path 1,4,3 node 8

This matrix is not reduced substract 11 from 2nd row and then this matrix is reduced.

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 1 & \infty & 0 & \infty & \infty \\ 0 & 3 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & 0 & \infty & \infty \end{bmatrix}$$

path 1,4,3 : node 8

Apply row reduction and column reduction.

$$r = 11+0 = 11$$

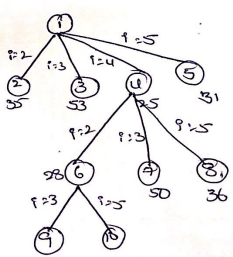
$$\hat{c}(8) = \hat{c}(4) + A(4,3) + r$$

$$= 25+0+11 = 36$$

since the minimum cost is 28, so select node 2 The matrix obtained for path (1,4,2) is considered as reduced cost matrix.

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 11 & \infty & 0 \\ 0 & \infty & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty \\ 11 & \infty & 0 & \infty & \infty \end{bmatrix}$$

Reduced cost matrix



Consider path (1,4,2,3) :- The 2nd row and 3rd column of reduced matrix are ∞. and element (3,1) also made ∞.

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty \\ 11 & \infty & \infty & \infty & \infty \end{bmatrix} \leftarrow 2$$

$\uparrow$ 11 $\qquad\qquad \uparrow_2$

This matrix is not reduced matrix. subtract 11 from 1st column and 2 from last column

path 1,4,2,3 node 9

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 0 \\ \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \end{bmatrix}$$

Apply row and column reduction

$r = 2+11 = 13$

$\hat{c}(9) = \hat{c}(6) + A(2,3) + r$.

$= 28 + 11 + 13 = 52$

Consider path (1,4,2,5) :- The 2nd row and 5th column of reduced matrix are made ∞ and (5,1) also ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty \end{bmatrix}$$

path 1,4,2,5 node 10

Apply row and column reduction $r = 0$

$\hat{c}(10) = \hat{c}(2) + A(2,5) + r$

$= 28 + 0 + 0 = 28$

since minimum cost is 28, so select node 5. The matrix obtained for path (1,4,2,5) is considered as reduced cost matrix.

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty \end{bmatrix}$$

Reduced cost matrix.

Consider the path (1,4,2,5,3):- The 5th row and 3rd column of reduced matrix are made ∞ and element (3,1) also made ∞.

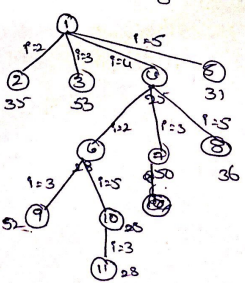
$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

path (1,4,2,5,3) node 11

Apply row reduction and column reduction r=0

$\hat{c}(11) = \hat{c}(5) + A(5,3) + r = 28+0+0 = 28$

complete state space tree generated by procedure LCBB



Path is
1→4→2→5→3→1
min cost :
10+6+2+7+3
= 28.

fig: state space tree generated by LCBB

NP-HARD AND NP-COMPLETE PROBLEMS:

Introduction:

The problems that we intend to solve, using a computer, can broadly be divided into two types P-class and NP-class problems.

The problems that can be solved in polynomial time are said to be in P-class.

Ex:- The problem of searching an element from the list. It can be solved in O(log n) time.

The problems that can be solved in the polynomial time, by non-deterministic machine, are called NP-class problems.

Ex:- knapsack problem whose time complexity is $(O(2^{n/2}))$

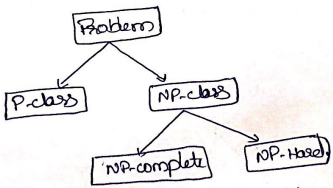


Fig:- classification of problems.

Tracable Problems:

They are problems which can be solved in a finite duration of time. The problems that can be solved in polynomial time are called tracable problems.

Polynomial time Algorithms:

An algorithm for a given problem size(n) is said to be a polynomial time algorithm if it's worst case complexity belongs to $O(n^k)$ for a fixed size small integer value k and an input size of n.

INTRACABLE PROBLEMS:

The problems that cannot be solved in the polynomial time are called intracable problems. In real time applications, there are some problems whose solution may take large amount of time or there might not be any known solution.

This might be due to the nature of the input, complexity of the algorithm used for solving etc. They are called intracable problems.

Ex:- Algorithms whose worst case complexity belongs to $O(2^n)$, $O(n^n)$ etc.

Problems can be divided into

* Decision problems
* optimization problems.

Decision problem is a problem for which the output is binary i.e., true/ false /no.

Optimization problem is a problem that finds an optimal value of a cost function.

NP-PROBLEMS:-

They are also called Deterministic polynomial Problems. NP problems can be solved by non-deterministic Algorithms.

NP class is a set of all decision problems solvable by non-deterministic algorithm in polynomial time.

Ex:- consider the problem of searching a particular element in a binary tree. Let the depth of binary tree be 'd'. suppose depth is 0, we have only one element in the tree

If the depth is 1, we have 3 elements. In general no of elements in the tree having depth 'd' is given by $(2^{d+1})-1$

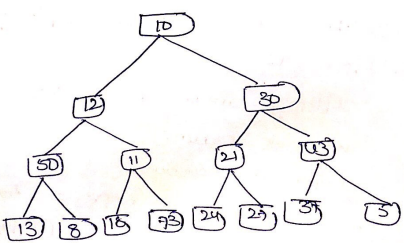


Fig:- Binary search problem representation.

A deterministic algorithm for this problem would be as follows

Search from the root node (10) and keep traversing in preorder comparing each element. Because search is sequential the worst case is $O(2^{h+1})$ comparisons.

A non deterministic algorithm for searching problem could be as follows:

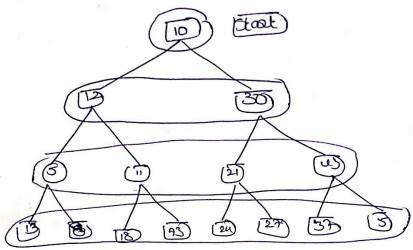


Fig: Binary search for a non deterministic algorithm.

For a non deterministic algorithm which can compare the search element with all the elements at every level in one unit of time, the search will have a worst case complexity of $O(d) \geq O(n)$: which is polynomial time.

The key point to note here is that search is not conducted in a serial fashion. Machines of such capability are non deterministic machines.

A non-deterministic algorithm is an algorithm which can be executed on a non-deterministic machine.

NP HARD AND NP-COMPLETE CLASSES

NP-complete problem:

There are large number of problems in real time applications belonging to NP-class.

It is possible to identify a small set of NP problems such that if any one problem in this set has a polynomial time solution on a deterministic machine, then the rest of all NP-problems will also have a polynomial time solution on a deterministic machine

This subset of NP problems having the above mentioned special property is called NP-complete problems.

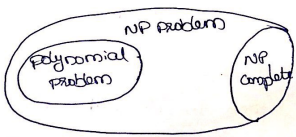
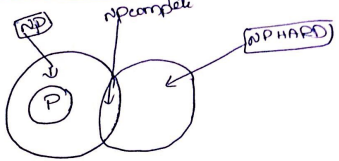


fig: NP complete problem representation

NP-HARD PROBLEMS:

A problem L is hard if and only if the satisfiability is reduced to L. If an NP-HARD problem can be solved in polynomial time then all NP-complete problems are solved in polynomial time.

A problem L is NP-complete if and only if L is NP-HARD and $L \in NP$



Satisfiability Problem:

We formulate satisfiability problem to determine whether a formulae is true for same assignment of truth values to the variables.

COOK's Theorem:

COOK's theorem states that satisfiability is in P if P=NP. We already know that satisfiability is in NP. Hence if P=NP, then satisfiability is in P. Then it remains to be shown that if satisfiability is in P then P=NP.

In order to prove this statement, we show how to obtain any polynomial time non deterministic decision algorithm A and input I, a formulae Q(A,I) such that Q is satisfiability if A has successful termination with input I.

If the length of I is 'n' and the time complexity of A is P(n) for some polynomial P() then the length of Q will be $O(P^3(n) \log n) = O(P^4(n))$. The time needed to construct Q will also be same.