Dynamic programming

Algorithm

```
if fib[n], fib[0]=0, fib[1]=1;
for(int i=2; i<n; i++)
  fib[i]=0;
int fibbonacci(int n)
  if(fib[n]==0 && n!=0)
    fib[n]= fib[n-1] + fib[n-2];
  else
    return fib[n];
}
```

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

In dynamic programming an optimal sequence of decisions is obtained by making explicit appeal to principle of optimality.

Principle of optimality

It states that an optimal sequence of decisions has the property that whatever the initial state and the decision are the remaining decisions must constitute an optimal decision sequence with regard to the state resulting from the first decision.

The difference b/w greedy and dynamic programming is that

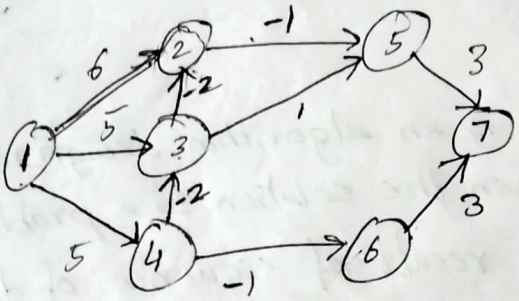
In greedy only one decision sequence is ever generated.

In DP many decision sequences may be generated. However sequences containing sub optimal subsequences cannot be optimal. (If the principle of optimality holds) and so will not be generated.

- Single source shortest path (general weights)

(Not allow -ve weight edges)

DP (allows -ve weight edges)



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

	$dist^k[1..7]$						
k	1	2	3	4	5	6	7
1	0	6	5	5	∞	∞	∞
2	0	6	3	5	5	4	∞
3	0						
4	0						
5	0						
6	0						
7	0						

k is stages

$$dist^k[u] = \min\{ dist^{k-1}[u], \min_{i}\{ dist^{k-1}[i] + cost[i,u] \} \}$$

$k = 2, 3, \ldots n-1$

i → adjacent vertices

k=2, u=2

$dist^2[2] = \min\{dist$

$= \min\{dist$

$= \min\{6$

$= \min\{$

$= \min\{$

$dist^2[2] = 3$

$dist^2[3] = \min\{dis$

$= \min$

$= \min$

$= 3$

$dist^2[4] =$

$=$

$=$

$dist^2[5] =$

$=$

$dist^2[5]$

$dist^2[6] =$

$dist^2[7]$

$dist^2[7]$

$dist^2[2] = \min\{dist'[2], \{dist'[1] + cost[1,2]\}\}$

$= \min\{dist'[2], \min\{dist'[1]+cost[1,2], dist'[3]+cost[3,2]\}\}$

$= \min\{6, \min\{0+6, 5+(-2)\}\}$

$= \min\{6, \min\{6,3\}\}$

$= \min\{6,3\}$

$dist^2[2] = 3$

$dist^2[3] = \min\{dist'[3], \min[dist'[4]+cost[4,3], dist'[1]+cost[1,3]\}$

$= \min\{5, \min\{5+(-2), 0+5\}\}$

$= \min\{5, \min\{3,5\}\}$

$= 3$

$dist^2[4] = \min\{dist'[4], \min\{dist'[1]+cost[1,4]\}$

$= \min\{5, \min\{0+5\}\}$

$= 5$

$dist^2[5] = \min\{dist'[5], \min\{dist'[2]+cost[2,5], dist'[3]+cost[3,5]\}$

$= \min\{\infty, \min[6+(-1), 5+1]\}$

$= \min\{\infty, \min\{5,6\}\}$

$dist^2[5] = 5$

$dist^2[6] = \min\{dist'[6], \min\{dist'[4]+cost[4,6]\}$

$= \min\{\infty, \min\{5+(-1)\}\}$

$= 4$

$dist^2[7] = \min\{dist'[7], \min\{dist'[5]+cost[5,7], dist'[6]+cost[3,7]\}$

$= \min\{\infty, \min\{\infty, \infty\}\}$

$dist^2[7] = \infty$

Multistage graph

A multistage is a directed graph in which the nodes can be divided into set of stages.

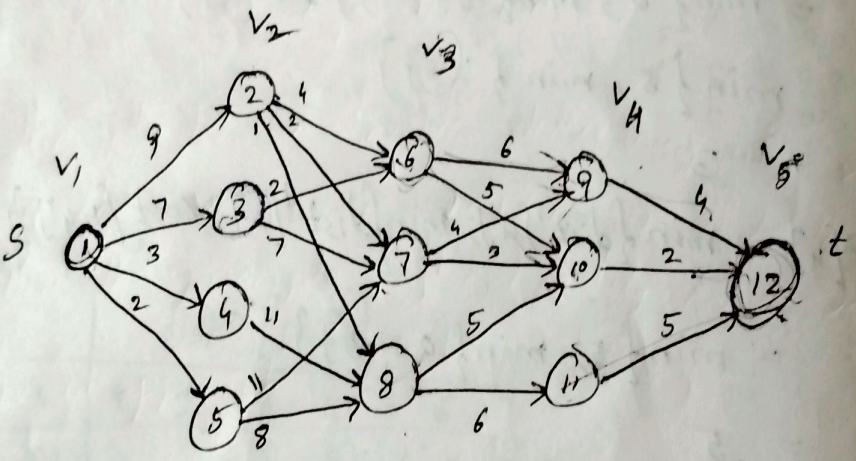
Given a multistage graph, a source vertex and destination vertex the problem is find the shortest path from source to the destination.

Always consider source at stage-1 and destination at last stage.

It can be solved by using 2 methods

Forward approach : sol starts from dest

Backward approach : sol starts from source



Forward approach

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

Cost (i, v) i ⇒ stage, v ⇒ vertex

Cost (5, 12) = 0

Cost (4, 9) = min { c(9,12) + cost(5,12) } = 4

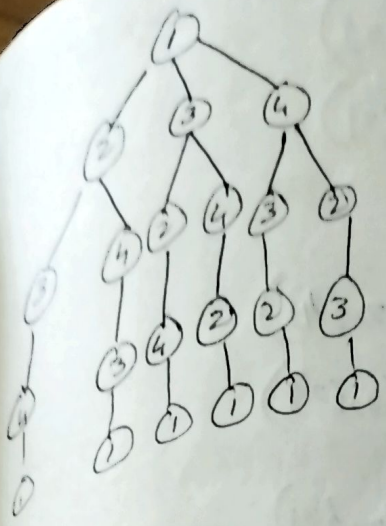
Cost (4, 10) = min { c(10,12) + cost(5,12) } = 2

Cost (4, 11) = min { c(11,12) + cost(5,12) } = 5

c - edge cost

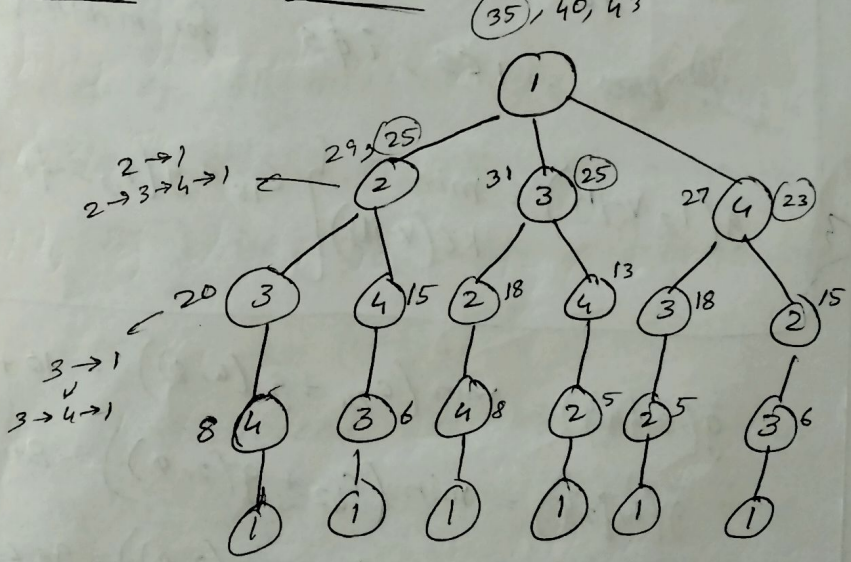
cost - node cost

d → because of which vertex we got cost.



- Consider city 1 as starting & end point.
- Generate all $(n-1)!$ permutations of cities.
- Calculate cost of every permutation and keep track of min cost permutation.
- Return permutation with the min cost.
- Time complexity - $O(n!)$

By using dynamic programming



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

$$= \min \begin{cases} c_{12} + g(2,\{3,4\}) \\ c_{13} + g(3,\{2,4\}) \\ c_{14} + g(4,\{2,3\}) \end{cases}$$

$$g(2,\{3,4\}) = \min \begin{cases} c_{23} + g(3,\{4\}) \\ c_{24} + g(4,\{3\}) \end{cases}$$

$$g(3,\{2,4\}) = \min \begin{cases} c_{32} + g(2,\{4\}) \\ c_{34} + g(4,\{2\}) \end{cases}$$

$$g(4,\{2,3\}) = \min \begin{cases} c_{42} + g(2,\{3\}) \\ c_{43} + g(3,\{2\}) \end{cases}$$

$$g(3,\{4\}) = \min(c_{34} + g(4,\phi)) = 12+8=20$$

$$g(4,\{3\}) = \min(c_{43} + g(3,\phi)) = 9+6=15$$

Formula

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

in general $i \notin s$, $1 \notin s$, $i \notin s$ for middle values.

$$g(1, V-\{1\}) = \min_{k \in (V-\{1\})} \{c_{1k} + g(k, s-\{k\})\} \quad (\text{If } i, \quad -(1)$$

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

$$g(4,\{2\}) = \min(c_{42} + g(2,\phi)) = 8+5=13 \qquad g(2,\phi) = c_{2}$$

$$g(2,\{3\}) = \min(c_{23} + g(3,\phi)) = 9+6=15 \qquad g(i,\phi) = c_{i1}$$

$$g(3,\{2\}) = \min(c_{32} + g(2,\phi)) = 13+5=18$$

$$g(4,\{2,3\}) = \min \begin{cases} 8+15 = 23 \\ 9+18 = 27 \end{cases} = 23$$

$$g(3,\{2,4\}) = \min \begin{cases} 13+18 = 31 \\ 12+13 = 25 \end{cases} = 25$$

$$g(2,\{3,4\}) = \min \begin{cases} 9+20 = 29 \\ 10+15 = 25 \end{cases} 25$$

$g(1,\{2,3,4\}) = \begin{cases} 15 + 25 = 40 \\ 20 + 23 = 43 \end{cases}$

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

$g(1,\{2,3,4\})$ → (because of 2 vertex)

Tour

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

$T(2,\{3,4\}) = 4$

$T(4,\{3\}) = 3$

$1 \to 2 \to 4 \to 3 \to 1$ min cost tour.

Time complexity = $2^n n^2$

→ Let N be the no. of $g(i,s)$ that have to be computed before equation (1).

For each value of $|s|$ there are $n-1$ choices for i

The no. of distinct sets s of size k not including 1 and i is $\binom{n-2}{k}$ i.e $^{n-2}C_k$. Hence N is calculated as $\sum_{k=0}^{n-2} (n-k-1)\, {}^{n-2}C_k \approx (n-1)2^{n-2}$

An algorithm that proceeds to find optimal tour by using eq (1) & (2) will require $\theta(n^2 2^n)$ time as computation of $g(i,s)$ with $|s| = k$ requires $k-1$ comparisions when solving eq (2).

in eq (1) we compare for min val.

Time complexity = $(n-1)\left((n-1)2^{n-2}\right)$

$= (n-1)^2 2^{n-2}$

$\theta(n^2 2^n)$

22.5.23

Articulation Point

A vertex v in a connected graph G is an articulation point if and only if the deletion of vertex v together with all the edges incident to v disconnects the graph into 2 or more non empty components.

Example

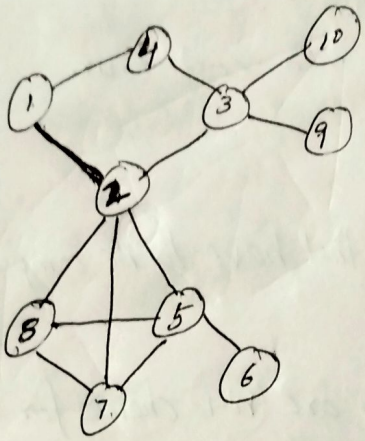
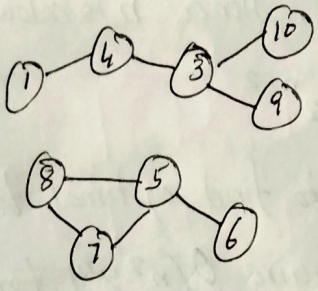


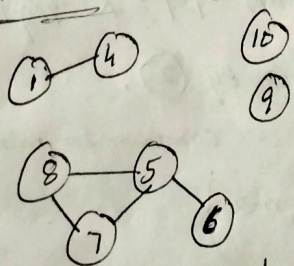
fig (1)

Articulation points are 2, 3, 5

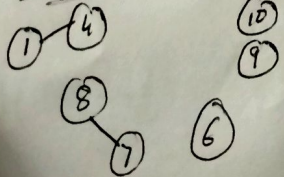
on deletion of vertex 2: 2 is an articulation point.



on deletion of vertex 3: 3 is an articulation point.

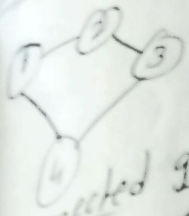


on deletion of vertex 5 5 is an articulation point.



Example

fig ②



There are no articulation points.

Biconnected graph

A graph G is biconnected if G has no articulation points.

fig ① is not biconnected

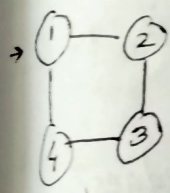
fig ② is biconnected.

The presence of articulation points is an undesirable feature in many of the cases.

Ex: In communication network which is represented by graph G with vertices representing the communication stations and edges - communication lines, then the failure of a communication station i that is an articulation point would result in the loss of communication to the other communication stations also.

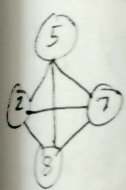
Biconnected component.

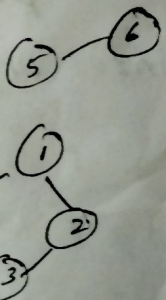
A maximal biconnected sub graph is a biconnected component.

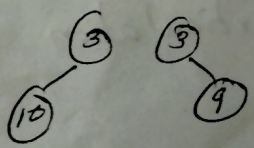


It has only one biconnected component.

Subgraphs of fig ①







These are all the biconnected components.

Algorithm for constructing a biconnected graph.

for each articulation point a do

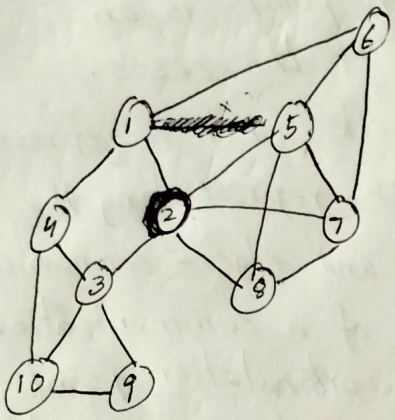
{

let $B_1, B_2, \ldots$ be biconnected components containing vertex a

let v_i , $v_i \neq a$ be a vertex in B_i $1 \leq i \leq k$;

Add to G the edges (v_i, v_{i+1}), $1 \leq i \leq k$;

}



Adding edges (4,10), (10,9) corresponding to articulation vertex 3.

Adding edge (1,6) corresponding to articulation point 2

edge (6,7) corresponding to articulation point 5

Note:

The edge v_i, v_{i+1} is added then 'a' is no more an articulation point.

By using the above algorithm adding edges corresponding to all articulation points, then G has no articulation points and so it is biconnected

tructing a biconnected graph.

point a do
biconnected components
a
be a vertex in B_i $1 \le i \le k$;
(v_i, v_{i+1}), $1 \le i \le k$;

Adding edges
(4,10), (10,9)
corresponding to
articulation vertex 3.

dding edge (1,6)
responding to articulation
nt 2
(6,7) corresponding
ulation point 5

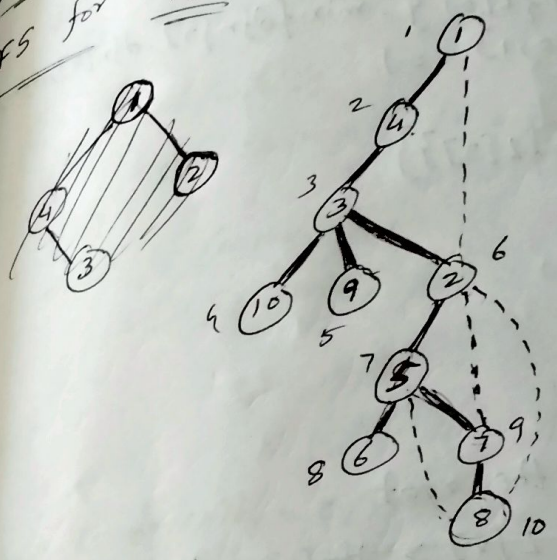
then 'a' is no

adding edges
n points, then
and so it is

If G has p articulation points and b biconnected components then according to the algorithm b-p new edges will be added to G.

Algorithm for Articulation points

DFS for fig (1)



→ The solid lines are called tree edges

→ The broken lines are back edges.

→ write numbers on vertices based on order in which we visited them. (depth first numbers)

→ find L values for all.

vertices	1	2	3	4	5	6	7	8	9	10
dfn	1	6	3	2	7	8	9	10	5	4
L (low values)	1	1	1	1	6	8	6	6	5	4

Lowest depth first search number reached from u using a path of descendants followed by atmost one backedge.

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

$L[10] = \min\{dfn[10]\} = 4$

$L[9] = \min\{dfn[9]\} = 5$

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

$L[8] = \min\{dfn[8], \min\{dfn[2], dfn[5]\}\} = \min\{10, 6, 7\} = 6$

$L[5] = \min\{dfn[5], \min\{L[6], L[7]\}, \min\{dfn[8]\}\} = \min\{7, 8, 6, 10\} = 6$

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

$= \min\{9, 6, 6\}$

$= 6$

$L[1] = \min\{dfn[1], \min\{L[4]\}, \min\{dfn[2]\}\}$

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

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

$L[2] = \min\{dfn[2], \min\{L[5]\}, \min\{dfn[7], dfn[8], dfn[1]\}\}$

$= \min\{6, 6, 9, 10, 1\} = 1$

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

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

$L[1] = \min\{1, 1, 6\} = 1$

If u is not the root

~~For any (u, v) if $L[v] \geq dfn[u]$~~, then u is an articulation point iff v has child w such that $L[w] \geq dfn[u]$

$u = 3, \quad w = 10$

$L[10] = 4 \qquad dfn[3] = 3$

$4 > 3$ 3 is articulation point

Algorithm Art(u,v)

Globally:
num = 1
dfn [0 . . . . . n]

{
dfn[u] := num;
L[u] := num;
num := num+1
for each vertex w adjacent to u do
{ if (dfn[w]=0) then
{ Art(w,u);
L[w] := min {L[w], L[u]};
}
else if (w≠v) then
L[u] := min{L[u], dfn[w]};
}
}

Time complexity = O(n+e)

Algorithm Bicomp(u,v)

{
dfn[u] := num;
L[u] := num
num := num+1;
for each vertex w adj to u do
{ if((v≠w) and (dfn[w]<dfn[u])) then
add (u,w) to top of stack s;
if(dfn[w]=0) then
{ if (L[w] ≥ dfn[u]) then
{ write ("New bicomponent");
repeat
{
delete an edge from top of stack s;
Let this edge be (x,y);
write (x,y);

```
} until (((x,y) = (u,w)) (or) ((x,y) = (w,u)));
}
BiComp(w,u);
l[u] := min(l[u], l[w]);
}
else if (w ≠ v) then l[u] := min(l[u], dfn[w]);
}
}
```

Optimal binary search tree

internal nodes ⇒ $a_1, a_2 \ldots\ldots a_n$

External nodes ⇒ $E_0, E_1 \ldots\ldots E_n$

$E_0 = x < a_1$

$E_i = a_i < x < a_{i+1}$

$E_n = x > a_n$

p_i = probability that $x = a_i$

q_i = prob$(a_i < x < a_{i+1})$

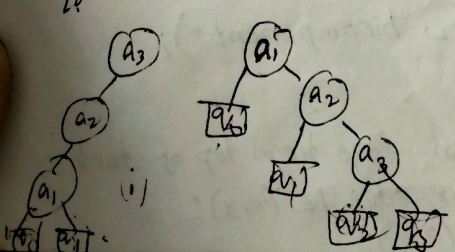
Cost of BST

$$\sum_{1 \le i \le n} p_i \times (\text{level}(a_i)) + \sum_{0 \le i \le n} q_i \times (\text{level}(E_i) - 1)$$

Ex.

a_1	a_2	a_3
$p_1 = 0.5$	$p_2 = 0.1$	$p_3 = 0.05$

$q_0 = 0.15$ $q_1 = 0.1$ $q_2 = 0.05$ $q_3 = 0.05$



$0.05\times1 + 0.1\times2 + 0.5\times3$

$+(0.15)3 + 0.1\times3 + 0.05\times2 + 0.05\times1$

$0.05 + 0.2 + 1.5 + 0.45 + 0.3 + 0.1 + 0.05$

$= 2.65$

cost $= 0.5\times1 + 0.1\times2 + 0.05\times3$

$+ 0.15\times1 + 0.1\times2 + 0.05\times3 + 0.05\times3$

$= 0.5 + 0.2 + 0.15 + 0.15 + 0.2 + 0.15 + 0.15$

$= 1.5$

cost $= 0.1\times1 + 0.5\times2 + 0.05\times2$

$+(0.15 + 0.1 + 0.05 + 0.05)\times2$

$= 0.1 + 1 + 0.1 + 0.7$

$= 1.9$

cost $= 0.05\times1 + 0.5\times2 + 0.1\times3$

$+(0.15\times2 + 0.1)\times3 + 0.05\times3 + 0.05\times1$

$= 0.05 + 1 + 0.3 + 0.3 + 0.3 + 0.05 + 0.05$

$= 2.15$

cost $= 0.5\times1 + 0.1\times3 + 0.05\times2$

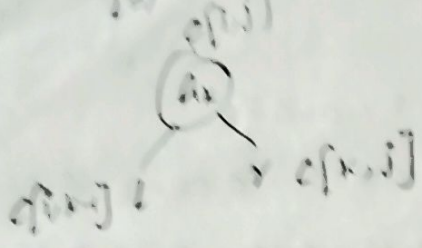
$+ 0.15\times1 + 0.1\times3 + 0.05\times3 + 0.05\times2$

$= 0.5 + 0.3 + 0.1 + 0.15 + 0.3 + 0.15 + 0.1$

$= 1.6$

Knuth's method

$$\text{cost}[i,m] = \sum_{i=i}^{m} p_i + (\text{level}(a_i)) + \sum_{i=0}^{m} q_i (\text{level}(i)-1)$$



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

if $(i=j)$ $c[i,j]=0$

the value of k for which we will get the min is the root.

$r(i,j) = k$.

$$w(i,j) = q(i) + \sum_{l=i+1}^{j} (p(l) + q(l))$$

w is cost of whole BST

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

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

Problem

$n=4$

$(a_1, a_2, a_3, a_4) = (7, 11, 19, 22)$

$p[1:4] = (3, 3, 1, 1)$

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

p's & q's are multiplied by 16.

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

$w(0,0) = q_0 = 2$, $w(1,1) = q_1 = 3$, $w(2,2) = q_2 = 1$,

$w(3,3) = q_3 = 1$, $w(4,4) = q_4 = 1$

(level(a_i)) + $\sum_{i=0}^{n}$ q_i (level(E_i) − 1)

j]

r c[x, j]

x−1] + c[x, j] } + w[i, j]

(i = j) c[i, j] = 0

for which we will get the min

(p[i] + q[i])

w is cost of whole BST

(j) + q(j)

q(i)

9, 22)

p's & q's are multiplied by 16.

3) = c(4,4) = 0

, w(2,2) = q_2 = 1 ,

q_4 = 1

$$\min_{0<x<4} \begin{cases} x=1 & c[0,0] + c[1,4] \\ x=2 & c[0,1] + c[2,4] \\ x=3 & c[0,2] + c[3,4] \\ x=4 & c[0,3] + c[4,4] \end{cases} + w[0,4]$$

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

$w_{01} = q_0 + p_1 + q_1 = 2 + 3 + 3 = 8$

$w_{12} = q_1 + p_2 + q_2 = 3 + 3 + 1 = 7$

$w_{23} = q_2 + p_3 + q_3 = 1 + 1 + 1 = 3$

$w_{34} = q_3 + p_4 + q_4 = 1 + 1 + 1 = 3$

$c[0,1] = \min \{ c[0,0] + c[1,1] \} + w[0,1] = 0 + 0 + 8 = 8$
$r(0,1) = 1$

$c[1,2] = \min \{ c[1,1] + c[2,2] \} + w[1,2] = 0 + 0 + 7 = 7$
$r[1,2] = 2$

$c[2,3] = \min \{ c[2,2] + c[3,3] \} + w[2,3] = 0 + 0 + 3 = 3$
$r[2,3] = 3$

$c[3,4] = \min \{ c[3,3] + c[4,4] \} + w[3,4] = 0 + 0 + 3 = 3$
$r[3,4] = 4$

$$c[0,2] = \min \begin{Bmatrix} c[0,0] + c[1,2] \\ c[0,1] + c[2,2] \end{Bmatrix} + w[0,2]$$

$w[0,2] = q_0 + p_1 + q_1 + p_2 + q_2 = 2+3+3+1+1 = 12$

$w[1,3] = q_1 + p_2 + q_2 + p_3 + q_3 = 3+3+1+1+1 = 9$

$w[2,4] = q_2 + p_3 + q_3 + p_4 + q_4 = 1+1+1+1+1 = 5$

$$c[0,2] = \min_{k=2}^{k=1} \begin{Bmatrix} 0+7+12 = 19 \\ 8+0+12 = 20 \end{Bmatrix} = 19$$

$r(0,2) = 1$

$$c[1,3] = \min_{k=3}^{k=2} \begin{Bmatrix} c[1,1] + c[2,3] \\ c[1,2] + c[3,3] \end{Bmatrix} + w[1,3]$$

$$= \min_{k=3}^{k=2} \begin{Bmatrix} 0+3+9 = 12 \\ 7+0+9 = 16 \end{Bmatrix} = 12$$

$r[1,3] = 2$

$$c[2,4] = \min_{k=4}^{k=3} \begin{Bmatrix} c[2,2] + c[3,4] \\ c[2,3] + c[4,4] \end{Bmatrix} + w[2,4]$$

$$= \min_{k=4}^{k=3} \begin{Bmatrix} 0+3+5 = 8 \\ 3+0+5 = 8 \end{Bmatrix} = 8$$

$r[2,4] = 3/4 = 3$

$w_{03} = q_0 + q_1 + p_1 + q_2 + p_2 + q_3 + p_3 = 2+3+3+3+1+1+1$

$= 14$

$w_{14} = q_1 + p_2 + q_2 + p_3 + q_3 + p_4 + q_4 = 3+3+1+1+1+1+1 = 11$

$$c[0,3] = \min \begin{Bmatrix} k=1: c[0,0] + c[1,3] \\ k=2: c[0,1] + c[2,3] \\ k=3: c[0,2] + c[3,3] \end{Bmatrix} + w[0,3]$$

$$= \min \begin{Bmatrix} 0+12+14 = 26 \\ 8+3+14 = 25 \\ 19+0+14 = 33 \end{Bmatrix} = 25$$

$r(0,3) = 2$

$$= \min \begin{cases} c[1,1]+c[2,4] \\ c[1,2]+c[3,4] \\ c[1,3]+c[4,4] \end{cases} + w[1,4]$$

$$= \min \begin{cases} 2 & 0+8+11 = 19 \\ 3 & 7+3+11 = 21 \\ 4 & 12+0+11 = 23 \end{cases} = 19$$

$r[1,4] = 2$

$w[0,4] = q_0+p_1+q_1+p_2+q_2+p_3+q_3+p_4+q_4 = 16$

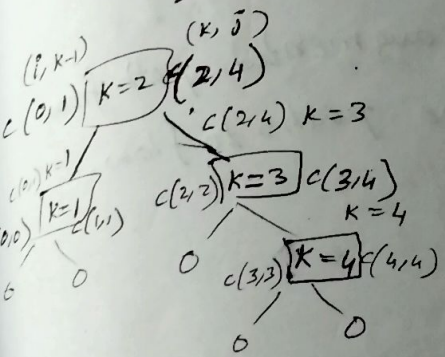
$$c[0,4] = \min \begin{cases} 1 & c[0,0]+c[1,4] \\ 2 & c[0,1]+c[2,4] \\ 3 & c[0,2]+c[3,4] \\ 4 & c[0,3]+c[4,4] \end{cases} + w[0,4]$$

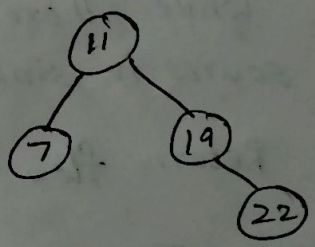
$$= \min \begin{cases} 0+19+16 = 35 \\ 8+8+16 = 32 \\ 19+3+16 = 38 \\ 25+0+16 = 41 \end{cases} = 32$$

$r[0,4] = 2$

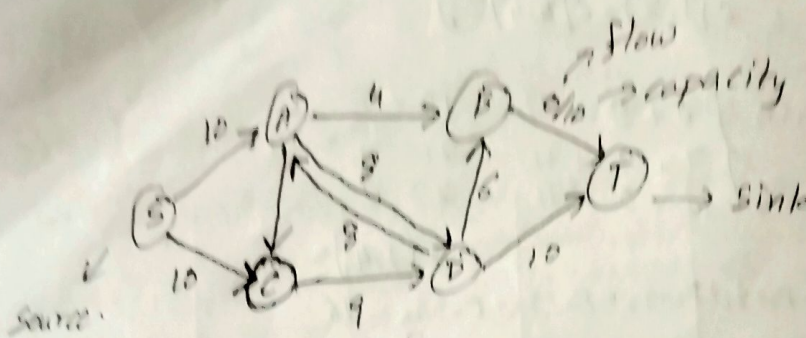
Now finally $c[0,4] = 32$ because of $k=2$, $r[0,4]=2$

so root is $k=2$ i.e 11

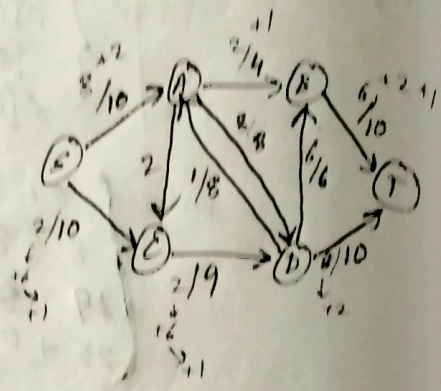




Ford Fulkerson Algorithm



Path	Flow (Max)
S→A→B→T	8
S→C→D→T	2
S→C→D→B→T	6
S→A→B→T	2
S→C→D→A→B→T	1



3 steps of ford fulkerson algorithm

i) Initially flow is 0

ii) while there is an augmented path from source to sink, add two paths to flow.

iii) Return flow

Time complexity

$O(E * F)$

- E → No. of edges
- F → Maximum flow.

Backtracking

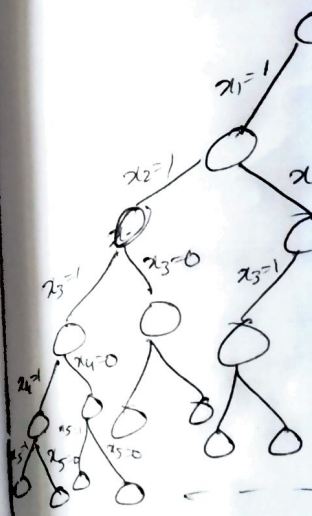
- Backtracking is one of the algorithm design techniques.
- Many problems which deal with searching for set of solutions or which ask for an optimal solution satisfies some constraints can be solved by using backtracking.
- The desired solution of the backtracking process is expressed as n-tuple $(x_1, x_2, \ldots, x_n)$ where value of $x_i \in S_i$
- The problem to be solved calls for finding one vector that maximises or minimizes a criterion function $P(x_1, x_2 \ldots x_n)$
- Many problems that are solved using backtracking require that all the solutions satisfy a complex set of constraints and these constraints are explicit and implicit.
- Explicit constraints are rules that restrict each x_i to take the values only from given set.

  Ex: $x_i \geq 0$ then $S_i = \{$all non -ve numbers$\}$

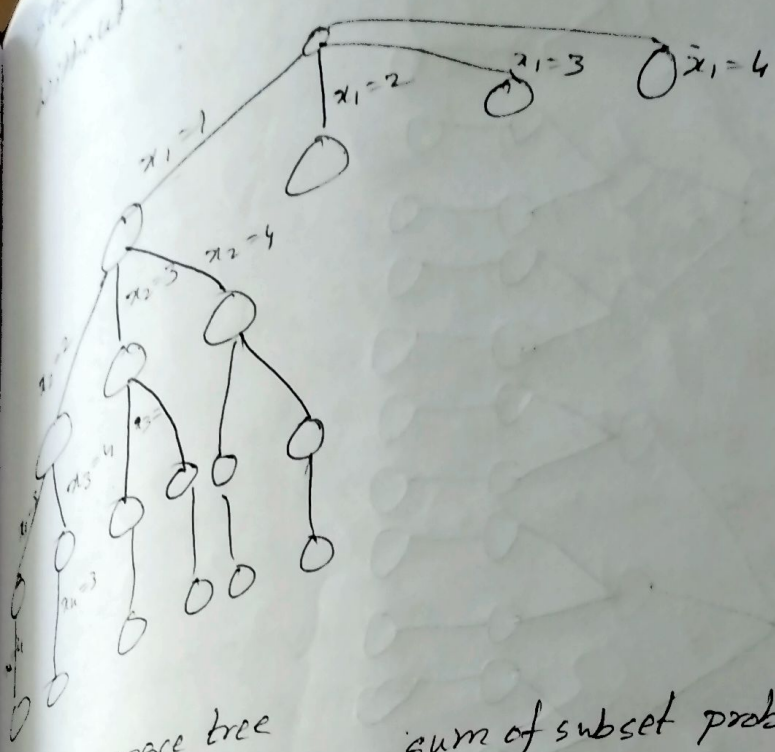
  $x_i = 0$ or 1 then $S_i = \{0, 1\}$

  These constraints depend on particular instance of problem being solved.
- ~~Implicit constraints~~
- All tuples that satisfy explicit constraints define a possible solution space for I (instance)
- Implicit constraints are the rules that determine which of the tuples in the solution space of I satisfy the criterion function.

  Thus, the implict constraints describe the way in which x_i's must relate to each other.



state space tree
without implicit conditions



total
24 leaf nodes.

state space tree
If we consider

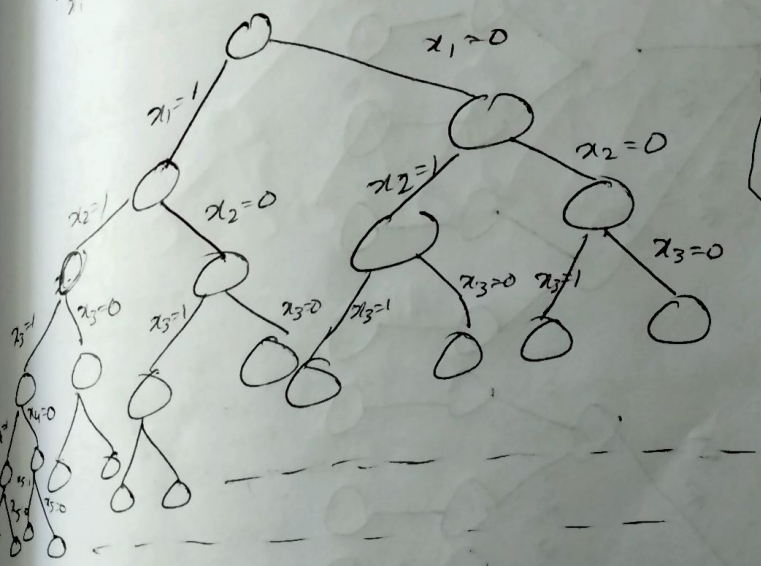
sum of subset problem.

$\{1, 2, 3, 5, 6\}$
$x_1\ x_2\ x_3\ x_4\ x_5$

sum = 6

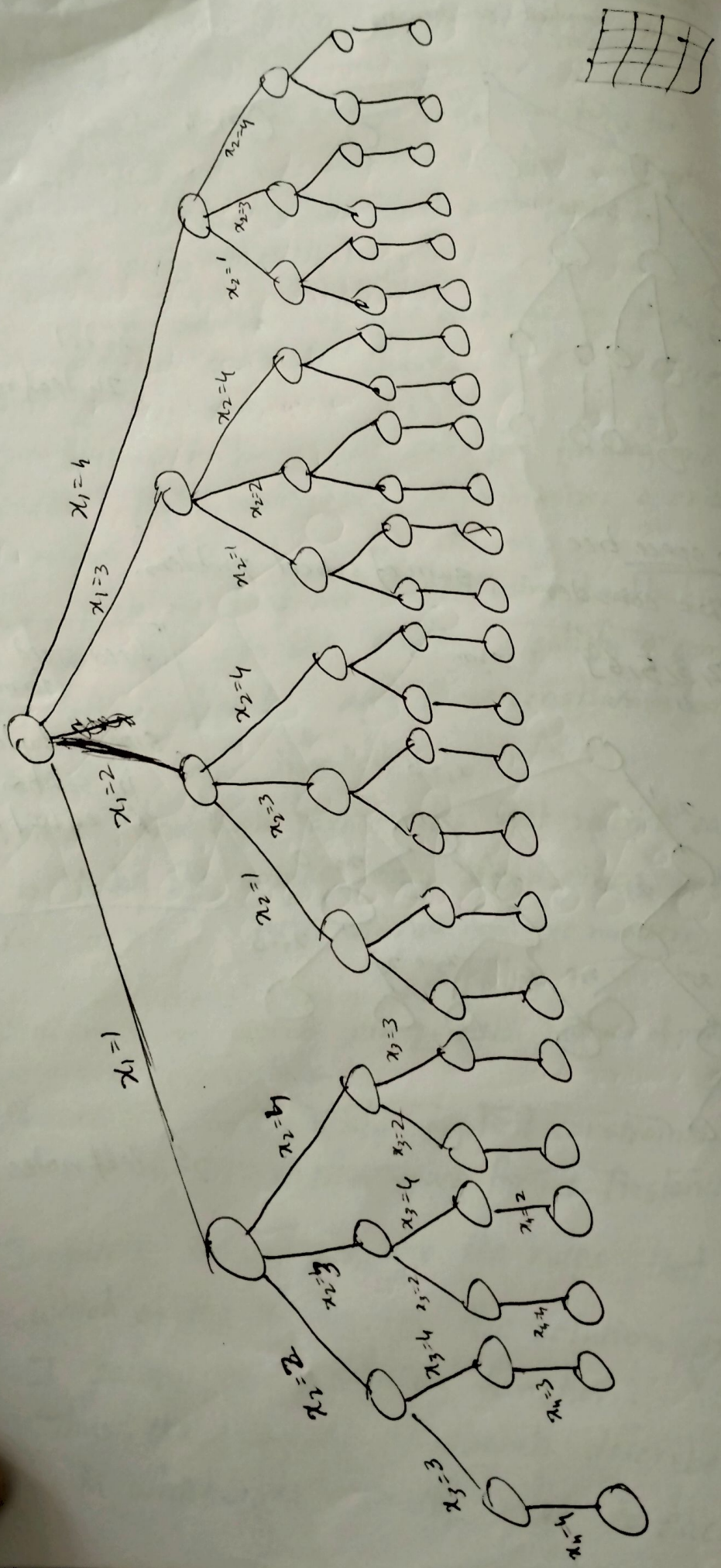
fixed sized formulation

size of tuples in solution set is equal.



25 leaf nodes

• State space tree for 4 queens problem



• Each node in the
• All paths from root
state space of the
• Solution states are
the path from
solution space.
• Answer states a
path from ro
member of
or simply t
satisfying th
• The tree orga
referred to

Live node :
It is a
all of wh
generated

E-Node
It is a
being

Dead node
It is
expl
have

Promis
A
a
It
sti
An
the

Each node in the tree defines a problem state.
All paths from root to other nodes defines the state space of the problem.
Solution states are those problem states s for which the path from root to s defines a tuple in the solution space.
Answer states are those solution states s with the path from root to s defines a tuple that is a member of set of solutions of the problem.
or simply those solution states which are satisfying the implicit constraints.
The tree organisation of the solution space is referred to as state space tree.

Live node:
It is a node that has been generated and all of whose children are not yet been generated.

E-Node
It is a live node whose children are currently being generated.

Deadnode
It is a node that is either not to be expanded further or for which all its children have been generated is known as a dead node.

Promising node
A node is promising if it eventually leads to a desired solution
It corresponds to a partial solution that is still feasible.
Any time the partial node becomes infeasible. then the branch will no longer be pursued.

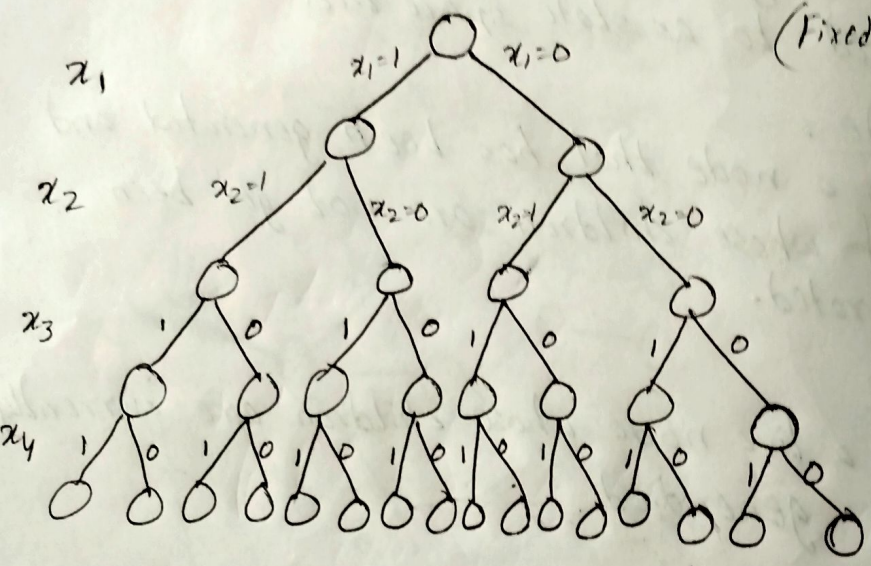
Non promising node

- A node is non-promising if it eventually leads to a state that cannot produce the desired solution.
- It corresponds to a partial solution that shows infeasibility to get a complete solution. such nodes are killed by a bounding function without further exploration.

Draw a state space tree for sum of subsets problem where $n=4$ $(w_1, w_2, w_3, w_4) = (11, 13, 24, 7)$

sum of subsets = 31.

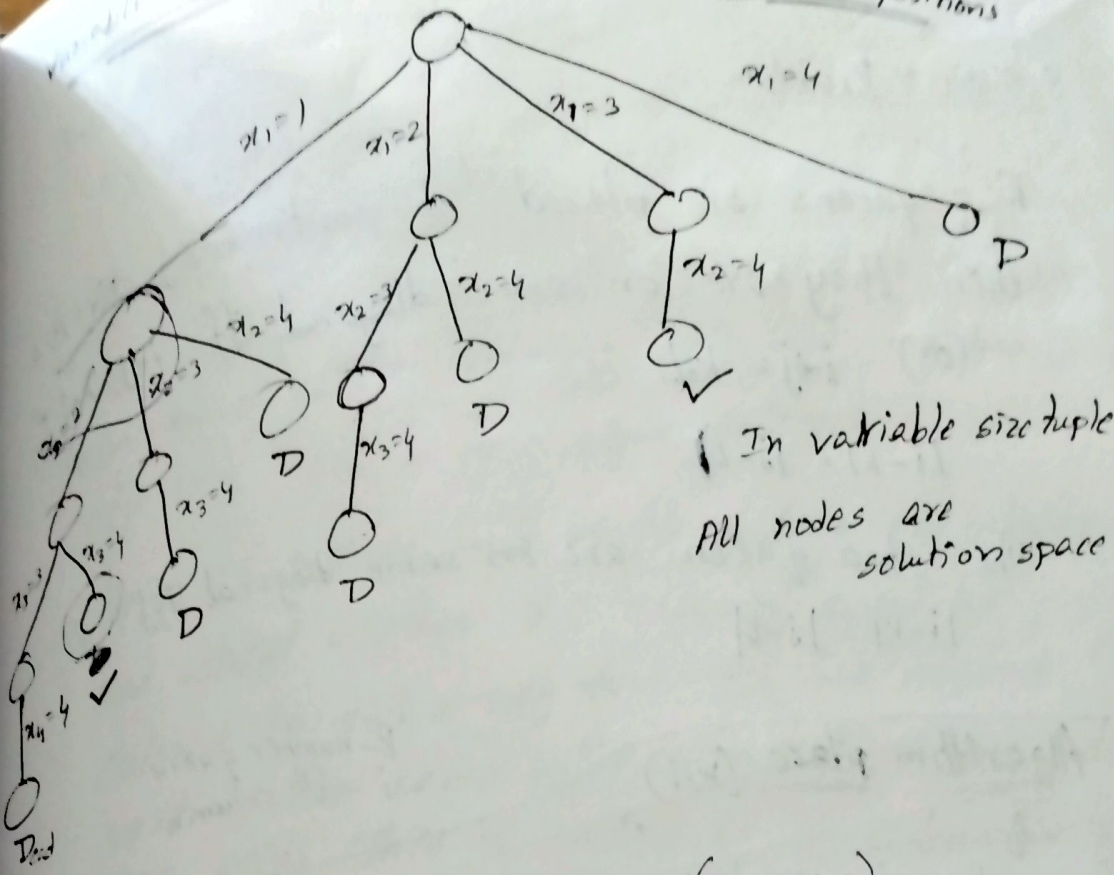
(Fixed size tuple)



All leaf nodes are solution space
Nodes which give answer are answer states

here x_1, x_2, x_3, x_4 are in increasing order

$x_1\ x_2\ x_3\ x_4$ are positions



In variable size tuple

All nodes are solution space

Algorithm Backtracking (k) (Recursive)

/* this scheme describes the backtracking process using recursion. On entering, the first k-1 values $x_1, x_2 \ldots x_{k-1}$ of the solution vector x_1 to x_n have been assigned $x[]$, n */ k-queen no.

```
{
  for each x[k] ∈ T(x[1], x[2] ... x[k-1]) do
  {
    if (B_k(x_1, x_2 ... x_k) ≠ 0) then
    {
    if (x[1], x[2] ..... x[k]) is a path to the
          answer node .
      then
          write (x[1:k])
    } if (k<n) then Backtracking (k+1);
  }
}
```

Iterative procedure also same way.

8 Queens Problem

If 2 queens are placed at positions (i,j) and (k,l) then they are on same diagonal iff $i-j=k-l$

(or) $i+j=k+l$.

$|i-k| = |j-l|$

The two queens are on same diagonal iff.

$|i-k| = |j-l|$

k - number queen
i - i^{th} column

Algorithm place (k,i)
{
for j:= 1 to k-1 do
if (x[j]= i) (or) Abs [x[j] -i) ~~Abs (j-k)~~

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;
}

O(k-1)

n-Queen's

Algorithm nqueens (k,n)
{
for i:=1 to n do
{ if place(k,i) then
{
x[k]:= i;
if (k=n) then write x[1:n];
} } else nqueens (k+1,n)

time complexity of place algorithm is $O(k-1)$.

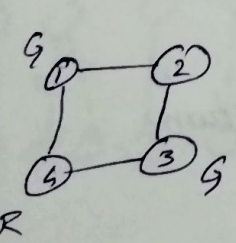
Initially algorithm n queens is called as nqueens(1,n).

Graph Colouring problem.

Given a graph with n vertices we need to find the min no. of colours, required such that no two adjacent vertices have same color.

→ Let G be a graph and m be a given +ve integer find whether the nodes of G can be colored in such a way that no two adjacent vertices are same color yet only m colors are used. This is called as m colourability decision problem.

→ m-colourability optimisation problem looks for smallest integer m for which the graph G can be coloured and this integer is referred as chromatic no. of the graph.

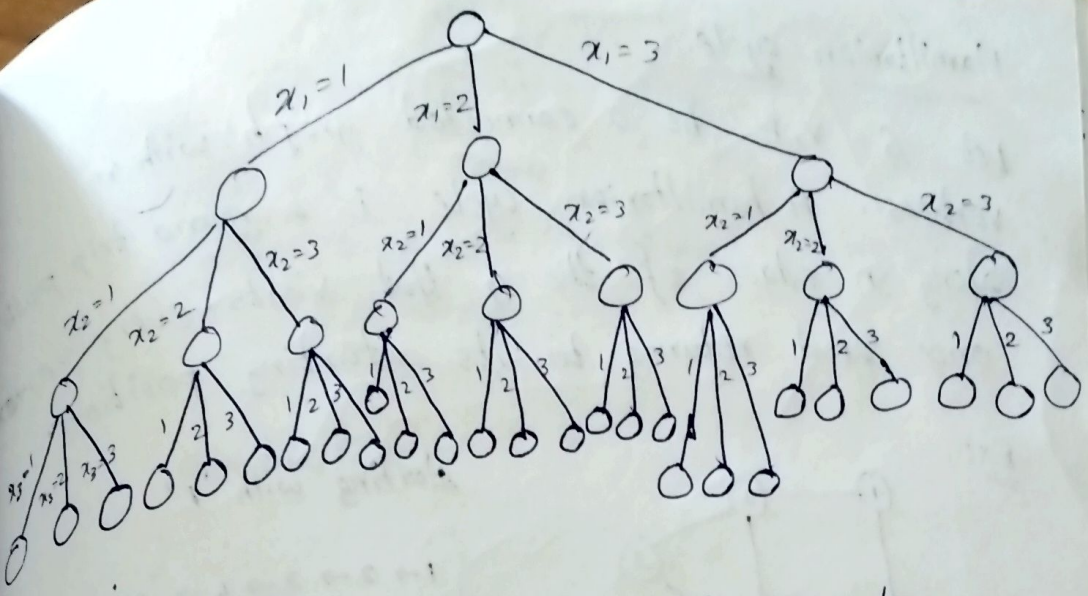


chromatic number = 2

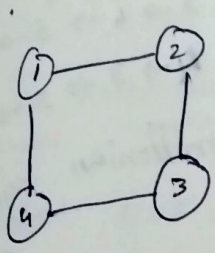
→ Suppose we represent a graph G by adjacency matrix $G[1:n, 1:n]$ where $G[i,j]=1$ if i,j is an edge in G. $G[i,j]=0$ if i,j is not an edge and the colours are represented by integers 1, 2, ... m. And the solutions are given by n tuple $(x_1, x_2 \ldots x_n)$ where x_i is color of node i.

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

- Draw the state space tree for m coloring when n=3 and m=3. (without adjacency)

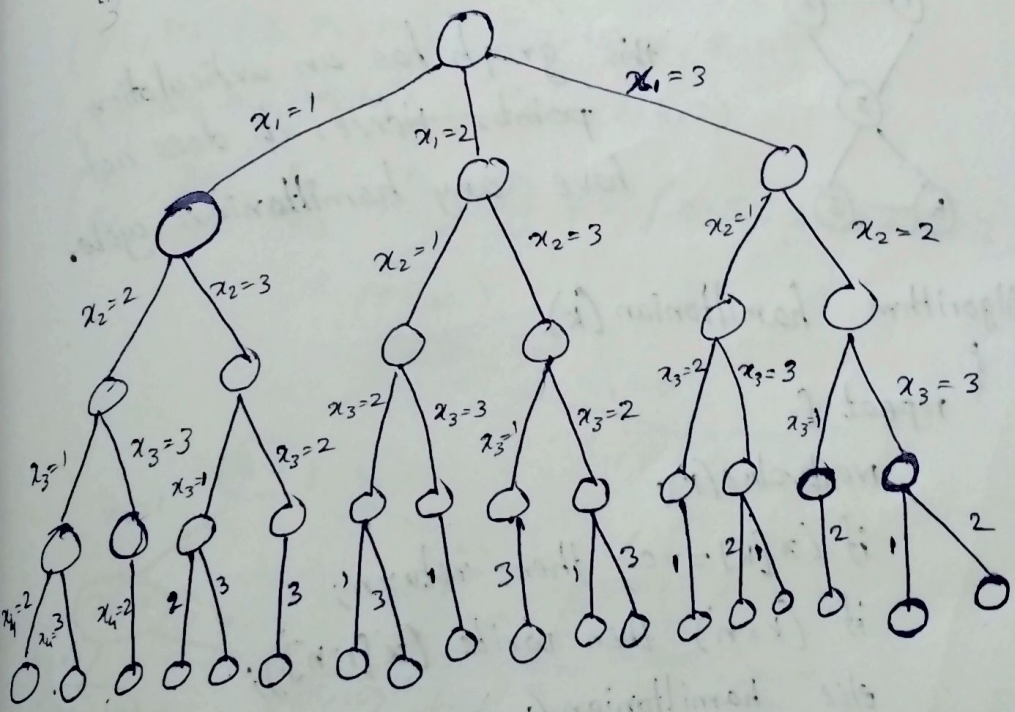


Ex:



$n = 4$
$m = 3$

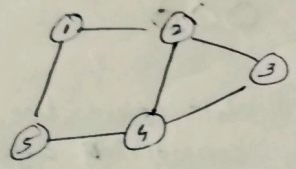
state space tree considering adjacency.



Hamiltonian cycle

Let $G = V, E$ be a connected graph with n vertices. A hamiltonian cycle is a round trip path along n edges of the G that visits every vertex once and returns to its starting position.

Ex:

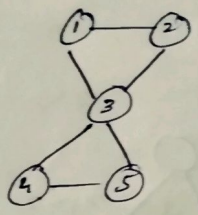


starting with '1'

1 → 2 → 3 → 4 → 5 → 1

1 → 5 → 4 → 3 → 2 → 1

are hamiltonian cycles.



this graph has an articulation point. Hence, it does not have any hamiltonian cycle.

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

Algorithm Nextvalue(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
    {
```

cted graph with n
le is a round trip path
that visits every vertex
starting position.
starting with '1'
→ 2 → 3 → 4 → 5 → 1
→ 5 → 4 → 3 → 2 → 1
hamiltonian cycles.
as an articulation
ence, it does not
hamiltonian cycle.

```
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);
}
```

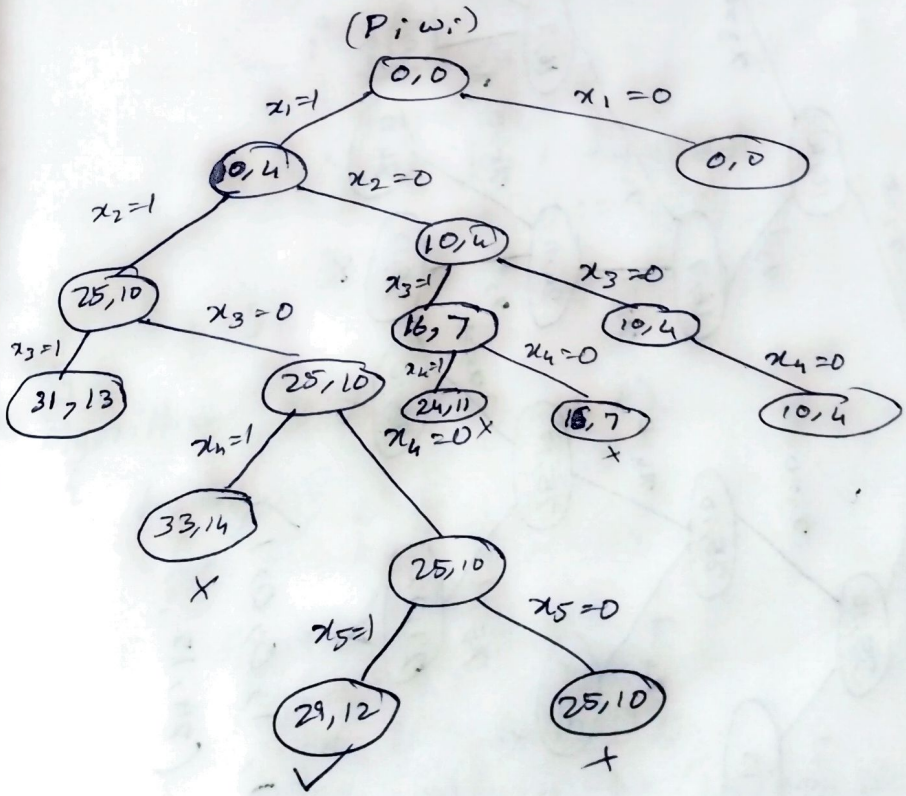
0-1 knapsack Using Backtracking

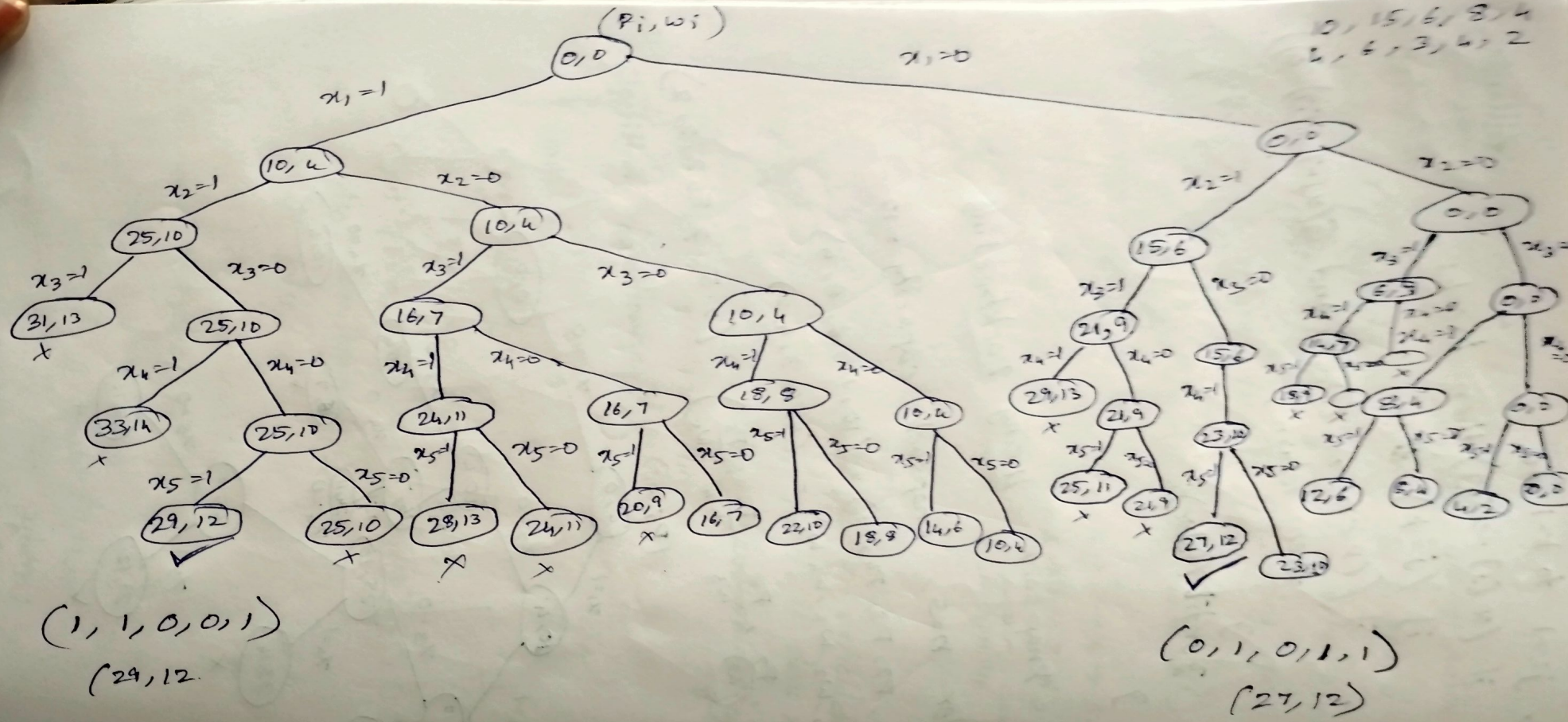
consider $n=5$, knapsack capacity $=12$.

$(P_1, P_2, P_3, P_4, P_5) = (10, 15, 6, 8, 4)$

$(w_1, w_2, w_3, w_4, w_5) = (4, 6, 3, 4, 2)$

$\left(\frac{P_1}{w_1}, \frac{P_2}{w_2}, \frac{P_3}{w_3}, \frac{P_4}{w_4}, \frac{P_5}{w_5}\right) = (2.5, 2.5, 2, 2, 2)$




(Pi, wi)
0, 15, 6, 8, 4
2, 6, 3, 4, 2
0,0
x1=1
x1=0
10,4
25,10
31,13
33,14
29,12
25,10
16,7
24,11
28,13
24,11
16,7
20,9
16,7
10,4
18,5
22,10
18,8
10,4
14,6
10,4
15,6
21,9
29,13
25,11
21,9
5,6
23,12
27,12
23,10
(1, 1, 0, 0, 1)
(29, 12
(0, 1, 0, 1, 1)
(27, 12)

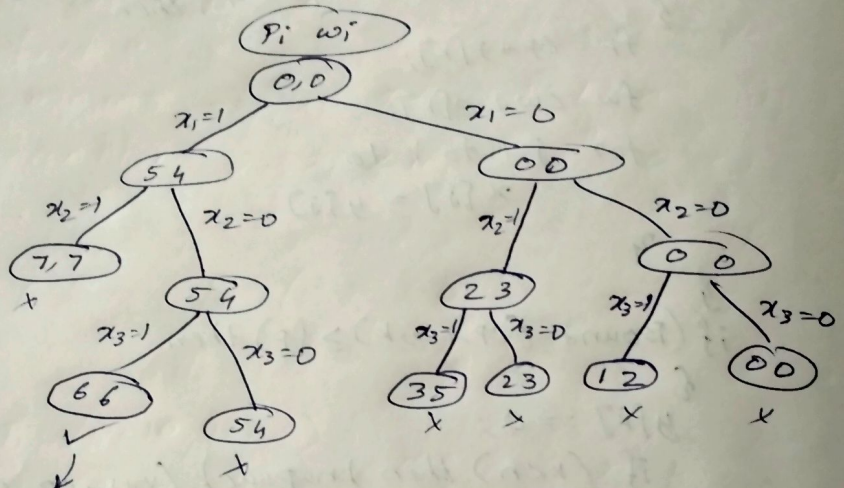
```
Algorithm Knapsack (k, cp, cw)                    [fp - final profit
{
  if (cw+w[k] ≤ m) then
  {
    y[k] := 1;
    if (k<n) then Knapsack(k+1, cp+p[k], cw+w[k]);
    if (cp+p[k] > fp) and (k==n) then
    {
      fp = cp+p[k];
      fw = cw+w[k];
      for j=1 to k do
          x[j] = y[j]
    }
  }
  if (Bound (cp, cw, k) ≥ fp) then
  {
    y[k] := 0;
    if (k<n) then knapsack (k+1, cp, cw);
    if ((cp > fp) and (k = n) then
    {
      fp := cp;
      fw := cw;
      for j := 1 to k do
          x[j] := y[j];
    }
  }
}
Algorithm Bound (cp, cw, k)
{
  b := cp; c := cw;
  for i := k+1 to n do
  {
    c := c + w[i];
    if (c < m)
    then b := b + p[i];
    else return b + (1-(c-m)/w[i]) * p[i]
  }
}
```

(2) Given $n=3$, $m=6$, $w=(2,3,4)$
$p=(1,2,5)$

$\frac{p_i}{w_i} = (0.5, 0.66, 1.25)$

$(p_1, p_2, p_3, p_4) = (5, 2, 1)$

$(w_1, w_2, w_3, w_4) = (4, 3, 2)$



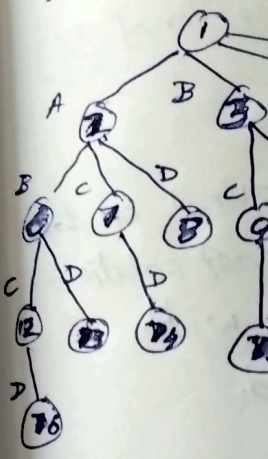
It is only possible value.

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

Branched Boun
in which al
generated
the E-node
there are

It follo
BFS in cre
state space
and uses q
data structu
to store nod

Example: Giv
set ABCD

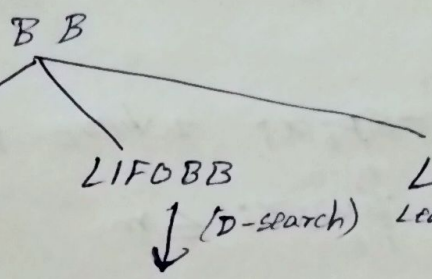


LCBB

This met
state sp
In this
node
explo
nodes

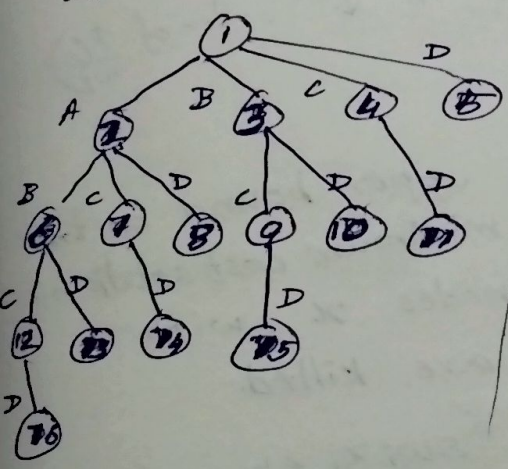
Branch and Bound refers to all state space search method in which all the children of the E-node are generated before any other live node can become the E-node.

There are 3 methods



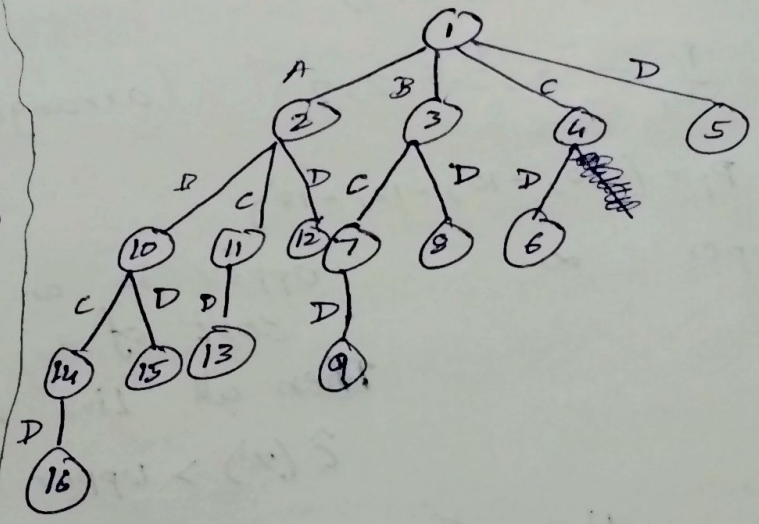
It follows BFS in creating state space tree and uses queue data structure. to store nodes.

Example: Given set ABCD



It follows stack approach to create state space tree

Example:
Given ABCD



LCBB

This method uses the cost function to explore state space tree.

In this method after the children of a particular node have been explored, the next node to be explored would be that node out of all unexplored nodes which has the least cost.

0/1 knapsack using LCBB

Branch and bound problem is a minimisation problem so the given knapsack problem has to be first converted into minimisation problem. i.e. for 0/1 knapsack problem, the problem is defined as

minimize $-\sum p_i x_i$ where $i = 1$ to n subject to condition $\sum_{i=1}^{n} w_i x_i \leq m$ $x_i = 0$ or 1 $1 \leq i \leq n$

- Let $n = 5$, $P_i = (10, 15, 6, 8, 4)$ $m = 12$

  $w_i = (4, 6, 3, 4, 2)$

- $P_i = (10, 10, 12, 18)$

  $w_i = (2, 4, 6, 9)$, $m = 15$, $n = 4$

  $\frac{P_i}{w_i} = (5, 2.5, 2, 2)$ (arrange in $\downarrow^{sin}$ order of $\frac{P_i}{w_i}$)

  $P_i = (-10, -10, -12, -18)$

upper $= \infty$ upper is an upper bound on the cost of a minimum cost solution. Then all live nodes x with $\hat{c}(x) >$ upper are killed.

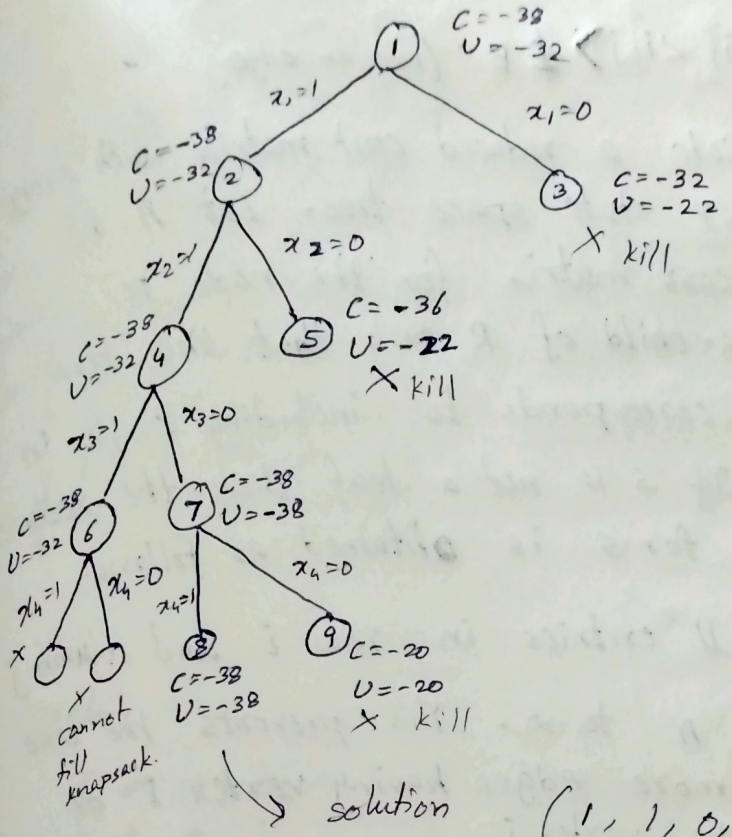
$U = \sum p_i x_i$ subject to $\sum w_i x_i \leq m$

$C = \sum p_i x_i$ subject to $\sum w_i x_i \leq m$ (fractional knapsack problem)

↳ including fractional values

(1) $C = \sum p_i x_i = -10 - 10 - 12 - 18 \times \frac{3}{9} = -32 - 6 = -38$

$U = \sum p_i x_i = -10 - 10 - 12 = -32$



Initial upper = -32

After 7th node upper = -38

kill nodes whose cost value is > upper

→ solution

(1, 1, 0, 1)

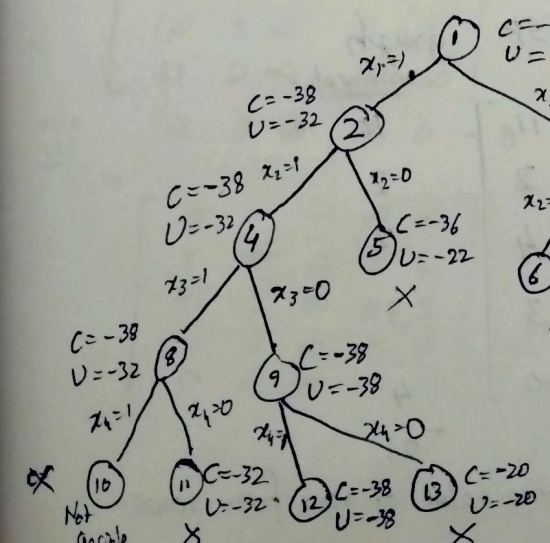
(-10, -10, 0, -18)

(2, 4, 0, 9)

0/1 knapsack FIFOBB

$m = 5$, $n = 4$, $P_i = (10, 10, 12, 18)$ upper = ∞

$w_i = (2, 4, 6, 9)$



2	3	4	5	6	7	8	9

At 9th upper = -38

→ ✓ soln = (1, 1, 0, 1)

TSP using LCBB

Let G(V,E) be a directed graph defining an instance of TSP. Let c_{ij} is cost of edge $\langle i,j \rangle$.

$c_{ij} = \infty$ if $\langle i,j \rangle \notin E$ (Not an edge)

We can associate a reduced cost matrix with every node in TSP state space tree. Let A be a reduced cost matrix for the node R. Let S be a child of R such that the tree edge (R,S) corresponds to including $\langle i,j \rangle$ in the tour. If S is not a leaf then the reduced cost matrix for S is obtained as follows:

1. Change all entries in row i and column j of matrix A to ∞. This prevents the use of any more edges having vertex i or entering the vertex j.
2. Set $A\langle j,1 \rangle$ to ∞ this prevents the use of the edge j,1.
3. Reduce all the rows and columns in resulting matrix except the rows and columns containing only ∞.

- Consider the following TSP graph

					min value
∞	20	30	10	11	10
15	∞	16	4	2	2
3	5	∞	2	4	2
19	6	18	∞	3	3
16	4	7	16	∞	4
					21

Subtract min value of row from rows.

Reduced Matrix

$$\begin{bmatrix} \infty & 10 & 20 & 0 & 1 \\ 12 & \infty & 14 & 2 & 0 \\ 1 & 3 & \infty & 0 & 2 \\ 16 & 3 & 15 & \infty & 0 \\ 12 & 0 & 3 & 12 & \infty \end{bmatrix}$$

$1 \quad 0 \quad 3 \quad 0 \quad 0 \Rightarrow 4$

min value.

reduced cost matrix

$$\begin{bmatrix} \infty & 10 & 17 & 0 & 1 \\ 12 & \infty & 11 & 2 & 0 \\ 0 & 3 & \infty & 0 & 2 \\ 15 & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & 12 & \infty \end{bmatrix}$$

$21+4 = 25$ → r = reduced cost.

Cost value of node 1 = 25

$\hat{c}(1) = 25$

$\hat{c}(2) = \hat{c}(1) + A(i,j) + r$ (edge cost, reduced cost; $A(i,j)$: $i=1, j=2$)

Node 2

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 11 & 2 & 0 \\ 0 & \infty & \infty & 0 & 2 \\ 15 & \infty & 12 & \infty & 0 \\ 11 & \infty & 0 & 12 & \infty \end{bmatrix}$$

$\hat{c}(2) = 25 + 10 + 0 = 35$

Node 3

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

min val

$11 \quad 0 \quad 0 \quad 0 \quad 0 = 11$

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

$\hat{c}(3) = \hat{c}(1) + A(i,j) + r$ ($A(i,j)$: $i=1, j=3$)

$= 25 + 17 + 11$

$\hat{c}(3) = 53$

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

$\hat{c}(4) = \hat{c}(1) + A(1,4) + r$

$= 25 + 0 + 0$

$\hat{c}(4) = 25$

Nodes

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

5

reduced

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

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

$= 25 + 145$

$\hat{c}(5) = 31$

min of $\hat{c}(2), \hat{c}(3), \hat{c}(4), \hat{c}(5) = 25$ of $\hat{c}(4)$

expand (4)

Node 6

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

$\hat{c}(6) = \hat{c}(4) + A(4,3) + r$

$= 25 + 3 + 0$

$\hat{c}(6) = 28$

Node 7

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

min 11

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

$= 25 + 12 + 13$

$\hat{c}(7) = 50$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 1 & \infty & \infty & \infty & 0 \\ \infty & 3 & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty \\ 0 & 0 & \infty & \infty & \infty \end{bmatrix} \quad \begin{matrix} \text{min} \\ 2 \end{matrix}$$

Node 8

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

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

Node 9

Node 9

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

min 11

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

Node 10

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

reduced

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

$\hat{c}(8) = \hat{c}(4) + A(4,5)$

Node 8

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 11 & \infty & \infty \\ 12 & \infty & 3 & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & 0 & \infty & \infty \end{bmatrix} \quad \text{min} = 11$$

$\hat{c}(8) = \hat{c}(4) + A(4,5) + r$

$= 25 + 0 + 11$

$= 36$

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

Node 6 min of $\hat{c}(6)$ $\hat{c}(7)$ $\hat{c}(8)$ = $\hat{c}(6) = 28$

Node 9

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty \\ 11 & \infty & \infty & \infty & \infty \end{bmatrix} \quad \text{min} = 11, \quad 2$$

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

$= 28 + 11 + 13$

$= 52$

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

Node 10

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

$\hat{c}(10) = \hat{c}(6) + A(2,5) + r$

$= 28 + 0 + 0$

$= 28$

min of $\hat{c}(9)$, $\hat{c}(10) = 28 = \hat{c}(10)$

Node 11

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

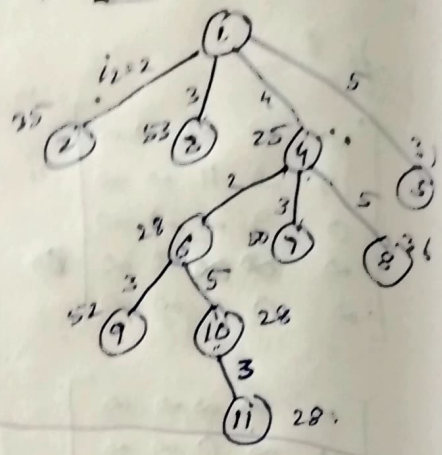
The path is

1 → 4 → 2 → 5 → 3 → 1

$\hat{c}(11) = \hat{c}(10) + A(5,3) + r$

$= 28 + 0 + 0$

$\hat{c}(11) = 28$



					min row
∞	7	3	12	8	3
3	∞	6	14	9	3
5	8	∞	6	18	5
9	3	5	∞	11	3
18	14	9	8	∞	8

obtained reduced cost matrix, state space tree & path

∞	4	0	9	5	
0	∞	3	11	6	
0	3	∞	1	13	
6	0	2	∞	8	
10	6	1	0	∞	
min	0	0	0	0	5

→

∞	4	0	9	0
0	∞	3	11	1
0	3	∞	1	8
6	0	2	∞	3
10	6	1	0	∞

$r = 22 + 5 = 27$

$\hat{c}(1) = 27$

Node 2

∞	∞	∞	∞	∞
∞	∞	3	11	1
0	∞	∞	1	8
6	∞	2	∞	3
10	∞	1	0	∞
min	0	1	0	1

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

$= 27 + 4 + 3$

$= 34$