

Dynamic Programming

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

Here program is not a computer program, it is a process of obtaining the plan to get optimal soln.

Divide & conquer

1. Applicable when subprograms are independent
2. Each subprog is solved separately & combined to get soln to original problem
3. Every instance of subproblem is recomputed, not stored, so cannot reuse the previous work.
- 4.
5. Uses top-down approach

Greedy Method

1. Only one decision sequence is generated

features of Dyn prog:-

1. principle of optimality:- For any optimal sequence of decisions, whatever the initial state & decision are, the remaining decisions must constitute an optimal decision sequence with regard to state resulting from the first decision.

Dyn prog

1. applicable when not independent.
2. Original problem is solved by using results of previous subproblems
3. only one instance of the subproblem is computed & stored, & reused when needed
4. More efficient than D&C
6. Uses bottom-up approach

Dyn prog

1. Many decision sequences may be generated.

2. Total no. of decision sequences is exponential in the no. of decisions. So, dyn programming algo's often have a polynomial complexity.

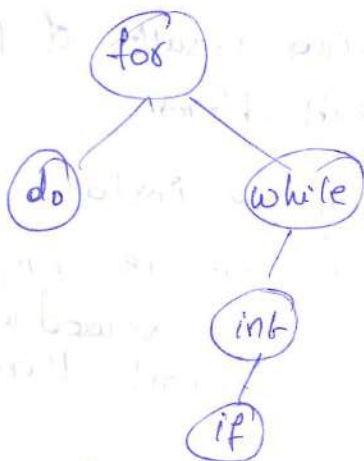
If there are d choices for each of the n decisions to be made, then there are d^n possible decision sequences.

3. Optimal solns to subproblems are retained to avoid recomputing their values. Values are stored in tables

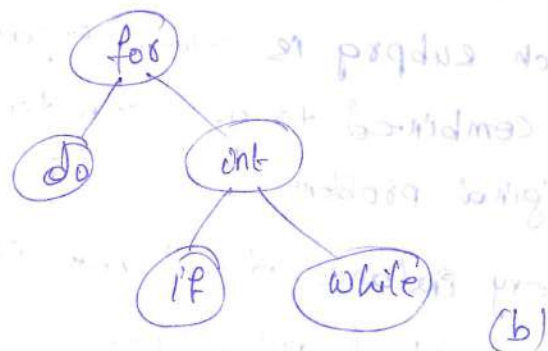
Applications:-

Optimal Binary Search trees

Given a fixed set of identifiers, goal is to create a binary search tree. Different binary trees for same identifier set have different performance characteristics.



(a)



(b)

Two possible search trees

(a) if

Fig(a) req four comparisons in worst case, $\epsilon_{\text{avg}} = \frac{12}{5}$

(b) " three " " " = $\frac{11}{5}$

Comparison

(b) "three"

Company

→ Avg no. of comparisons, calculation assumes that each identifier is searched for with equal prob and that no unsuccessful searches are made.

→ But in practical situation, different ids to be searched with different frequencies or probs. Also unsuccessful searches are present.

→ Assume the given set of ids $\{a_1, a_2, \dots, a_n\}$ with $a_1 < a_2 < \dots < a_n$ (2)

let $p(i)$ be the prob with which we search for a_i

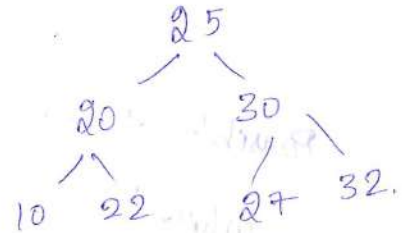
$q(i)$ " " that id ' x ' being searched for

$$a_i < x < a_{i+1}, 0 \leq i \leq n$$

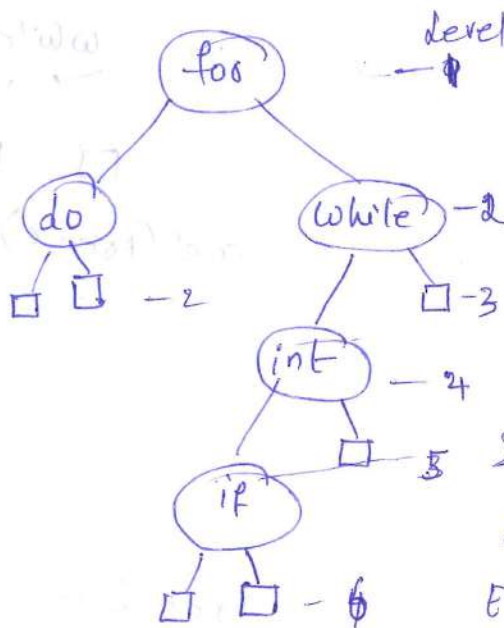
$$\text{Assume } a_0 = -\infty, a_{n+1} = +\infty$$

$\sum_{0 \leq i \leq n} q(i)$ is the prob of an unsuccessful search

$$\sum_{1 \leq i \leq n} p(i) + \sum_{0 \leq i \leq n} q(i) = 1$$



→ Add external nodes



level → Add fictitious node in place of every empty subtree. &

2 If a binary search tree represents n identifiers, then there will be exactly n internal nodes & $n+1$ external nodes.

Internal node represents a point where a successful search terminates.

External node represents a point where a unsuccessful search terminates.

→ The identifiers not in BST can be partitioned into $n+1$ equivalence

classes $E_i, 0 \leq i \leq n$.

E_0 - all ids x such that $x < a_1$

E_i - " " " $a_i \leq x < a_{i+1}$
 $1 \leq i \leq n$

E_n - " " " $x > a_n$

Cost of a binary search tree =

$$\sum_{1 \leq i \leq n} p(i) * \text{level}(a_i) + \sum_{0 \leq i \leq n} q(i) * (\text{level}(E_i) - 1)$$

Goal is to define an OBST for id set $\{a_1, a_2, \dots, a_n\}$ to be a BST for which cost is min.

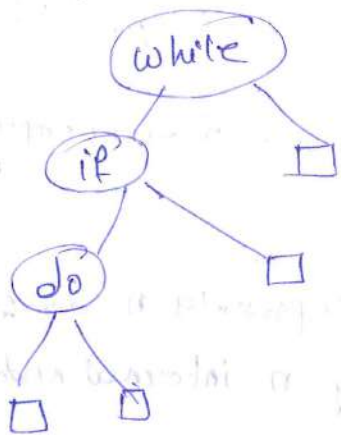
Ex. Define a OBST for $\{a_1, a_2, a_3\} = (\text{do}, \text{if}, \text{while})$ with equal probabilities

No. of Possible BST for n-node
 $= \frac{2n!}{n+1}$

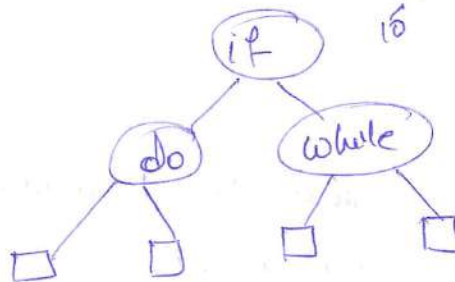
Soln:- Given equal probabilities

i.e $p(i) = q(i) = \frac{1}{7} \quad \forall i = 1, 2, 3$

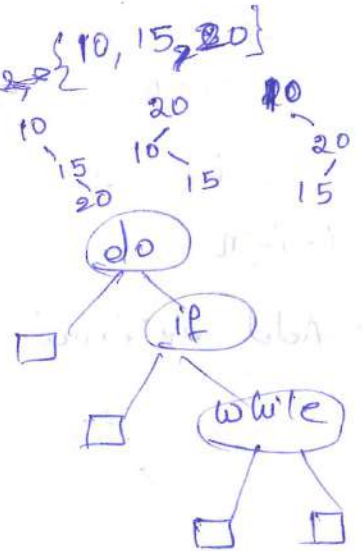
Possible trees are



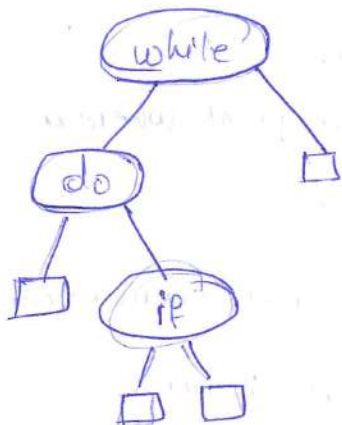
a) $\text{Cost}(\text{tree a}) = \frac{15}{7}$



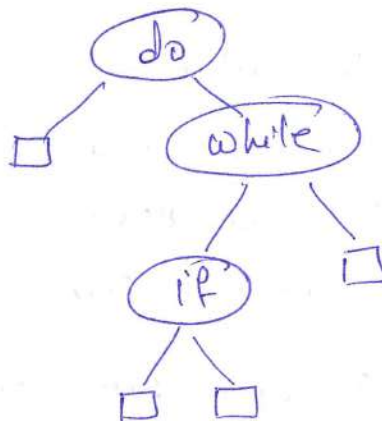
b) $\text{Cost}(\text{tree b}) = \frac{13}{7}$



c) $\text{Cost}(\text{tree c}) = \frac{15}{7}$



d) $\text{Cost}(\text{tree d}) = \frac{15}{7}$



e) $\text{Cost}(\text{tree e}) = \frac{15}{7}$

Tree b is optimal

If probs are changed to $p(1) = .5, p(2) = .1, p(3) = .05,$
 $q(0) = .15, q(1) = .1, q(2) = .05, q(3) = .05$

$\text{Cost}(\text{tree a}) = 2.65$ $\text{Cost}(\text{tree b}) = 1.9$

c = 1.5

d = 2.05

e = 1.6

→ Apply dyn prog to OBST

Make a decision to which of a_i 's should be assigned to the root node of the tree.

If we choose a_k , then internal nodes for $a_1, a_2, \dots, a_{k-1} \in$
external " " $E_0, E_1, \dots, E_{k-1}$ lie in
the left subtree l of a_k . The remaining nodes will be in the right
subtree r of a_k



$$\text{Cost}(l) = \sum_{1 \leq i < k} p(i) * \text{level}(a_i) + \sum_{0 \leq i < k} q(i) * (\text{level}(E_i) - 1)$$

and

$$\text{cost}(r) = \sum_{k < i \leq n} p(i) * \text{level}(a_i) + \sum_{k < i \leq n} q(i) * (\text{level}(E_i) - 1)$$

→ If $w(i,j)$ is sum $q(i) + \sum_{k=i+1}^j (q(k) + p(k))$
⇒ Expected cost of search tree $= p(k) + \text{cost}(l) + \text{cost}(r) + w(0, k-1) + w(k, n)$

If tree is optimal, then cost should be min., i.e. $\text{cost}(l) \leq \text{cost}(r)$
should be min.

Let $c(i,j)$ - cost of an OBST $t_{i,j}$, containing $a_{i+1}, \dots, a_j \in$
 $E_i, \dots, E_j$

$$\Rightarrow \text{cost}(l) = c(0, k-1)$$

$$\text{cost}(r) = c(k, n)$$

In addition, k must be chosen such that
 $p(k) + c(0, k-1) + c(k, n) + w(0, k-1) + w(k, n)$ is minimum.

$$\Rightarrow c(i,j) = \min_{i < k \leq j} \{ c(i, k-1) + c(k, j) \} + w(i,j) \quad \text{--- (1)}$$

$r(i,j)$ is the value of k that minimizes $c(i,j)$ in (1)

Compute $c(i,j)$ such that $j-i=2$, then for all $c(i,j)$ with $j-i=3, \dots$

Ex:- Use dyn programming, to generate OBST for
 $n=4, (a_1, a_2, a_3, a_4) = (\text{do}, \text{it}, \text{int}, \text{white})$. Probs are multiplied by 16 for convenience.
 $P(1:4) = (3, 3, 1, 1)$ $\Rightarrow P(1:4) = (\frac{1}{2}, \frac{1}{2}, \frac{1}{6}, \frac{1}{6})$ $\rightarrow$ If this type of probs are given, multiply by LCM.
 $q(0:4) = (2, 3, 1, 1)$ $\Rightarrow P(1:4) = (0.5, 0.5, 0.16, 0.16)$ etc. LCM.

Soln: Initially $w(i,i) = q(i)$ $0 \leq i \leq 4$

For first row

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

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

2 parts
 1) Table construction
 2) OBST

$i \rightarrow$

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

$j-i=0$

$j-i=1$

$j-i=2$

$j-i=3$

$j-i=4$

for all rows except row 1

$$w_{ij} = P(j) + q(j) + w_{i,j-1}$$

$$c_{ij} = w_{ij} + \min_{i \leq k \leq j} \{c_{ik} + c_{kj}\}, \quad r = k \text{ (with mincost } c_{ij})$$

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

$$w(0,1) = P(1) + q(1) + w(0,0) = 8$$

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

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

$$w(1,2) = P(2) + q(2) + w(1,1) = 7$$

$$c(1,2) = w(1,2) + \min \{c(1,1) + c(2,2)\} = 7$$

$$r(0,2) = 2$$

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

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

$$r(2,3) = 3$$

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

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

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

$$w_{02} = p(2) + q(2) + w(0,1)$$

$$= 3 + 1 + 8 = 12$$

$$c_{02} = 12 + \min_{i,j} \begin{cases} k=1 & c(0,0) + c(1,2) \checkmark \\ k=2 & c(0,1) + c(2,2) \end{cases}$$

$$= 12 + \min \begin{cases} 0 + 7 \\ 8 + 0 \end{cases} = 19$$

$$r_{02} = 1$$

$$w_{13} = p(3) + q(3) + w(1,2)$$

$$= 1 + 1 + 7 = 9$$

$$c_{13} = 9 + \min_{i,j} \begin{cases} k=2 & c_{11} + c_{23} \\ k=3 & c_{12} + c_{33} \end{cases} = 9 + \min \begin{cases} 0 + 3 \checkmark \\ 7 + 0 \end{cases}$$

$$= 12$$

$$r_{13} = 2$$

$$w_{24} = p(4) + q(4) + w(2,3)$$

$$= 1 + 1 + 3 = 5$$

$$c_{24} = 5 + \min_{i,j} \begin{cases} k=3 & c_{22} + c_{34} \\ k=4 & c_{23} + c_{44} \end{cases} = 5 + \min \begin{cases} 0 + 3 \checkmark \\ 3 + 0 \end{cases}$$

$$= 8$$

$$r_{24} = 3$$

$$w_{03} = p(3) + q(3) + w(0,2)$$

$$= 1 + 1 + 12 = 14$$

$$c_{03} = 14 + \min_{i,j} \begin{cases} k=1 & c_{00} + c_{13} \\ k=2 & c_{01} + c_{23} \\ k=3 & c_{02} + c_{33} \end{cases} = 14 + \min \begin{cases} 0 + 12 \\ 8 + 3 \checkmark \\ 19 + 0 \end{cases}$$

$$= 14 + 11 = 25$$

$$r_{03} = 2$$

$$w_{14} = p(4) + q(4) + w(1,3) = 1 + 1 + 9 = 11$$

$$c_{14} = 11 + \min_{i,j} \begin{cases} k=2 & c_{11} + c_{24} \\ k=3 & c_{12} + c_{34} \\ k=4 & c_{13} + c_{44} \end{cases} = 11 + \min \begin{cases} 0 + 8 \checkmark \\ 7 + 3 \\ 12 + 0 \end{cases} = 19$$

$$r_{14} = 2$$

$$w_{04} = p(4) + q(4) + w(0,3) \\ = 1 + 1 + 14 = 16$$

$$C_{04} = 16 + \min \begin{cases} k=1 & C_{00} + C_{14} \\ k=2 & C_{01} + C_{24} \\ k=3 & C_{02} + C_{34} \\ k=4 & C_{03} + C_{44} \end{cases} = 16 + \min \begin{cases} 0 + 19 \\ 8 + 8 \checkmark \\ 19 + 3 \\ 25 + 0 \end{cases} \\ = 16 + 16 = 32$$

$$r_{04} = 2$$

$C_{04} = 32$ is the min cost of a BST for (a_1, a_2, a_3, a_4)

As $r_{04} = 2$ root is a_2

i.e left subtree is t_{01} & right subtree is t_{24}

t_{01} has root a_1 & subtrees t_{00} & t_{11}

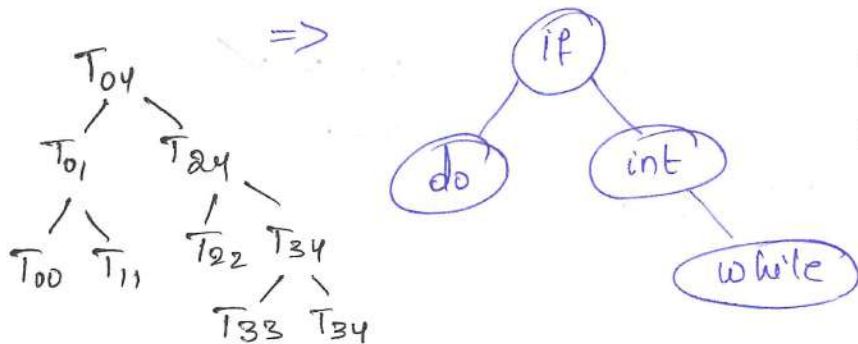
t_{24} " " a_3 " " t_{22} & t_{34}

As $r_{04} = 2$, a_2 is root for T_{04}

As $r_{01} = 1$, a_1 " " T_{01}

As $r_{24} = 3$, a_3 " " T_{24}

As $r_{24} = 4$, a_4 " " T_{34}



Total time to evaluate all $c(i,j)$'s & $r(i,j)$'s is $O(n^3)$

Algorithm OBST (P, q, n)

/* Given n distinct identifiers $a_1 < a_2 < \dots < a_n$ and probs $p[i], 1 \leq i \leq n$, and $q[i], 0 \leq i \leq n$, this algo computes $c[i,j]$ of OBSTs t_{ij} for identifiers $a_{i+1} \dots a_j$. It also computes $r[i,j]$, root of t_{ij} , $w[i,j]$ - weight of t_{ij} */

for $i := 0$ to $n-1$ do

 // Initialize

$w[i,i] := q[i]; r[i,i] := 0; c[i,i] := 0.0;$

// OBSTs with one node

$w[i, i+1] := q[i] + q[i+1] + p[i+1];$

$r[i, i+1] := i+1;$

$c[i, i+1] := q[i] + q[i+1] + p[i+1];$

}

$w[n,n] := q[n]; r[n,n] := 0; c[n,n] := 0.0;$

for $m := 2$ to n do // Find optimal trees with m nodes.

for $i := 0$ to $n-m$ do

{ $j := i+m;$

$w[i,j] := w[i,j-1] + p[j] + q[j];$

$k := \text{Find}(c, r, i, j);$

$c[i,j] := w[i,j] + c[i,k-1] + c[k,j];$

$r[i,j] := k;$

}
write(~~w~~ $c[0,n], w[0,n], r[0,n]$);

}

Algorithm Find(c, r, i, j)

{ $\min := \infty;$

for $m := r[i,j-1]$ to $r[i+1,j]$ do

if $(c[i,m-1] + c[m,j]) < \min$ then

{ $\min := c[i,m-1] + c[m,j]; l := m;$

return $l;$

}

0/1 Knapsack:-

Given a knapsack (bag or container) with

M - capacity of knapsack

n - no. of objects

w - an array consisting of weights $w_1, w_2, w_3 \dots w_n$.

P - " " Profits $P_1, P_2, P_3 \dots P_n$

X - an array " " " 0 or 1

$X = 0$ - i th object has not been selected

$= 1$ " " been selected.

Main objective is to place the objects into the knapsack so the max profit is obtained.

$$\text{Maximize } \sum_{i=1}^n P_i X_i \quad \text{subject to } \sum_{i=1}^n w_i X_i \leq M.$$

Dyn Prog for 0/1 Knapsack:-

Let i represent i th obj in question to be selected or not.
with to get max profit with weight $w_i \leq P_i$ with remaining capacity j .

$V[i, j]$ is the profit obtained by considering all i objs with capacity j .

Case 1: If there are no objects (i.e. $i=0$) or capacity of knapsack itself is zero (i.e. $j=0$) the profit = 0,

$$V[i, j] = 0 \quad \text{if } (i=0 \text{ or } j=0)$$

Case 2: $V[i, j] = V[i-1, j]$ if $w_i > j$ (weight of i th obj is greater than remaining capacity j)

In this case obj is not selected, b'coz it cannot fit into the knapsack.

Case 3:- w_i is less than j , two alternatives

(6)

- Reject i^{th} obj. $\Rightarrow$ Profit $V[i, j] = V[i-1, j]$

- Select i^{th} " $\Rightarrow$ " $V[i, j] = V[i-1, j-w_i] + P_i$

whichever yields max profit is selected. considered.

$$\Rightarrow V[i, j] = \max \left\{ \underline{V[i-1, j]}, \underline{V[i-1, j-w_i] + P_i} \right\} \text{ if } w_i \leq j$$

$$V[i, j] = \begin{cases} 0 & \text{if } i=j=0 \\ V[i-1, j] & \text{if } w_i > j \\ \max(V[i-1, j], V[i-1, j-w_i] + P_i) & \text{if } w_i \leq j \end{cases}$$

$V[n, M] \rightarrow$ gives optimal profit obtained.

Ex. Apply bottom-up dynamic programming technique to the following instance of the knapsack problem with capacity $M=5$

Item	Weight	Value
1	2	\$ 12
2	1	\$ 10
3	3	\$ 20
4	2	\$ 15

Soln :-

$N=4$

$w_1=2, w_2=1, w_3=3, w_4=2$

$P_1=12, P_2=10, P_3=20, P_4=15$

Soln matrix consists of $N+1$ rows $\leq M+1$ columns

Step 1:- In bottom up approach, start from initial soln $i=0$ or $j=0$

$i=0$, no objs are there

		0	1	2	3	4	5
i	0	0	0	0	0	0	0
	1	0					
	2	0					
	3	0					
$N=4$	4	0					
		j					M

Profit-table

Step 2

when $i=1$, $w_i=2$, $P_i=12$.

check whether P_i can be selected or not.

w_i j

$$2 > 1 \quad v[1,1] = v[0,1] = 0$$

$$2 \quad 2 \quad v[1,2] = \max(v[0,2], v[0,0] + 12) = \max(0, 0+12) = 12$$

$$2 \quad 3 \quad v[1,3] = \max(v[0,3], v[0,1] + 12) = \max(0, 0+12) = 12$$

$$2 \quad 4 \quad v[1,4] = \max(v[0,4], v[0,2] + 12) = \max(0, 0+12) = 12$$

$$2 \quad 5 \quad v[1,5] = \max(v[0,5], v[0,3] + 12) = \max(0, 0+12) = 12$$

Step 3

when $i=2$, $w_i=1$, $P_i=10$

w_i j

$v[2, \cdot]$

$$1 \quad 1 \quad v[2,1] = \max(v[1,1], v[1,0] + 10) = \max(0, 0+10) = 10$$

$$1 \quad 2 \quad v[2,2] = \max(v[1,2], v[1,1] + 10) = \max(12, 0+10) = 12$$

$$1 \quad 3 \quad v[2,3] = \max(v[1,3], v[1,2] + 10) = \max(12, 12+10) = 22$$

$$1 \quad 4 \quad v[2,4] = \max(v[1,4], v[1,3] + 10) = \max(12, 12+10) = 22$$

$$1 \quad 5 \quad v[2,5] = \max(v[1,5], v[1,4] + 10) = \max(12, 12+10) = 22$$

Step 4 when $i=3$, $w_i=3$, $P_i=20$

w_i j

$$3 > 1 \quad v[3,1] = \max(v[2,1], [2, \cdot] = 10) = 10$$

$$3 > 2 \quad v[3,2] = \max(v[2,2], [2, \cdot] = 12) = 12$$

$$3 \quad 3 \quad v[3,3] = \max(v[2,3], [2,0] + 20) = \max(22, 0+20) = 22$$

$$3 \quad 4 \quad v[3,4] = \max(v[2,4], [2,1] + 20) = \max(22, 10+20) = 30$$

$$3 \quad 5 \quad v[3,5] = \max(v[2,5], [2,2] + 20) = \max(22, 12+20) = 32$$

Step 5

(7)

When $i=4$, $w_i=2$, $P_i=15$

w_i j

$$\begin{aligned} 2 > 1 & \quad v[4,1] = v[3,1] = 10 \\ 2 \quad 2 & \quad v[4,2] = \max(v[3,2], v[3,0]+15) = \max(12, 0+15) = 15 \\ 2 \quad 3 & \quad v[4,3] = \max(v[3,3], v[3,1]+15) = \max(22, 10+15) = 25 \\ 2 \quad 4 & \quad v[4,4] = \max(v[3,4], v[3,2]+15) = \max(30, 12+15) = 30 \\ 2 \quad 5 & \quad v[4,5] = \max(v[3,5], v[3,3]+15) = \max(32, 22+15) = 37 \end{aligned}$$

Profit table

		$j \rightarrow M$					
		0	1	2	3	4	5
i	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

$N=4$ $\rightarrow$ optimal soln.

$$V[N, M] = V[4, 5] = 37$$

Objects selected: - Now remaining capacity $j=5$
 Instead $(x_1, x_2, x_3, x_4) = (0, 0, 0, 0) \rightarrow$ indicate no obj's selected

Step 1:- $V[4, 5] = 37$ $V[3, 5] = 32$ since 2 values are different, thus

indicates object $i=4$ is selected. so, $x(4)=1$

$\Rightarrow$ remaining capacity $j = 5 - w_4 = 5 - 2 = 3$

$i = i-1$ i.e. $i=3$, so consider $V[i, j]$ i.e. $V[3, 3]$

Step 2:- $V[3, 3] = 22$ $V[2, 3] = 22 \rightarrow$ object 3 is not selected $x(3)=0$

Steps :- $i=i-1$ $i=2$ with earlier remaining capacity

$V[2, 3] = 22$ $V[1, 3] = 12 \Rightarrow x(2)=1$

$\Rightarrow$ remaining capacity $j = 3 - w_2 = 3 - 1 = 2$

$i = i - 1 = 1$

so consider $v[i, j]$ i.e. $v[1, 2]$

step 4: $v[1, 2]$ $v[0, 2] \Rightarrow x(1) = 1$
1 2 0

$j = 2 - w_1 = 2 - 2 = 0$

$i = i - 1 = 0$

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

ALGORITHM

KNAPSACK (n, m, w, p, v)

1/k To find optimal soln for knapsack problem using dynamic programming

n - no. of obj's selected

m - capacity of knapsack

w - weights of all the objects

p - profits " "

o/p - v - optimal soln

for $i \leftarrow 0$ to n do

for $j \leftarrow 0$ to m do

if ($i = 0$ or $j = 0$)

$v[i, j] = 0$

else if ($w[i] > j$)

$v[i, j] = v[i - 1, j]$

else

$v[i, j] = \max(v[i - 1, j], v[i - 1, j - w[i]] + p[i])$

end if

end for

end for

return

0/1 Knapsack Problem

(7a)

Dynamic programming solution

Method 2 -

Let S^i represent the possible states resulting from 2^i decision sequences for $x_1, \dots, x_i$.

A state refers to a pair (P_j, w_j)

w_j - Total weight of objects included in the knapsack.

P_j - Corresponding profit.

To obtain S^{i+1} , the possibilities for x_{i+1} are $x_{i+1} = 0$ or $x_{i+1} = 1$.

- When $x_{i+1} = 0$, the resulting states are same as S^i ,

- " " = 1,

adding (P_{i+1}, w_{i+1}) to each state in S^i .

Can the resultant set be S_1^i , i.e. $S_1^i = \{(P, w) / P = P_i, w = w_i\} \in S^i$

- S^{i+1} can be computed by merging the states in S^i and S_1^i together, and then purging or discarding

by dominance rule.

Dominance rule: If S^{i+1} contains two pairs (P_j, w_j) and (P_k, w_k) with the property that $P_j < P_k$ and $w_j \geq w_k$, then the pair (P_j, w_j) can be discarded

i.e. Dominated tuples gets purged.

i.e. (P_k, w_k) dominates (P_j, w_j)

Ex: 1:- Consider $n=3$, $(w_1, w_2, w_3) = (2, 3, 4)$

$$(p_1, p_2, p_3) = (1, 2, 5) \text{ s.t. } m=6$$

$$S^0 = \{(0,0)\} \quad S_i = S_i + (p_{i+1}, w_{i+1})$$

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

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

$$S_1^1 = \{(2,3), (3,5)\}$$

$$S^2 = \{(0,0), (1,2), (2,3), (3,5)\}$$

$$S_1^2 = \{(5,4), (6,6), (7,7), (8,9)\}$$

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

Total weight exceeds m
so discard this two

$(3,5)$ is discarded

using purging rule with $(5,4)$ after merging.

Consider last tuple in S^3 .

$$(6,6) \in S^3$$

$$\notin S^2 \Rightarrow x_3 = 1$$

$$(6 - p_3, 6 - w_3)$$

$$(1, 2) \in S^2$$

$$\in S^1 \Rightarrow x_2 = 0$$

$$(1, 2) \in S^1$$

$$\notin S^0 \Rightarrow x_1 = 1$$

$\Rightarrow$ Optimal solution is $(x_1, x_2, x_3) = (1, 0, 1)$

$$\rightarrow P = (1, 2, 3, 3) \text{ , } w = (2, 2, 2, 3, 15) \text{ , } C = 42 \text{ , } n = 4$$

Optimal soln $(1, 0, 1, 1)$

Find
~~Print~~ optimal soln.

(3)

for $i \leftarrow 0$ to $n-1$ do

$x[i] \leftarrow 0$

end for

$i = n$

$j = m$

while ($i \neq 0$ and $j \neq 0$)

if ($v[i][j] \neq v[i-1][j]$)

$x[i] = 1$

$j = j - w[i]$

end if

$i = i - 1$

end while

All-pairs shortest paths

Let $G = (V, E)$ be a directed graph with n vertices.

Let cost be a cost or adjacency matrix for G , such that $\text{cost}(i, i) = 0$
 $1 \leq i \leq n$.

$\text{cost}(i, j)$ is the length (cost) of edge $\langle i, j \rangle$ if $\langle i, j \rangle \in E(G)$

$\text{cost}(i, j) = \infty$ if $i \neq j$ and $\langle i, j \rangle \notin E(G)$

The all pairs-shortest problem is to determine a matrix A such that $A(i, j)$ is the length of a shortest path from i to j .

$C(i, j) \geq 0$, for every edge $\langle i, j \rangle$, we require G have no cycles with $-ve$ length.

shortest i to j path in G , $i \neq j$, originated at i &

goes through some intermediate vertices & terminates at j .

i.e. if k is some " vertex, then i to k &

k to j must be shortest paths.

Dynamic programming approach:-

If 'k' is an intermediate vertex on this length shortest path in G going through no vertex with index greater than $k-1$. Similarly the k to j path is the shortest k to j path in G going through no vertex of index greater than $k-1$.

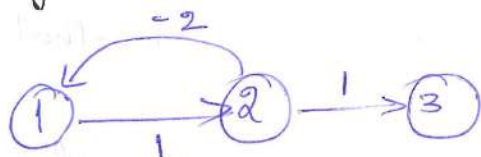
Let $A^k(i, j)$ - represent the length of a shortest path from i to j going through no vertex of index greater than k .

$$A^k(i, j) = \min \left\{ \min_{1 \leq k \leq n} \{ A^{k-1}(i, k) + A^{k-1}(k, j) \}, \text{cost}(i, j) \right\}$$

$$A^0(i, j) = \text{cost}(i, j) \quad 1 \leq i \leq n, 1 \leq j \leq n.$$

$$A^k(i, j) = \min \{ A^{k-1}(i, j), A^{k-1}(i, k) + A^{k-1}(k, j) \}, \quad k \geq 1$$

Ex



$$A^0 = \begin{bmatrix} 0 & 1 & \infty \\ -2 & 0 & 1 \\ \infty & \infty & 0 \end{bmatrix}$$

$$A^2(1, 3) \neq \min \{ A^1(1, 3), A^1(1, 2) + A^1(2, 3) \} = 2$$

Instead we see that $A^2(1, 3) = -\infty$

The length of the path $1, 2, 1, 2, 1, 2, \dots, 1, 2, 3$

Due to the presence of the cycle $(2, 1)$ which has a length of -1

→ A^n can be computed by first computing A^1, A^2 , then A^3 and soon.

Since there is no vertex in G with index greater than n ,

$A(i, j) = A^n(i, j)$. Function Allpaths computes $A^n(i, j)$

Algorithm Allpaths (cost, A, n)

cost[1:n, 1:n] is the cost adjacency matrix of a graph with n vertices. A[i, j] is the cost of a shortest path from vertex i to vertex j. cost[i, i] = 0.0, for $1 \leq i \leq n$.

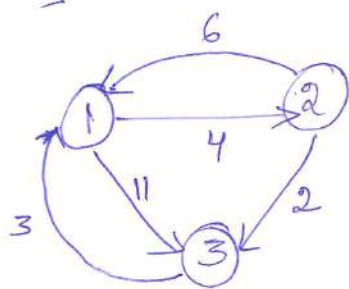
```

{
  for i := 1 to n do
    for j := 1 to n do
      A[i, j] := cost[i, j];
    for k := 1 to n do
      for i := 1 to n do
        for j := 1 to n do
          A[i, j] := min (A[i, j], A[i, k] + A[k, j]);
        }
      }
}

```

$\rightarrow \Theta(n^3)$

Ex



A^0	1	2	3
1	0	4	11
2	6	0	2
3	3	6	0

A^1	1	2	3
1	0	4	11
2	6	0	2
3	3	7	0

A^2	1	2	3
1	0	4	6
2	6	0	2
3	3	7	0

A^3	1	2	3
1	0	4	6
2	5	0	2
3	3	7	0

$$A^1(1, 2) = \min \{ A^0(1, 2), A^0(1, 1) + A^0(1, 2) \}$$

$$= \min \{ 4, 0 + 4 \} = 4$$

$$A^1(1, 3) = \min \{ A^0(1, 3), A^0(1, 1) + A^0(1, 3) \}$$

$$= \min \{ 11, 0 + 11 \} = 11$$

$$A^1(2,1) = \min \{ A^0(2,1), A^0(2,1) + A^0(1,1) \}$$

$$= \min \{ 6, 6 + 0 \} = 6$$

$$A^1(2,3) = \min \{ A^0(2,3), A^0(2,1) + A^0(1,3) \}$$

$$= \min \{ 2, 6 + 11 \} = 2$$

$$A^1(3,1) = \min \{ A^0(3,1), A^0(3,1) + A^0(1,3) \}$$

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

$$A^1(3,2) = \min \{ A^0(3,2), A^0(3,1) + A^0(1,2) \}$$

$$= \min \{ 2, 3 + 4 \} = 7$$

$$A^2(1,2) = \min \{ A^1(1,2), A^1(1,2) + A^1(2,2) \}$$

$$= \min \{ 4, 4 + 0 \} = 4$$

Traveling Salesperson Problem:-

(15)

This is a permutation problem.

Let $G = (V, E)$ be a directed graph with edge costs c_{ij}

where $c_{ij} > 0 \forall i \text{ and } j$
 $= \infty$ if $\langle i, j \rangle \notin E$.

$|V| > n$ and assume $n > 1$.

The tour of G is a directed simple cycle that includes every vertex in V . Cost of a tour is the sum of the cost of the edges on the tour. The TSP finds a tour of min cost.

Application:- 1. "Route a postal van to pickup mail from mail boxes located at 'n' different sites. An $n+1$ vertex graph represents this situation. One vertex represents a post office from which the postal van starts & to which it must return. Edge $\langle i, j \rangle$ is assigned a cost equal to the distance from site i to j . Minimize the tour length.

2. Robot arm to tighten the nuts on some piece of machinery on an assembly line. Arm should start from the initial position successively move to each of the remaining nuts & return to the initial pos. Part of path of the arm is the tour, vertices represent the nuts. Minimize time needed for arm to complete the task.

3. Production environment in which several commodities are manufactured on the same set of machines. Manufacture proceeds in cycles. In each prod cycle, n diff commodities are produced. When the machines are changed from production of commodity 'i' to 'j', a change over ^{cost} c_{ij} is incurred. Find a sequence in which it minimizes the sum of change over costs. It is necessary to include the cost of starting the next cycle. This is just change over cost from last to first commodity.

This is TSP on an ' n ' vertex graph with edge cost c_{ij} 's being the changeover cost from commodity i to j .

⇒ Let tour is a simple path that starts & ends at vertex 1.

Every tour consists of an edge $\langle 1, k \rangle$ for some $k \in V - \{1\}$ and a path from vertex k to 1. The path from vertex k to 1 goes through each vertex in $V - \{1, k\}$ exactly once.
 (one)

If the tour is optimal, then the path from k to 1 must be a shortest k to 1 path going thru all vertices in $V - \{1, k\}$

Let $g(i, S)$ be the length of the shortest path starting at vertex i going thru all vertices in S , and terminating at vertex 1.

$$g(1, V - \{1\})$$

$g(1, V - \{1\})$ is the length of an optimal salesperson tour.

From the principle of optimality

$$g(1, V - \{1\}) = \min_{2 \leq k \leq n} \{c_{1k} + g(k, V - \{1, k\})\} \quad \text{--- (1)}$$

Generalize the above eqn (for $i \notin S$)

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

Eq (1) can be solved for $g(1, V - \{1\})$ if we know $g(k, V - \{1, k\})$ $\forall k$.

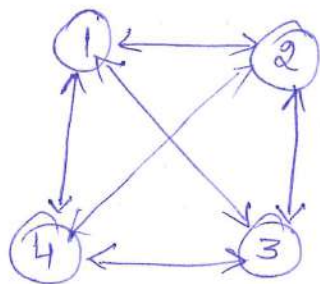
g values can be obtained by (2), to obtain $g(i, S) \forall |S| = 1$

$$g(i, \emptyset) = c_{i1}, \quad 1 \leq i \leq n. \quad \left| \quad \text{then " } g(i, S) \forall |S| = 2 \text{ } \right. \begin{matrix} \text{Base} \\ \text{Step} \end{matrix}$$

when $|S| < n-1$, the values of i and S for which $g(i, S)$ is needed are such that $i \neq 1$, $1 \notin S$, and $i \notin S$.

Consider the directed graph.

Edge lengths are given



0	10	15	20
5	0	9	10
6	13	0	12
8	8	9	0

Step 1

Given adj matrix

Given adj matrix

c_{ij}

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

Let tour starts at v_1

Step 1

$|S| = 0$

$$g(i, \emptyset) = c_{i1}$$

$$\Rightarrow g(1, \emptyset) = 0$$

$$i=2 \quad g(2, \emptyset) = 5$$

$$i=3 \quad g(3, \emptyset) = 6$$

$$i=4 \quad g(4, \emptyset) = 8$$

Step 2

$|S| = 1$

$i \notin S$ S can be $\{2\}$ or $\{3\}$ or $\{4\}$

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

$$g(2, \{3\}) = c_{23} + g(3, \emptyset) = 15$$

$$i=2 \quad g(2, \{3\}) = c_{23} + g(3, \emptyset) = 15$$

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

$$i=3 \quad g(3, \{2\}) = c_{32} + g(2, \emptyset) = 13 + 5 = 18$$

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

$$i=4 \quad g(4, \{2\}) = c_{42} + g(2, \emptyset) = 8 + 5 = 13$$

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

11 Step 3 compute $g(i, s)$ with $|s|=2, i \neq 1, 1 \notin s$ and $i \notin s$

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

$$= \min \{9 + 20, 10 + 15\}$$

$$= \min \{29, 25\} = 25 \quad \& \quad j=4 \quad \text{ie which } j \text{ gives min value,}$$

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

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

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

$$= 25$$

$$\& \quad j=4$$

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

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

$$= \min \{23, 27\} = 23 \quad \& \quad j=2$$

Step 4 compute $g(i, i)$ with $|s|=3$

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

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

$$= \min \{35, 40, 43\} = 35 \quad \& \quad j=2$$

$\Rightarrow$ optimal tour has length 35,

To construct tour of this length, consider the val of j that minimizes R.H.S.

$$\text{ie } J(i, s) = J(1, \{2, 3, 4\}) = 2, \quad \text{Tour starts at 1, \& goes to 2} \quad 1 \rightarrow 2$$

$$\text{consider } g(2, \{3, 4\}) \text{ ie } J(2, \{3, 4\}) = 4, \text{ so next edge is } \langle 2, 4 \rangle$$

$$1 \rightarrow 2 \rightarrow 4$$

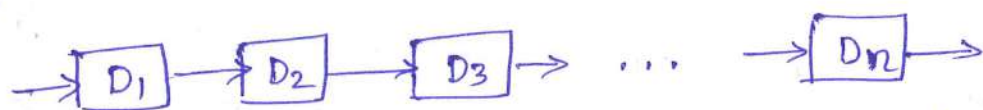
$$\text{now, } g(4, \{3\}) \text{ ie } J(4, \{3\}) = 3 \quad \Rightarrow 1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 1 \text{ is the optimal tour.}$$

Time complexity = $\Theta(n^2 2^n)$ $\rightarrow$ This is better than enumerating all $n!$ different tours & find

Space " = $O(n 2^n)$ $\rightarrow$ This is too large even for modest values of n .

Reliability Design

The problem is to design a system that is composed of several devices connected in series.



n devices D_i , $1 \leq i \leq n$, connected in series.

Let r_i be the reliability of device D_i (i.e. r_i is the prob that device 'i' will function properly).

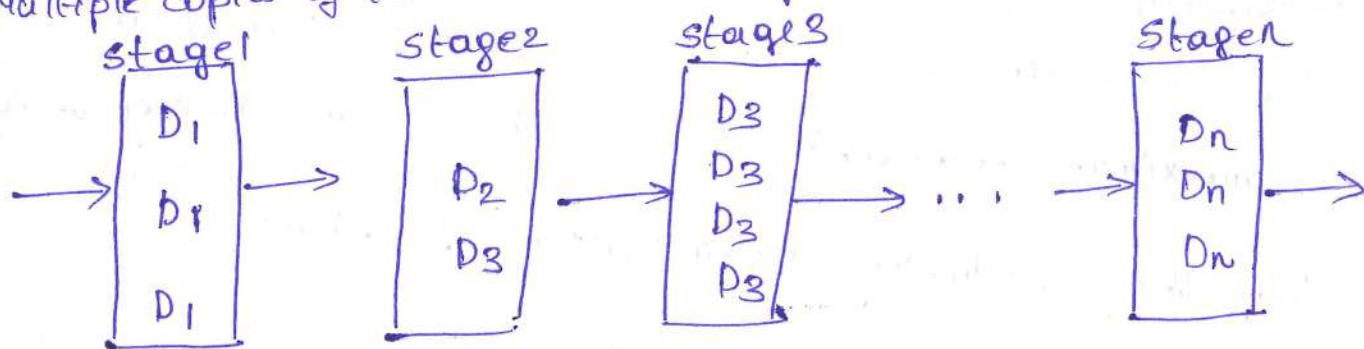
Then, the reliability of the entire system is $\prod r_i$.

Even if the individual devices are very reliable, the reliability of the system may not be very good.

Ex: if $n=10$, and $r_i = .99$, $1 \leq i \leq 10$, then $\prod r_i = 0.904$

Hence, it is desirable to duplicate devices.

Multiple copies of the same device type are connected in parallel



Multiple devices connected in parallel in each stage.

→ All stages are connected using switching ~~devices~~ circuits.

These switching circuits determine with which devices in any given group are functioning properly. They can make use of one such device at each stage.

If stage 'i' contains m_i copies of device D_i , then the prob that all m_i have a malfunction is $(1-r_i)^{m_i}$,
 hence the reliability of stage 'i' becomes $1 - (1-r_i)^{m_i}$.

Eg. if $r_i = .99$ and $m_i = 2$, the stage reliability = 0.999.
 In practical situations, the stage reliability is a little less than $1 - (1-r_i)^{m_i}$, because the switching circuits themselves are not reliable.

Also, failures of copies of the same device may not be fully independent.

Let Reliability of stage 'i' is given by a function $\phi_i(m_i)$, $1 \leq i \leq n$
 The reliability of the system of stages is $\prod_{i=1}^n \phi_i(m_i)$

Our problem is to use device duplication to maximize reliability.

Let c_i be the cost of each unit of device i and let c be the maximum allowable cost of the system being designed.

Solve the following maximization problem:

$$\text{Maximize } \prod_{1 \leq i \leq n} \phi_i(m_i)$$

$$\text{Subject to } \sum_{1 \leq i \leq n} c_i m_i \leq c$$

Dynamic programming solution:-

Assume each $c_i > 0$, each m_i must be in the range $1 \leq m_i \leq u_i$,

$$\text{where } u_i = \left\lceil (c + c_i - \sum_{j=1}^n c_j) / c_i \right\rceil$$

An optimal solution $m_1, m_2, \dots, m_n$ is the result of a sequence⁽¹³⁾ of decisions, one decision for each m_i .

Let $f_i(x)$ represent the maximum value of $\sum_{1 \leq j \leq i} \phi(m_j)$ subject

to the constraints $\sum_{1 \leq j \leq i} c_j m_j \leq x$ and $1 \leq m_j \leq u_j, 1 \leq j \leq i$.

then value of optimal soln is

$$f_n(c) = \max_{1 \leq m_n \leq u_n} \{ \phi_n(m_n) + f_{n-1}(c - c_n m_n) \}$$

for any $f_i(x)$, $i \geq 1$, this equation generalizes to

$$f_i(c) = \max_{1 \leq m_i \leq u_i} \{ \phi_i(m_i) + f_{i-1}(x - c_i m_i) \}$$

Let S_i consist of tuples of the form (f, x) where $f = f_i(x)$

There is at most one tuple for each different x that results from a sequence of decisions on $m_1, m_2, \dots, m_n$.

The dominance rule (f_1, x_1) dominates (f_2, x_2) iff $f_1 \geq f_2$ and $x_1 \leq x_2$ holds for the problem too. Hence dominated tuples can be discarded from S^i .

→ Design a three stage system with device types D_1, D_2, D_3 .

The costs are \$30, \$15, \$20 respectively. The costs of the system is to be no more than \$105. The reliability of each device type is .9, .8 and .5 respectively.

→ Assume that if stage 'i' has m_i devices of type i in parallel, then $\phi_i(m_i) = 1 - (1 - r_i)^{m_i}$. → $\phi_i(m_i)$ - reliability func.

Given $c_1 = 30, c_2 = 15, c_3 = 20, c = 105,$

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

$u_1 = 2, u_2 = 3, u_3 = 3$

$u_i = \text{Upper bound} = \left\lfloor (c + c_i - \sum_{j=1}^n c_j) / c_i \right\rfloor$

$u_1 = \left\lfloor (105 + 30 - (30 + 15 + 20)) / 30 \right\rfloor$

$= 2$

$u_2 = \left\lfloor (105 + 15 - (30 + 15 + 20)) / 15 \right\rfloor$

$= 3$

$u_3 = \left\lfloor (105 + 20 - (30 + 15 + 20)) / 20 \right\rfloor$

$= \underline{\underline{3}}$

$\phi_i(m_i) = 1 - (1 - r_i)^{m_i}$, $\phi_i(m_i)$ is a reliability function

Each pair in this is represented by (r, x) where r is reliability, x is a cost.

Using s_j^i to represent all tuples of obtainable from s^{i-1}

by choosing $m_i = 1$

→ Beginning with $s^0 = \{(1, 0)\}$

Since $1 \leq m_i \leq u_i$ i.e. $1 \leq m_1 \leq 2$,

$m_1 = 1$ or 2 , it means $j = 1$ or 2 .

So, by using s_j^i calculate s_1^1 here $i = 1, j = 1$, i.e., $m_1 = 1$

$\phi_1(m_1) = 1 - (1 - r_1)^{m_1} = 1 - (1 - 0.9)^1 = 1 - 0.1 = 0.9$

$s^0 = \{(1, 0)\}$

In each tuple (r, x) reliability is multiplied with

previous reliability & cost is added to previous cost.

$$S_1' = \{(0.9, 30)\}$$

(14)

Here 0.9 is multiplied with 1 and $c_1 = 30$ is added to 0.

$$\underline{S_2^1} \quad i=1, \quad j=2 \quad \text{i.e.} \quad m_1 = 2$$

$$\phi_2^1(m_1) = 1 - (1 - x_1)^2 = 1 - (1 - 0.9)^2 = 1 - (0.01) = 0.99$$

0.99 is multiplied with 1 and cost $2c_1 = 60$ (since $j=2$) is added to 0.

$$S_2' = \{(0.99, 60)\}$$

S' is obtained by merging sets S_1' , S_2'

$$S' = \{(0.9, 30), (0.99, 60)\}$$

→ Calculate S_1^2, S_2^2, S_3^2 from S'

$$\underline{S_1^2} \quad j=1, \quad \text{i.e.} \quad m_2 = 1, \quad i=2$$

$$\phi_2(m_2) = 1 - (1 - x_2)^{m_2} = 0.8$$

$$S_1^2 = \{(0.9 \times 0.8, 30 + 15), (0.99 \times 0.8, 60 + 15)\} \\ = \{(0.72, 45), (0.792, 75)\}$$

$$\underline{S_2^2} \quad i=2, \quad j=2, \quad \text{i.e.} \quad m_2 = 2$$

$$\phi_2(m_2) = 1 - (1 - x_2)^2 = 0.96$$

Since $j=2$, $2c_2 = 2 \times 15 = 30$ is added to S' pairs.

$$\Rightarrow S_2^2 = \{(0.9 \times 0.96, 30 + 30), (0.99 \times 0.96, 60 + 30)\} \\ = \{(0.864, 60), (0.9504, 90)\}$$

41 $S_3^2 \quad i=2, j=3 \quad \text{i.e. } m_2=3$

$$\phi_2(m_2) = 1 - (1 - r_2)^{m_2} = 0.992$$

Since $j=3$, $3c_2 = 3 \times 15 = 45$

$$S_3^2 = \{ (0.9 \times 0.992, 30+45), (0.99 \times 0.992, 60+45) \}$$

$$= \{ (0.8928, 75), (0.98208, 105) \}$$

$$S^2 = S_1^2 \cup S_2^2 \cup S_3^2$$

$$= \{ (0.72, 45), (0.864, 60), (0.9504, 90), (0.8928, 75), (0.98208, 105) \}$$

→ where $(0.792, 75)$ is removed from S^2 b'coz of purge rule.

Observe pairs, cost is in increasing order

i.e. $45 < 60 < 90 < 75$ here $90 < 75$, condition fails.

so, $(0.9504, 90)$ is not in appropriate order, so remove it.

$$S^2 = \{ (0.72, 45), (0.864, 60), (0.8928, 75), (0.98208, 105) \}$$

Calculate S_1^3, S_2^3, S_3^3 from S^2 .

$$S_1^3 \quad i=3, j=1, m_3=1$$

$$\phi_3(m_3) = 1 - (1 - r_3)^{m_3} = 0.5$$

$c_3 = 20$ is added to S^2 ,

$$S_1^3 = \{ (0.72 \times 0.5, 45+20), (0.864 \times 0.5, 60+20), (0.8928 \times 0.5, 75+20), (0.98208 \times 0.5, 105+20) \}$$

last tuple exceeds given cost 105, remove it.

$$S_1^3 = \{ (0.36, 65), (0.432, 80), (0.4464, 95) \}$$

$$\underline{S_2^3} = i=3, j=2, \text{ i.e. } m_3=2.$$

$$\text{Since } j=2, 2C_3 = 2 \times 20 = 40$$

$$\phi_3(m_3) = 1 - (1 - r_3)^{m_3} = 0.75$$

$$S_2^3 = \{ (0.72 \times 0.75, 45+40) \quad (0.864 \times 0.75, 60+40) \} \\ = \{ (0.54, 85), (0.648, 100) \}$$

$$\underline{S_3^3} \quad i=3, j=3 \quad \text{i.e. } m_3=3$$

$$\text{Since } j=3, 3C_3 = 3 \times 20 = 60$$

$$\phi_3(m_3) = 1 - (1 - r_3)^{m_3} = 0.875$$

$$S_3^3 = \{ (0.72 \times 0.875, 45+60) \} = \{ (0.63, 105) \}$$

$$\underline{S^3} = S_1^3 \cup S_2^3 \cup S_3^3$$

$$= \{ (0.36, 65), (0.432, 80), (0.4464, 95), (0.54, 85), \\ (0.648, 100), (0.63, 105) \}$$

Apply Pareto & Dominance rule,

$$S^3 = \{ (0.36, 65), (0.432, 80), (0.54, 85), (0.648, 100) \}$$

$(0.648, 100)$ is present in S_2^3 i.e. $i=3$ & $j=2$ so, $m_3=2$.

- it is obtained from $(0.864, 60)$

$(0.864, 60)$ is present in S_2^2 , $i=2$ and $j=2$, so $m_2=2$

- it is obtained from $(0.9, 30)$

$(0.9, 30)$ is present in S_1^1 , $i=1$ and $j=1$ so $m_1=1$.

$$\therefore m_1=1, m_2=2, m_3=2.$$

21

$$i=0$$

$$s^0$$

$$s^0$$

$$i=1, j=m_1=1$$

$$s_1^1$$

$$j=m_1=2$$

$$s_2^1$$

$$s^1$$

$$i=2, j=m_2=1$$

$$s_1^2$$

$$j=m_2=2$$

$$s_2^2$$

$$j=m_2=3$$

$$s_3^2$$

$$s^2$$

$$i=3, j=m_3=1$$

$$s_1^3$$

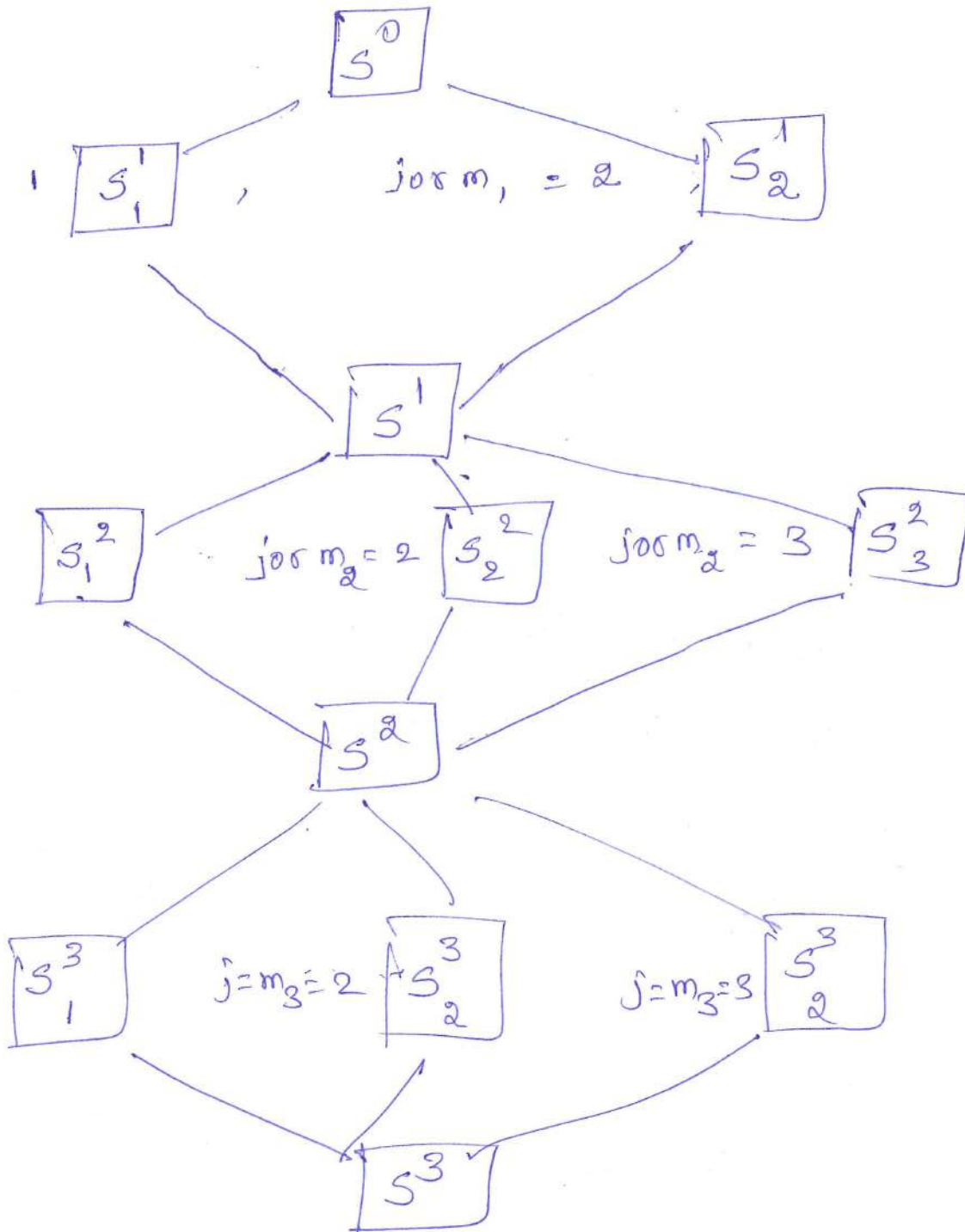
$$j=m_3=2$$

$$s_2^3$$

$$j=m_3=3$$

$$s_2^3$$

$$s^3$$



Dynamic programming

A problem must have two characteristics to be suitable for dyn prog,

1. An optimal substructure: - A specified problem has optimal substructure property if its optimal solution can be derived from optimal solns of its subproblems.

2. Overlapping subproblems: - The problem in question should be able to be broken down into a cascade of reusable subproblems or a recursive algo can be repeatedly solve subproblems without generating new ones.

- If problem doesnot posses these properties, DP cannot be applied to the problem.

Dyn. prog uses memorization, storing & tabulating solns so that they can be retrieved quickly, saving the time of repeating the computation.

Top-down approach:-

- Soln to a problem is recursive,
- Allow memorization of soln to the overlapping subproblem
- If the subproblem has not been solved previously find the soln of SP, & store soln to memorization cache
- New subproblems are solved by referring to the memorized data cache

Adv:- Straightforward to understand

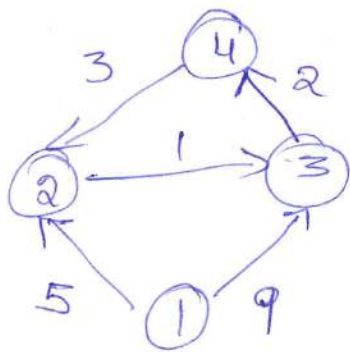
- Debugging can be easily done

Disadv:- Recursion can use lot of cache mem, which can lead to stack overflow errors

Bottom-up approach:-

- called as tabulation or table filling.
- Create a record of stored subproblems & solution and fill the table.
- Then use the tabulated solved subproblems for bottom-up reformulation of the problem.
- First solve smaller subproblems, with those solutions, solve the bigger sps.
- 'For' loop can be used to iterate the subproblems

→ Solve All pairs shortest path



$$A^0 = \begin{bmatrix} 0 & 5 & 9 & \infty \\ \infty & 0 & 1 & \infty \\ \infty & \infty & 0 & 2 \\ \infty & 3 & \infty & 0 \end{bmatrix}$$

$$A^1 = \begin{bmatrix} 0 & 5 & 9 & \infty \\ \infty & 0 & 1 & \infty \\ \infty & \infty & 0 & 2 \\ \infty & 3 & \infty & 0 \end{bmatrix}$$

$$A^2 = \begin{bmatrix} 0 & 5 & 6 & \infty \\ \infty & 0 & 1 & \infty \\ \infty & \infty & 0 & 2 \\ \infty & 3 & 4 & 0 \end{bmatrix}$$

$$A^3 = \begin{bmatrix} 0 & 5 & 6 & 8 \\ \infty & 0 & 1 & 3 \\ \infty & \infty & 0 & 2 \\ \infty & 3 & 4 & 0 \end{bmatrix}$$

$$A^4 = \begin{bmatrix} 0 & 5 & 6 & 8 \\ \infty & 0 & 1 & 3 \\ \infty & 5 & 0 & 2 \\ \infty & 3 & 4 & 0 \end{bmatrix}$$

0/1 knapsack using dynamic programming

(17)

$$P = (11, 21, 31, 33)$$

$$W = (2, 12, 23, 15) \quad C = 42, \quad n = 4$$

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

$$S_1^0 = \{(11, 2)\}$$

$$S^1 = \{(0, 0), (11, 2)\}$$

$$S_1^1 = \{(21, 12), (32, 14)\}$$

$$S^2 = \{(0, 0), (11, 2), (21, 12), (32, 14)\}$$

$$S_1^2 = \{(31, 23), (42, 25), (52, 35), (63, 37)\}$$

$$S^3 = \{(0, 0), (11, 2), (21, 12), \underbrace{(32, 14)}_{\text{Dominated}}, (31, 23), (42, 25), (52, 35), (63, 37)\}$$

$$S_1^3 = \{(33, 15), (44, 17), (54, 27), (64, 38), (75, 40), \underbrace{(85, 50), (96, 52)}_{\text{Exceeds capacity } m=42}\}$$

$$S^4 = \{(0, 0), (11, 2), (21, 12), \underbrace{(31, 23)}_4, \underbrace{(42, 25)}_3, \underbrace{(52, 35)}_2, (63, 37), (33, 15), (44, 17), (54, 27), (64, 38), (75, 40)\}$$

↳ dominance (order of cancellation compared to $(33, 15)$)

$$(75, 40) \in S^4 \\ \notin S^3 \Rightarrow x_4 = 1$$

$$(75-33, 40-15)$$

$$(42, 25) \in S^3 \\ \notin S^2 \Rightarrow x_3 = 1$$

$$(42-31, 25-23)$$

$$(11, 2) \in S^2 \\ \in S^1 \Rightarrow x_2 = 0$$

$$(11, 2) \in S^1 \\ \notin S^0 \Rightarrow x_1 = 1$$

$$\Rightarrow (1, 0, 1, 1)$$