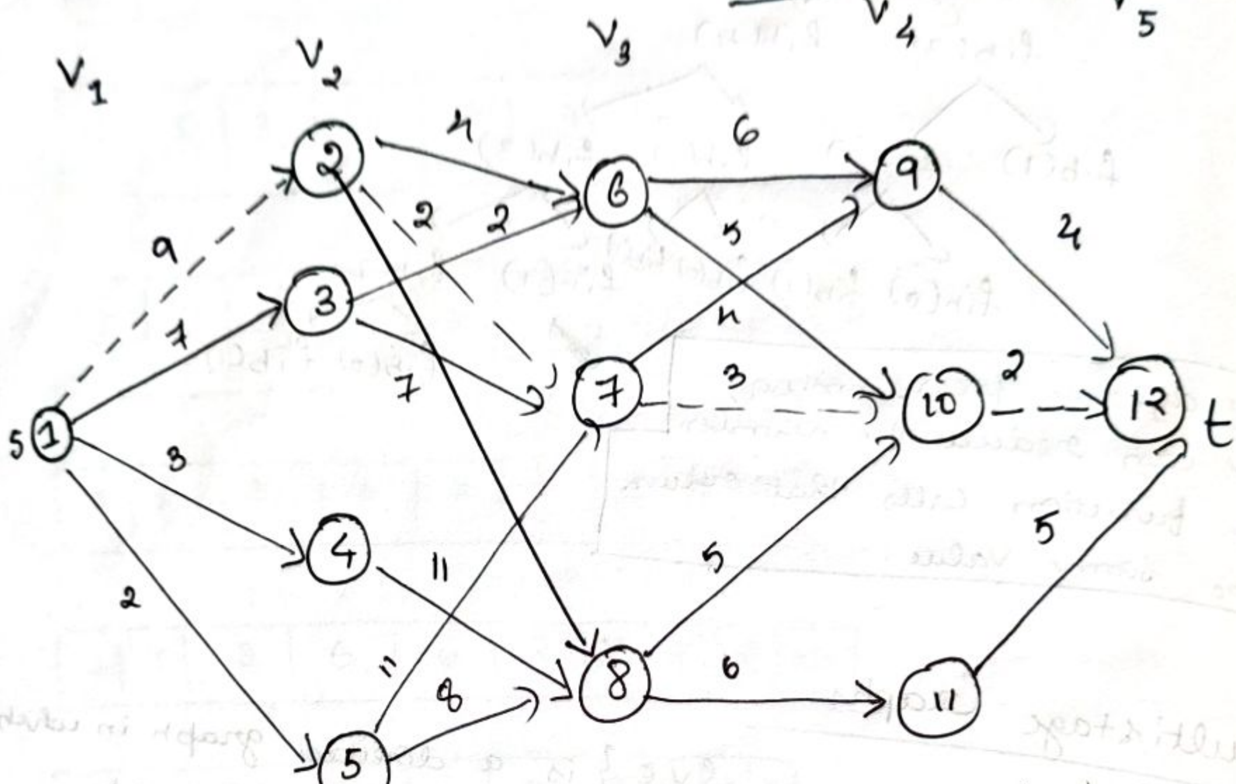
Formula to calculate cost is:

$$cost(i, j) = \min \left\{ c(j, l) + cost(i+1, l) \right.$$

i → stage number.
j → current vertex
l → next vertex.

Five-stage graph.



Calculate minimum cost path from 's' to 't'.

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
~~destination~~ d	2/3	7	6	8	8	10	10	10	12	12	12	12.

Stage 3.

6. $cost(3,6) = \min\begin{cases} c(6,9) + cost(4,9) \\ c(6,10) + cost(4,10) \end{cases}$

$= \min\{6+4, 5+2\}$

$= \min\{10, 7\} = \underline{7}$.

7. $cost(3,7) = \min\begin{cases} c(7,9) + cost(4,9), \\ c(7,10) + cost(4,10) \end{cases}$

$\min\{4+4, 3+2\}$.

$\min\{8, 5\} = \underline{5}$.

8. $cost(3,8) = \min\begin{cases} c(8,10) + cost(4,10) \\ c(8,11) + cost(4,11) \end{cases}$

$\min\{5+2, 6+5\}$.

$\min\{7, 11\} = \underline{7}$.

Stage 2.

2. $cost(2,2) = \min\begin{cases} c(2,6) + cost(3,6) \\ c(2,7) + cost(3,7) \\ c(2,8) + cost(3,8) \end{cases}$

$= \min\{4+7, 2+5, 1+7\}$.

$= \min\{11, 7, 8\}$.

3. $cost(2,3) = \min\begin{cases} c(3,6) + cost(3,6) \\ c(3,7) + cost(3,7) \end{cases}$

$= \min\{2+7, 7+5\}$

$= \min\{9, 12\} = \underline{9}$.

5. $cost(2,5) = \min\begin{cases} c(5,7) + cost(3,7) \\ c(5,8) + cost(3,8) \end{cases}$

$= \min\{11+5, 8+7\}$.

$\min\{16, 15\} = 15$//.

Stage 1:

1.

$$c(1,1) = \min \begin{cases} c(1,2) + cost(2,2) \\ c(1,3) + cost(2,3) \\ c(1,4) + cost(2,4) \\ c(1,5) + cost(2,5) \end{cases}$$

$c(1,5) + cost(2,5)$

$\min\{9+7, 7+9, 3+18, 2+15\}$. $= 16$.

$\min\{16, 16, 21, 17\}$.

Stage 2.

4. $c(2,4) = \min\{c(4,8) + c(3,8)\}$.

$\min\{11+7\} = 18$.

stage no, vertex

Minimum cost path, $d(1,1) = 2$.

$d(2,2) = 7$

$d(3,7) = 10$.

$d(4,10) = 12$.

$\rightarrow 1 \rightarrow 2 \rightarrow 7 \rightarrow 10 \rightarrow 12$.

or.

$d(1,1) = 3$

$d(2,3) = 6$.

$d(3,6) = 10$

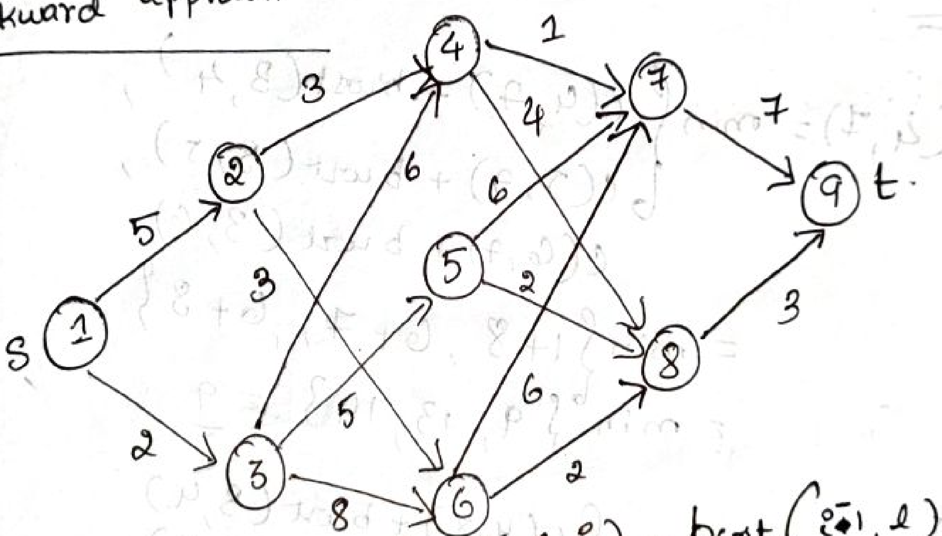
$d(4,10) = 12$.

$d(5,12) = 12$.

$1 \rightarrow 3 \rightarrow 6 \rightarrow 10 \rightarrow 12$.

Algorithm FGraph (G , k , n , p)
// The input is a k-stage graph G = (V,E) with n vertices
// indexed in order of stages. E is a set of edges and c[i,j]
// is the cost of ⟨i, j⟩. p[1 : k] is a minimum cost path.
{
cost [n] := 0.0;
for j := n-1 to 1 step -1 do
{ // Compute cost[j]
Let r be a vertex such that ⟨j, r⟩ is an edge of G and c[j, r] + cost[r] is minimum;
cost [j] := c [j, r] + cost[r];
} d[j] := r;
// Find a minimum cost path.
p[1] := 1 ; p[k] := n;
for j := 2 to k-1 do p[j] := d[p[j-1]];
}

Backward approach.



$$bcost(i,j) = \min \{ c(l,j) + bcost(i-1, l).$$

v	1	2	3	4	5	6	7	8	9
Cost	0	5	2	8	7	8	9	9	12
d	1	1	1	2/3	3	2	4	5	8

Stage 3.

4

$$bcost(3,4) = \min \begin{cases} c(2,4) + bcost(2,2), \\ c(3,4) + bcost(2,3) \end{cases}$$

$$= \min \{3+5,\ 6+2\}$$

$$= \min \{8, 8\} = \underline{8}$$

5

$$bcost(3,5) = \min \{c(3,5) + bcost(2,3)\}$$

$$= \min \{5 + 2\}$$

$$= \min \{7\} = \underline{7}.$$

6

$$bcost(3,6) = \min \begin{cases} c(2,6) + bcost(2,2), \\ c(3,6) + bcost(2,3) \end{cases}$$

$$= \min \{3+5,\ 8+2\}$$

$$= \min \{8, 10\}. = \underline{8}$$

Stage 4:

7

$$bcost(4,7) = \min \begin{cases} c(4,7) + bcost(3,4), \\ c(5,7) + bcost(3,5), \\ c(6,7) + bcost(3,6) \end{cases}$$

$$= \min \{1+8,\ 6+7,\ 6+8\}.$$

$$= \min \{9, 13, 14\}. = \underline{9}$$

8.

$$bcost(4,8) = \min \begin{cases} c(4,8) + bcost(3,4) \\ c(5,8) + bcost(3,5) \\ c(6,8) + bcost(3,6) \end{cases}$$

$$= \min \{4+8,\ 2+7,\ 2+8\}$$

$$\min \{12, 9, 10\}. = \underline{9}$$

Stage 5.

9

$$bcost(5,9) = \min \begin{cases} c(7,9) + bcost(4,7). \\ c(8,9) + bcost(4,8). \end{cases}$$

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

$$= \min \{16, 12\}. = 12$$

Minimum cost path,

$d(5,9) = 8$.

$d(4,8) = 5$.

$d(3,5) = 3$.

$d(2,3) = 1$

~~Path ⇒ 1 → 3 → 5 → 8~~

Path ⇒ 9 → 8 → 5 → 3 → 1

Cost = 12.

Time complexity $\Theta(|V| + |E|)$.

Algorithm BGraph(G, k, n, p).

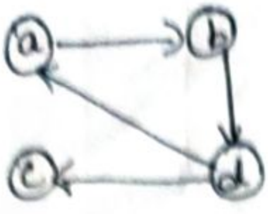
// Same function as FGraph.

```
{   bcost[1] := 0.0;
    for j := 2 to n do.
    {
      // Compute bcost[j]
         Let r be such that {r, j} is an edge
      of G and bcost[r] + c[r, j] is minimum;
      bcost[j] := bcost[r] + c[r, j];
      d[j] := r;
    }
  // Find a minimum - cost path.
    p[1] := 1 ; p[k] = n;
  for j := k-1 to 2 do p[j] := d[p[j+1]];

}
```

1	2	3	4	5	6	7	8
10	9	9	6	6	1	0	

Warshall's algorithm using dynamic Programming



(a) Digraph

$$A = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 0 & 0 \\ b & 0 & 0 & 0 & 1 \\ c & 0 & 0 & 0 & 0 \\ d & 1 & 0 & 1 & 0 \end{array}$$

(b) Adjacency matrix

$$T = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 1 & 1 & 1 & 1 \\ b & 1 & 1 & 1 & 1 \\ c & 0 & 0 & 0 & 0 \\ d & 1 & 1 & 1 & 1 \end{array}$$

(c) Transitive closure.

Definition:

Through a

$$R^{(0)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 0 & 0 \\ b & 0 & 0 & 0 & 1 \\ c & 0 & 0 & 0 & 0 \\ d & 1 & 0 & 1 & 0 \end{array}$$

$$R^{(1)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 0 & 0 \\ b & 0 & 0 & 0 & 1 \\ c & 0 & 0 & 0 & 0 \\ d & 1 & 1 & 1 & 0 \end{array}$$

Keep same

Through B

Keep same

$$R^{(2)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 0 & 1 \\ b & 0 & 0 & 0 & 1 \\ c & 0 & 0 & 0 & 0 \\ d & 1 & 1 & 1 & 1 \end{array}$$

Keep same

Keep same

Through c

$$R(3) = \begin{array}{c|cccc} & & & & \\ \hline & 0 & 0 & 0 & 1 \\ & 0 & 0 & 0 & 1 \\ & 0 & 0 & 0 & 0 \\ & 1 & 1 & 1 & 1 \end{array}$$

Through D

$$R(4) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

ALGORITHM

ALGORITHM Warshall ($A[1..n, 1..n]$)

// Implements Warshall's algorithm for computing the transitive

// Input: The adjacency matrix A of a digraph with 'n' vertices.

// Output: The transitive closure of the digraph.

$R^{(0)} \leftarrow A$

for $k \leftarrow 1$ to n do

  for $i \leftarrow 1$ to n do

    for $j \leftarrow 1$ to n do

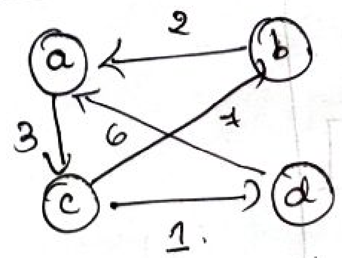
      $R^{(k)}[i,j] \leftarrow R^{(k-1)}[i,j]$ or

      $(R^{(k-1)}[i,k]$ and $R^{(k-1)}[k,j])$

return $R^{(n)}$

Floyd's Algorithm

All Pairs Shortest paths using Floyd's algorithm



ⓐ Digraph

Step 1:

$$\begin{array}{c|cccc} & a & b & c & d \\ \hline a & & & & \\ b & & & & \\ c & & & & \\ d & & & & \end{array}$$

$$\begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{matrix} a & b & c & d \end{matrix} \\ \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & \infty & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & \infty & 0 \end{bmatrix} \end{array}$$

ⓑ weight matrix.

$$D^{(0)} = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \left[\begin{array}{cccc} 0 & \infty & 3 & \infty \\ 2 & 0 & \infty & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & \infty & 0 \end{array}\right] \end{array}$$

→ weight matrix

$$D^{(1)} = \begin{array}{c} \\ \\ a\ 0 \\ b\ 2 \\ c\ \infty \\ d\ 6 \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{cccc} 0 & \infty & 3 & \infty \end{array} \\ \left[\begin{array}{cccc} 0 & \infty & 3 & \infty \\ 2 & 0 & 5 & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & 9 & 0 \end{array}\right] \end{array}$$

$$D^{(2)} = \begin{array}{c} \\ \infty \\ 0 \\ 7 \\ \infty \end{array} \begin{array}{c} \begin{array}{cccc} 2 & 0 & 5 & \infty \end{array} \\ \left[\begin{array}{cccc} 0 & \infty & 3 & \infty \\ 2 & 0 & 5 & \infty \\ 9 & 7 & 0 & 1 \\ 6 & \infty & 9 & 0 \end{array}\right] \end{array}$$

$$D^{(3)} = \begin{array}{c} \\ 3 \\ 5 \\ 0 \\ 9 \end{array} \begin{array}{c} \begin{array}{cccc} 9 & 7 & 0 & 1 \end{array} \\ \left[\begin{array}{cccc} 0 & 10 & 3 & 4 \\ 2 & 0 & 5 & 6 \\ 9 & 7 & 0 & 1 \\ 6 & 16 & 9 & 0 \end{array}\right] \end{array}$$

$$D^{(4)} = \begin{array}{c} \\ 4 \\ 6 \\ 1 \\ 0 \end{array} \begin{array}{c} \begin{array}{cccc} 6 & 16 & 9 & 0 \end{array} \\ \left[\begin{array}{cccc} 0 & 10 & 3 & 4 \\ 2 & 0 & 5 & 6 \\ 7 & 7 & 0 & 1 \\ 6 & 16 & 9 & 0 \end{array}\right] \end{array}$$

Knapsack Problem using dynamic Programming.

$$F(i,j) = \begin{cases} \max\{F(i-1,j),\ v_i + F(i-1,j-w_i)\} & \text{if } j-w_i \geq 0 \\ F(i-1,j) & \text{if } j-w_i < 0 \end{cases}$$

Ex:- let us consider the instance given by following data.

item	weight	value
1	2	\$12
2	1	\$10
3	3	\$20
4	2	\$15

Capacity, $W=5$

10	9	9	6	4	1	10

i \ j	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	12	12	12	12
2	0	10	12	22	22	22
3	0	10	12	22	30	32
4	0	10	15	25	30	37

$1 \geq 1$

$$F(i,j) = \text{Max}\begin{cases} F(i-1,j),\ v_i + F(i-1, j-w_i) & \text{if } j \geq w \\ F(i-1,j) & \text{if } \underbrace{j < w}_{3<2\times} \end{cases}$$

$$F(1,1) = \overset{j<w,\ 1<2\checkmark}{\text{Max}}\{F(0,1)\}\ \cancel{12+} = \underline{\underline{0}}$$

$$F(1,2) = \text{Max}\{F(0,2),\ 12+F(0,0)\}$$
$$= \text{Max}\{0,\ 12+0\} = \underline{\underline{12}}$$

$$\boxed{F(1,3) = \text{Max}\{F(0,3),\ 12+F(0,1)\} \quad \text{Max}\{0,\ 12+0\} = \underline{\underline{12}}}$$

$$F(1,4) = \text{Max}\{F(0,4),\ 12+F(0,2)\}$$
$$= \text{Max}\{0,\ 12+0\} = \underline{\underline{12}}$$

$$F(1,5) = \text{Max}\{F(0,5),\ 12+F(0,3)\}$$
$$= \text{Max}\{0,\ 12+0\} = \underline{\underline{12}}$$

$i=2,\ w=1,\ v=10$

$$F(2,1) = \text{Max}\{F(1,1),\ 10+F(1,0)\}$$
$$= \text{Max}\{0,\ 10+0\} = \underline{\underline{10}}$$

$$F(2,2) = \text{Max}\{F(1,2),\ 10+F(1,1)\}$$
$$= \text{Max}\{12,\ 10+0\} = \underline{\underline{12}}$$

$$F(2,3) = \text{Max}\{F(1,3),\ 10+F(1,2)\}$$
$$= \text{Max}\{12,\ 10+12\} = \underline{\underline{22}}$$

$$F(2,4) = \text{Max}\{F(1,4),\ 10+F(1,3)\}$$
$$= \text{Max}\{12,\ 10+12\} = \underline{\underline{22}}$$

$$F(2,5) = \text{Max}\{F(1,5),\ 10+F(1,4)\}$$
$$= \text{Max}\{12,\ 10+12\} = 22$$

item 3.

$i = 3, \quad w = 3, \quad v = 20.$

$F(3,1) = Max\{F(2,1)\}, \cancel{20+F(2,} = 10.$

$F(3,2) = Max\{F(2,2)\} = 12.$

$F(3,3) = Max\{F(2,3), \; 20 + F(2,0)\}.$

$Max\{ 22, \; 20+0 \} = 22.$

$F(3,4) = Max\{F(2,4), \; 20 + F(2,1)\}.$

$= Max\{ 22, \; 20+10\} = 30.$

$F(3,5) = Max\{F(2,5), \; 20 + F(2,2)\}.$

$= Max\{ 22, \; 20+12\} = 32.$

item 4

$i = 4 \quad w = 2 \quad v = 15.$

$F(4,1) = Max\{F(3,1)\} =$

$F(4,2) = Max\{F(3,1), \; 15 + F(3,0)\}$

$= Max\{ 10, \; 15+0\} = 15.$

$F(4,3) = Max\{F(3,1), \; 15 + F(3,1)\}.$

$= Max\{ 10, \; 15 + 10\} = 25$

$37 = (4,5) = 37 - 15 = 22$ ↗ 1st position. Profit rem → 1

$5 - 2 = 3$ Capacity rem.

$22 = (2,3) = 22 - 10 = 12$ → 2

$3 - 1 = 2$

$12 = (1,2) = 12 - 12 = 0$ → 4.

$2 - 2 = 0$

$(x_1, x_2, x_3, x_4) = (1, 1, 0, 1).$

Knapsack Algorithm using memory function.

Apply the memory function method to solve the following ~~instance~~ instance of the knapsack problem with capacity, $m=5$.

Item	weight	value.
1	2	\$12
2	1	\$10
3	3	\$20
4	2	\$15

$$F(i,j) = \max \begin{cases} F(i-1,j),\ v_i + F(i-1, j-w_i) & \text{if } j > w_i \\ F(i-1,j) & \text{if } j < w_i \end{cases}$$

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	12	12	12	12
2	0	–	12	22	–	22
3	0	–	–	22	–	32
4	0	–	–	–	–	37

Step 1. $i=4, j=5, v_4=15, w_4=2.$

$$F(4,5) = \max\begin{cases} F(3,5),\ 15 + F(3,3) \\ 32,\ 15+22 \end{cases} = \boxed{37}$$

Step 2 $i=3, j=5, w_3=3, v_3=20$

$$F(3,5) = \max\begin{cases} F(2,5),\ 20+F(2,2) \\ 22,\ 20+12 \end{cases} = 32.$$

Step 3 $i=3, j=3$ $w_3=3, v_3=20$

$$F(3,3) = \max\begin{cases} F(2,3),\ 20+F(2,0) \\ 22,\ 20+0 \end{cases} = 22.$$

Step 4 $i=2, j=5, w_2=1, v_2=10$

$$F(2,5) = \max\begin{cases} F(1,5),\ 10+F(1,4) \\ 12,\ 10+12 \end{cases} = 22$$

Step 5 $i=2, j=2, w_2=1, v_2=10$

$$F(2,2) = \max\begin{cases} F(1,2),\ 10+F(1,1) \\ 12,\ 10+0 \end{cases} = 12.$$

Step 6. $i=2, j=3, w_2=1, v_2=10$

$$F(2,3) = \text{Max}\left\{ \underset{12}{F(1,3)}, \underset{10+12}{10+F(1,2)} \right\} = 22.$$

Step 7 $i=1, j=5, w_1=2, v_1=12.$

$$F(1,5) = \text{Max}\left\{ \underset{0}{F(0,5)}, \underset{12+0}{12+F(0,3)} \right\} = 12.$$

Step 8 $i=1, j=4, w_1=2, v_1=12.$

$$F(1,4) = \text{Max}\left\{ \underset{0}{F(0,4)}, \underset{12+0\ =12}{12+F(0,2)} \right\} = 12.$$

Step 9

$$F(1,2) = \text{Max}\left\{ \underset{0}{F(0,2)}, \underset{12+0\ =12}{12+F(0,0)} \right\} = 12.$$

Step 10

$$F(1,1) = \text{Max}\left\{ F(0,1) \right\}$$ [redacted] $= 0$

Step 11

$$F(1,3) = \text{Max}\left\{ \underset{0}{F(0,3)}, \underset{12+0\ =12}{12+F(0,1)} \right\} = 12$$

Step 12.

$$F(1,2) = \text{Max}\left\{ F(0,2), 12+F(0,0) \right\}$$

Memory function Algorithm.

Algorithm MFKnapsack (i,j)

// Implements the memory function method for the knapsack problem.

// Input : A non negative integer 'i' indicating the number of the first items being considered and a non negative integer 'j' indicating the knapsack capacity.

// Output : The value of an optimal feasible subset of the first i items.

// Note : Uses as global variables input arrays Weights [1...n], Values [1...n], and table F [0..n, 0..W] whose entries are initialized with −1's except for row 0 and column 0 initialized with 0's.

if F[i, j] < 0

  if j < Weights[i]

    value <- MFKnapsack (i-1, j)

  else

    value <- max (MFKnapsack (i-1, j),

      Values[i] + MFKnapsack (i-1, j - Weights[i]))

  F [i, j] <- value

return F[i, j]

Conclusion:

$37 \Rightarrow (4, 5) = 37 - 15 = \underline{22}$, $5 - 2 = 3$

$22 \Rightarrow (2, 3) = 22 - 10 = 1\underline{2}$, $3 - 1 = 2$

$12 \Rightarrow (\underline{1}, \underline{2}) = 12 - 12 = \underline{0}$ $2 - 2 = 0$

$\underline{(x_1, x_2, x_3, x_4) = (1, 1, 0, 1)}$

Updated Prims Algorithm

$c(i,j) = \min_{i \le k \le j} \{ \dots \sum_{s=i} \dots$

$+ \sum_{s=k+1}^{j} p_s \cdot (\text{level of } a_s \text{ in } T_{k+1}$

> Standard formula to obtain the optimal binary tree is
> Pre condition is $\boxed{c(i,i) = p_i}$ and $\boxed{c(i, i-1) = 0}$

$$c(i,j) = \min_{i \le k \le j} \{ c(i, k-1) + c(k+1, j) \} + \sum_{s=i}^{j} P_s \text{ for } 1 \le i \le j \le n.$$

Construct an optimal binary tree for the following data.

Key	A (1)	B (2)	C (3)	D (4)
Probability	0.1	0.2	0.4	0.3

$\boxed{\begin{array}{l} c(i,i) = p. \\ c(i,i-1) = 0 \end{array}}$

Main table

i \ j →	0	1	2	3	4
1	0	0.1	0.4	1.1	1.7
2		0	0.2	0.8	1.4
3			0	0.4	1.0
4				0	0.3
5					0

root table

	0	1	2	3	4
1		1	2	3	3
2			2	3	3
3				3	3
4					4
5					

C

let us compute $c(1,2)$ for 2 nodes

$$= \min \begin{cases} k=1, & c(1,0)+c(2,2)+0.3 \\ k=2, & c(1,1)+c(3,2)+0.3 \end{cases}$$

$$\min \begin{Bmatrix} 0+0.2+0.3 \\ 0+0+0.3 \end{Bmatrix}$$

$$\min \{0.5, 0.3\} \Rightarrow \boxed{k=2}$$

Let us compute $(2,3)$ for 2 nodes.

$$= \min \begin{cases} k=2, & c(2,1)+c(3,3)+0.6 \\ k=3, & c(2,2)+c(4,3)+0.6 \end{cases}$$

$$= \min \begin{Bmatrix} 0+0.4+0.6 \\ 0.2+0+0.6 \end{Bmatrix}$$

$$= \min \{1.0, 0.8\} = 0.8 \text{ for } k=3$$

Let us compute $(3,4)$ for 2 nodes

$$= \min \begin{cases} k=3, & c(3,2)+c(4,4)+0.7 \\ k=4, & c(3,3)+c(5,4)+0.7 \end{cases}$$

$$= \min \{0+0.3+0.7 \quad 0.4+0+0.7\}$$

$$= \min \{1.0, 1.1\} = 1.0 \text{ for } k=3.$$

For 3 nodes.

$$c(1,3) = \min \begin{cases} k=1, & c(1,0)+c(2,3)+0.7 \\ k=2, & c(1,1)+c(3,3)+0.7 \\ k=3, & c(1,2)+c(4,3)+0.7 \end{cases}$$

$$= \min \begin{Bmatrix} 0+0.8+0.7 \\ 0.1+0.4+0.7 \\ 0.4+0+0.7 \end{Bmatrix}$$

$$= \min \{1.5, 1.2, 1.1\} = 1.1 \text{ for } k=3.$$

$$c(2,4) = \min \begin{cases} k=2, & c(2,1) + c(3,4) + 1.09 \\ k=3, & c(2,2) + c(4,4) + 1.09 \\ k=4, & c(2,3) + c(5,4) + 1.09 \end{cases}$$

$$\min \{ 0+1+0.9,\ 0.2+0.3+0.9,\ 0.8+0+0.9 \}$$

$$\min \{ 1.9,\ 1.4,\ 1.7 \} = 1.4 \text{ for } k=3$$

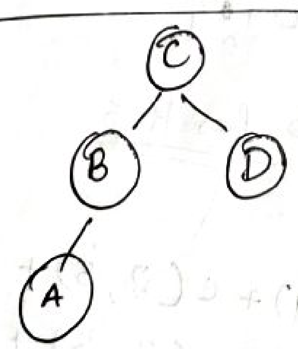
For n = 4 nodes.

$$c(1,4) = \min \begin{cases} k=1, & c(1,0) + c(2,4) + 1 \\ k=2, & c(1,1) + c(3,4) + 1 \\ k=3, & c(1,2) + c(4,4) + 1 \\ k=4, & c(1,3) + c(5,4) + 1 \end{cases}$$

$$= \min \{ 0+1.4+1,\ 0.1+1+1,\ 0.4+0.3+1,\ 1.1+0+1 \}$$

$$= \min \{ 2.4,\ 2.1,\ 1.7,\ 2.1 \} = 1.7 \text{ for } k=3.$$

- Optimal binary search tree.



ALGORITHM optimalBST(P[1...n])

for i ← 1 to n do

  C[i, i-1] ← 0

  C[i, i] ← P[i]

  R[i, i] ← i

C[n+1, n] ← 0

for d ← 1 to n-1 do

  for i ← to n-1 do

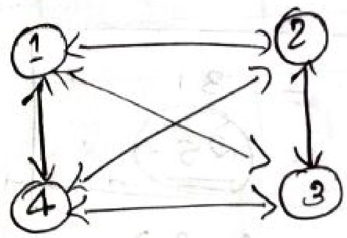
```
j <- i+d
minval <- ∞
for k <- i to j do
    if c[i, k-1] + c[k+1, j] < minval
        minval <- c[i, k-1] + c[k+1, j].
            kmin <- k.
R[i,j] <- kmin
sum <- P[i].
for s <- i+1 to j do
    sum <= sum + P[s]
    c[i, j] <- minval + sum
return c[1, n] . R.
```

Travelling Sales Person problem. $(O(n^2 2^n))$

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

$i \to$ starting vertex ; $S \to$ Remaining vertices.

Eg:



$$\begin{bmatrix} 0 & 10 & 15 & 20 \\ 5 & 0 & 9 & 10 \\ 6 & 13 & 0 & 12 \\ 8 & 8 & 9 & 0 \end{bmatrix}$$

$g(1, \phi) = c_{11} = 0$

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

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

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

Vertex 1.

$$g(1, \{2,3,4\}) = \min \begin{cases} c_{12} + g(2, \{3,4\}) \\ c_{13} + g(3, \{2,4\}) \\ c_{14} + g(4, \{2,3\}) \end{cases}$$

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

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

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

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

$$\cancel{\Rightarrow g(1,\{2,3,4\}) = \min}$$

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

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

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

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

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

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

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

$$\boxed{g(1,\{2,3,4\}) = \min\begin{cases}10+25 = 35 \\ 15+25 = 40 \\ 20+23 = 43\end{cases} = 35}$$

$1 \rightarrow 2 \rightarrow 4 \xrightarrow{2\rightarrow} 1$ (Minimum cost path)

$10+10+9+6 = \underline{\underline{35}}$.

Space and Time Trade - Offs.

<1> Sorting by counting.

62	31	84	96	19	47

list to be stored.

0	0	0	0	0	0

1st Pass: i=0

~~2~~ 3	0	1	1	0	0.

2nd Pass i=1

3	~~0~~ 1	2	2	0	1

3rd Pass i=2

3	1	~~3~~ 4	3	0	1.

4th Pass i=3

3	1	4	5	0	1.

5th Pass i=4

3	1	4	5	0	2.

∴ The sorted list is;

19	31	47	62	84	96.
0	1	2	3	4	5

//

Pattern BARBER
2 4 8 2 1 3

Character c	A	B	C	D	E	F	...	R	...	Z	_
Shift (c).	4	2	6	6	1	6	6	3	6	6	6

② JIM – SAW – ME – IN – A – BARBER SHOP

BARBER

BARBER.

BARBER

BARBER

~~BARBER~~

~~BARBER~~

BARBER.

$\boxed{\text{BARBER.}}$

② Character c	A	E	R	N	....
Shift (c)	2	3	1	4	4.

FAIL – MEANS – FIRST – ATTEMPT – IN – LEARN.

EARN.

EARN

EARN.

EARN

EARN.

EARN

EARN

EARN

ALGORITHM ShiftTable (P[0..m-1])

// Fills the shift table used by Horspool's and Boyer-Moore algorithm.

// Input: Pattern P[0..m-1] and an alphabet of possible characters.

// Output: Table[0.. size-1] indexed by alphabets characters and

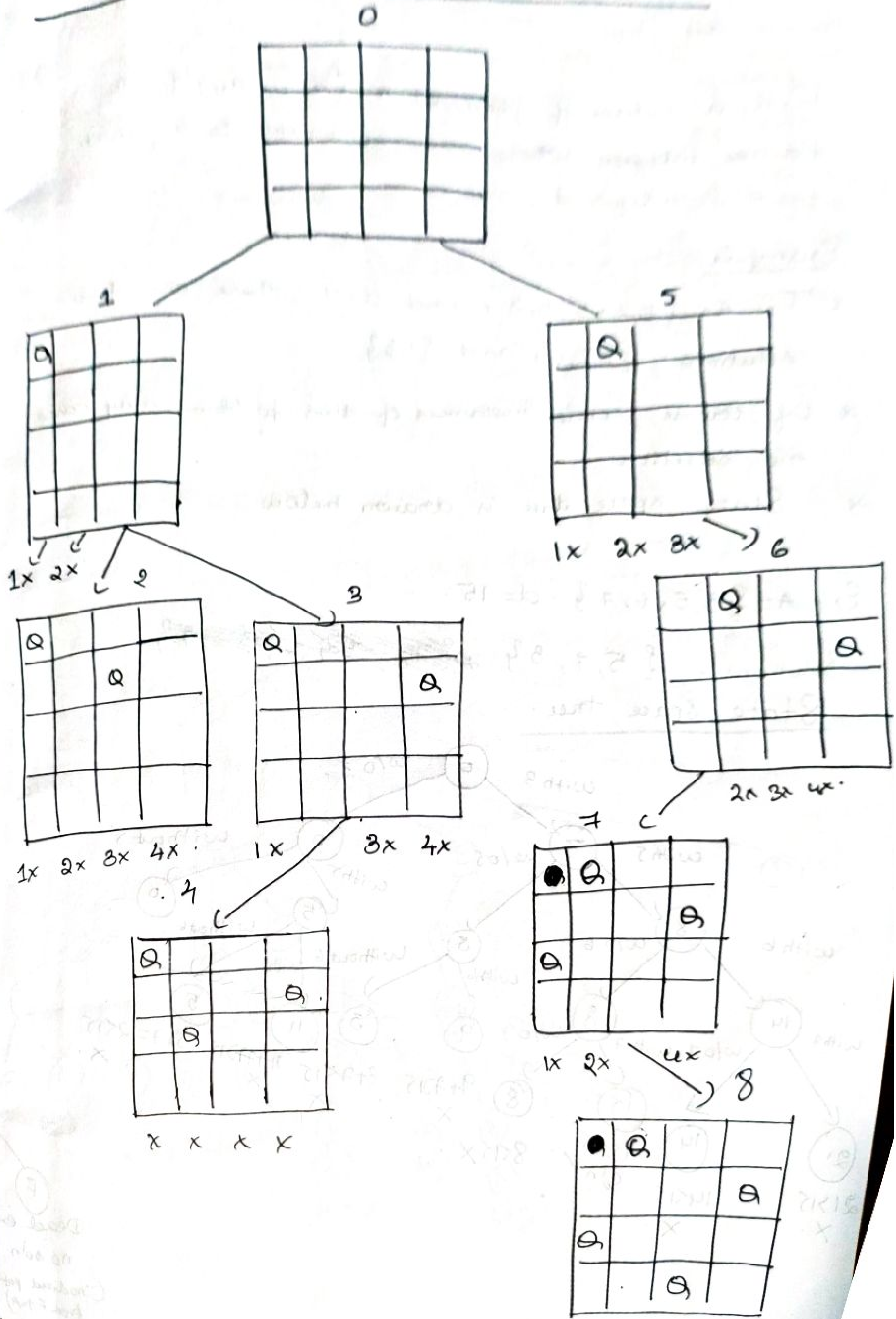
[redacted] filled with shift sizes computed by formula.

for i ← 0 to size-1 do Table[i] ← m

for j ← 0 to m-2 do Table[P[j]] ← m-1-j

return Table.

State-space tree for 4-queen problem



Sum of Subsets problem.

Problem definition:

Find a subset of given set $A = \{a_1, \ldots a_n\}$ of n positive integers whose sum is equal to a given positive integer d.

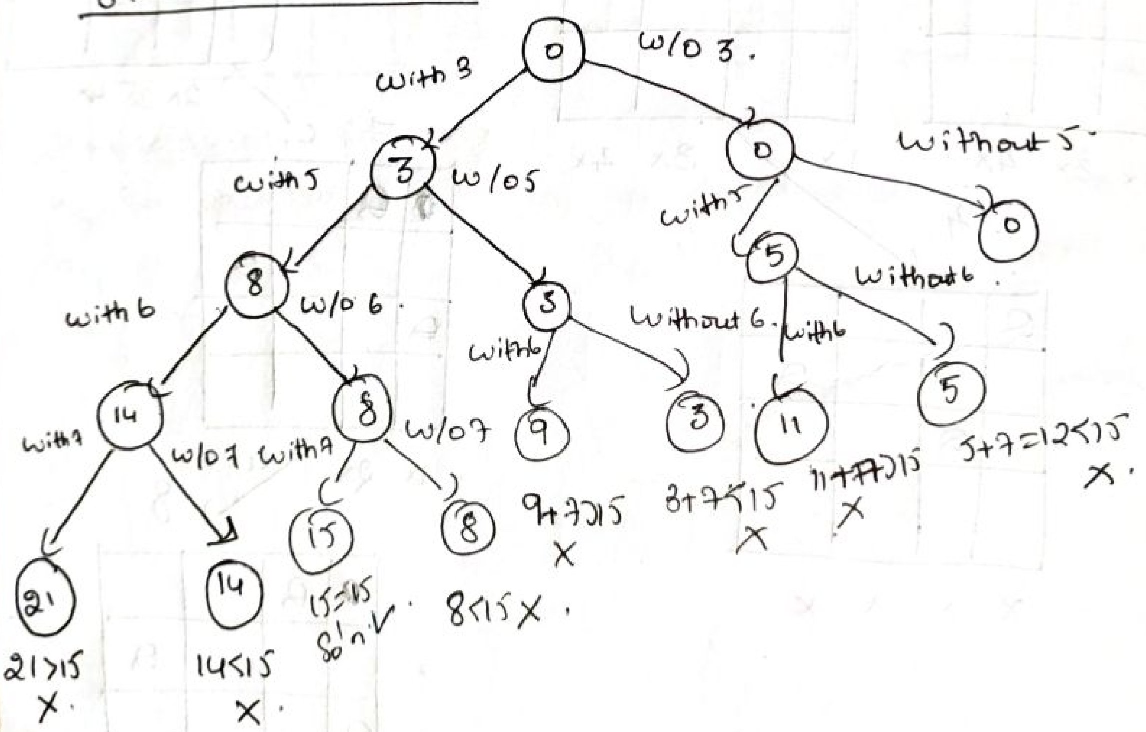
Example:

* For $A = \{1, 2, 5, 6, 8\}$ and $d = 9$, there are two solutions: $\{1, 2, 6\}$ and $\{1, 8\}$.
* Of course, some instances of this problem may have no solutions.
* State space tree is drawn below.

Ex:- $A = \{3, 5, 6, 7\}$ $d = 15$.

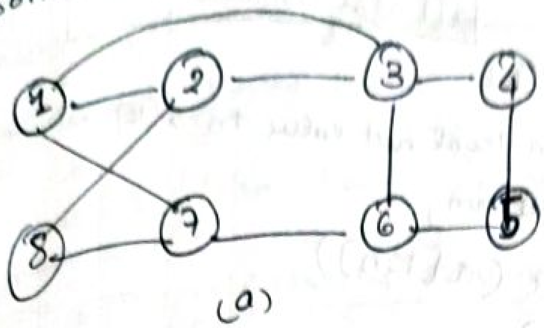
Solutions:- $\{5, 7, 3\}$

State space tree.

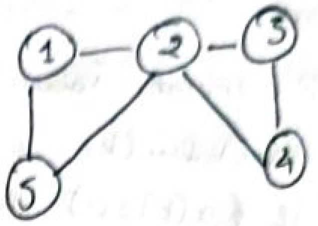


Hamiltonian Cycle

Let $G = (V, E)$ be a connected graph with n vertices. A Hamiltonian cycle is a round-trip path along n edges of G that visits every vertex once and returns to its starting position. In other words, if a Hamiltonian cycle begins at some vertex $v_1 \in G$ and the vertices.

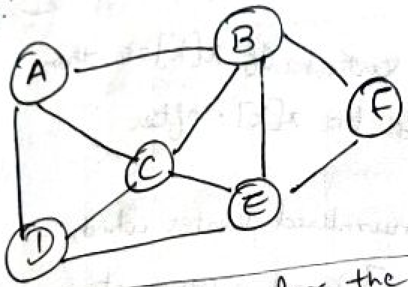


(a)



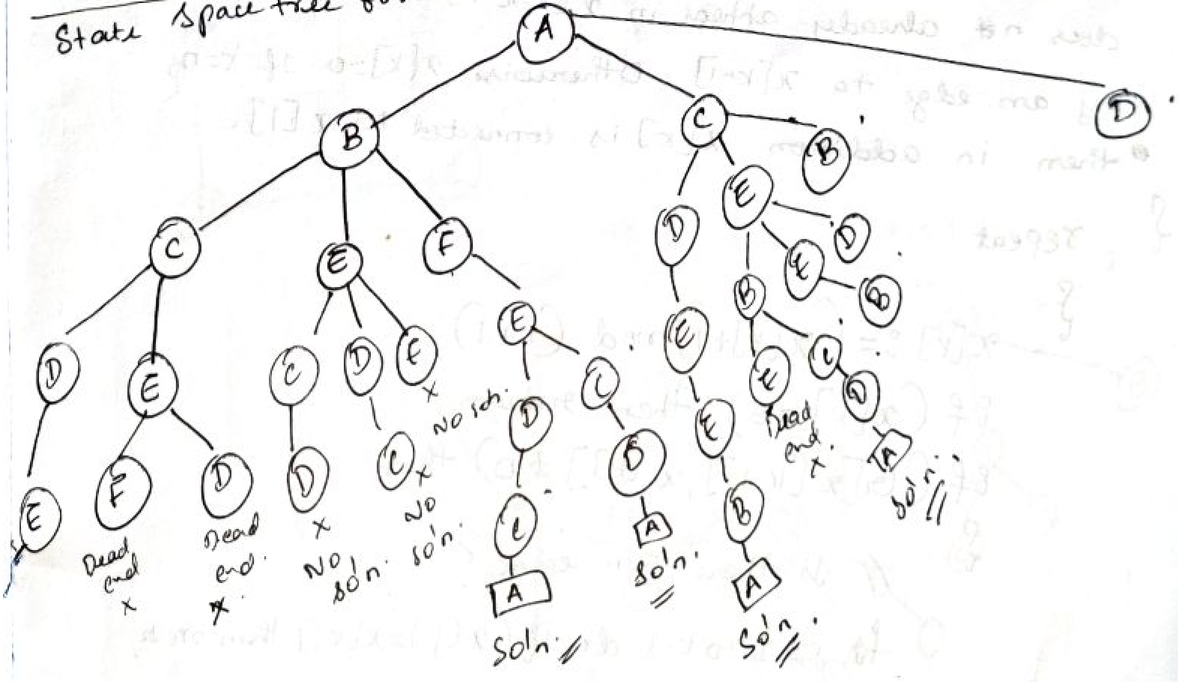
~~1→2→3→4→5→6~~

Eg:



$\boxed{A \to B \to F \to E \to C \to D \to A.}$

State Space tree for the above graph. (Using DFS)



Graph Coloring Problem

Let $G=(V,E)$ be a graph, In graph coloring problem we have to find out whether all the vertices of the given graph are colored or not, with the constraint that no two adjacent vertices have the same color.

The minimum number of colors required to color all the vertices of the given graph, with the constraint that no two adjacent vertices have the same colour.

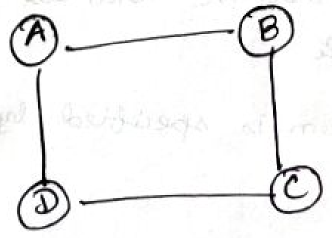
This problem has two versions:

1) m-colorability decision problem.

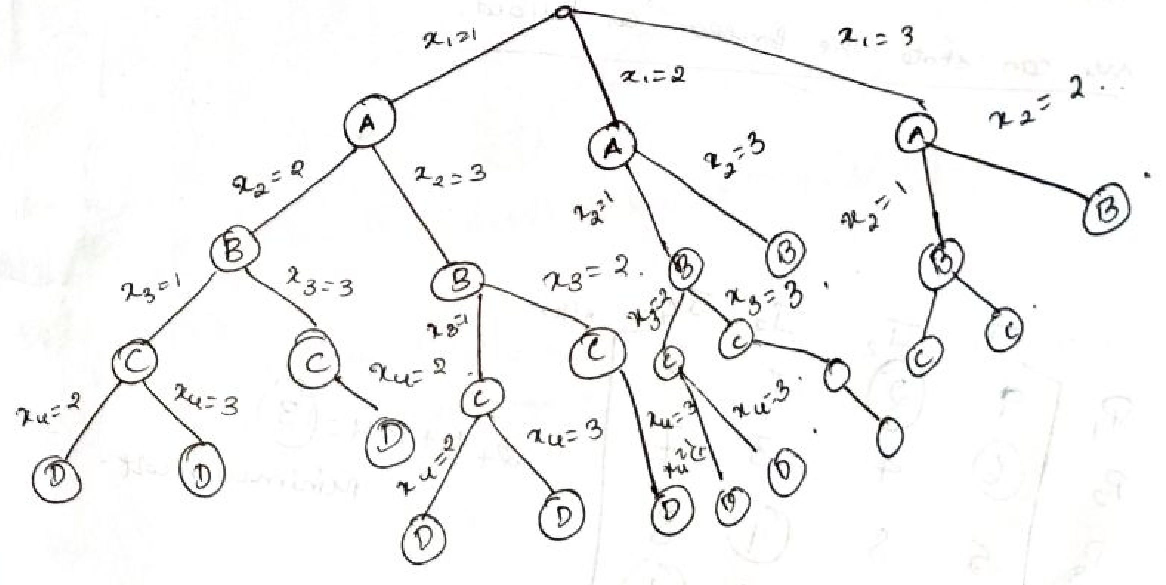
By using 'm' number of colors whether all the vertices of the given graph can be colored or not.

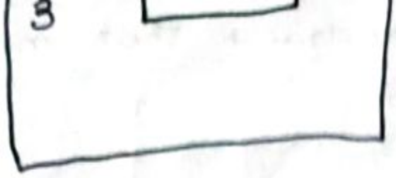
2) m-colorability optimization problem.

-> Either requires maximum result or minimum result.
-> Here we require minimum result, minimum number of colors with the given constraint.



$m = 3$
1, 2, 3.





4-planar graph.

1 -> 2, 3, 4.
2 -> 1, 3, 4, 5.
4 -> 1, 2, 3, 5.
5 -> 4, 2,
3 -> 1, 2, 4.

Branch and Bound.

1) Assignment Problem

Problem Definition.

Assigning 'n' people to 'n' jobs so that the total cost of the assignment is as small as possible.

An instance of the assignment problem is specified by an nxn cost matrix C

We can state the problem as follows:

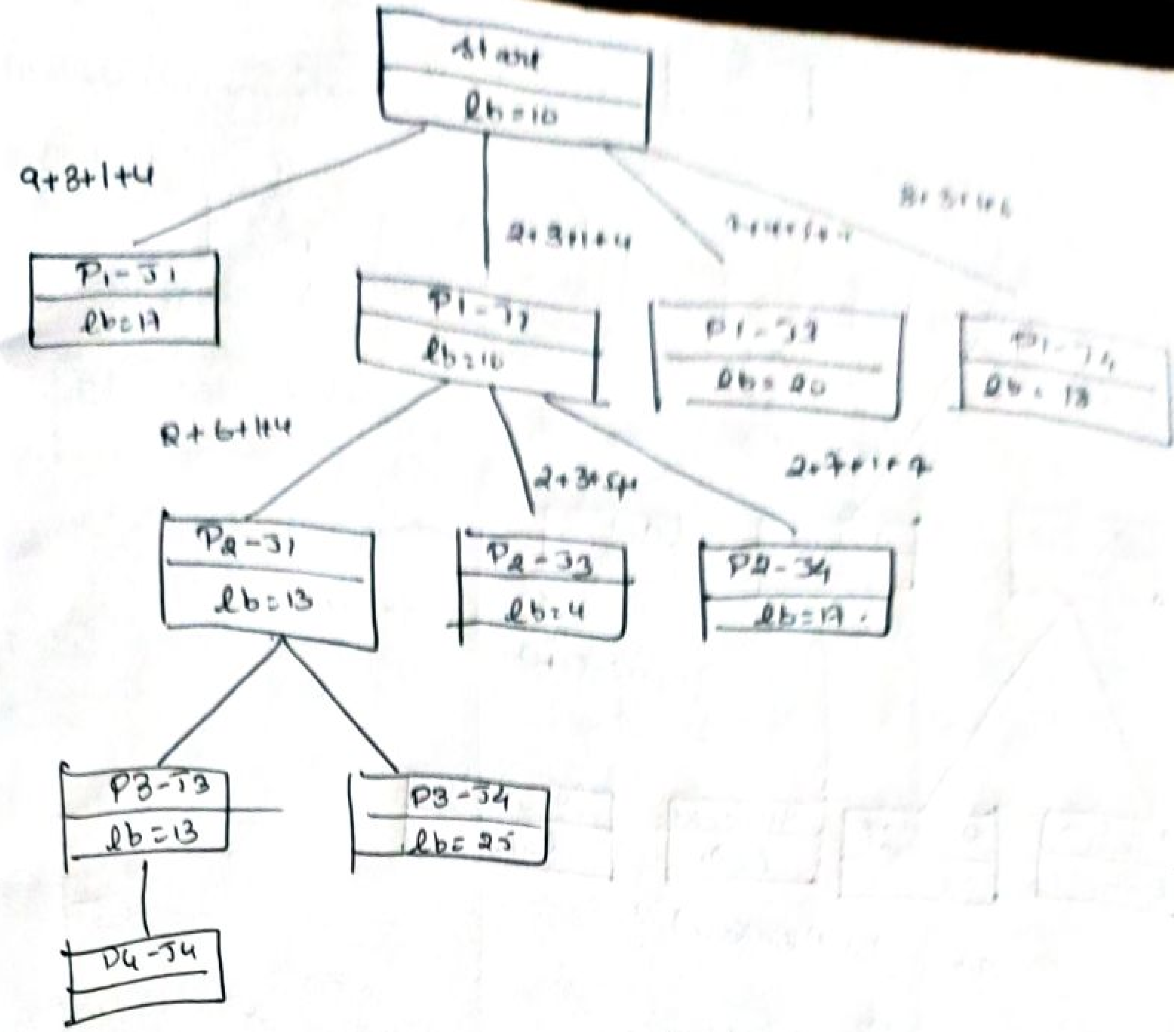
$$\begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_4 \end{array} \begin{array}{cccc} J_1 & J_2 & J_3 & J_4 \\ \end{array}$$

$$\begin{array}{c|cccc} & J_1 & J_2 & J_3 & J_4 \\ \hline P_1 & 9 & 2 & 7 & 8 \\ P_2 & 6 & 4 & 3 & 7 \\ P_3 & 5 & 8 & 1 & 8 \\ P_4 & 7 & 6 & 9 & 4 \end{array}$$

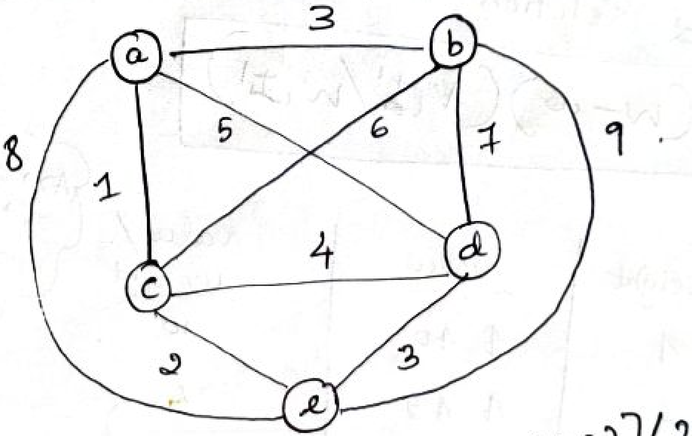
After.

$2+6+1+4=13$

Minimum cost.



Branch & Bound - TSP.



for a =

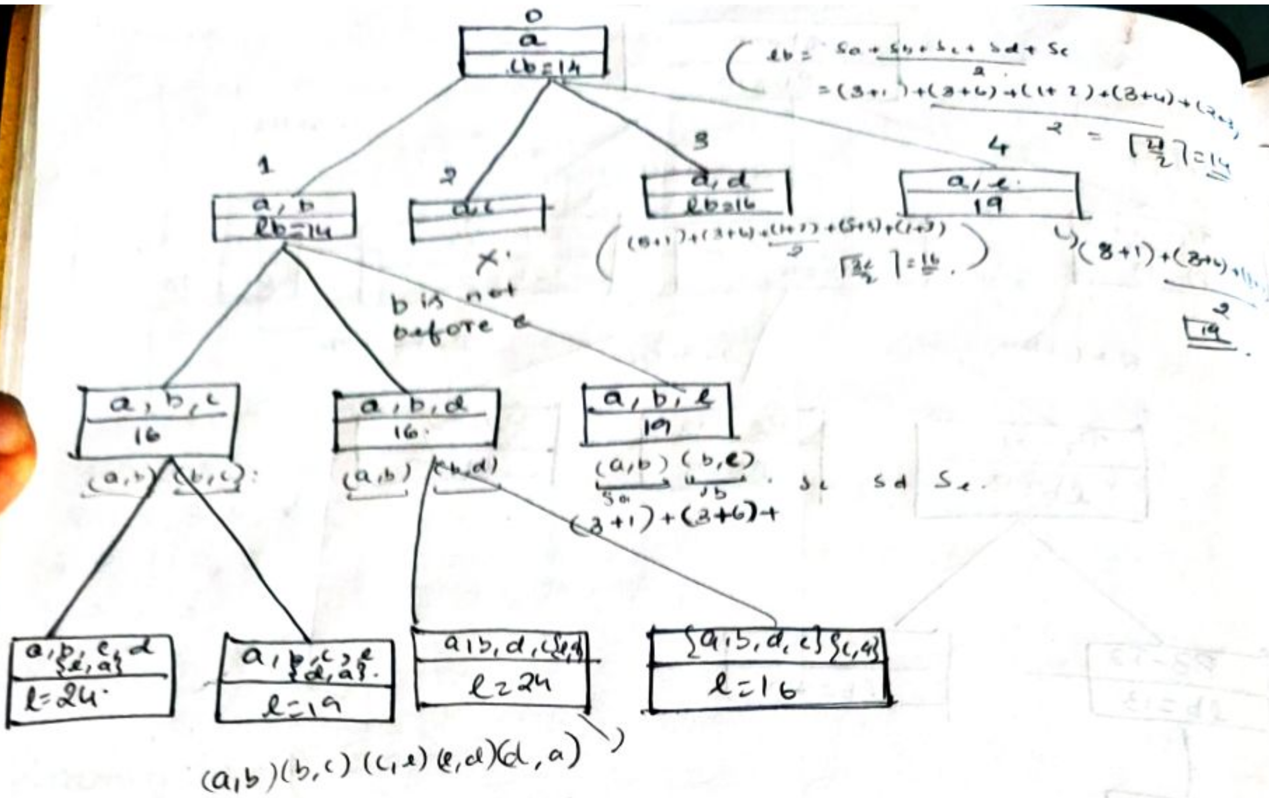
$$lb = \left\lceil [(1+3)+(3+6)+(1+2)+(3+4)+(2+3)]/2 \right\rceil = 14$$

Assumptions:

1) 'a' is the starting city.
2) City 'b' is visited before visiting city 'c'.

$$lb = \frac{S_a + S_b + S_c + S_d + S_e}{2}$$

$$= \frac{(1+3)+(3+6)+(1+2)+(3+4)+(2+3)}{2} = \left\lceil \frac{28}{2} \right\rceil = 14$$



0/1 knapsack problem

Branch and Solution to 0/1 knapsack Problem.

$$\boxed{ub = v + (W - w)(v_{i+1} / w_{i+1})}$$

Arrange the item in descending order of value/wt

Q)

item	weight	value	value/weight
1	4	\$ 40	10
2	7	\$ 42	6
3	5	\$ 25	5
4	3	\$ 12	4

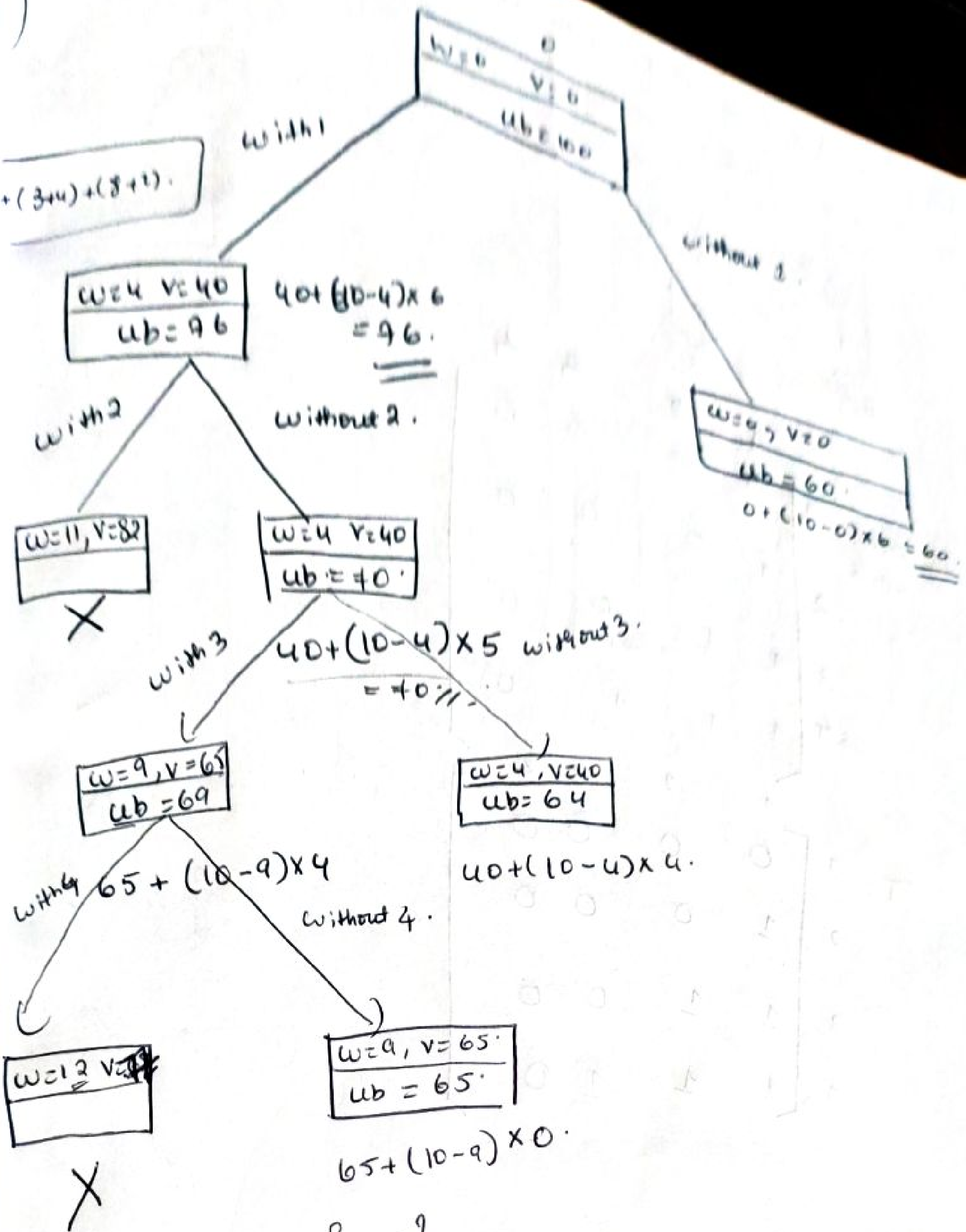
The knapsack's capacity $W = 10$.

Let us apply branch and bound algorithm

(i) $v = 0,\ w = 0$.

$$ub = 0 + (10 - 0) \times 10.$$

$$ub = 100.$$



Items added {1, 3}.

Imp NP-Complete and NP-Hard Problems.

P, NP.

NP → Non deterministic Polynomial time.

P → Polynomial time deterministic.