mood-book

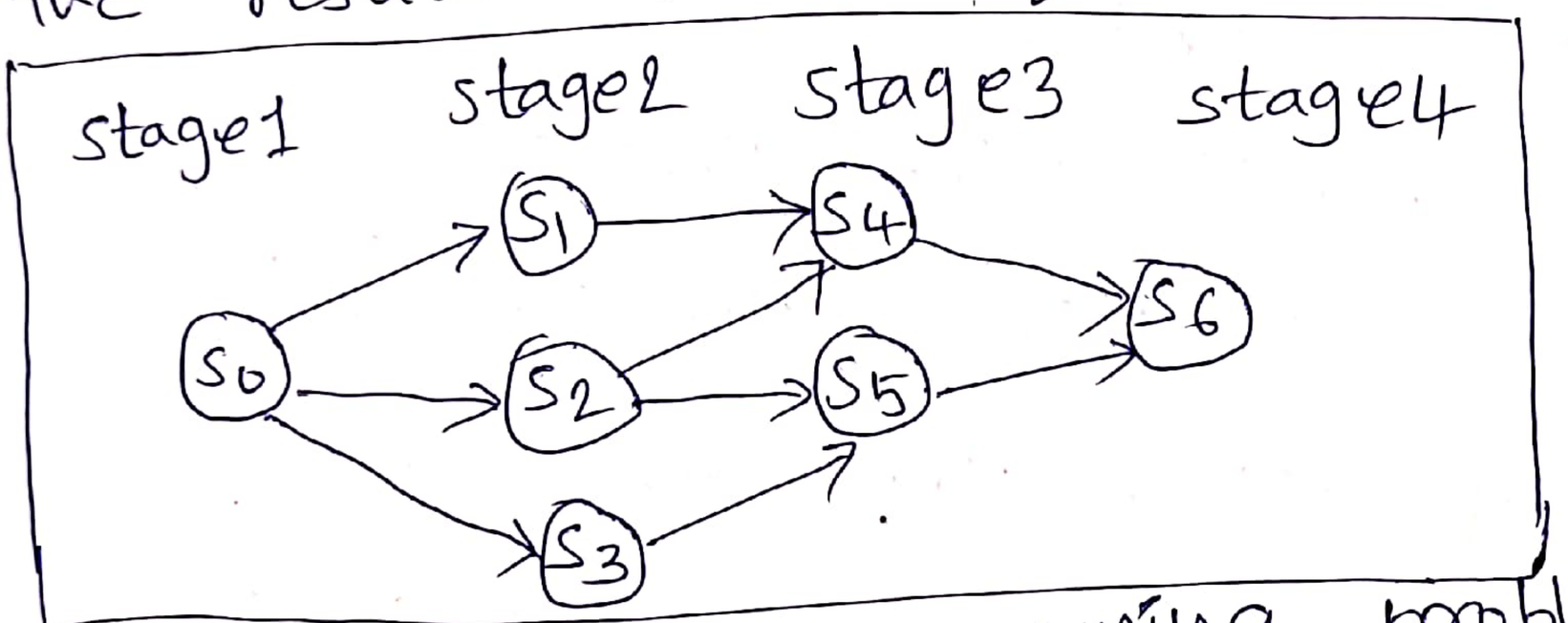
UNIT - III

DYNAMIC PROGRAMMING

In Greedy method, we will make one decision at a time

In Dynamic programming, we will make more than one decision at a time

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



A dynamic programming problem can be divided into a number of stages, where an optimal decision must be made at each stage

The decision made at each stage influences the decision to be taken next.

Each stage has a number of states associated with it.

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

Applications of Dynamic programming

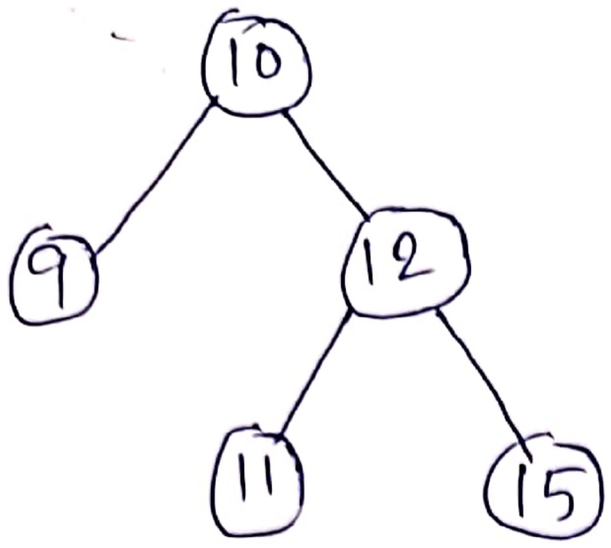
1) Optimal Binary search trees (OBST).
2) 0/1 knapsack problem
3) All pairs shortest paths problem
4) Travelling sales person problem (TSP)
5) Reliability design problem.

OPTIMAL BINARY SEARCH TREES (OBST)

Property of Binary search tree

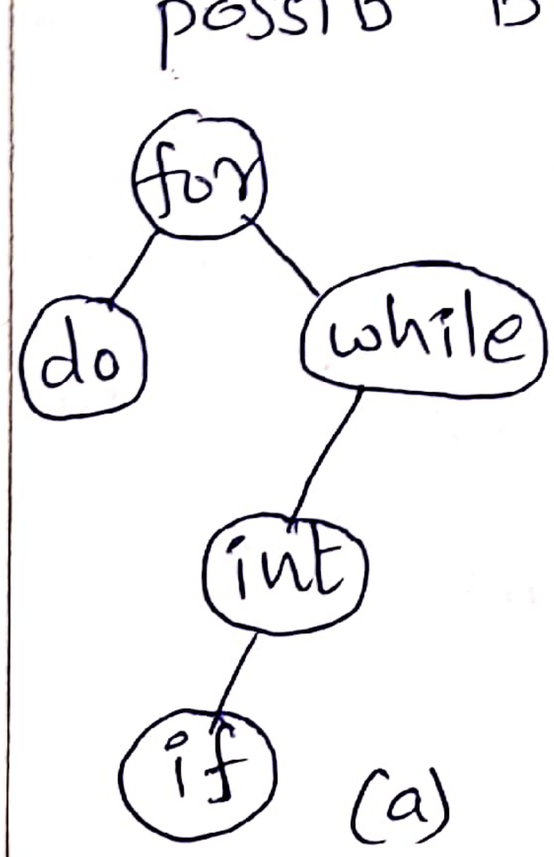
- Left child value must be less than (numerically or alphabetically) the value of parent node
- Right child node value $\geq$ parent node value.

Eg

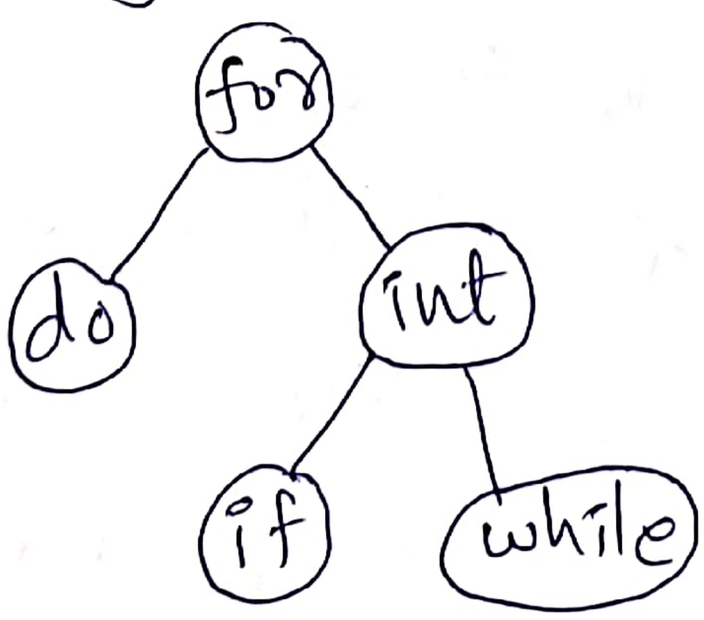


If we want to search for an element in binary search tree, first that element is compared with root node. If element is less than root node then search will continue in left subtree. If the element is greater than root node then search will continue in the right subtree. If element is equal to root node then print the successful search (element found) and terminate search procedure.

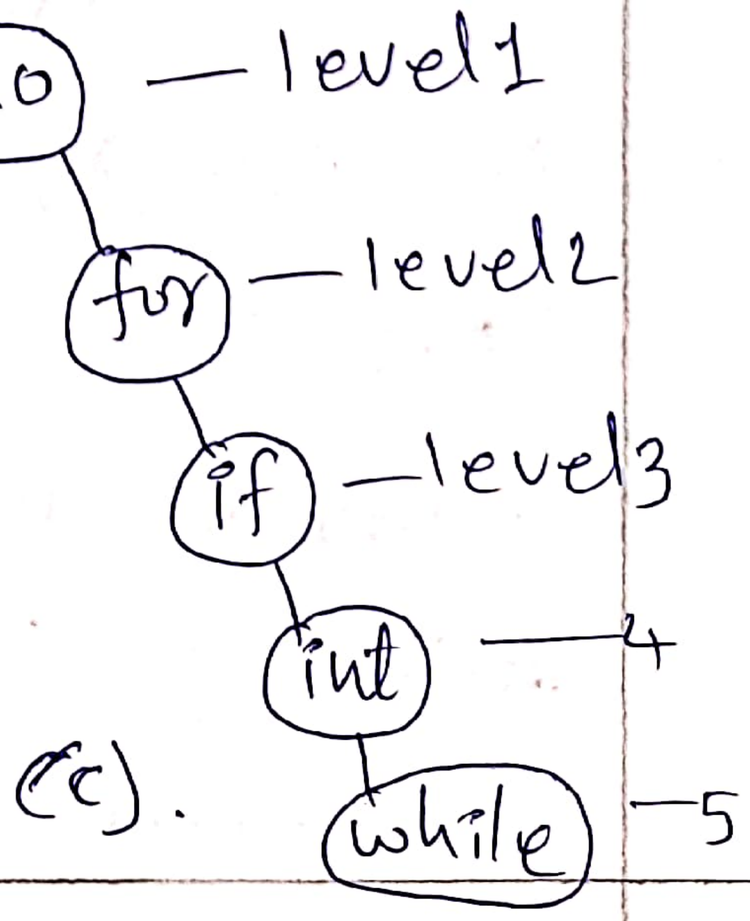
→ Eg Given set of identifiers
{do, for, if, int, while}.
possib binary search trees



(a)



(b)



(c).

Average search time

= Average no. of comparisons

$$= \frac{\text{Total no. of comparison}}{\text{No. of elements}}$$

For (a) $= \frac{1+2+2+3+4}{5} = \frac{12}{5}$

For (b) $= \frac{1+2+2+3+3}{5} = \frac{11}{5}$ ← optimal binary search tree.

For (c) $= \frac{1+2+3+4+5}{5} = \frac{15}{5}$

∴ As its search time is less out three binary search trees, (a) is optimal binary search tree.

→ In the above example we assumed that each identifier is searched for with equal probability.

→ The no. of comparisons required to search for an element is = level of that element in the binary search tree.

→ But in a general situation, we can expect different identifiers to be searched for with different frequencies (or probabilities) or different number of times.

In addition, we can expect unsuccessful searches also to be made.

Let us assume that the given set of identifiers is $\{a_1, a_2, \ldots, a_n\}$ with $a_1 < a_2 < \cdots < a_n$

→ Let $P(a_i)$ or $P(i)$ be the probability with which we search for (a_i) an internal node [successful search, node or element which is present in the database]

→ Let $q(E_i)$ or $q(i)$ be the probability that the identifier x being searched for is such that $a_i < x < a_{i+1}$, $0 \le i \le n$ [unsuccessful search for an external element which is not present in the database].

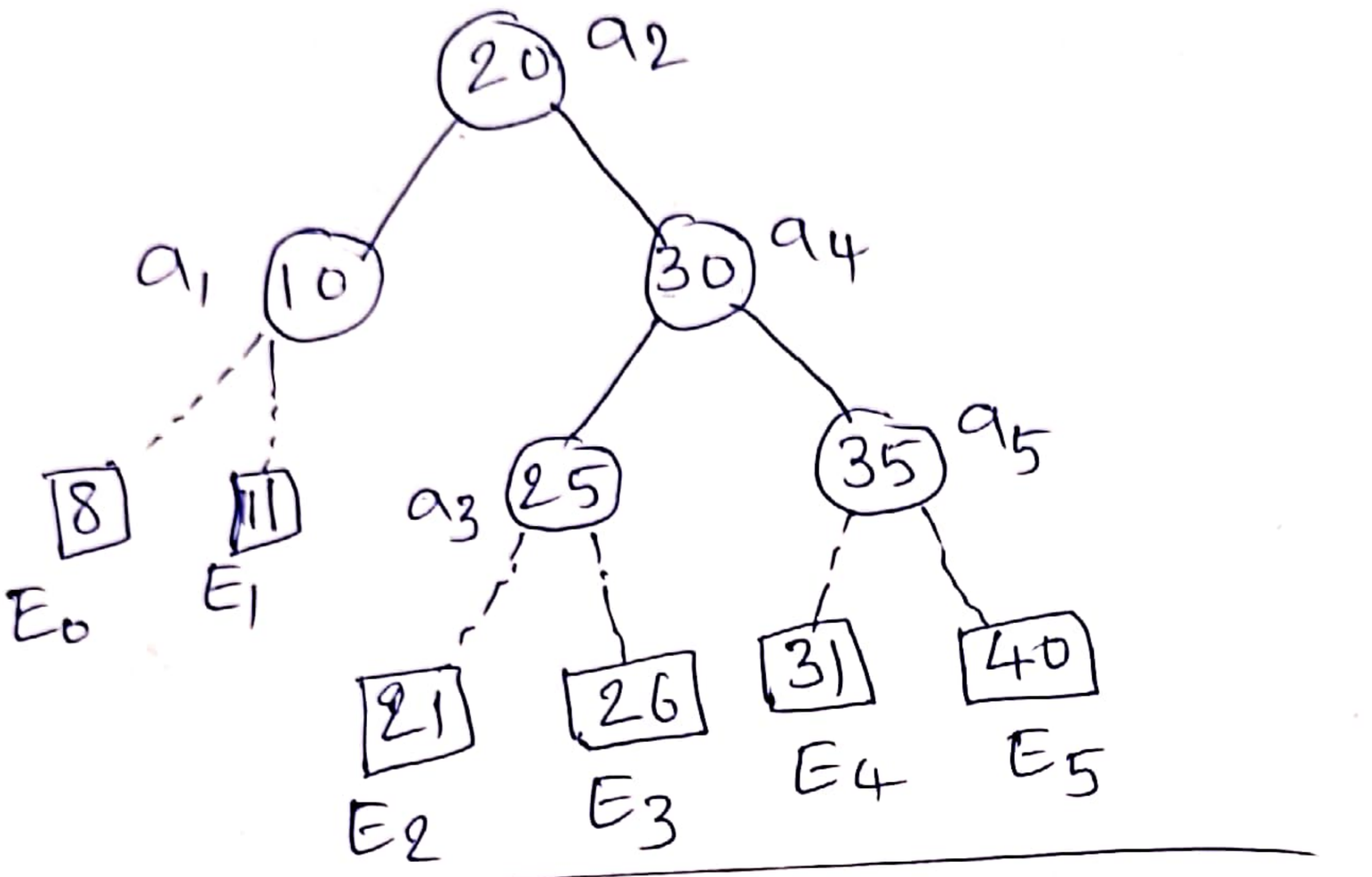
sum of the probabilities = 1

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

or

$$\sum_{1 \le i \le n} p(a_i) + \sum_{0 \le i \le n} q(E_i) = 1$$

If a binary search tree represents n identifiers, then there will be exactly n internal nodes and n+1 external nodes (fictious nodes).



a_1	a_2	a_3	a_4	a_5
10	20	25	30	35

8	11	21	26	31	40
E_0	E_1	E_2	E_3	E_4	E_5.

search for 11 terminates at 10 = level 2

= no. of comparison = 2 = level(11) − 1 = 3−1.

level of fictious node 11 = 3

Expected search cost of binary search tree is given by the formula cost(Tree) =

$$\sum_{1 \le i \le n} p(a_i) \times level(a_i) + \sum_{0 \le i \le n} q(E_i) \times [level(E_i) - 1]$$

No. of comparisons required for an external element E_i = level $(E_i) - 1$.

Eg Give identifier set $(a_1, a_2, a_3) = (do, if, while)$ with equal probability of search for all internal and external elements. Find optimal binary search tree.

Sol:- Given $n = 3$ identifiers

There will $n = 3$ internal nodes and $n+1 = 3+1 = 4$ external nodes.

Total no. of possible binary search trees

$$= \frac{1}{n+1} {}^{2n}C_n \qquad {}^{n}C_r = \frac{n!}{(n-r)!\,r!}$$

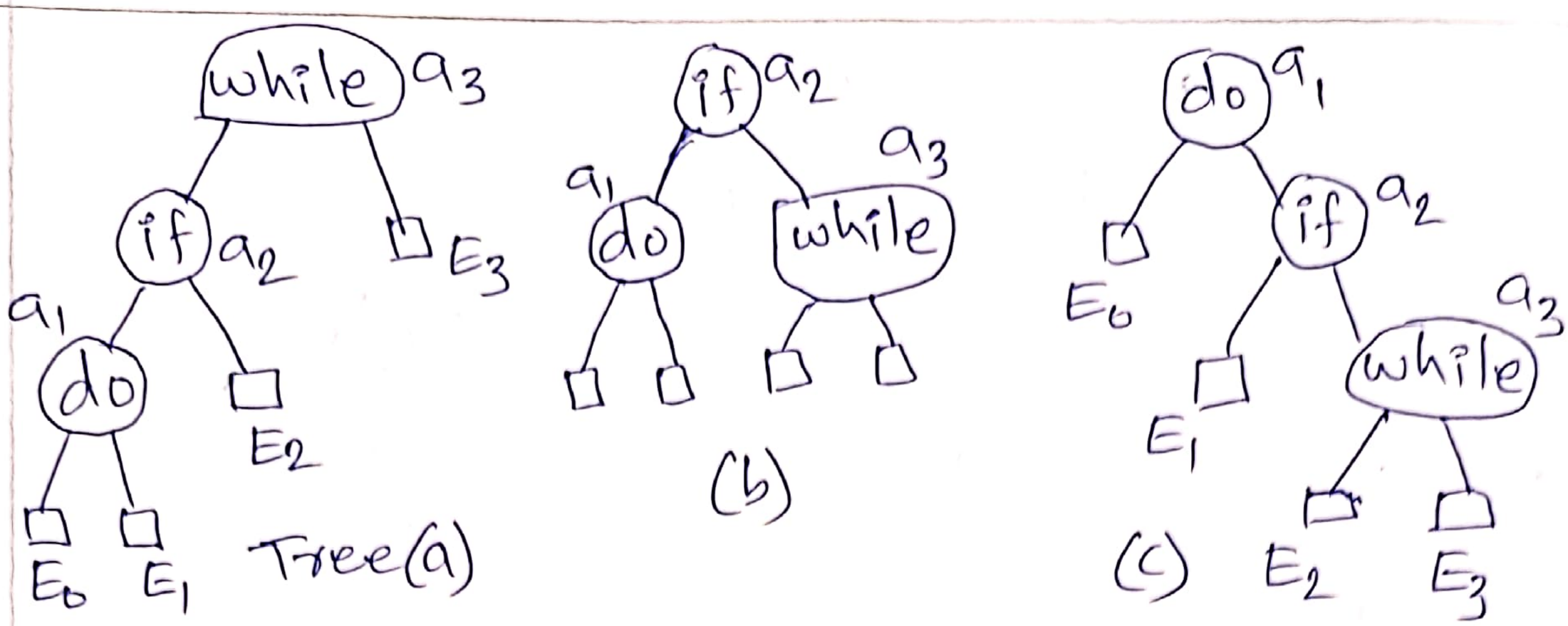
$$= \frac{1}{3+1} {}^{6}C_n$$

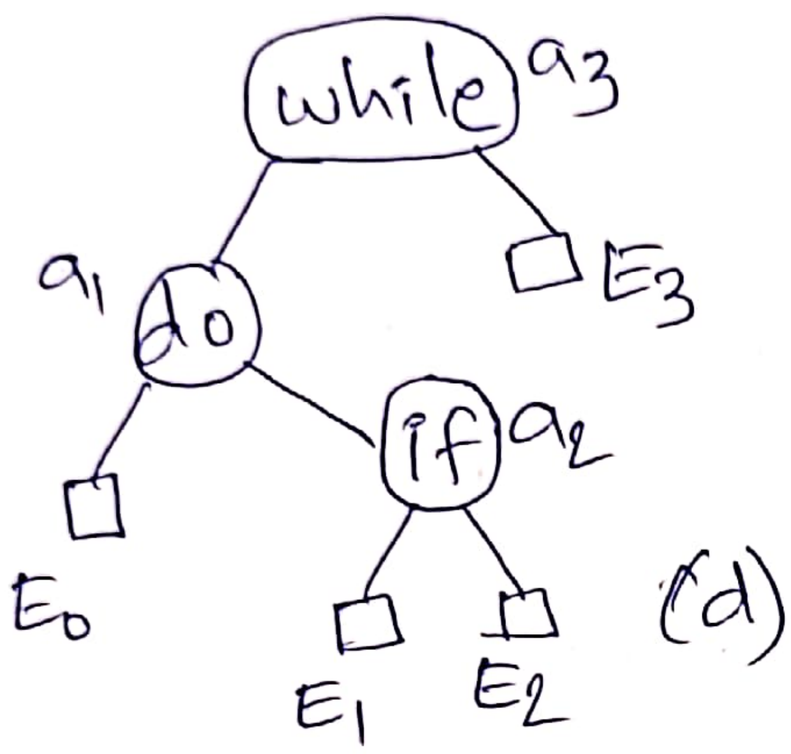
$$= \frac{1}{4} \times \frac{6 \times 5 \times 4 \times 3!}{(6-3)!\,3!}$$

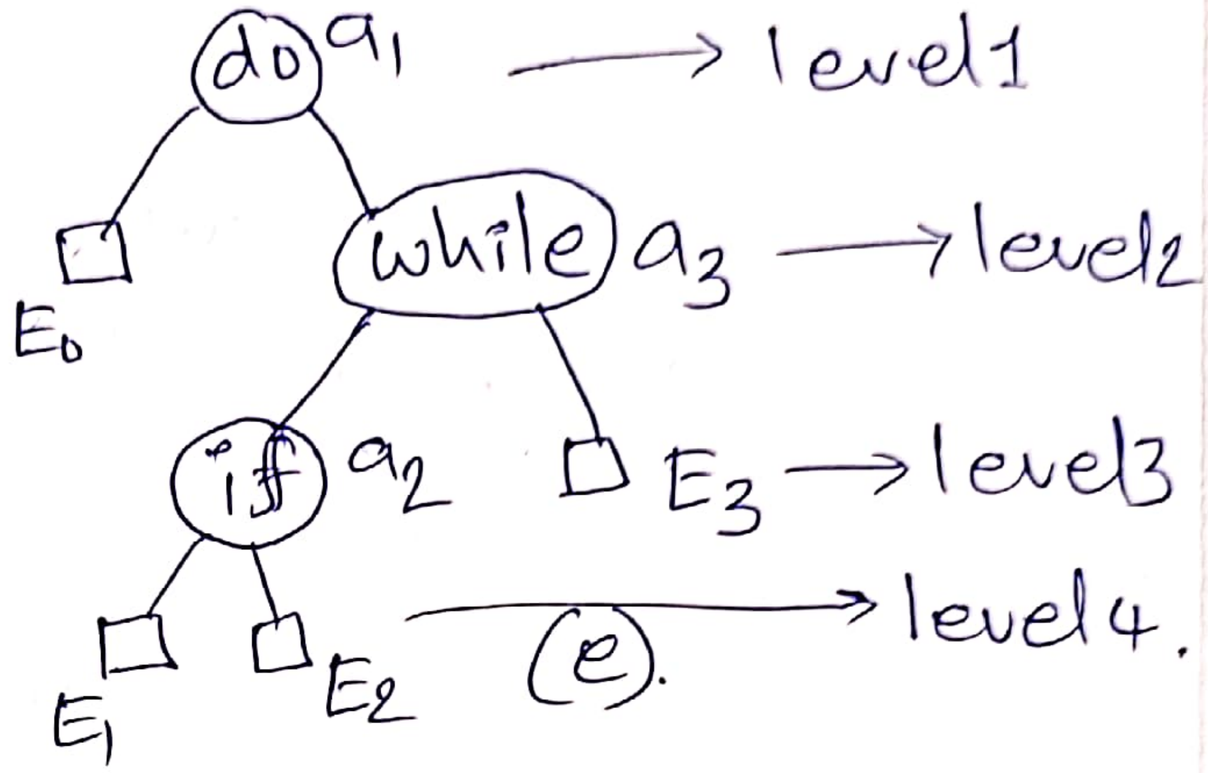
$$= \frac{1}{4} \times \frac{6 \times 5 \times 4}{3!}$$

$$= 5.$$

$$P(a_1) = P(a_2) = P(a_3) = q(E_0) = q(E_1) = q(E_2) = q(E_3) = \frac{1}{7}$$







$$cost(T) = \sum_{1 \le i \le n} p(i) \times level(a_i) + \sum_{0 \le i \le n} q(i) \times [level(E_i) - 1]$$

$$= p(1)\,level(a_1) + p(2)\,level(a_2) + p(3)\,level(a_3) +$$

$$q(0)[level(E_0)-1] + q(1)[level(E_1)-1] + q(2)[level(E_2)-1] +$$

$$q(3)[level(E_3)-1].$$

$$cost(Tree\ a) = \frac{1}{7}\times 3 + \frac{1}{7}\times 2 + \frac{1}{7}\times 1 + \frac{1}{7}(4-1) + \frac{1}{7}(4-1)$$

$$+ \frac{1}{7}(3-1) + \frac{1}{7}\times(2-1)$$

$$cost(a) = \frac{1}{7}(3+2+1+3+3+2+1) = 15/7.$$

$$COST(b) = \frac{1}{7}(1+2+2) + \frac{1}{7}(2+2+2+2) = \frac{13}{7} \sim \text{optimal.}$$

$cost(c) = \frac{1}{7}(1+2+3) + \frac{1}{7}(1+2+3+3) = 15/7.$

$cost(d) = \frac{1}{7}(1+2+3) + \frac{1}{7}(1+2+3+3) = 15/7.$

$cost(e) = \frac{1}{7}(1+2+3) + \frac{1}{7}(1+2+3+3) = 15/7.$

cost the binary search tree (b) is minimum. hence it is optimal binary search tree.

→ solve the above problem by taking

$p(1) = 0.5$, $p(2) = 0.1$; $p(3) = 0.05$

$q(0) = 0.15$, $q(1) = 0.1$, $q(2) = 0.05$, and $q(3) = 0.05$.

$cost(a) = 0.05\times3 + 0.1\times2 + 0.05\times1 + 0.15\times3 + 0.1\times3 + 0.05\times2 + 0.05\times1 = \underline{2.65}$

$cost(b) = 0.5\times2 + 0.1\times1 + 0.05\times2 + 0.15\times2 + 0.1\times2 + 0.05\times2 + 0.05\times2 = \underline{1.9}.$

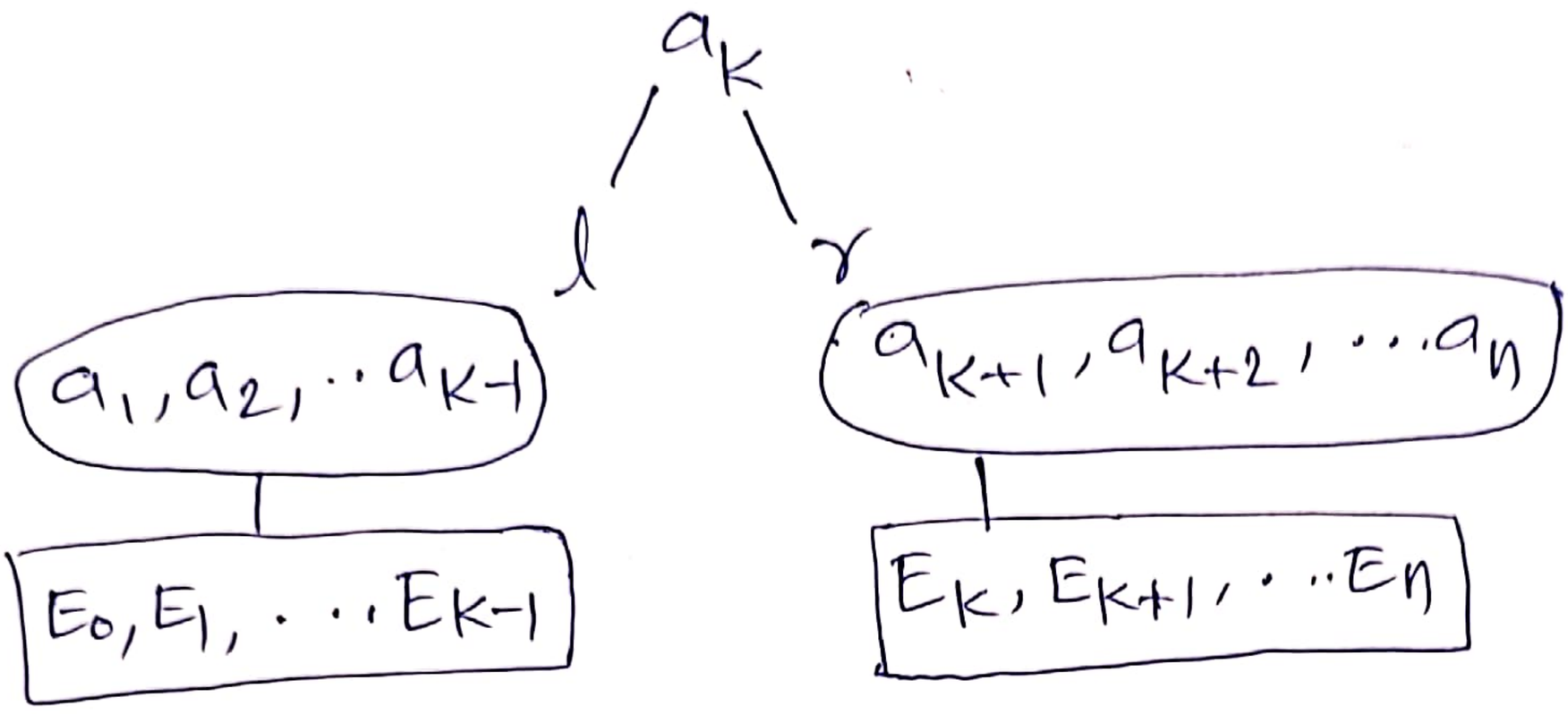
$cost(c) = 0.5\times1 + 0.1\times2 + 0.05\times3 + 0.15\times1 + 0.1\times2 + 0.05\times3 + 0.05\times3 = \underline{1.5}$ ✓

$cost(d) = 0.5\times2 + 0.1\times3 + 0.05\times1 + 0.15\times2 + 0.1\times3 + 0.05\times3 + 0.05\times1 = \underline{2.05}$

$cost(e) = 0.5\times1 + 0.1\times3 + 0.05\times2 + 0.15\times1 + 0.1\times3 + 0.05\times3 + 0.05\times2 = 1.6.$

cost(c) is minimum. Hence with these probabilities of search (c) is optimal binary search tree

Which of the a_i's should be placed at root node of the tree



search cost of left subtree l

$$cost(l) = \sum_{1 \le i < k} p(i) \times level(a_i) + \sum_{0 \le i < k} q(i)\left[level(E_i) - 1\right]$$

cost of right subtree r

$$cost(r) = \sum_{k < i \le n} p(i) \times level(a_i) + \sum_{k < i \le n} q(i)\left[level(E_i) - 1\right].$$

we use $w(i,j)$ to represent the sum

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

$c(i,j)$ represents the cost of an optimal binary search tree t_{ij} containing

$a_{i+1}, \ldots a_j$ and $E_i, \ldots, E_j$

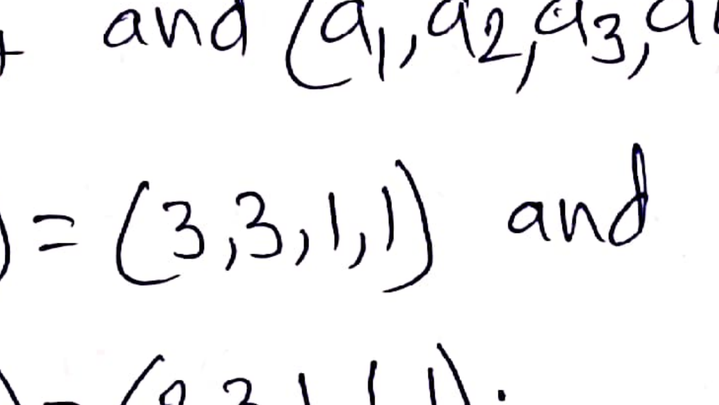
$c(i,i)=0$ and $w(i,i)=q(i)$ } for $0 \le i \le n$

$r(i,i)=0$

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

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

$T_{i,j} = T_{i,k-1}, T_{k,j}$



$r(i,j)=k \rightarrow$ is the value of k that minimizes $c(i,j)$

Eg Let $n=4$ and $(a_1,a_2,a_3,a_4) = (do, if, int, while)$

Let $p(1:4) = (3,3,1,1)$ and

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

Find the values of $w(i,j)$'s, $c(i,j)$'s, $r(i,j)$'s and find OBST.

sol:- $p(1:4) = (3,3,1,1) \rightarrow = 8$ $3/16, 3/16, 1/16, 1/16$

$q(0:4) = (2,3,1,1,1) \rightarrow = 8.$ $2/16, 3/16, 1/16, 1/16, 1/16$

↓

probabilities of search for elements

No. of possible binary search trees $n=4$.

$$= \frac{1}{n+1} {}^{2n}C_n$$

$$= \frac{1}{4+1} {}^{8}C_4$$

$$= \frac{1}{5} \times \frac{8 \times 7 \times 6 \times 5 \times 4!}{(8-4)!\,4!}$$

$$= \frac{8 \times 7 \times 6}{4!}$$

$$= \frac{8 \times 7 \times 6^{2}}{24_{3}} = 14$$

But by using tabular method we can construct optimal binary search tree directly.

In order to solve the above problem, first we will draw one table by taking 'i' corresponding to rows, 'j' corresponding to columns.

The cells in the table can be indicated as

$$w_{j,j+i}, \quad c_{j,j+i}, \quad r_{j,j+i}$$

Initially we have, $w(i,i) = q(i)$, $c(i,i) = 0$, $r(i,i) = 0$, $0 \le i \le 4$

$w_{00} = q(0) = 2$ $\quad$ $c_{00} = r_{00} = 0$

$w_{11} = q(1) = 3$ $\quad$ $c_{11} = r_{11} = 0$

$w_{22} = q(2) = 1$ $\quad$ $c_{22} = r_{22} = 0$

$w_{33} = q(3) = 1$ $\quad$ $c_{33} = r_{33} = 0$

$w_{44} = q(4) = 1$ $\quad$ $c_{44} = r_{44} = 0$.

computation of $w(0,4), c(0,4), r(0,4)$ $\longrightarrow j$

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

$\downarrow i$

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

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

$$w(1,2) = P(2) + Q(2) + w(1,1)$$
$$= 3+1+3$$
$$= 7.$$

$$w(2,3) = P(3) + Q(3) + w(2,2)$$
$$= 1+1+1$$
$$= 3.$$

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

$= 1+1+1$

$= 3$

$w(0,2) = p(2) + q(2) + w(0,1)$

$= 3+1+8$

$= 12$

$w(1,3) = p(3) + q(3) + w(1,2)$

$= 1+1+7$

$= 9.$

$w(2,4) = p(4) + q(4) + w(2,3)$

$= 1+1+3$

$= 5$

$w(0,3) = p(3) + q(3) + w(0,2)$

$= 1+1+12$

$= 14$

$w(1,4) = p(4) + q(4) + w(1,3)$

$= 1+1+9$

$= 11$

$w(0,4) = p(4) + q(4) + w(0,3)$

$= 1+1+14$

$= 16.$

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

$c(0,1) = \min_{0<k\leq 1} \{c(0,0) + c(1,1)\} + w(0,1)$

$= 0+0+8$

$= 8.$ $\quad r(0,1) = 1.$

$$c(1,2) = \min_{1<K\leq 2} \{c(1,1) + c(2,2)\} + w(1,2)$$

$$= 0+0+7$$

$$= 7.$$

$$r(1,2) = 2$$

$$c(2,3) = \min_{2<K\leq 3} \{c(2,2) + c(3,3)\} + w(2,3).$$

$\underline{K=3}$

$$= 0+0+3$$

$$= 3.$$

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

$$c(3,4) = \min_{3<K\leq 4} \{c(3,3) + c(4,4)\} + w(3,4)$$

$K=4$

$$= 0+0+3$$

$$= 3.$$

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

$$c(\underset{i}{0},\underset{j}{2}) = \min_{0<K\leq 2} \{\underbrace{c(0,0)+c(1,2)}_{K=1\ a_1}, \underbrace{c(0,1)+c(2,2)}_{K=2\ a_2}\} + w(0,2)$$

$$= \min\{0+7, 8+0\} + 12$$

$$= 19.$$

$$r(0,2) = 1. \ a_1$$

$$c(1,3) = \min_{1<K\leq 3} \{\underbrace{c(1,1)+c(2,3)}_{K=2}, \underbrace{c(1,2)+c(3,3)}_{K=3.}\} + w(1,3).$$

$$= \min\{0+3, 7+0\} + 9.$$

$$= 12$$

$$r(1,3) = 2 \quad a_2.$$

$$c(2,4) = \min_{2<k\leq 4} \{ c(2,2) + c(3,4),\ c(2,3) + c(4,4) \} + w(2,4)$$

($k=3$: $c(2,2)+c(3,4)$; $k=4$: $c(2,3)+c(4,4)$)

$$= \min\{0+3,\ 3+0\} + 5$$

$$= 8$$

$$r(2,4) = 3 \quad a_3.$$

$$c(0,3) = \min_{0<k\leq 3} \{ c(0,0) + c(1,3),\ c(0,1) + c(2,3),\ c(0,2) + c(3,3) \} + w(0,3)$$

$$= \min\{0+12,\ \underline{8+3},\ 19+0\} + 14$$

($k=1$: $0+12$; $k=2$: $8+3$; $k=3$: $19+0$)

$$= 25$$

$$r(0,3) = 2 \quad a_2$$

$$c(1,4) = \min_{1<k\leq 4} \{ c(1,1) + c(2,4),\ c(1,2) + c(3,4),\ c(1,3) + c(4,4) \} + w(0,3)$$

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

$$= 19.$$

$$r(1,4) = 2 \quad a_2.$$

$$c(0,4) = \min_{0<k\leq 4} \{ c(0,0) + c(1,4),\ c(0,1) + c(2,4),\ c(0,2) + c(3,4),\ c(0,3) + c(4,4) \} + w(0,4).$$

$$= \min\{0+19,\ \underline{8+8},\ 19+3,\ 25+0\} + 16.$$

$$= \min\{19, 16, 22, 25\} + 16.$$

$$= 32$$

$$r(0,4) = 2 \quad a_2.$$

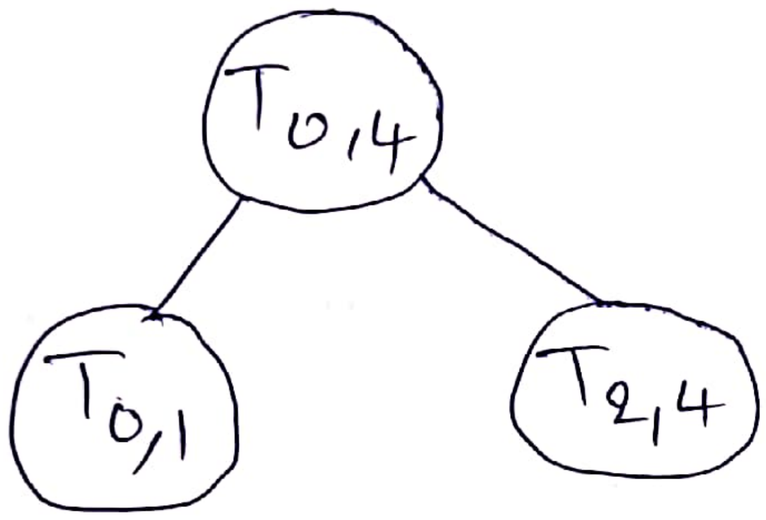
Now observe the table last cell i.e 4th row and 0th column, contains $r_{04}=2$ i.e $r_{04}=a_2$ (2 corresponds to second node a_2)

$r_{04}=k$, then $k=2$

Let T be the optimal binary search tree

$$T_{i,j} = T_{i,k-1}, T_{k,j}$$

$T_{0,4}$ is divided into two parts $T_{0,1}$ & $T_{2,4}$ ($\because k=2$)

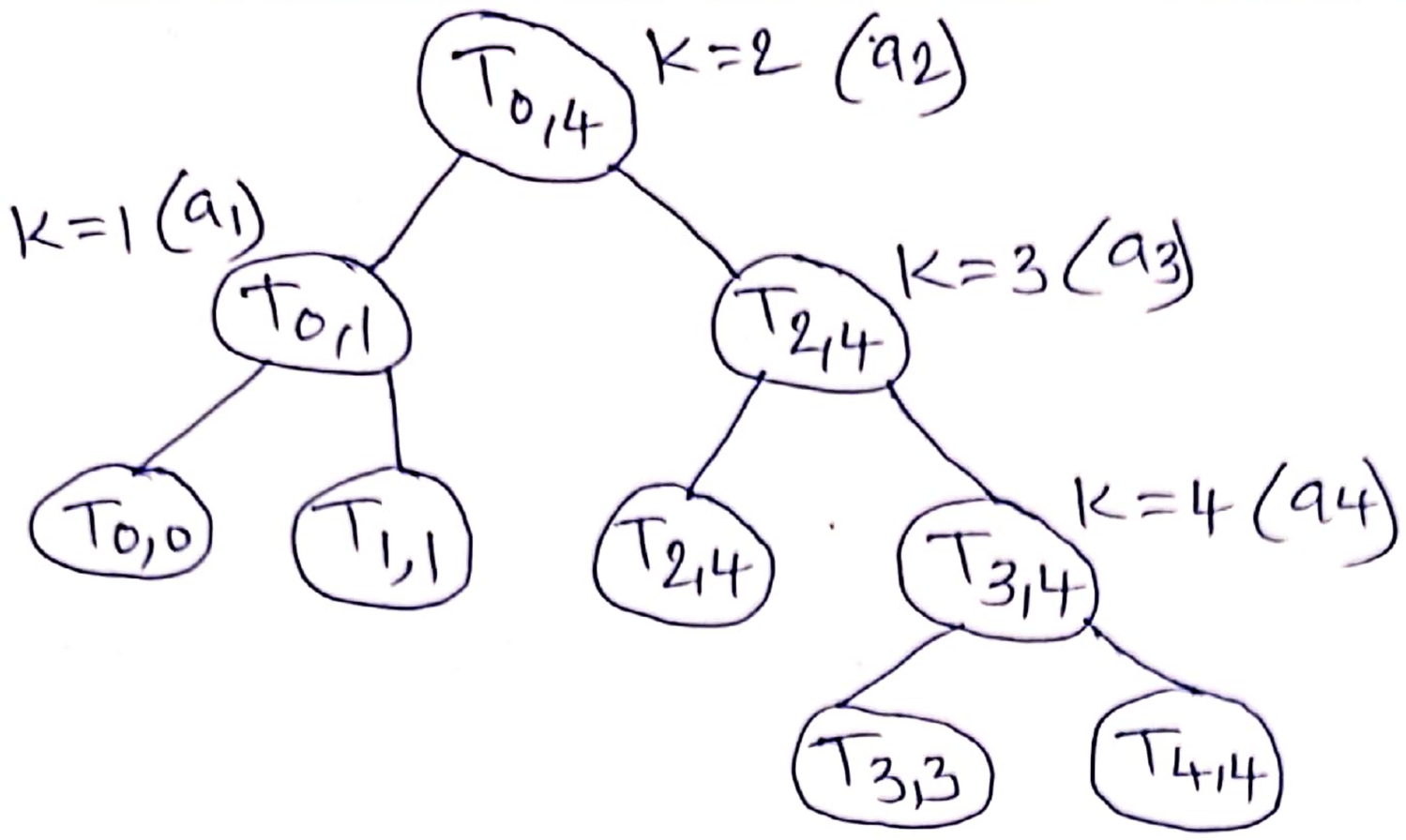


Now $r_{0,1}=1 \Rightarrow k=1$ (a_1) $r_{24}=3 \Rightarrow k=3$. (a_3)

$\therefore T_{0,1}$ is divided into $T_{0,0}$ & $T_{1,1}$

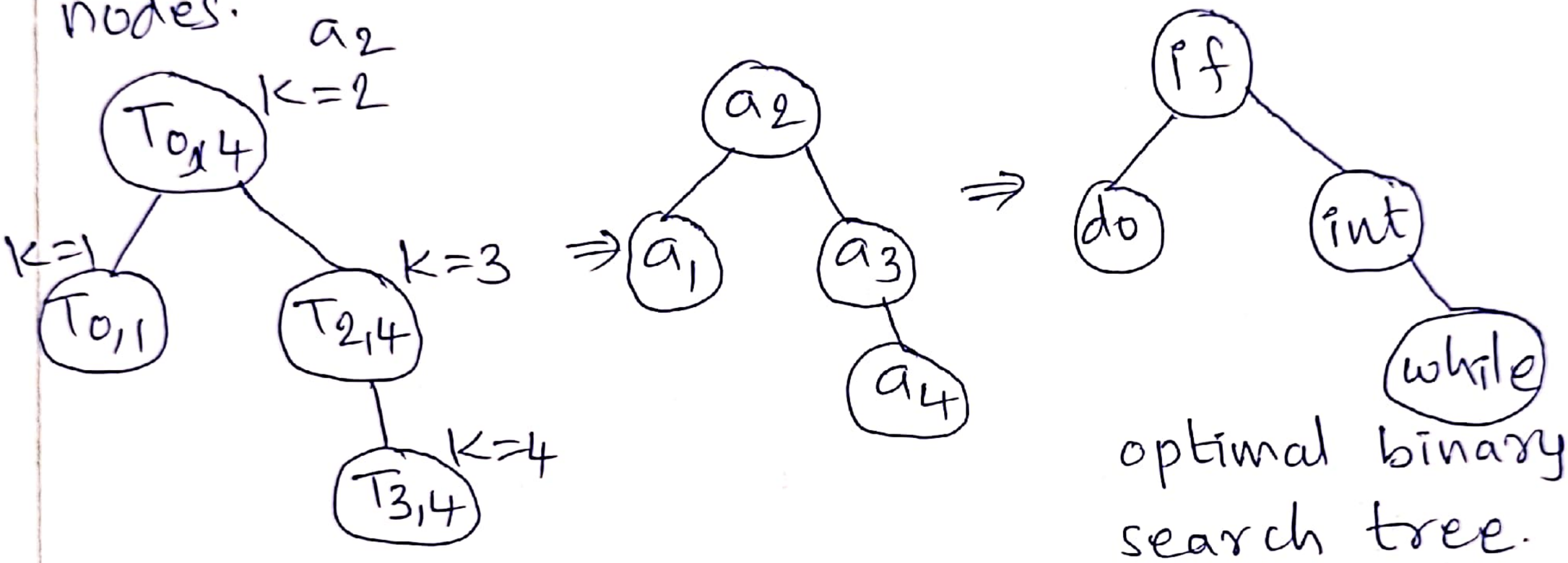
$T_{2,4}$ is divided into $T_{2,2}$ & $T_{3,4}$.

$r_{34}=4 \Rightarrow T_{3,4}$ is divided into $T_{3,3}$ & $T_{4,4}$ ($k=4$) (a_4)



Since $r_{00}, r_{11}, r_{22}, r_{33}$ and r_{44} is 0

These are external nodes, we can neglect these nodes.



optimal binary search tree.

cost of optimal binary search tree is 32

$$cost(T) = \left[\frac{3}{16}\times 2 + \frac{3}{16}\times 1 + \frac{1}{16}\times 2 + \frac{1}{16}\times 3\right] + \left[\frac{2}{16}\times 2 + \frac{3}{16}\times 2 + \frac{1}{16}\times 2 + 1/16\times 3 + \frac{1}{16}\times 3\right]$$

$$cost(T) = \frac{1}{16}[6 + 3 + 2 + 3 + 4 + 6 + 2 + 3 + 3]$$

$$cost(T) = \frac{1}{16}\times 32.$$

Eg consider four elements $(a_1, a_2, a_3, a_4) = (do, if, int, while)$, with $P(a_1) = 1/4$, $P(a_2) = 1/8$, $P(a_3) = P(a_4) = 1/16$ and $q(E_0) = 1/8$, $q(E_1) = 3/16$, $q(E_2) = q(E_3) = 1/16$. construct an optimal binary search tree

Sol:- $n = 4$ elements given

$(a_1, a_2, a_3, a_4) = (do, if, int, while)$.

Highest value of denominator in probabilities is 16. To get $p(a_i)$ and $p(E_i)$ values multiply with 16.

$p(a_1) = 1/4 \times 16 = 4$

$p(a_2) = 1/8 \times 16 = 2$

$p(a_3) = p(a_4) = 1/16 \times 16 = 1$

$q(E_0) = 1/8 \times 16 = 2$

$q(E_1) = 3/16 \times 16 = 3$

$q(E_2) = q(E_3) = 1/16 \times 16 = 1$

$q(E_4) = 1/16 \times 16 = 1.$

$\therefore P(1:4) = (4, 2, 1, 1)$ $q(0:4) = (2, 3, 1, 1, 1)$

solve the problem similar to the previous problem.

Algorithm for finding a minimum-cost binary search tree

```
Algorithm OBST(p, q, n)
// Given n distinct identifiers a1 < a2 < ... < an
// and probabilities p[i], 1 ≤ i ≤ n, and q[i],
// 0 ≤ i ≤ n, this algorithm computes the
// cost c[i, j] of optimal binary search
```

```
//trees t_ij for identifiers a_{i+1}, ..., a_j. It also
// computes r[i,j], the root of t_ij.
// w[i,j] is the 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;
   // optimal trees 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:= Find(c,r,i,j);
      // A value of l in the range r[i,j-1] ≤ l ≤ r[i+1,j].
      // that minimizes c[i,l-1]+c[l,j];
      c[i,j]:=w[i,j]+c[i,k-1]+c[k,j];
      r[i,j]:=k;
```

```
}
write(c[0,n], w[0,n], x[0,n]);
}
Algorithm  Find(c, r, i, j)
{
  min:=∞;
 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];
        d:=m;
    }
 return d;
}
```

The total time to evaluate all the c(i,j)'s and r(i,j)'s is $O(n^3)$.

0/1 KNAPSACK PROBLEM

knapsack means a bag.

$x_p = 0$ or 1 but not a fraction

If enough space is there in the knapsack we place the next objed completely, otherwise we don't place., but in this 0/1 knapsack problem objects cannot be divided into fractions.

$x_i = 0$ means we cannot place next object (ornament) in the bag.

$x_i = 1$ means we can place next ornament completely in the bag.

This problem contains either 0 or 1, that's why this problem is called as 0/1 knapsack problem. In this we can't place (divide) fractional weights in the knapsack.

Formula for calculating profit and weight

$$S_1^i = \{(P_i, W_i) \mid (P - P_i, W - W_i) \in S^{i-1}\}$$

where P - Total profit, W - Total weight.

If we try to remove the ith item then the profit of ith item will be reduced from the total profit and weight of ith item is removed from the total weight which belongs to the profit and weight of $(i-1)^{th}$ item.

$S_0^0 = \{0, 0\}$

$S_1^i = S^{i-1} + (P_i, W_i)$ addition of next item i

$S^i = S^{i-1} \cup S_1^i$ (merging operation)

Eg solve the following instance of 0/1 knapsack problem by using dynamic programming.

Given $n=3$ objects (items) with knapsack capacity $M=6kg$ and

profits $(P_1, P_2, P_3) = (1, 2, 5)$

weight $(w_1, w_2, w_3) = (2, 3, 4)$.

sol:- $S_0^0 = \{(0,0)\}$ initially no objects placed in the knapsack total profit = 0, weight = 0.

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

After placing object 1 in the knapsack.

$\rightarrow S_1^1 = S^0 + (P_1, w_1)$

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

$= \{(1,2)\}$.

$S^1 = S^0 \cup S_1^1$ $\quad S^i = S^{i-1} \cup S_1^i$

$= \{(0,0)\} \cup \{(1,2)\}$

$= \{(0,0), (1,2)\}$.

$\rightarrow S_1^2 = S^1 + (P_2, W_2)$

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

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

$S^2 = S^1 \cup S_1^2$

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

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

$\rightarrow S_1^3 = S^2 + (P_3, W_3)$

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

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

$S^3 = S^2 \cup S_1^3$

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

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

(3,5) and (5,4) are dominated pairs; (7,7) and (8,9) exceed the knapsack capacity.

Total weight exceeding knapsack capacity $M = 6$ kg. Discard these pairs.

(3,5), (5,4)

This pair giving less profit with more weight, by using purging (dominance) rule discard this solution (profit, wt) pair.

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

Purging Rule (Dominance Rule) If S^{i-1} has a pair (P_j, w_j) and S^i has a pair (P_k, w_k) and $P_j \leq P_k$ and $w_j \geq w_k$ then the

less profit — more weight

pair (P_j, w_j) is discarded from S^{i-1}

From S^2 we can remove (3,5)

After applying Purge rule, we will check the following condition inorder to find solution

→ If $(P_i, w_i) \in S^n$ and $(P_i, w_i) \notin S^{n-1}$ then

$x_n = 1$

otherwise $x_n = 0$.

$M = 6$.

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

$\therefore x_3 = 1$ → $Item_3$ is placed completely in knapsack

subtract (P_3, w_3) from $(6,6)$

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

$(1,2) \in S^2$ but $(1,2) \notin S^1$ also

$\therefore x_2 = 0$ item$_2$ cannot be placed in bag.

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

$\therefore x_1 = 1 \rightarrow$ item$_1$ is entirely placed in bag.

$\therefore x_1 = 1,\ x_2 = 0,\ x_3 = 1$

Total profit obtained $= \sum P_i x_i = P_1 x_1 + P_2 x_2 + P_3 x_3$

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

$= 1 + 0 + 5$

$= 6.$

Total weight of items placed in knapsack

$\sum w_i x_i = w_1 x_1 + w_2 x_2 + w_3 x_3$

$= 2 \times 1 + 3 \times 0 + 4 \times 1$

$= 2 + 0 + 4$

$= 6.$

$\therefore$ optimal (maximum profit giving) solution tuple $(x_1, x_2, x_3) = (1, 0, 1)$.

DKP — Dynamic knapsack

Algorithm DKP(p, w, n, m)

{

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

for i = 1 to n-1 do

{

$S_1^{i-1} = \{(P,W) \mid (P-P_i, W-w_i) \in S^{i-1}$ and $W \leq m\}$;

S^i = MergePurge(S^{i-1}, S_1^{i-1});

}

(P_x, W_x) = last pair in S^{n-1};

$(P_y, W_y) = (P'+P_n, W'+w_n)$ where W' is the largest W in any pair in S^{n-1} such that $W+w_n \leq m$;

// Trace back for $x_n, x_{n-1}, \ldots, x_1$

if ($P_x > P_y$) then $x_n = 0$;

else $x_n = 1$;

TraceBackFor($x_{n-1}, x_{n-2}, \ldots, x_1$);

}

The time needed to compute all the S^i's

$$T(n) = O(2^n)$$

where n is the no. of objects given.

ALL PAIRS SHORTEST PATHS PROBLEM

Let $G=(V,E)$ be a directed graph consisting of n vertices. The problem is to find the shortest path between all pairs of vertices.

Determine a matrix A such that $A(i,j)$ is the length of shortest path from vertex i to vertex j.

Assume that this path contains no cycles

If k is an intermediate vertex on this path, then the subpaths from i to k and from k to j are the shortest paths from i to k and k to j respectively otherwise the path i to j is not shortest path.

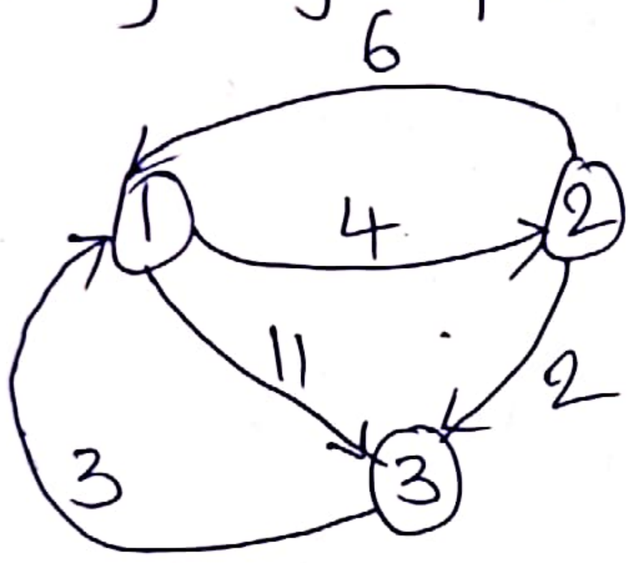
If k is an intermediate vertex with highest index, then the path i to k is the shortest path going through no vertex with index greater than k-1. similarly the path k to j is shortest path going through no vertex with index greater than k-1.

The shortest path can be computed by using the following recursive method.

$$A^k(i,j) = c(i,j) \quad \text{if } k=0$$
$$= \min\{A^{k-1}(i,j), A^{k-1}(i,k) + A^{k-1}(k,j)\}, \text{ if } k \geq 1.$$

$A^k(i,j)$ → shortest path from i to j going through k.

Eg Find all pairs shortest paths for the following graph by using dynamic programming



Sol:- cost adjacency matrix

$$A^0(i,j) = c(i,j) = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 4 & 11 \\ 6 & 0 & 2 \\ 3 & \infty & 0 \end{bmatrix} \end{array}$$

There is no edge from 3 to 2.

Take $c(3,2) = \infty$

So while calculating minimum (shortest path) it will be discarded.

step1: For $k=1$ i.e going from i to j through the intermediate vertex $k=1$ ($\because k-1=0$).

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

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

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

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

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

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

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

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

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

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

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

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

$$\therefore A^1(i,j) = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 4 & 11 \\ 6 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix} \end{array}$$

step2: For $k=2$ i.e going from i to j through the intermediate vertex $k=2$

$k-1 = 2-1 = 1$

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

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

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

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

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

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

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

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

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

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

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

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

$$\therefore A^2(i,j) = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 4 & 6 \\ 6 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix} \end{array}$$

Step 3: For $k=3$, i.e, going from i to j through the intermediate vertex $k=3$ $(k-1=2)$.

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

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

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

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

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

$A^3(2,1) = \min\{A^2(2,1), A^2(2,3)+A^2(3,1)\} = \min\{6, 2+3\} = 5$

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

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

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

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

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

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

$$\therefore A^3(i,j) = \begin{matrix} \\ 1 \\ 2 \\ 3 \end{matrix}\begin{matrix} \begin{matrix} 1 & 2 & 3 \end{matrix} \\ \begin{bmatrix} 0 & 4 & 6 \\ 5 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix} \end{matrix} \quad \begin{bmatrix} 0,\ 1\to2,\ 1\to2\to3 \\ 2\to3\to1,\ 0,\ 2\to3 \\ 3\to1,\ 3\to1\to2,\ 0 \end{bmatrix}$$

Algorithm AllPaths(cost, A, n) Floyd-warshall algorithm

// 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 $1 \le i \le n$

```
{
  for i:=1 to n do
      for j:=1 to n do
          A[i,j]:=cost[i,j]; // copy cost into A^0(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]);

}
```

Time complexity $T(n) = n^2 + n^3$

$T(n) = O(n^3)$.

TRAVELLING SALES PERSON PROBLEM (TSP)

Here the sales person starts at a place and visits all other places exactly once and comes back to the starting point. The problem is to find a shortest route connecting all the places

Applications Suppose we have to route a postal van to pick up mail from mail boxes located at n different sites. An n+1 vertex graph can be used to represent this situation. One vertex represents the post office from which the postal van starts and to which it must return. The route taken by the postal van is a tour and it should have minimum cost (length).

The following formulas are used for solving this problem.

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

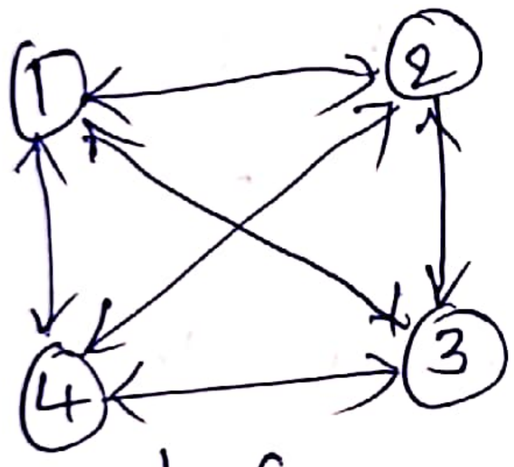
2) $g(i, S) = \min_{j \in S} \{ c_{ij} + g(j, S - \{j\}) \}$.

S → set of remaining vertices to be visited.

$g(i,S)$ means i is starting node and the remaining nodes in S are to be visited.

$\min_{j \in S}$ is considered as the intermediate node, $g(j, S-\{j\})$ means j is already visited so next we have to traverse nodes in $S-\{j\}$ with j as starting point.

Eg Solve the following instance of travelling sales person problem by using dynamic programming.



Graph G

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 10 & 15 & 20 \\ 5 & 0 & 9 & 10 \\ 6 & 13 & 0 & 12 \\ 8 & 8 & 9 & 0 \end{bmatrix} \end{array}$$

cost adjacency matrix c.

Assume that sales person starts at vertex 1.

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

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

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

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

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

The sales person starts at vertex 1
Next he can visit vertex 2 or vertex 3 or vertex 4.
So we have to find

$$g(1,\{2,3,4\}) = \min_{j \in S} \begin{Bmatrix} c_{12} + g(2,\{3,4\}), \\ c_{13} + g(3,\{2,4\}), \\ c_{14} + g(4,\{2,3\}) \end{Bmatrix}$$

$$\rightarrow \boxed{g(i,s) = \min_{j \in S} \{c_{ij} + g(j, s-\{j\})\}} \rightarrow \text{cost function}$$

$|s| = 1$

The salesperson starts at vertex 1, from that he can move to vertex 2. If he visits vertex 2 from that vertex he can next visit either vertex 3 or vertex 4.

$1 \rightarrow 2 \rightarrow 3$ $g(2,3) = \min_{j \in S} \{c_{23} + g(3,\phi)\} = 9+6 = 15$ $2 \rightarrow 3 \rightarrow 1$

$1 \rightarrow 2 \rightarrow 4$ $g(2,4) = \min_{j \in S} \{c_{24} + g(4,\phi)\} = 10+8 = 18$ $2 \rightarrow 4 \rightarrow 1$

$1 \rightarrow 3 \rightarrow 2$ $g(3,2) = \min_{j \in S} \{c_{32} + g(2,\phi)\} = 13+5 = 18$ $3 \rightarrow 2 \rightarrow 1$

$1 \rightarrow 3 \rightarrow 4$ $g(3,4) = \min_{j \in S} \{c_{34} + g(4,\phi)\} = 12+8 = 20$ $3 \rightarrow 4 \rightarrow 1$

$1 \rightarrow 4 \rightarrow 2$ $g(4,2) = \min_{j \in S} \{c_{42} + g(2,\phi)\} = 8+5 = 13$ $4 \rightarrow 2 \rightarrow 1$

$1 \rightarrow 4 \rightarrow 3$ $g(4,3) = \min_{j \in S} \{c_{43} + g(3,\phi)\} = 9+6 = 15$ $4 \rightarrow 3 \rightarrow 1$

$|S| = 2$ → $|S|$ set of remaining two vertices

$1 \to 2 \to \{3,4\}$
$1 \to 3 \to \{2,4\}$
$1 \to 4 \to \{2,3\}$

$g(2,\{3,4\}) = \min_{j \in S} \{ c_{23} + g(3,4), c_{24} + g(4,3) \}$
$i \quad S$; $j=3$; $j=4$ ✓ ; min

$= \min\{9+20, 10+15\}$ ✓

$= 25$

$g(3,\{2,4\}) = \min_{j \in S} \{ c_{32} + g(2,4), c_{34} + g(4,2) \}$
$j=2$; $j=4$ ✓ ; min

$= \min\{13+18, 12+13\}$ ✓

$= 25$

$g(4,\{2,3\}) = \min_{j \in S} \{ c_{42} + g(2,3), c_{43} + g(3,2) \}$
$j=2$ ✓ ; $j=3$

$= \min\{8+15, 9+18\}$ ✓

$= 23$

→ $|S| = 3$

$g(1,\{2,3,4\}) = \min_{j \in S} \{ c_{12} + g(2,\{3,4\}), c_{13} + g(3,\{2,4\}), c_{14} + g(4,\{2,3\}) \}$
$j=2$ ✓ ; $j=3$; $j=4$

$1 \to 2 \to 4 \to 3 \to 1$
$1 \to 3 \to 4 \to 2 \to 1$
$1 \to 4 \to 2 \to 3 \to 1$

$= \min\{10+25, 15+25, 20+23\}$ ✓

$= \min\{35, 40, 43\}$ ✓

$= 35$ km. length of shortest path
optimal tour of the graph

cost function $g(i,S)$ is the length of the optimal salesperson tour

$J(i,S)$ is the value of intermediate vertex j that minimizes $g(i,S)$.

$J(1,\{2,3,4\}) = 2$. Thus the tour starts from 1 and goes to 2 $\underline{1 \to 2}$

$J(2,\{3,4\}) = 4$. Thus the next edge is $\underline{2 \to 4}$

$J(4,\{3\}) = 3$. Next edge of the tour is (shortest path) is $\underline{4 \to 3}$

All the vertices visited. and the salesperson returns back to starting vertex 1 $3 \to 1$

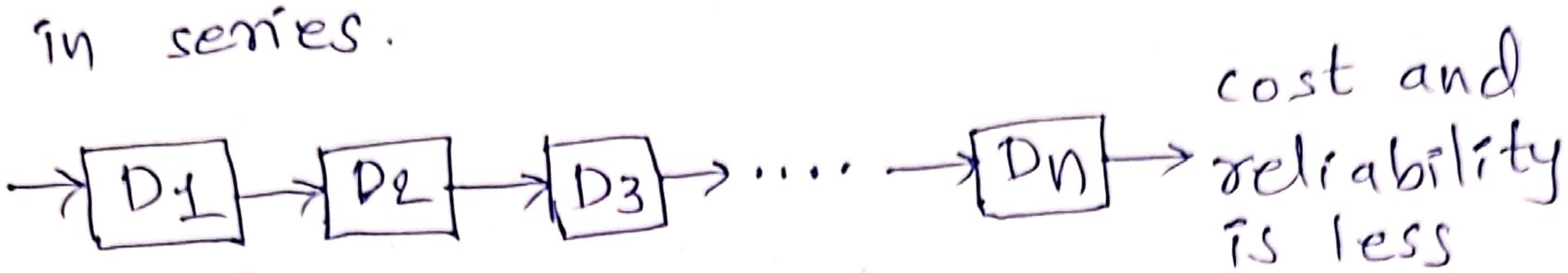
$\therefore$ optimal (shortes minimum length) tour is

$$\boxed{1 \to 2 \to 4 \to 3 \to 1}$$

$10+10+9+6 = 35$ km length or cost of the optimal tour.

RELIABILITY DESIGN PROBLEM

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



n devices D_i, $1 \le i \le n$ connected in series

Let r_i be the reliability of device D_i is r_i

Reliability of a device is the probability that the device will function properly

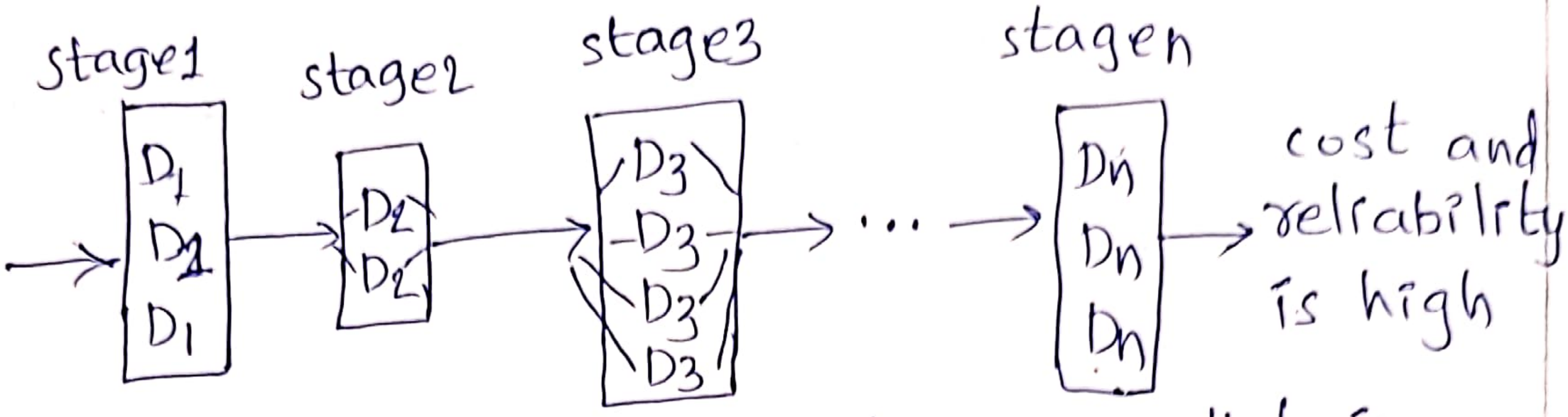
The reliability of the entire system is

$$\Pi r_i = r_1 \times r_2 \times \cdots \times r_n$$

In this if one device fails, then the entire system will fail.

→ Our problem is to use device duplication to maximize reliability. Multiple copies of the same device type are connected in parallel through the use of switching circuits. The switching circuits determine which devices in any given group are functioning properly. They then make use.

of one such device at each stage.



Multiple devices connected in parallel in each stage

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

$$\boxed{1-(1-r_i)^{m_i}} \rightarrow \text{probability that atleast one device function properly at stage i.}$$

If $r_p = 0.99$ and $m_i = 2$, the stage reliability becomes $1-(1-0.99)^2 = 1-(0.01)^2 = 1-0.0001 = 0.9999$

Let us assume that the reliability of stage i is given by a function

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

Our problem is to use device duplication to maximize reliability

The maximization is to be carried out under a cost constraint.

Let c_i be the cost of each unit of device D_i

Let C be the maximum allowable cost of the system. being designed.

We wish to solve the following maximization problem

$$\text{maximize} \prod_{1 \le i \le n} \phi_i(m_i) \quad \text{(reliability function)}$$

$$\text{subject to} \sum_{1 \le i \le n} c_i m_i \le C \quad \text{(total allowable cost)}$$

$$m_i \ge 1, \text{ for } 1 \le i \le n.$$

each $c_i > 0$

Each m_i must be in the range $1 \le m_i \le u_i$

where

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

upper bound on the no. of devices that can be placed at each stage i.

(connected)

Eg We are to design a three stage system with device types D_1, D_2, D_3. The costs are \$30, \$15, \$20 respectively. The cost of the system is to be no more than \$105. The reliability of each device type is 0.9, 0.8, 0.5 respectively

Sol:- Given data

$c_1 = 30,\ c_2 = 15,\ c_3 = 20,\ C = 105$

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

$$U_i = \text{upper bound} = \left\lfloor \left(C + c_i - \sum_{j=1}^{n} c_j \right) / c_i \right\rfloor$$

$$U_1 = \lfloor [105 + 30 - (30 + 15 + 20)] / 30 \rfloor = \lfloor (135 - 65)/30 \rfloor = \lfloor 2.333 \rfloor = 2$$

$$U_2 = \lfloor [105 + 15 - (30 + 15 + 20)] / 15 \rfloor = \lfloor (120 - 65)/15 \rfloor = \lfloor 3.666 \rfloor = 3.$$

$$U_3 = \lfloor [105 + 20 - (30 + 15 + 20)] / 20 \rfloor = \lfloor (125 - 65)/20 \rfloor = \lfloor 3 \rfloor = 3.$$

$\therefore U_1 = 2,\ U_2 = 3,\ U_3 = 3.$

Reliability function $\boxed{\phi_i(m_i) = 1 - (1 - r_i)^{m_i}}$

Each pair in the solution is represented by $\boxed{(r, x)}$ where r is reliability, x is cost.

Using S_j^i to represent all tuples obtainable from S^{i-1} by choosing $\boxed{m_i = j}$ $i \rightarrow$ stage no. $j \rightarrow$ no. of devices

Beginning with $S^0 = \{(1,0)\}$. $\rightarrow \{(r,x)\}$

reliability (r) values will be multiplied and cost (x) values will be added one by one.

stage 1

since $1 \le m_i \le u_i$ i.e $1 \le m_1 \le 2$ $u_1 = 2$

At stage 1 we can connect 1 or 2 devices D_1.

$m_1 = 1$ 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 $m_1 = 1$

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

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

In each tuple (r,x) reliability will be multiplied with previous reliability and cost will be added to previous cost.

$S_1^1 = \{(1\times0.9, 0+30)\} = \{(0.9, 30)\}$ $\rightarrow$ one D_1 at stage 1 solution.

If two devices are connected at stage 1, then

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

$\phi_1(m_1) = 1-(1-r_1)^2 = 1-(1-0.9)^2 = 1-0.01 = 0.99$

$S_2^1 = \{(1\times 0.99,\ 0+2c_1)\} = \{(0.99,\ 0+2\times 30)\} = \{(0.99, 60)\}$.

$\underline{s^1}$ can be obtained by merging the sets

S_1^1, S_2^1 $\boxed{s^1 = S_1^1 \cup S_2^1}$

$s^1 = \{(0.9, 30), (0.99, 60)\}$.

$\underline{\text{At stage 2}}$

Similarly S_1^2, S_2^2, S_3^2 can be calculated from s^1 as follows

As $u_2=3, \Rightarrow m_2=1$ or 2 or 3

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

$\phi_2(m_2) = 1-(1-r_2)^{m_2} = 1-(1-0.8)^1 = 1-0.2 = 0.8$

$1c_2 = 1\times \$15 = \$15$

$S_1^2 = \{(0.9\times 0.8,\ 30+15),\ (0.99\times 0.8,\ 60+15)\}$ ← from s^1

$S_1^2 = \{(0.72, 45), (0.792, 75)\}$

$\underline{S_2^2}$ $i=2, j=2$. i.e $m_2=2$ $r_2=0.8$

$\phi_2(m_2) = 1-(1-r_2)^{m_2} = 1-(1-0.8)^2 = 1-(0.2)^2 = 0.96$

$2c_2 = 2\times 15 = \$30$.

$S_2^2 = \{(0.9\times0.96, 30+30), (0.99\times0.96, 60+30)\}$

$S_2^2 = \{(0.864, 60), (0.9504, 90)\}$

S_3^2 $i=2, j=3$ i.e $m_2=3$

$\phi_2(m_2) = 1-(1-r_2)^{m_2} = 1-(1-0.8)^3 = 1-(0.2)^3 = 0.992$

$j=3$, $3C_2 = 3\times15 = \$45$

$S_3^2 = \{(0.9\times0.992, 30+45), (0.99\times0.992, 60+45)\}$

$S_3^2 = \{(0.8928, 75), (0.98208, 105)\}$

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

$S^2 = \{(0.72, 45), (0.7923, 75), (0.864, 60), (0.9504, 90),$
$(0.8928, 75), (0.98208, 105)\}$.

Removed by purging rule

cost is in increasing order

$(0.9504, 90)$ is removed from S^2 since it leaves only \$15 which is not enough to allow atleast one device at stage 3 $m_3=(C_3=20)$.

$(0.98208, 105)$ will be removed from S^2 since cost $C=105$, we can't add m_3.

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

At stage 3

As $u_3 = 3$, $m_3 = 1$ or 2 or 3

s_1^3, s_2^3, s_3^3 can be calculated from s^2 as follows.

$\boxed{s_j^r \rightarrow m_i = j}$

$s_1^3 \rightarrow m_3 = 1 \rightarrow {}^1C_3 = 1 \times 20 = \$20$

$i = 3, j = 1 \quad r_3 = 0.5$

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

from s^2

$s_1^3 = \{(0.72 \times 0.5, 45 + 20), (0.864 \times 0.5, 60 + 20),$

$(0.8928 \times 0.5, 75 + 20)\}.$

$s_1^3 = \{(0.36, 65), (0.432, 80), (0.44 64, 95)\}.$

$s_2^3 \rightarrow m_3 = 2 \quad i = 3, j = 2$

$\downarrow$

${}^2C_3 = 2 \times 20 = \$40$

$\phi_3(m_3) = 1 - (1 - r_3)^{m_3} = 1 - (1 - 0.5)^2 = 1 - 0.25 = 0.75$

$s_2^3 = \{(0.72 \times 0.75, 45 + 40), (0.864 \times 0.75, 60 + 40),$

$(0.8928 \times 0.75, 75 + 40)\}$

$\downarrow$ 115 $\times$ $\geq$ 105 allowable cost.

$s_2^3 = \{(0.54, 85), (0.648, 100)\}.$

$s_3^3 \quad m_3 = 3$

$\phi_3(m_3) = 1 - (1 - r_3)^{m_3} = 1 - (1 - 0.5)^3 = 1 - 0.125 = 0.875$

$s_3^3 = \{(0.72 \times 0.875, 45 + 60)\} = \{(0.03, 105)\}.$

$S^3 = S_1^3 \cup S_2^3 \cup S_3^3$

$S^3 = \{(0.36, 65), (0.432, 80), (0.4464, 95), (0.54, 85),$
(less reliability, more cost)
$(0.648, 100), (0.63, 105)\}$.

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

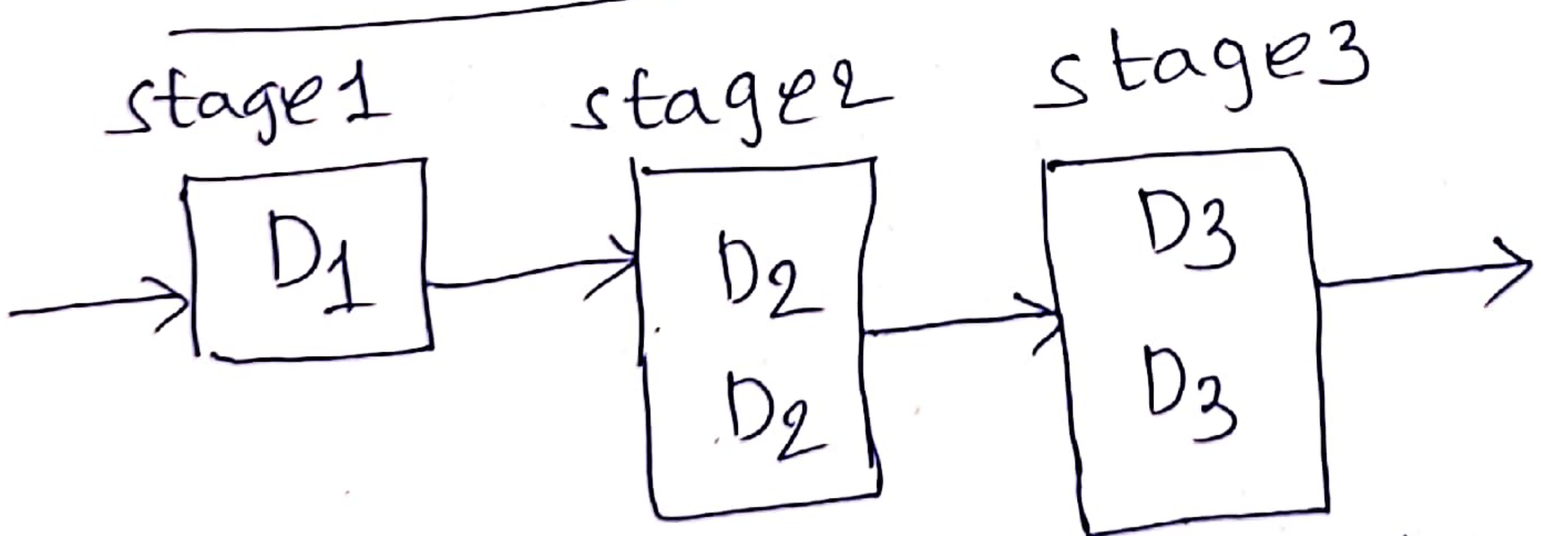
The best system design is reliability 0.648 and cost 100

$(0.648, 100)$ pair is present in $S_2^3 \Rightarrow i=3$ and $j=2$ so $m_3=2$

The pair $(0.648, 100)$ is obtained from $(0.864, 60)$ which is present in $S_2^2 \Rightarrow i=2, j=2$ so $m_2=2$

$(0.864, 60)$ pair is obtained from $(09, 30)$ which is in $S_1^1 \Rightarrow i=1, j=1 \rightarrow m_1=1$

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



$\$30 + 2 \times 15 = \$30 + 2 \times 20 = 40$.

$30 + 30 + 40 = \$100 \rightarrow$ cost of the whole system.

Reliability of the system $= 0.648$

In 64% of the times system will function properly.