

UNIT III

Dynamic Programming: General method, applications- Optimal binary search trees, 0/1 knapsack problem, All pairs shortest path problem, Traveling sales person problem, Reliability design.

DYNAMIC PROGRAMMING

Dynamic programming is a name, coined by Richard Bellman in 1955. Dynamic programming, as greedy method, is a powerful algorithm design technique that can be used when the solution to the problem may be viewed as the result of a sequence of decisions. In the greedy method we make irrevocable decisions one at a time, using a greedy criterion. However, in dynamic programming we examine the decision sequence to see whether an optimal decision sequence contains optimal decision subsequence.

When optimal decision sequences contain optimal decision subsequences, we can establish recurrence equations, called *dynamic-programming recurrence equations*, that enable us to solve the problem in an efficient way.

Dynamic programming is based on the principle of optimality (also coined by Bellman). The principle of optimality states that no matter whatever the initial state and initial decision are, the remaining decision sequence must constitute an optimal decision sequence with regard to the state resulting from the first decision. The principle implies that an optimal decision sequence is comprised of optimal decision subsequences. Since the principle of optimality may not hold for some formulations of some problems, it is necessary to verify that it does hold for the problem being solved. Dynamic programming cannot be applied when this principle does not hold.

The steps in a dynamic programming solution are:

- ☐ Verify that the principle of optimality holds
- ☐ Set up the dynamic-programming recurrence equations
- ☐ Solve the dynamic-programming recurrence equations for the value of the optimal solution.
- ☐ Perform a trace back step in which the solution itself is constructed.

ALL PAIRS SHORTEST PATHS

In the all pairs shortest path problem, we are to find a shortest path between every pair of vertices in a directed graph G . That is, for every pair of vertices (i, j) , we are to find a shortest path from i to j as well as one from j to i . These two paths are the same when G is undirected.

When no edge has a negative length, the all-pairs shortest path problem may be solved by using Dijkstra's greedy single source algorithm n times, once with each of the n vertices as the source vertex.

The all pairs shortest path problem is to determine a matrix A such that $A(i, j)$ is the length of a shortest path from i to j . The matrix A can be obtained by solving n single-source

problems using the algorithm shortest Paths. Since each application of this procedure requires $O(n^2)$ time, the matrix A can be obtained in $O(n^3)$ time.

The dynamic programming solution, called Floyd's algorithm, runs in $O(n^3)$ time. Floyd's algorithm works even when the graph has negative length edges (provided there are no negative length cycles).

The shortest i to j path in G, $i \neq j$ originates at vertex i and goes through some intermediate vertices (possibly none) and terminates at vertex j. If k is an intermediate vertex on this shortest path, then the subpaths from i to k and from k to j must be shortest paths from i to k and k to j, respectively. Otherwise, the i to j path is not of minimum length. So, the principle of optimality holds. 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, we obtain:

$$A^k(i, j) = \{\min \{\min \{A^{k-1}(i, k) + A^{k-1}(k, j)\}, c(i, j)\} \mid 1 \leq k \leq n\}$$

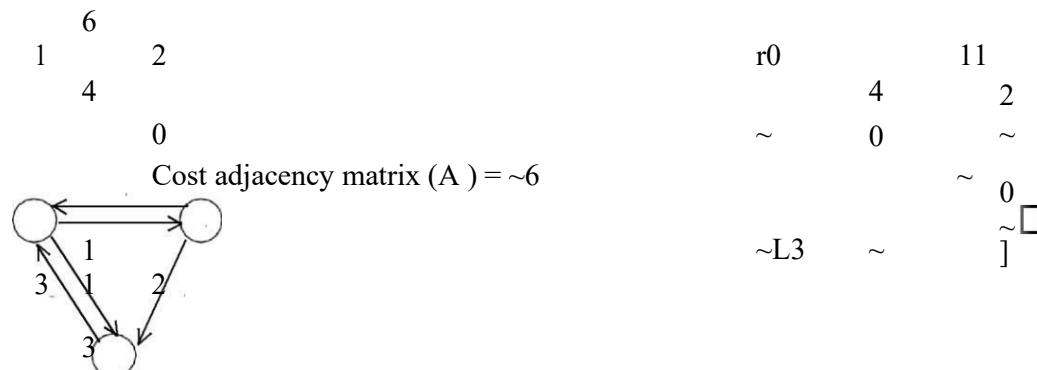
Algorithm All Paths (Cost, A, n)

```
// cost [1:n, 1:n] is the cost adjacency matrix of a graph which
// 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]; // copy cost into A.
        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]);
    }
```

Complexity Analysis: A Dynamic programming algorithm based on this recurrence involves in calculating $n+1$ matrices, each of size $n \times n$. Therefore, the algorithm has a complexity of $O(n^3)$.

Example 1:

Given a weighted digraph $G = (V, E)$ with weight. Determine the length of the shortest path between all pairs of vertices in G. Here we assume that there are no cycles with zero or negative cost.



General formula: $\min \{A^{k-1}(i, k) + A^{k-1}(k, j), c(i, j)\} \quad 1 < k < n$

Solve the problem for different values of $k = 1, 2$

and 3 **Step 1:** Solving the equation for, $k = 1$;

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

$$2) = \min \{(A^0(1, 1) + A^0(1, 2)), c(1, 2)\} = \min \{(0 + 4), 4\} = 4$$

$$A^1(1, 3) = \min \{(A^0(1, 1) + A^0(1, 3)), c(1, 3)\} = \min \{(0 + 11), 11\} = 11 \quad A^1(2,$$

$$1) = \min \{(A^0(2, 1) + A^0(1, 1)), c(2, 1)\} = \min \{(6 + 0), 6\} = 6$$

$$A^1(2, 2) = \min \{(A^0(2, 1) + A^0(1, 2)), c(2, 2)\} = \min \{(6 + 4), 0\} = 0 \quad A^1(2,$$

$$3) = \min \{(A^0(2, 1) + A^0(1, 3)), c(2, 3)\} = \min \{(6 + 11), 2\} = 2 \quad A^1(3, 1) = \min$$

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

$$+ A^0(1, 2)), c(3, 2)\} = \min \{(3 + 4), 0\} = 7 \quad A^1(3, 3) = \min \{(A^0(3, 1) + A^0(1,$$

$$3)), c(3, 3)\} = \min \{(3 + 11), 0\} = 0$$

$A^1(1)$

$$= \begin{matrix} & \sim 0 & 4 & 11 \\ & \sim & & \sim \\ & \sim 6 & 0 & 2 \sim \quad \square \\ & \sim L & & 0 \\ 3 & 7 & & \sim U \end{matrix}$$

Step 2: Solving the equation for, $K = 2$;

$A^2(1,$

$A^2(1,$

$A^2(1,$

$A^2(2,$

$A^2(2,$

$A^2(2,$

$A^2(3,$

$A^2(3,$

$A^2(3,$

$$1) = \min \{(A^1(1, 1) + A^1(2, 1), c(1, 1)\} = \min \{(4 + 6), 0\} + A^1(2, 2) = 0$$

$$2) = \min \{(A^1(1, 2) + A^1(2, 2), c(1, 2)\} = \min \{(4 + 0), 4\} + A^1(2, 3) = 4$$

$$3) = \min \{(A^1(1, 3) + A^1(2, 3), c(1, 3)\} = \min \{(4 + 2), 11\} = 6$$

$$1) = \min \{(A^1(2, 1) + A^1(2, 1), c(2, 1)\} = \min \{(0 + 6), 6\} = 6$$

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

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

$$\begin{aligned}
 1) &= \min \{(A(3, 2) + A(2, 1), c(3, 1))\} = \min \{(7 + 3) = 10\} \\
 2) &= \min \{(A(3, 2) + A(2, 2), c(3, 2))\} = \min \{(7 + 7) = 14\} \\
 3) &= \min \{(A(3, 2) + A(2, 3), c(3, 3))\} = \min \{(7 + 0) = 7\}
 \end{aligned}$$

$$\begin{aligned}
 A(2) &= \begin{matrix} \sim 0 & 4 & 6 \\ \sim & & 2 \end{matrix} \\
 &\quad \begin{matrix} \sim 6 \\ 3^L & 0 & \sim \\ & & 0 \\ & & \sim \end{matrix} \\
 &\quad \quad \quad 7
 \end{aligned}$$

Step 3: Solving the equation for, $k = 3$;

$$\begin{aligned}
 A^3(1, 3) &= \min \{A^2(1, 3) + A^2(1, 3), c(1, 3)\} = \min \{(6 + 0) = 6\} \\
 A^3(1, 3) &= \min \{A^2(1, 3) + A^2(1, 3), c(1, 3)\} = \min \{(6 + 4) = 10\} \\
 A^3(1, 3) &= \min \{A^2(1, 3) + A^2(1, 3), c(1, 3)\} = \min \{(6 + 6) = 12\} \\
 A^3(2, 3) &= \min \{A^2(2, 3) + A^2(2, 3), c(2, 3)\} = \min \{(2 + 6) = 8\} \\
 A^3(2, 3) &= \min \{A^2(2, 3) + A^2(2, 3), c(2, 3)\} = \min \{(2 + 0) = 2\} \\
 A^3(2, 3) &= \min \{A^2(2, 3) + A^2(2, 3), c(2, 3)\} = \min \{(2 + 2) = 4\} \\
 A^3(3, 3) &= \min \{A^2(3, 3) + A^2(3, 3), c(3, 3)\} = \min \{(0 + 3) = 3\} \\
 A^3(3, 3) &= \min \{A^2(3, 3) + A^2(3, 3), c(3, 3)\} = \min \{(0 + 7) = 7\}
 \end{aligned}$$

107

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

$$\begin{aligned}
 A(3) &= \begin{matrix} 4 & 6 \\ \sim 0 & \sim \\ & 0 & 2 \end{matrix} \\
 &\quad \begin{matrix} \sim 5 & 3 & 7 \\ & 0 & \sim \end{matrix}
 \end{aligned}$$

TRAVELLING SALESPERSON PROBLEM:

Let $G = (V, E)$ be a directed graph with edge costs C_{ij} . The variable c_{ij} is defined such that $c_{ij} > 0$ for all i and j and $c_{ij} = a$ if $\langle i, j \rangle \in E$. Let $|V| = n$ and assume $n > 1$. A tour of G is a directed simple cycle that includes every vertex in V . The cost of a tour is

the sum of the cost of the edges on the tour. The traveling sales person problem is to find a tour of minimum cost. The tour is to be a simple path that starts and ends at vertex 1.

Let $g(i, S)$ be the length of shortest path starting at vertex i , going through all vertices in S , and terminating at vertex 1. The function $g(1, V - \{1\})$ is the length of an optimal

salesperson tour. From the principal of optimality it follows that:

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

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

Generalizing equation 1, we obtain (for $i \in S$)

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

The Equation can be solved for $g(1, V - \{1\})$ if we know $g(k, V - \{1, k\})$ for all choices of k .

Complexity Analysis:

For each value of $|S|$ there are $n - 1$ choices for i . The number of distinct sets S of

size k not including 1 and i is $\binom{n-2}{k}$.

Hence, the total number of $g(i, S)$'s to be computed before computing $g(1, V - \{1\})$ is:

$$\sum_{k=1}^{n-2} \binom{n-2}{k} = 2^{n-2} - 1$$

$$k \sim 0$$

~ ~

To calculate this sum, we use the binominal theorem:

$$\sum_{k=0}^n \binom{n}{k} ((n-2) + i)^k ((n-2) - i)^{n-k}$$

According to the binominal theorem:

$$\sum_{k=0}^n \binom{n}{k} ((n-2) + i)^k ((n-2) - i)^{n-k} = ((n-2) + i)^n + ((n-2) - i)^n$$

Therefore,

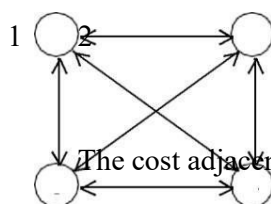
$$\sum_{k=0}^n \binom{n}{k} ((n-2) + i)^k ((n-2) - i)^{n-k} = 2^n ((n-2)^2 + 1)$$

This is $\Phi(n 2^{n-2})$, so there are exponential number of calculate. Calculating one $g(i, S)$ require finding the minimum of at most n quantities. Therefore, the entire algorithm is $\Phi(n^2 2^{n-2})$. This is better than enumerating all $n!$ different tours to find the best one. So, we have traded on exponential growth for a much smaller exponential growth.

The most serious drawback of this dynamic programming solution is the space needed, which is $O(n 2^n)$. This is too large even for modest values of n .

Example 1:

For the following graph find minimum cost tour for the traveling salesperson problem:



The cost adjacency matrix =

3 4

r			
0	1	20	10
~	0	15	~
~	0	9	~
	1	12	
5	3	0	~
~			
6			01
~	8	9]
~			

Let us start the tour from vertex 1:

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

More generally writing:

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

Clearly, $g(i, T) = c_{i1}$, $1 \leq i \leq n$. So,

$$g(2, T) = C_{21} = 5$$

$$g(3, T) = C_{31} = 6$$

$$g(4, \sim) = C_{41} = 8$$

Using equation – (2) we obtain:

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

$$\begin{aligned} g(2, \{3, 4\}) &= \min \{c_{23} + g(3, \{4\}), c_{24} + g(4, \{3\})\} \\ &= \min \{9 + g(3, \{4\}), 10 + g(4, \{3\})\} \end{aligned}$$

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

$$\begin{aligned} g(4, \{3\}) &= \min \{c_{43} + g(3, \sim)\} \\ &= 9 + 6 = 15 \end{aligned}$$

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

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

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

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

$$\begin{aligned} \text{Therefore, } g(3, \{2, 4\}) &= \min \{13 + 18, 12 + 13\} = \min \{41, 25\} = 25 \\ g(4, \{2, 3\}) &= \min \{c_{42} + g(2, \{3\}), c_{43} + g(3, \{2\})\} \end{aligned}$$

$$g(2, \{3\}) = \min \{c_{23} + g(3, \sim)\} = 9 + 6 = 15$$

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

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

$$g(1, \{2, 3, 4\}) = \min \{c_{12} + g(2, \{3, 4\}), c_{13} + g(3, \{2, 4\}), c_{14} + g(4, \{2, 3\})\} = \min \{10 + 25, 15 + 25, 20 + 23\} = \min \{35, 40, 43\} = 35$$

The optimal tour for the graph has length = 35 The

optimal tour is: 1, 2, 4, 3, 1.

OPTIMAL BINARY SEARCH TREE

Let us assume that the given set of identifiers is $\{a_1, \dots, a_n\}$ with $a_1 < a_2 < \dots < a_n$. Let $p(i)$ be the probability with which we search for a_i . Let $q(i)$ be the probability that the identifier x being searched for is such that $a_i < x < a_{i+1}$, $0 \leq i \leq n$ (assume $a_0 = -\infty$ and $a_{n+1} = +\infty$). We have to arrange the identifiers in a binary search tree in a way that minimizes the expected total access time.

In a binary search tree, the number of comparisons needed to access an element at depth 'd' is $d + 1$, so if ' a_i ' is placed at depth ' d_i ', then we want to minimize:

$$\sum_{i=1}^n P(i) (1 + d_i)$$

Let $P(i)$ be the probability with which we shall be searching for ' a_i '. Let $Q(i)$ be the probability of an un-successful search. Every internal node represents a point where a successful search may terminate. Every external node represents a point where an unsuccessful search may terminate.

The expected cost contribution for the internal node for ' a_i ' is:

$$P(i) * \text{level}(a_i)$$

Unsuccessful search terminate with $I = 0$ (i.e at an external node). Hence the cost contribution for this node is:

$$Q(i) * \text{level}((E_i) - 1)$$

110

The expected cost of binary search tree is:

$$\sum_{i=1}^n P(i) * \text{level}(a_i) + \sum_{i=1}^n Q(i) * \text{level}((E_i) - 1)$$

Given a fixed set of identifiers, we wish to create a binary search tree organization. We may expect different binary search trees for the same identifier set to have different performance characteristics.

The computation of each of these $c(i, j)$'s requires us to find the minimum of m quantities. Hence, each such $c(i, j)$ can be computed in time $O(m)$. The total time for all $c(i, j)$'s with $j - i = m$ is therefore $O(nm - m^2)$.

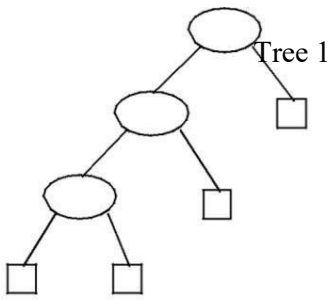
The total time to evaluate all the $c(i, j)$'s and $r(i, j)$'s is therefore:

$$\sim (nm - m^2) = O(n^3) \quad 1 < m < n$$

Example 1: The possible binary search trees for the identifier set $(a_1, a_2, a_3) = (\text{do}, \text{if}, \text{stop})$ are as follows. Given the equal probabilities $p(i) = Q(i) = 1/7$ for all i , we have:

stop

if



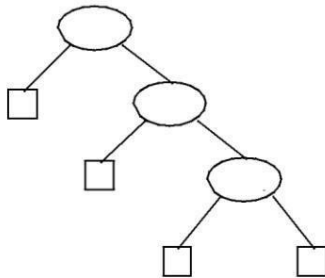
do

Tree 2

do

if

st
o
p



Tree 3

$$\text{Cost (tree \# 1)} = \sim (1 \times 1 + 1 \times 2 + 1 \times 3 + \dots) \sim 7$$

$$1+2+3+1+2+3+3+6+9+15$$

$$\text{Cost (tree \# 3)} = \sim \frac{1}{1} \times 1 + 1 \times 2 + 1 \times 3 + \dots \sim 7$$

$$\text{Cost (tree \# 4)} = \sim \frac{1}{1} \times 1 + 1 \times 2 + 1 \times 3 + \dots \sim 7$$

$$\text{Cost (tree \# 2)} = \sim \frac{1}{1} \times 1 + 1 \times 2 + 1 \times 3 + \dots \sim 7$$

Huffman coding tree solved by a greedy algorithm has a limitation of having the data only at the leaves and it must not preserve the property that all nodes to the left of the root have keys, which are less etc. Construction of an optimal binary search tree is harder, because the data is not constrained to appear only at the leaves, and also because the tree must satisfy the binary search tree property and it must preserve the property that all nodes to the left of the root have keys, which are less.

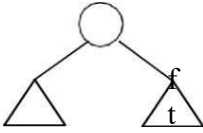
A dynamic programming solution to the problem of obtaining an optimal binary search tree can be viewed by constructing a tree as a result of sequence of decisions by holding the principle of optimality. A possible approach to this is to make a decision as which of the a_i 's be arraigned to the root node at 'T'. If we choose ' a_k ' then it is clear that the internal nodes for a_1, a_2, \dots, a_{k-1} as well as the external nodes for the classes $E_0, E_1,$

\dots, E_{k-1} will lie in the left sub tree, L, of the root. The remaining nodes will be in the right subtree, ft. The structure of an optimal binary search tree is:

a_k

L

K



$$\begin{aligned}
 \text{Cost (L)} &= \sum_{i=1}^K P(i) * \text{level}(a_i) \sim \sum_{i=1}^K Q(i) * (\text{level}(E_i) - 1) \\
 &\sim 1 \quad 0 \\
 &\quad n \quad n
 \end{aligned}$$

$$\begin{aligned}
 \text{Cost (ft)} &= \sum_{i=K+1}^n P(i) * \text{level}(a_i) \sim \sum_{i=K+1}^n Q(i) * (\text{level}(E_i) - 1) \\
 &\sim K \quad K
 \end{aligned}$$

The $C(i, J)$ can be computed as:

$$\begin{aligned}
 C(i, J) &= \min \{C(i, k-1) + C(k, J) + P(K) + w(i, K-1) + w(K, J)\} \quad i < k < J \\
 &= \min_{i < k < J} \{C(i, K-1) + C(K, J) + w(i, J)\} \quad \text{--} \quad (1)
 \end{aligned}$$

$$\text{Where } W(i, J) = P(J) + Q(J) + w(i, J-1) \quad \text{--} \quad (2)$$

Initially $C(i, i) = 0$ and $w(i, i) = Q(i)$ for $0 \leq i \leq n$.

Equation (1) may be solved for $C(0, n)$ by first computing all $C(i, J)$ such that $J - i = 1$. Next, we can compute all $C(i, J)$ such that $J - i = 2$, Then all $C(i, J)$ with $J - i = 3$ and so on.

$C(i, J)$ is the cost of the optimal binary search tree ' T_{ij} '. During computation we record the root $R(i, J)$ of each tree ' T_{ij} '. Then an optimal binary search tree may be constructed from these $R(i, J)$. $R(i, J)$ is the value of ' K ' that minimizes equation (1).

We solve the problem by knowing $W(i, i+1)$, $C(i, i+1)$ and $R(i, i+1)$, $0 \leq i < 4$;

Knowing $W(i, i+2)$, $C(i, i+2)$ and $R(i, i+2)$, $0 \leq i < 3$ and repeating until $W(0, n)$, $C(0, n)$ and $R(0, n)$ are obtained.

The results are tabulated to recover the actual tree.

Example 1:

Let $n = 4$, and $(a_1, a_2, a_3, a_4) = (\text{do}, \text{if}, \text{need}, \text{while})$ Let $P(1:4) = (3, 3, 1, 1)$ and $Q(0:4) = (2, 3, 1, 1, 1)$

Solution:

Table for recording $W(i, j)$, $C(i, j)$ and $R(i, j)$:

Column Row	0	1	2	3	4
0	2,0,0	3,0, 0 1,0, 0		1, 0, 0,	1,0,0
1	8,8,1	7,7, 2 3,3, 3		3, 3, 4	
2	12, 19, 1	9, 12, 2 5, 8, 3			
3	14, 25, 2	11, 19, 2			
4	16, 32, 2				

This computation is carried out row-wise from row 0 to row 4. Initially, $W(i, i) = Q(i)$ and $C(i, i) = 0$ and $R(i, i) = 0$, $0 \leq i < 4$. Solving for $C(0, n)$:

First, computing all $C(i, j)$ such that $j - i = 1$; $j = i + 1$ and as $0 \leq i < 4$; $i = 0, 1, 2$ and 3 ; $i < k \leq J$. Start with $i = 0$; so $j = 1$; as $i < k \leq j$, so the possible value for $k = 1$

$$W(0,1)=P(1)+Q(1)+W(0,0)=3+3+2=8$$

$$C(0, 1) = W(0, 1) + \min \{C(0, 0) + C(1, 1)\} = 8$$

$$R(0, 1) = 1 \text{ (value of 'K' that is minimum in the above equation).}$$

Next with $i = 1$; so $j = 2$; as $i < k \leq j$, so the possible value for $k = 2$

$$W(1, 2) = P(2) + Q(2) + W(1, 1) = 3 + 1 + 7 = 11$$

$$C(1, 2) = W(1, 2) + \min \{C(1, 1) + C(2, 2)\} = 11 + \min \{0 + 0\} = 11$$

$$\begin{aligned} (1, 2) &= (1, 2) + C(2, 2) = 7 \\ R(1, 2) &= 2 \end{aligned}$$

Next with $i = 2$; so $j = 3$; as $i < k \leq j$, so the possible value for $k = 3$

$$\begin{aligned} W(2, 3) &= P(3) + Q(3) + W(2, 2) = 1 + 1 + 1 = 3 \\ C(2, 3) &= W(2, 3) + \min \{C(2, 2) + C(3, 3)\} = 3 + 3 = 6 \\ \text{ft}(2, 3) &= 3 \end{aligned}$$

Next with $i = 3$; so $j = 4$; as $i < k \leq j$, so the possible value for $k = 4$

$$\begin{aligned} W(3, 4) &= P(4) + Q(4) + W(3, 3) = 1 + 1 + 3 = 5 \\ C(3, 4) &= W(3, 4) + \min \{C(3, 3) + C(4, 4)\} = 5 + 6 = 11 \\ \text{ft}(3, 4) &= 5 \end{aligned}$$

Second, Computing all $C(i, j)$ such that $j - i = 2$; $j = i + 2$ and as $0 \leq i < 3$; $i = 0, 1, 2$; $i < k \leq J$. Start with $i = 0$; so $j = 2$; as $i < k \leq J$, so the possible values for $k = 1$ and 2 .

$$\begin{aligned} W(0, 2) &= P(2) + Q(2) + W(0, 1) = 3 + 1 + 8 = 12 \\ C(0, 2) &= W(0, 2) + \min \{(C(0, 0) + C(1, 2)), (C(0, 1) + C(2, 2))\} = 12 \\ &\quad + \min \{(0 + 7, 8 + 0)\} = 19 \text{ ft}(0, 2) = 1 \end{aligned}$$

Next, with $i = 1$; so $j = 3$; as $i < k \leq j$, so the possible value for $k = 2$ and 3 .

$$\begin{aligned} W(1, 3) &= P(3) + Q(3) + W(1, 2) = 1 + 1 + 7 = 9 \\ C(1, 3) &= W(1, 3) + \min \{[C(1, 1) + C(2, 3)], [C(1, 2) + C(3, 3)]\} \\ &= 9 + \min \{(0 + 3), (7 + 0)\} = 9 + 3 = 12 \\ \text{ft}(1, 3) &= 2 \end{aligned}$$

Next, with $i = 2$; so $j = 4$; as $i < k \leq j$, so the possible value for $k = 3$ and 4 .

$$\begin{aligned} W(2, 4) &= P(4) + Q(4) + W(2, 3) = 1 + 1 + 3 = 5 \\ C(2, 4) &= W(2, 4) + \min \{[C(2, 2) + C(3, 4)], [C(2, 3) + C(4, 4)]\} \\ &= 5 + \min \{(0 + 3), (3 + 0)\} = 5 + 3 = 8 \text{ ft}(2, 4) = 3 \end{aligned}$$

Third, Computing all $C(i, j)$ such that $J - i = 3$; $j = i + 3$ and as $0 \leq i < 2$; $i = 0, 1$; $i < k \leq J$. Start with $i = 0$; so $j = 3$; as $i < k \leq j$, so the possible values for $k = 1, 2$ and 3 .

$$\begin{aligned} W(0, 3) &= P(3) + Q(3) + W(0, 2) = 1 + 1 + 12 = 14 \\ C(0, 3) &= W(0, 3) + \min \{[C(0, 0) + C(1, 3)], [C(0, 1) + C(2, 3)]\} \\ &= 14 + \min \{(0 + 12), (1 + 8)\} = 14 + 9 = 23 \text{ ft}(0, 3) = 1 \end{aligned}$$

$$[C(0,2)+C(3,3)] + 0 = 14 + 11 = 25$$

$$ft(0,3) = 14 + \min \{(0+12), (8+3), (19+0)\} = 14 + 3 = 17$$

Start with $i = 1$; so $j = 4$; as $i < k \leq j$, so the possible values for $k = 2, 3$ and 4 .

$$W(1,4) = P(4) + Q(4) + W(1,3) = 1 + 1 + 9 = 11 = W(1,4)$$

$$C(1,4) = (1,4) + \min \{[C(1,1) + C(2,4)], [C(1,2) + C(3,4)], [C(1,3) + C(4,4)]\}$$

$$= 11 + \min \{(0+8), (7+3), (12+0)\} = 11 + 0 = 11$$

Fourth, Computing all $C(i, j)$ such that $j - i = 4$; $j = i + 4$ and as $0 \leq i < 1$; $i = 0$; $i < k \leq J$.

Start with $i = 0$; so $j = 4$; as $i < k \leq j$, so the possible values for $k = 1, 2, 3$ and 4 .

$$W(0,4) = P(4) + Q(4) + W(0,3) = 1 + 1 + 14 = 16$$

$$C(0,4) = W(0,4) + \min \{[C(0,1) + C(3,4)], [C(0,2) + C(2,4)], [C(0,3) + C(1,4)]\}$$

$$= 16 + \min [0 + 19, 8 + 8, 19 + 3, 25 + 0] = 16 + 16 = 32$$

From the table we see that $C(0, 4) = 32$ is the minimum cost of a binary search tree for (a_1, a_2, a_3, a_4) . The root of the tree 'T04' is 'a2'.

Hence the left sub tree is 'T01' and right sub tree is T24. The root of 'T01' is 'a1' and the root of 'T24' is a3.

The left and right sub trees for 'T01' are 'T00' and 'T11' respectively. The root of T01 is 'a1'

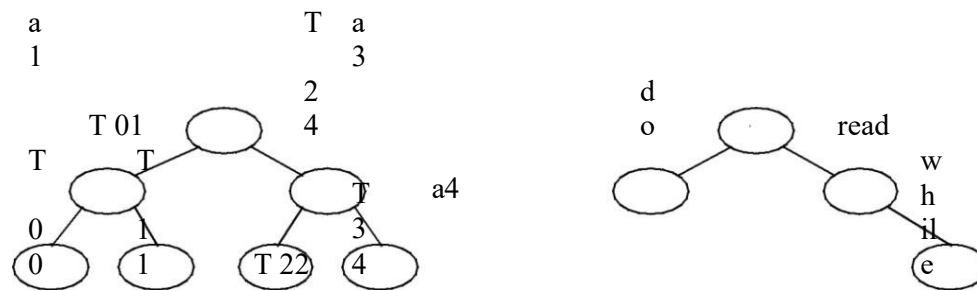
The left and right sub trees for T24 are T22 and T34 respectively.

The root of T24 is 'a3'.

The root of T22 is null

The root of T34 is

a2
T 04 if



Example 2:

Consider four elements a_1, a_2, a_3 and a_4 with $Q_0 = 1/8, Q_1 = 3/16, Q_2 = Q_3 = Q_4 = 1/16$ and $p_1 = 1/4, p_2 = 1/8, p_3 = p_4 = 1/16$. Construct an optimal binary search tree. Solving for $C(0, n)$:

First, computing all $C(i, j)$ such that $j - i = 1; j = i + 1$ and as $0 \leq i < 4; i = 0, 1, 2$ and 3 ;

$i < k \leq J$. Start with $i = 0$; so $j = 1$; as $i < k \leq j$, so the possible value for $k = 1$

$$W(0,1)=P(1)+Q(1)+W(0,0)=4+3+2=9$$

$C(0, 1) = W(0, 1) + \min \{C(0, 0) + C(1, 1)\} = 9 + [(0 + 0)] = 9$ ft $(0, 1) = 1$ (value of 'K' that is minimum in the above equation).

Next with $i = 1$; so $j = 2$; as $i < k \leq j$, so the possible value for $k = 2$

$$W(1,2)=P(2)+Q(2)+W(1,1)=2+1+3=6$$

$$C(1, 2) = W(1, 2) + \min \{C(1, 1) + C(2, 2)\} = 6 + [(0 + 0)] = 6$$
 ft $(1, 2)=2$

Next with $i = 2$; so $j = 3$; as $i < k \leq j$, so the possible value for $k = 3$

$$W(2, 3) = P(3) + Q(3) + W(2, 2) = 1 + 1 + 3 = 3$$

$$C(2, 3) = W(2, 3) + \min \{C(2, 2) + C(3, 3)\} = 3 + [(0 + 0)] = 3$$

$$ft(2, 3) = 3$$

Next with $i = 3$; so $j = 4$; as $i < k \leq j$, so the possible value for $k = 4$

$$W(3,4) = P(4) + Q(4) + W(3,3) = 1 + 1 + 3 = 3$$

$$C(3, 4) = W(3, 4) + \min \{C(3, 3) + C(4, 4)\} = 3 + [(0 + 0)] = 3$$

$$ft(3, 4) = 4$$

Second, Computing all $C(i, j)$ such that $j - i = 2; j = i + 2$ and as $0 \leq i < 3; i = 0, 1, 2; i < k \leq J$

Start with $i = 0$; so $j = 2$; as $i < k \leq j$, so the possible values for $k = 1$ and 2 .

$$W(0,2)=P(2)+Q(2)+W(0,1)=2+1+9=12$$

$$C(0,2) = W(0,2) + \min \{(C(0,0) + C(1,2)), (C(0,1) + C(2,2))\} = 12 + \min \{(0 + 6, 9 + 0)\} = 12 + 6 = 18$$

$$ft(0,2) = 1$$

Next, with $i = 1$; so $j = 3$; as $i < k \leq j$, so the possible value for $k = 2$ and 3 .

$$W(1,3) = P(3) + Q(3) + W(1,2) = 1 + 1 + 6 = 8$$

$$C(1,3) = W(1,3) + \min \{[C(1,1) + C(2,3)], [C(1,2) + C(3,3)]\}$$

$$= 8 + \min \{(0 + 3), (6 + 0)\} = 8 + 3 = 11$$

$$ft(1,3) = 2$$

Next, with $i = 2$; so $j = 4$; as $i < k \leq j$, so the possible value for $k = 3$ and 4 .

$$W(2,4) = P(4) + Q(4) + W(2,3) = 1 + 1 + 3 = 5$$

$$C(2,4) = W(2,4) + \min \{[C(2,2) + C(3,4)], [C(2,3) + C(4,4)]\} = 5 + \min \{(0 + 3), (3 + 0)\} = 5 + 3 = 8$$

Third, Computing all $C(i, j)$ such that $J - i = 3$; $j = i + 3$ and as $0 \leq i < 2$; $i = 0, 1$; $i < k \leq J$. Start with $i = 0$; so $j = 3$; as $i < k \leq j$, so the possible values for $k = 1, 2$ and 3 .

$$W(0,3) = P(3) + Q(3) + W(0,2) = 1 + 1 + 12 = 14$$

$$C(0,3) = W(0,3) + \min \{[C(0,0) + C(1,3)], [C(0,1) + C(2,3)], [C(0,2) + C(3,3)]\} = 14 + \min \{(0 + 11), (9 + 3), (18 + 0)\} = 14 + 11 = 25$$

$$ft(0,3) = 1$$

Start with $i = 1$; so $j = 4$; as $i < k \leq j$, so the possible values for $k = 2, 3$ and 4 .

$$W(1,4) = P(4) + Q(4) + W(1,3) = 1 + 1 + 8 = 10$$

$$C(1,4) = W(1,4) + \min \{[C(1,1) + C(2,4)], [C(1,2) + C(3,4)], [C(1,3) + C(4,4)]\} = 10 + \min \{(0 + 8), (6 + 3), (11 + 0)\} = 10 + 6 = 16$$

$$ft(1,4) = 2$$

Fourth, Computing all $C(i, j)$ such that $J - i = 4$; $j = i + 4$ and as $0 \leq i < 1$; $i = 0$; $i < k \leq J$. Start with $i = 0$; so $j = 4$; as $i < k \leq j$, so the possible values for $k = 1, 2, 3$ and 4 .

$$W(0,4) = P(4) + Q(4) + W(0,3) = 1 + 1 + 14 = 16$$

$$C(0,4) = W(0,4) + \min \{[C(0,0) + C(1,4)], [C(0,1) + C(2,4)], [C(0,2) + C(3,4)], [C(0,3) + C(4,4)]\}$$

$$= 16 + \min [0 + 18, 9 + 8, 18 + 3, 25 + 0] = 16 + 17 = 33$$

$$R(0,4) = 2$$

Table for recording $W(i, j)$, $C(i, j)$ and $R(i, j)$

Column					
Row	0	1	2	3	4
0	2,0,0	1,0,0	1,0,0	1,0,0	1,0,0
1	9,9,1	6,6,2	3,3,3	3,3,4	
2	12,18,1	8,11,2	5,8,3		
3	14,25,2	11,18,2			
4	16,33,2				

From the table we see that $C(0, 4) = 33$ is the minimum cost of a binary search tree for (a_1, a_2, a_3, a_4)

The root of the tree 'T04' is 'a2'.

Hence the left sub tree is 'T01' and right sub tree is T24. The root of 'T01' is 'a1' and the root of 'T24' is a3.

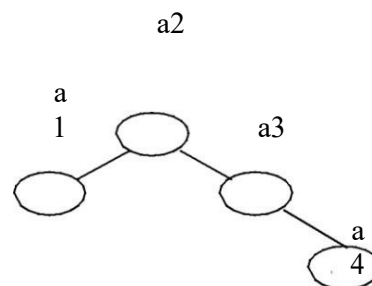
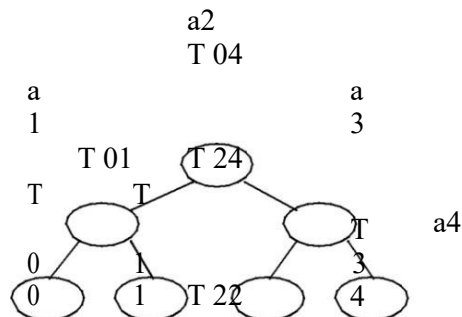
The left and right sub trees for 'T01' are 'T00' and 'T11' respectively. The root of T01 is 'a1'

The left and right sub trees for T24 are T22 and T34 respectively.

The root of T24 is 'a3'.

The root of T22 is null.

The root of T34 is a4.



0/1 – KNAPSACK:

We are given n objects and a knapsack. Each object i has a positive weight w_i and a positive value V_i . The knapsack can carry a weight not exceeding W . Fill the knapsack so that the value of objects in the knapsack is optimized.

A solution to the knapsack problem can be obtained by making a sequence of decisions on the variables x_1, x_2, \dots, x_n . A decision on variable x_i involves determining which of the values 0 or 1 is to be assigned to it. Let us assume that decisions on the x_i are made in the order x_n, x_{n-1}, \dots, x_1 . Following a decision on x_n , we may be in one of two possible states: the capacity remaining in $m - w_n$ and a profit of p_n has accrued. It is clear that the remaining decisions x_{n-1}, \dots, x_1 must be optimal with respect to the problem state resulting from the decision on x_n . Otherwise, x_n, \dots, x_1 will not be optimal. Hence, the principle of optimality holds.

$$F_n(m) = \max \{f_{n-1}(m), f_{n-1}(m - w_n) + p_n\} \quad \text{--} \quad 1$$

For arbitrary $f_i(y)$, $i > 0$, this equation generalizes to:

$$F_i(y) = \max \{f_{i-1}(y), f_{i-1}(y - w_i) + p_i\} \quad \text{--} \quad 2$$

Equation-2 can be solved for $f_n(m)$ by beginning with the knowledge $f_0(y) = 0$ for all y and $f_i(y) = -\infty$, $y < 0$. Then f_1, f_2, \dots, f_n can be successively computed using equation-2.

When the w_i 's are integer, we need to compute $f_i(y)$ for integer y , $0 \leq y \leq m$. Since $f_i(y) = -\infty$ for $y < 0$, these function values need not be computed explicitly. Since each f_i can be computed from f_{i-1} in $\Theta(m)$ time, it takes $\Theta(mn)$ time to compute f_n . When the w_i 's are real numbers, $f_i(y)$ is needed for real numbers y such that $0 < y \leq m$. So, f_i cannot be explicitly computed for all y in this range. Even when the w_i 's are integer, the explicit $\Theta(mn)$ computation of f_n may not be the most efficient computation. So, we explore **an alternative method for both cases**.

The $f_i(y)$ is an ascending step function; i.e., there are a finite number of y 's, $0 = y_1 < y_2 < \dots < y_k$, such that $f_i(y_1) < f_i(y_2) < \dots < f_i(y_k)$; $f_i(y) = -\infty$, $y < y_1$; $f_i(y) = f_i(y_k)$, $y \geq y_k$; and $f_i(y) = f_i(y_j)$, $y_j \leq y \leq y_{j+1}$. So, we need to compute only $f_i(y_j)$, $1 \leq j < k$. We use the ordered set $S^i = \{(f(y_j), y_j) \mid 1 \leq j < k\}$ to represent $f_i(y)$. Each number of S^i is a pair (P, W) , where $P = f_i(y_j)$ and $W = y_j$. Notice that $S^0 = \{(0, 0)\}$. We can compute S^{i+1} from S^i by first computing:

$$S^{i+1} = \{(P, W) \mid (P - p_i, W - w_i) \in S^i\}$$

Now, S^{i+1} can be computed by merging the pairs in S^i and S^{i+1} together. Note that if S^{i+1} contains two pairs (P_j, W_j) and (P_k, W_k) with the property that $P_j \leq P_k$ and $W_j > W_k$, then the pair (P_j, W_j) can be discarded because of equation-2. Discarding or purging rules such as this one are also known as dominance rules. Dominated tuples get purged. In the above, (P_k, W_k) dominates (P_j, W_j) .

RELIABILITY DESIGN

The problem is to design a system that is composed of several devices connected in series. Let r_i be the reliability of device D_i (that is r_i is the probability 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 r_i 's are very close to one), the reliability of the system may

not be very good. For example, if $n = 10$ and $r_i = 0.99$, $1 \leq i \leq 10$, then $\prod_{i=1}^{10} r_i = .904$. Hence, it is desirable to duplicate devices. Multiply copies of the same device type are connected in parallel.

If stage i contains m_i copies of device D_i . Then the probability that all m_i have a malfunction is $(1 - r_i)^{m_i}$. Hence the reliability of stage i becomes $1 - (1 - r_i)^{m_i}$.

The reliability of stage 'i' is given by a function $f_i(m_i)$.

Our problem is to use device duplication. This maximization is to be carried out under a cost constraint. 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.

We wish to solve:

$$\text{Maximize } \prod_{i=1}^n f_i(m_i)$$

$$1 \leq i \leq n$$

$$\text{Subject to } \sum_{i=1}^n c_i m_i \leq C$$

$$1 \leq i \leq n$$

$$m_i \geq 1 \text{ and integer, } 1 \leq i \leq n$$

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

$$u_i = \left\lfloor \frac{C}{c_i} \right\rfloor$$

The upper bound u_i follows from the observation that $m_j \geq 1$

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

Let $f_i(x)$ represent the maximum value of

Subject to the constraints:

$$\prod_{j=1}^i f_j(m_j)$$

$$f_j^i(x) = C_j - m_j x \quad \text{and } 1 \leq m_j \leq u_j, 1 \leq j \leq i$$

The last decision made requires one to choose m_n from $\{1, 2, 3, \dots, u_n\}$

Once a value of m_n has been chosen, the remaining decisions must be such as to use the remaining funds $C - C_n m_n$ in an optimal way.

The principle of optimality holds on

$$f_n^i(x) = \max_{m_n} \{ f_{n-1}^i(x - C_n m_n) \mid 1 \leq m_n \leq u_n \}$$

for any $f_i(x_i)$, $i > 1$, this equation generalizes to

$$f_n^i(x) = \max_{m_i} \{ f_{n-1}^i(x - C_i m_i) \mid 1 \leq m_i \leq u_i \}$$

clearly, $f_0(x) = 1$ for all x , $0 \leq x \leq C$ and $f(x) = -\infty$ for all $x < 0$. 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 result 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) if $f_1 \geq f_2$ and $x_1 \leq x_2$. Hence, dominated tuples can be discarded from S^i .

Example 1:

Design a three stage system with device types D1, D2 and D3. The costs are \$30, \$15 and \$20 respectively. The Cost of the system is to be no more than \$105. The reliability of each device is 0.9, 0.8 and 0.5 respectively.

Solution:

We assume that if stage I has m_i devices of type i in parallel, then $0 \leq m_i \leq u_i$ where

Since, we can assume each $c_i > 0$, each m_i must be in the range $1 \leq m_i \leq u_i$. Where:

$$u_i = \frac{105 - (30 + 15 + 20)}{c_i} = \frac{105 - 65}{c_i} = \frac{40}{c_i}$$

Using the above equation compute u_1, u_2 and u_3 .

$$u_1 = \frac{105 - (30 + 15 + 20)}{30} = \frac{40}{30} = 1.33 \approx 1$$

$$u_2 = \frac{105 - (30 + 15 + 20)}{15} = \frac{40}{15} = 2.66 \approx 2$$

$$u_3 = \frac{105 - (30 + 15 + 20)}{20} = \frac{40}{20} = 2$$

We use S^i - * i : stage number and J : no. of devices in stage $i = m_i$ S^0

$= \{f_0(x), x\}$ initially $f_0(x) = 1$ and $x = 0$, so, $S^0 = \{1, 0\}$

Compute S^1, S^2 and S^3 as follows:

S^1 depends on u_1 value, as $u_1 = 1$, so

$$S1 = \{S1, S^1\}$$

$S2 =$ depends on $u2$ value, as $u2 = 3$,
so

$$S2 = \{S^2, S^2, S^2\}$$

$S3 =$ depends on $u3$ value, as $u3 = 3$,
so

$$S3 = \{S^3, S^3, S^3\}$$

$$S = \{S^1, S^2, S^3\}$$

Now find $f^1(x)$

$f^1(x) = \{01(1)f_0 \sim, 01(2)f_0\}$ With devices $m1 = 1$ and $m2 = 2$ Compute $01(1)$ and $01(2)$ using the formula: $0i(mi) = 1 - (1 - ri)mi$

$$f^1_1 = 1 - (1 - 0.9)^1 = 0.9$$

$$f^1_2 = 1 - (1 - 0.9)^2 = 0.99$$

$$S \sim f^1_1 \sim x, x \sim$$

$$S \sim 0.9, 30$$

$$S \sim 10.99, 30 + 30 = (0.99, 60)$$

$$\text{Therefore, } S^1 = \{(0.9, 30), (0.99, 60)\}$$

Next find $2 \sim f$

$$S_1 = \{02(1) * f^1_1, 02(2) * f^1_2, 02(3) * f^1_3\}$$

$$f^2_1 = 1 - (1 - 0.8) = 1 - 0.2 = 0.8$$

$$f^2_2 = 1 - (1 - 0.8)^2 = 0.96$$

$$f^2_3 = 1 - (1 - 0.8)^3 = 0.992$$

$$= \{(0.8(0.9), 30 + 15), (0.8(0.99), 60 + 15)\} = \{(0.72, 45), (0.792, 75)\} = \{(0.96(0.9), 30 + 15 + 15), (0.96(0.99), 60 + 15 + 15)\}$$

$$= \{(0.864, 60), (0.9504, 90)\}$$

$$= \{(0.992(0.9), 30 + 15 + 15 + 15), (0.992(0.99), 60 + 15 + 15 + 15)\}$$

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

$$S2 = \{S^2, S^2, S^2\}$$

$$S = \{S^1, S^2, S^3\}$$

By applying Dominance rule to S^2 :

$$\text{Therefore, } S2 = \{(0.72, 45), (0.864, 60), (0.8928, 75)\} \text{ Dominance Rule:}$$

Page 22 of 25

$$= \sim iC + Ci - \sim CJ r / Ci I \sim$$

Using the above equation compute u1, u2 and u3.

$$\begin{array}{rcl}
 u_1 & \frac{105}{3} & = 35 \\
 u_2 & \frac{105}{15} & = 7 \\
 u_3 & \frac{105}{20} & = 5.25
 \end{array}$$

$$\begin{aligned}
 f_3(105) &= \max \{ \sim 3(m_3), f_2(105 - 20m_3) \} \quad 1 < m_3 \leq u_3 \\
 &= \max \{ 3(1) f_2(105 - 20), 63(2) f_2(105 - 20 \times 2), \sim 3(3) f_2(105 - 20 \times 3) \} = \max \\
 &\quad \{ 0.5, 0.75 f_2(65), 0.875 f_2(45) \} \\
 &= \max \{ 0.5 \times 0.8928, 0.75 \times 0.864, 0.875 \times 0.72 \} = 0.648. \\
 &= \max \{ 2(m_2), f_1(85 - 15m_2) \} \quad 1 \leq m_2 \leq u_2 \\
 &= \max \{ 2(1), f_1(85 - 15), \sim 2(2), f_1(85 - 15 \times 2), \sim 2(3), f_1(85 - 15 \times 3) \} = \\
 &\quad \max \{ 0.8 f_1(70), 0.96 f_1(55), 0.992 f_1(40) \} \\
 &= \max \{ 0.8 \times 0.99, 0.96 \times 0.9, 0.99 \times 0.9 \} = 0.8928 \\
 f_1(70) &= \max \{ \sim 1(m_1), f_0(70 - 30m_1) \} \quad 1 \leq m_1 \leq u_1 \\
 &= \max \{ \sim 1(1) f_0(70 - 30), t_1(2) f_0(70 - 30 \times 2) \} \\
 &= \max \{ \sim 1(1) \times 1, t_1(2) \times 1 \} = \max \{ 0.9, 0.99 \} = 0.99 \\
 f_1(55) &= \max \{ t_1(m_1), f_0(55 - 30m_1) \} \quad 1 \leq m_1 \leq u_1 \\
 &= \max \{ \sim 1(1) f_0(50 - 30), t_1(2) f_0(50 - 30 \times 2) \} \\
 &= \max \{ \sim 1(1) \times 1, t_1(2) \times -\infty \} = \max \{ 0.9, -\infty \} = 0.9
 \end{aligned}$$

$$\begin{aligned}
f_1(40) &= \max_{1 \leq m_1 \leq u_1} \{ \sim 1(m_1). f_0(40 - 30m_1) \} \\
&= \max \{ \sim 1(1) f_0(40 - 30), t_1(2) f_0(40 - 30 \times 2) \} \\
&= \max \{ \sim 1(1) \times 1, t_1(2) \times -\infty \} = \max \{ 0.9, -\infty \} = 0.9
\end{aligned}$$

$$\begin{aligned}
f_2(65) &= \max_{1 \leq m_2 \leq u_2} \{ 2(m_2). f_1(65 - 15m_2) \} \\
&= \max \{ 2(1) f_1(65 - 15), \underline{62(2) f_1(65 - 15 \times 2)}, \sim 2(3) f_1(65 - 15 \times 3) \} = \max \{ 0.8 f_1(50), \\
&\quad 0.96 f_1(35), 0.992 f_1(20) \} \\
&= \max \{ 0.8 \times 0.9, 0.96 \times 0.9, -\infty \} = 0.864
\end{aligned}$$

$$\begin{aligned}
f_1(50) &= \max_{1 \leq m_1 \leq u_1} \{ \sim 1(m_1). f_0(50 - 30m_1) \} \\
&= \max \{ \sim 1(1) f_0(50 - 30), t_1(2) f_0(50 - 30 \times 2) \} \\
&= \max \{ \sim 1(1) \times 1, t_1(2) \times -\infty \} = \max \{ 0.9, -\infty \} = 0.9 \quad f_1(35) = \max_{\sim 1(m_1). f_0(35 - 30m_1)}
\end{aligned}$$

$$\begin{aligned}
&1 \leq m_1 \leq u_1 \\
&= \max \{ \sim 1(1). \underline{f_0(35 - 30)}, \sim 1(2). f_0(35 - 30 \times 2) \} \\
&= \max \{ \sim 1(1) \times 1, t_1(2) \times -\infty \} = \max \{ 0.9, -\infty \} = 0.9
\end{aligned}$$

$$\begin{aligned}
f_1(20) &= \max_{1 \leq m_1 \leq u_1} \{ \sim 1(m_1). f_0(20 - 30m_1) \} \\
&= \max \{ \sim 1(1) f_0(20 - 30), t_1(2) f_0(20 - 30 \times 2) \} \\
&= \max \{ \sim 1(1) \times -, \sim 1(2) \times -\infty \} = \max \{ -\infty, -\infty \} = -\infty
\end{aligned}$$

$$\begin{aligned}
f_2(45) &= \max_{1 \leq m_2 \leq u_2} \{ 2(m_2). f_1(45 - 15m_2) \} \\
&= \max \{ 2(1) f_1(45 - 15), \sim 2(2) f_1(45 - 15 \times 2), \sim 2(3) f_1(45 - 15 \times 3) \} = \max \{ 0.8 f_1(30), \\
&\quad 0.96 f_1(15), 0.992 f_1(0) \} \\
&= \max \{ 0.8 \times 0.9, 0.96 \times -, 0.99 \times -\infty \} = 0.72
\end{aligned}$$

$$\begin{aligned}
f_1(30) &= \max_{1 \leq m_1 \leq u_1} \{ \sim 1(m_1). f_0(30 - 30m_1) \} \quad 1 < m_1 \sim u_1 \\
&= \max \{ \sim 1(1) f_0(30 - 30), t_1(2) f_0(30 - 30 \times 2) \} \\
&= \max \{ \sim 1(1) \times 1, t_1(2) \times -\infty \} = \max \{ 0.9, -\infty \} = 0.9 \quad \text{Similarly, } f_1(15) = -,
\end{aligned}$$

$$f_1(0) = -.$$

The best design has a reliability = 0.648 and

Cost = $30 \times 1 + 15 \times 2 + 20 \times 2 = 100$.

Tracing back for the solution through S^i 's we can determine that: $m_3 = 2$, $m_2 = 2$ and $m_1 = 1$.