UNIT 3
DYNAMIC PROGRAMMING

* Dynamic programming:

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

Ex: Knapsack problem

→ The solution to the knapsack problem can be viewed as the result of sequence of decisions.

→ We have to decide the values of x_i, $1 \le i \le n$. First we make a decision on x_1, then on x_2, then on x_3, ... so on. A optimal sequence of decisions maximizes the objective function $\sum P_i x_i$.

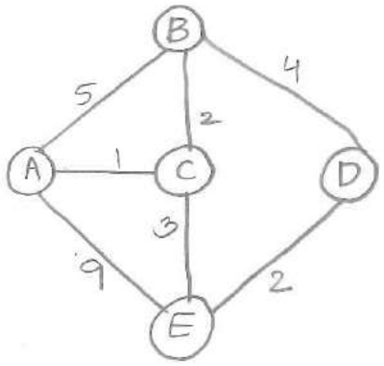
→ For some of the problems that may be viewed in this way. an optimal sequence of decisions can be formed by making the decisions one at a time and never making wrong decision. This is true for all problems that can be solved by greedy method

→ For many other problems, it is not possible to make step-wise decisions in such a manner that the sequence of decisions made is optimal.

*** Principle of optimality:**

→ It states that ~~at~~ an optimal sequence of decision has the property that whatever the initial state and decision are, the remaining decision must contain an optimal decision sequence with regard to the state resulting from the first decision.

→ For example,



Find the shortest path from vertex A to vertex D

A − C − E − D = 1 + 3 + 2 = 6.

*** Comparison between dynamic programming and greedy method:**

Dynamic programming	Greedy method.
1. In this method, the solution to a problem can be viewed as result of sequence of	1. Greedy method is the most forward technique for constructing solution to

decisions	an optimal problem through a sequence of steps.
2. In this method more than one decision is made at a time.	2. Only one decision is made at a time.
3. Dynamic programming considers all possible sequences in order to obtain the optimal solution	3. Greedy method considers an optimal solution without revising the previous solution.
4. It is more expensive than greedy method.	4. Greedy method is comparitively less expensive.

* Comparison between dynamic programming and divide and conquer:

Dynamic programming	Divide and Conquer
→ It is a method in which a solution to a problem can be viewed as the result of sequence of decisions.	→ It is a method in which solution to a problem can be obtained by dividing it into several subproblems
→ Dynamic programming algorithm solves every sub problem just once and saves the result in a table. From this, solution to the problem is obtained.	→ In divide and conquer, every subproblem is solved and the results are combined to get the original solution.

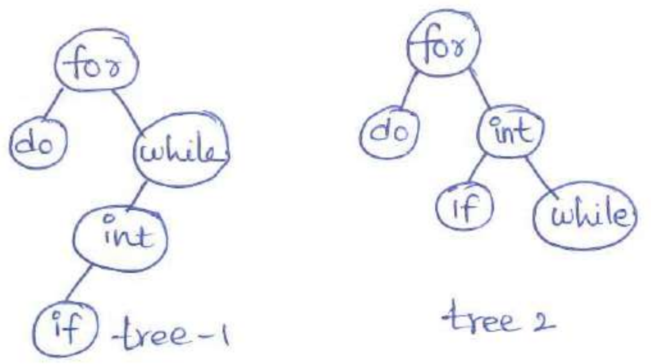
→ Dynamic programming best when all sub problems are dependent.	best when all subproblems are independent.

* Applications of dynamic programming:

1. Matrix chain multiplication
2. Optimal binary search trees.
3. 0/1 knapsack problem
4. All pairs shortest path problems
5. Travelling sales person problem
6. Reliability design problem.

*Optimal binary search tree:

→ Optimal binary search tree is a binary search tree for which the nodes are arranged such that the tree cost is minimum. Consider the 2 following possible BST.



→ The above tree consist of set of identifiers for, do, while, int, if.

→ we can create different BST for same identifier set.

→ In the 1st tree, we require 4 comparisons to find the identifier in the worst case.

→ In the 2nd tree, we requires '3' comparisons to find an identifier in the worst case.

→ Average no.of comparisons required for 1st tree is

$$\frac{1+2+2+3+4}{5} = \frac{12}{5} = 2.4$$

→ Average no.of comparisons required for 2nd tree is

$$\frac{1+2+2+3+3}{5} = \frac{11}{5} = 2.2.$$

→ Second BST is consideed as OBST where its cost is 2.2.

→ In a general situation, we can expect different identifiers to be seacched for different probabilities. In addition, we can expect unsuccessful searches also to be made.

→ To obtain a cost function for binary search trees it is useful to add external nodes.

→ External nodes are denoted with square symbol.

→ Internal nodes are denoted with circular symbol.

→ If a binary search tree represents 'n' identifiers then there will be exactly 'n' internal nodes and 'n+1' external nodes.

→ 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 following formula is used to find the cost of a binary search tree.

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

where $P(i)$ is probability of internal nodes

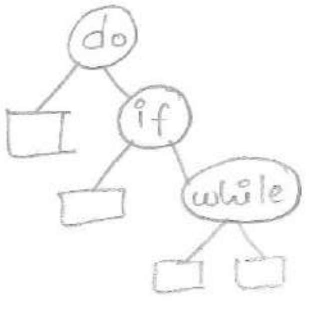
a_i $(a_1, a_2, \dots a_n)$ represents internal nodes.

$q(i)$ is probability of external nodes

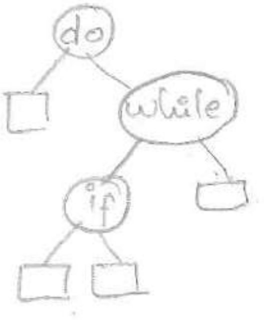
E_i $(E_1, E_2, \dots E_n)$ represents external nodes

1Q) Consider the identifier set $(a_1, a_2, a_3) = (\text{do}, \text{if}, \text{while})$ and probabilities of internal nodes are $P(1) = 0.5$, $P(2) = 0.1$, $P(3) = 0.05$ and probabilities of external nodes are $q(0) = 0.15$, $q(1) = 0.1$, $q(2) = 0.05$, $q(3) = 0.05$. Construct possible binary search trees and find OBST.

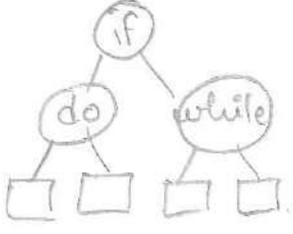
Sol:- Construct possible BST's for the given identifiers (do, if, while).



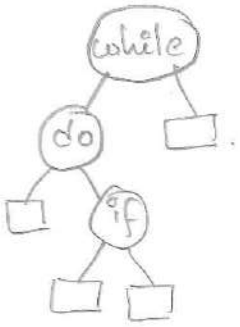
tree 1



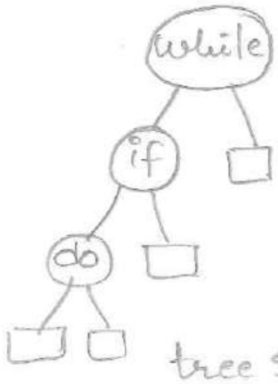
tree 2



tree 3



tree 4



tree 5

$\text{cost(tree 1)} = [P(1) \cdot \text{level}(a_1) + P(2) \cdot \text{level}(a_2) + P(3) \cdot \text{level}(a_3)]$

$+ [q(0)\ \text{level}(E_0 - 1) + q(1)\ \text{level}(E_1 - 1) + q(2)\ \text{level}(E_2 - 1)$

$+ q(3)\ \text{level}(E_3 - 1)]$

$= [0.5 \times 1 + 0.1 \times 2 + 0.05 \times 3] + [0.15 \times 1 + 0.1 \times 2 + 0.05 \times 3 + 0.05 \times 3]$

$= 0.85 + 0.65$

$= 1.5.$

$\text{cost (tree2)} = (0.5 \times 1 + 0.1 \times 3 + 0.05 \times 2) + (0.15 \times 1 + 0.1 \times 2 + 0.05 \times 3 + 0.05 \times 3)$

$= 0.9 + 0.65$

$= 1.55$

$\text{cost(tree3)} = (0.5 \times 2 + 0.1 \times 1 + 0.05 \times 2) + (0.15 \times 2 + 0.1 \times 2 + 0.05 \times 2 + 0.05 \times 2)$

$= 1.9$

$$\text{cost (tree 4)} = [0.5\times 2 + 0.1\times 3 + 0.05\times 1] + [0.15\times 1 + 0.1\times 2 + 0.05\times 3 + 0.05\times 3]$$

$$= 1.35 + 0.65.$$
$$= 2.$$

$$\text{cost (trees)} = [0.5\times 3 + 0.1\times 2 + 0.05\times 1] + [0.15\times 1 + 0.1\times 2 + 0.05\times 3 + 0.05\times 3]$$

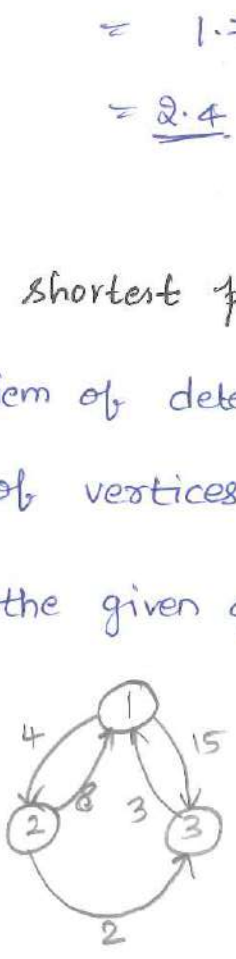
$$= (1.5 + 0.2 + 0.05) + (0.65)$$
$$= 1.75 + 0.65$$
$$= \underline{2.4}$$

```
  1.5
  0.2
  0.05
  ----
  1.75
 +0.65
  ----
  2.40
```

* All pairs shortest path problem:

→ The problem of determining the shortest path among all pairs of vertices of a given graph is

Consider the given graph



→ The above graph consisting of three vertices 1,2,3. so we have to find shortest paths b/w all types of vertices 1,2,3.

→ All pairs are $A(1,1), A(1,2), A(1,3), A(2,1), A(2,2), A(2,3), A(3,1), A(3,2), A(3,3)$

→ ~~In adjacen~~ Represent the given matrix in adjacency matrix form.

$$A^0 = \begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & 0 & 4 & 15 \\ 2 & 8 & 0 & 2 \\ 3 & 3 & \infty & 0 \end{array}$$

$$\boxed{\begin{array}{l} A^0 = w[i,j] \\ A^k_{(i,j)} = \min\left\{A^{k-1}_{(i,j)},\ A^{k-1}_{(i,k)} + A^{k-1}_{(k,j)}\right\} \end{array}}$$

Step-2: Take vertex 1 as intermediate vertex ($k=1$) and find shortest path among all pairs of vertices

$i=1, j=1, k=1$

$$A^1_{(1,1)} = \min\left\{A^{k-1}_{(1,1)},\ A^{k-1}_{(1,1)} + A^{k-1}_{(1,1)}\right\}$$

$$= \min\left\{A^0_{(1,1)},\ A^0_{(1,1)} + A^0_{(1,1)}\right\}$$

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

$$= 0.$$

$i=1, j=2, k=1$

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

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

$$= 4$$

$i=1, j=3, k=1$

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

$$= \min\{15, 0+15\}$$

$$= 15$$

→ ~~In adjacen~~ Represent the given matrix in adjacency matrix form.

$$A^0 = \begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & 0 & 4 & 15 \\ 2 & 8 & 0 & 2 \\ 3 & 3 & \infty & 0 \end{array}$$

$$\boxed{\begin{array}{l} A^0 = w[i,j] \\ A^k_{(i,j)} = \min\left\{A^{k-1}_{(i,j)},\ A^{k-1}_{(i,k)} + A^{k-1}_{(k,j)}\right\} \end{array}}$$

Step-2: Take vertex 1 as intermediate vertex $(k=1)$ and find shortest path among all pairs of vertices

$i=1, j=1, k=1$

$$A^1_{(1,1)} = \min\left\{A^{k-1}_{(1,1)},\ A^{k-1}_{(1,1)} + A^{k-1}_{(1,1)}\right\}$$

$$= \min\left\{A^0_{(1,1)},\ A^0_{(1,1)} + A^0_{(1,1)}\right\}$$

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

$$= 0.$$

$i=1, j=2, k=1$

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

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

$$= 4$$

$i=1, j=3, k=1$

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

$$= \min\{15, 0+15\}$$

$$= 15$$

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

$= \min\{8, 8+0\}$

$= 8$

$A^1_{(2,2)} =$

$i=2, j=2, k=1$

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

$= \min\{0, 8+4\}$

$= 0.$

$A^1_{(2,3)} =$

$i=2, j=3, k=1$

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

$= \min\{2, 8+15\}$

$= 2.$

$A^1_{(3,1)} =$

$i=3, j=1, k=1$

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

$= \min\{3, 3+0\}$

$= 3$

$A^1_{(3,2)} =$

$i=3, j=2, k=1$

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

$= \min\{\infty, 3+4\}$

$= 7$

$A^1_{(3,3)} \Rightarrow i=3, j=3, k=1$

$= \min(0, 3+15)$

$= 0.$

$$A^1 = \begin{bmatrix} 0 & 4 & 15 \\ 8 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix}$$

<u>Step-3</u>: Take vertex '2' as a intermediate and find shortest path between all pairs of vertices.

$A^2_{(1,1)} = \min\{0, 4+8\} = 0$

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

$A^2_{(1,3)} = \min\{15, 4+2\} = 6.$

$A^2_{(2,1)} = \min\{8, 0+8\} = 8$

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

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

$A^2_{(3,1)} = \min\{3, 7+8\} = 3$

$A^2_{(3,2)} = \min\{7, 7+0\} = 7$

$A^2_{(3,3)} = \min\{0, 7+2\} = 0.$

$$A^2 = \begin{bmatrix} 0 & 4 & 6 \\ 8 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix}$$

Step-4: Take vertex 3 as intermediate and find shortest path between all pairs of vertices.

$A^3_{(1,1)} = \min\{0, 6+3\} = 0$

$A^3_{(1,2)} = \min\{4, 6+7\} = 4$

$A^3_{(1,3)} = \min\{6, 6+0\} = 6$

$A^3_{(2,1)} = \min\{8, 2+3\} = 5$

$A^3_{(2,2)} = \min\{0, 2+7\} = 0$

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

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

$A^3_{(3,2)} = \min\{7, 0+7\} = 7$

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

$$A^3 = \begin{bmatrix} 0 & 4 & 6 \\ 5 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix}$$

* 0/1 knapsack problem:

→ It contains n items with profits $(P_1, P_2, \ldots P_n)$ and weights $(w_1, w_2, \ldots w_n)$. knapsack means empty bag it has a capacity of m units. and The objective function of this problem is to get maximum profit by filling empty back with given items. subject to constraints $0 \le x_i \le 1$ and $m \le \Sigma w_i$, $i \le n$.

→ we can solve this problem by using dynamic programming method

→ It uses the following steps.

Step-1:

Computational formulae

$$S_1^i = S^{i-1} + (P_i, w_i)$$

→ It is used to add tuples

$$S^i = S^{i-1} + S_1^i$$

→ This formula is used for merging operation.

Step-2:

purging rule

→ It decides which tuple is to be discard and which tuple is to be accept

→ If one of S^{i-1} and S_1^i has a pair (P_j, w_j) and other (P_k, w_k) and $P_j \le P_k$, $w_j > w_k$ then discard (P_j, w_j)

bag

Step-3:

e

Selection of items:

In this step, we can decide which item is to be placed in the bag based on following constraints if $(P_i, w_i) \in S^n$ & $(P_i, w_i) \notin S^{n-1}$ then $x_n = 1$ else $x_n = 0$.

<u>Problems:</u>

1) Consider the instance $n=3$, $m=6$, $(P_1, P_2, P_3) = (1,2,5)$ $(w_1, w_2, w_3) = (2,3,4)$. Find the maximum profit by using dynamic programming method.

<u>Sol:-</u> <u>step-1:</u> Initially

$$S_1^0 = (0,0) \quad , S^0 = (0,0)$$
$$= (P_i, w_i)$$

$\boxed{i=1} \Rightarrow S_1^i = S^{i-1} + (P_i, w_i)$

$$= S^0 + (P_1, w_1)$$
$$= (0,0) + (1,2)$$
$$= (1,2)$$

merging $\boxed{i=1}$.

$$S^i = S^{i-1} + S_1^i$$
$$S^1 = S^0 + S_1^1$$
$$= (0,0) + (1,2)$$
$$S^1 = \{(0,0), (1,2)\}$$

$\boxed{i=2} \Rightarrow S_1^i = S^{i-1} + (P_i, w_i)$

$$= S^1 + (P_2, w_2)$$

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

$= \{(2,3),(3,5)\}$

$S^i = S^{i-1} + S_1^i$

$S^2 = S^1 + S_1^2$

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

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

$\boxed{i=3}$

$S_1^i = S^{i-1} + (P_i, w_i)$

$S_1^3 = S^2 + (P_3, w_3)$

$\{(0,0),(1,2),(2,3),(3,5)\} + \{(5,4)\}$

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

$S^i = S^{i-1} + S_1^i$

$S^3 = S^2 + S_1^3$

$= \{(0,0)(1,2),(2,3),(3,5)\} + \{(5,4),(6,6),(7,7),(8,9)\}$

$= \{(0,0),(1,2),(2,3),(3,5),(5,4),(6,6),(7,7),(8,9)\}$

<u>step-2</u>: Apply the purging rule on tuples of

$S^3 = \{(0,0),(1,2),(2,3),(3,5),(5,4),(6,6),(7,7),(8,9)\}$

discard tuples (7,7), (8,9) because tuple values exceeded the size of the knapsack size m=6.

→ Apply the purging rule on (3,5) and (5,4)

(3,5) & (5,4)

(P_j, w_j) & (P_K, W_K)

$(P_j < P_k)$ and $(w_j > w_k)$

$(3 < 5)$ and $(5 > 4)$ so discard $(3,5)$ tuple.

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

Step-3:

Decide which items are placed in the bag

$(P_i\, w_i) = S^n$ & $(P_i\, w_i) \notin S^{n-1}$

$n = 3$

$(6,6) \in S^3$ and $(6,6) \notin S^2$

T & T = T

$x_n = 1 = x_3 = 1$

$\boxed{(P_3, w_3) = (5,4)}$

$(6,6) - (5,4) = (1,2)$

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

T & F

$x_2 = 0$.

$(1,2) \in S^1$ & $(1,2) \notin S^0$

T & T = T

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

$\Sigma P_i x_i = P_1 x_1 + P_2 x_2 + P_3 x_3$

$= 1 \cdot 1 + 2 \cdot 0 + 5 \cdot 1$

$= 1 + 0 + 5$

$= 6$.

* Travelling salesperson problem:

→ It is to find a ~~tool~~ tour of minimum ~~cot~~ cost. A ~~tool~~ tour of graph 'G' is a directed simple cycle that includes every vertex in 'V'.

→ The cost of the tour is the sum of the cost of the edges on the tour.

→ Suppose we have to route a postal van to ~~beco~~ pick up mail from mail boxes located at 'l' different sites.

→ One vertex represents the post office from which the postal van starts and to which it must return.

→ Edge (i,j) is assigned a cost equal to the distance from site 'i' to site 'j'.

→ The route taken by postal van is a tour and we are interested to find a tour of minimum length.

→ We can solve travelling salesperson problem by using dynamic programming method.

→ It uses the following computational formulae..

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

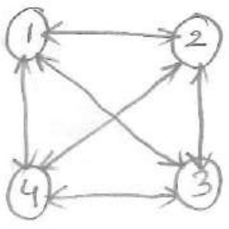
$g(i,\phi) = c_{i1}$, $1 \le i \le n$

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

<u>problem:</u>

1) consider the directed graph and its adjacent matrix and find tour of minimum cost of travelling sales person problem.

<u>Sol:-</u>



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

Initially

$g(i,\phi) = c_{i1}$ $1 \le i \le 4$

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

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

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

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

→ Take $|S|=1$ that means set S contains only one vertex. From one vertex we can visit all the vertices 2, 3, 4

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

$g(2,\{3\}) = \min\limits_{j \in S} \{c_{23} + g(3, \{3\}-\{3\})\}$

$= \min\limits_{j \in S} \{9 + g(3,\phi)\}$

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

$= 15$

d

$g(2,\{4\}) = \min\limits_{j\in\{4\}} \{c_{24} + g(4, \{4\}-\{4\})\}$

$= \min\{10+g(4,\phi)\}$

$= \min\{10+8\}$

$= 18$

$g(3,\{2\}) = \min\limits_{j\in\{2\}} \{c_{32}+g(2,\{2\}-\{2\})\}$

$= \min\limits_{j\in\{2\}} (13+5)$

$= 18$

$g(3,\{4\}) = \min\limits_{j\in\{4\}} \{c_{34}+g(4,\{4\}-\{4\})\}$

$= \min (12+g(4,\phi)\}$

$= 12+8$

$= 20.$

$g(4,\{2\}) = \min\limits_{j\in\{2\}} \{c_{42}+ g(2, \{2\}-\{2\})\}$

$= \min\{8+g(2,\phi)\}$

$= 13$

From

all the

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

$= \min\{9+ g(3,\phi)\}$

$= 15.$

→ Take $|S|=2$ that means set S contains only two vertices.

$$g(2,\{3,4\}) = \min_{j\in\{3,4\}} \left(c_{23}+g(3,\{3,4\}-\{3\}),\ c_{24}+g(4,\{3,4\}-\{4\})\right)$$

$$= \min\{9+g(3,\{4\}),\ 10+g(4,\{3\})\}$$

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

$$= 25.$$

$$g(3,\{2,4\}) = \min_{j\in\{2,4\}} \left(c_{32}+g(2,\{2,4\}-\{2\}),\ c_{34}+g(4,\{2,4\}-\{4\})\right)$$

$$= \min\{13+g(2,\{4\}),\ 12+g(4,\{2\})\}$$

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

$$= 25$$

$$g(4,\{2,3\}) = \min_{j\in\{2,3\}} \left(c_{42}+g(2,\{2,3\}-\{2\}),\ c_{43}+g(3,\{2,3\}-\{3\})\right)$$

$$= \min\{8+g(2,\{3\}),\ 9+g(3,\{2\})\}$$

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

$$= 2\cancel{4}3$$

$$g(1,\{2,3,4\}) = \min_{j\in\{2,3,4\}} \{c_{12}+g(2,\{2,3,4\}-\{2\}),\ c_{13}+g(3,\{2,3,4\}-\{3\}),$$

$$c_{14}+g(4,\{2,3,4\}-\{4\}))$$

$$= \min\{10+g(2,\{3,4\}),\ 15+g(3,\{2,4\}),\ 20+g(4,\{2,3\})\}$$

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

$$= \min\{35,40,43\}$$

$$= 35$$

path of the four with minimum cost 35

$c_{12} + g(2, \{3,4\})$

$= c_{12} + c_{24} + g(4, \{3\})$

$= c_{12} + c_{24} + c_{43} + g(3, \phi)$

$= c_{12} + c_{24} + c_{43} + c_{31}$

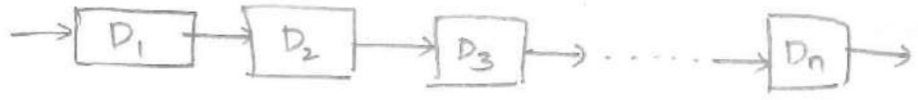
$= 10 + 10 + 9 + 6$

$= 35$

path is $1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 1$

* Reliability design problem:

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



→ Let 'r_i' be the reliability of the device 'D_i' [i.e r_i is the probability ~~of the entire system~~ that device 'i' will function properly].

→ Then the reliability of entire system is 'Πr_i'.

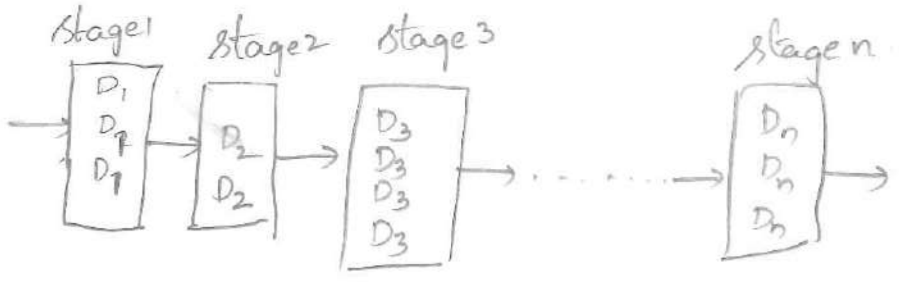
→ Even if the individual devices are very reliable then the reliability of system may not be very good.

for example: If $n=10$, $r_i = 0.99$, $1 \le i \le 10$. then

$\Pi r_i = 0.99 \times 3.14 = 0.904$

→ Hence it is required to use duplicate devices.

→ Multiple copies of same device type are connected in parallel through the use of switching circuit.



Problem: Design a 3 stage system with device types D_1, D_2, D_3. The cost are 30, 15, 20 \$ respectively. The cost of system should no more than 105 \$. The reliability of each device type is 0.9, 0.8, 0.5 respectively.

Sol:- Our problem is to use device duplication to maximize reliability. This maximization is to be carried out under ~~a~~ cost constraint.

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

→ Maximize $\prod_{1 \le i \le n} \phi_i(m_i)$

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

$m_i \ge 1$ and integer, $1 \le i \le n$

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

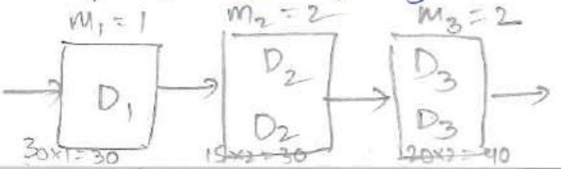
$\phi(m_i) = 1 - (1 - r_i)^{m_i}$

→ $c_1 = 30,\ c_2 = 15,\ c_3 = 20$

System cost = 105

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

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



$= 30 + 30 + 40$

$= 100$

→ The best design has a reliability of 0.648 and cost of 100 dollars (\$)