Design And Analysis Of Algorithms

ALVIRA
160321733013

Assignment - II

Set - I

Answer the following Questions:

1. Using divide and conquer write a recursive algorithm for finding both the minimum and maximum elements in an array A of n elements. What is the run time

Ans: Let $P = (n, a[i], \ldots, a[j])$ denote an arbitary instance. Here n is the number of elements in the list $a[i], \ldots, a[j]$ from where the maximum and minimum element is to be found. Let small(P) be true when $n \leq 2$.

→ The maximum and minimum are $a[i]$ if $n = 1$

→ If $n = 2$, the problem can be solved by making one comparision.

→ If $n > 2$, the list has more than two elements, P has to be divided into smaller instances.

$P_1 = (\lfloor n/2 \rfloor, a[1], \ldots a[\lfloor n/2 \rfloor])$ and

$P_2 = (n - \lfloor n/2 \rfloor, a[\lfloor n/2 \rfloor + 1], \ldots, a[n])$.

After having divided P into two smaller sub problems, they can be solved by recursively invoking the same divide-and-conquer algorithm

The solutions for P_1 and P_2 can be combined to obtain the solution of P. If MAX(P) and MIN(P) are the maximum and minimum of the elements in P, then MAX(P) is the larger of MAX(P_1) and MAX(P_2). Also, MIN(P) is the smaller of MIN(P_1) and MIN(P_2).

```
1 Algorithm MaxMin (i, j, max, min)
2 // a[1:n] is a global array. Parameters i and
    j are integers.
3 // 1 ≤ i ≤ j ≤ n. The effect is to set max and min to
4 // the largest and smallest values in a[i:j],
    respectively.
5 {
6 if (i = j) then max := min := a[i]; // small (P)
```

```
7  else if (i=j-1) then // Another case of small(P)
8. {
9  if (a[i] < a[j]) then
10 {
11 max := a[j]; min := a[i];
12 }
13 else
14 {
15 max := a[i]; min := a[j];
16 }
17 }
18 else
19 {// If P is not small, divide P into subproblems
20 // Find where to split the set
21 mid := ⌊(i+j)/2⌋;
22 // solve the subproblems
23 MaxMin (i, mid, max, min);
24 MaxMin (mid+1, j, max1, min1);
25 // combine the solutions
26 if (max < max1) then max := max1;
27 if (min > min1) then min := min1;
28 }
29 }
```

In the analysis of algorithm, $T(n)$ represents the number of element comparisions needed, then the resulting recurrence relation is

$$T(n) = \begin{cases} 0 & n=1 \\ 1 & n=2 \\ T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil) + 2 & n>2 \end{cases}$$

when n is a power of two, $n = 2^k$ for some positive integer k, then

$$T(n) = T(n/2) + T(n/2) + 2$$

$$= 2T(n/2) + 2$$

$$= 2(2T(n/4) + 2) + 2$$

$$= 2^2 T(n/2^2) + 2^2 + 2$$

$$= 2^2(2T(n/8) + 2) + 2^2 + 2^1$$

$$= 2^3 T(n/2^3) + 2^3 + 2^2 + 2^1$$

$$= 2^{k-1} T\left(n/2^{k-1}\right) + 2^{k-1} + 2^{k-2} + \cdots 2^1$$

$$= 2^{k-1} T\left(\frac{2^k}{2^{k-1}}\right) + 2^{k-1} + 2^{k-2} + \cdots 2^1 \quad [\because n = 2^k]$$

$$= 2^{k-1} T(2) + \left(2^{k-1} + 2^{k-2} + \cdots 2\right)$$

$$= 2^{k-1} T(2) + \frac{2^{k+1+1} - 1}{2-1}$$

$$\left[\because x^{n-1} + x^{n-2} + \cdots 1 = \frac{x^{n+1}-1}{(x-1)}\right]$$

$$= 2^{k-1} T(2) + 2^k - 1 \quad [\because T(2) = 1]$$

$$= \frac{2^k}{2} + 2^k - 1 \quad [\because 2^k = n]$$

$$= \frac{n}{2} + n - 1$$

$$T(n) = \frac{n + 2n - 2}{2} = \frac{3n-2}{2}$$

Hence $\frac{3n-2}{2}$ is the best, avg and worst case number of comparisions when n is a power of two.

2 Briefly explain the Brute force knapsack problem with an example

Ans: 'Brute Force' is a straight forward approach to solving a problem, usually directly based on the problem's statement and definitions of the concepts involved.

The "force" implied by the strategy's definition is that of a computer.

Example:

Compute a^n for a given number 'a' and a non negative integer n. By the definition of exponentiation,

$$a^n = \underbrace{a * \ldots * a}_{n \text{ times}}$$

This suggests computing a^n by multiplying 'a' 'n' times.

Brute-Force strategy is applicable to a very wide variety of problems. It is used for many elementary but important algorithmic tasks such as computing the sum of 'n' numbers, finding the largest element in the list. For some problems such as sorting, searching, matrix multiplication, string matching, the brute-force approach yields reasonable algorithms of at least some practical value with no limitation on instance size.

The expense of designing a more efficient algorithm may be unjustifiable if only a few

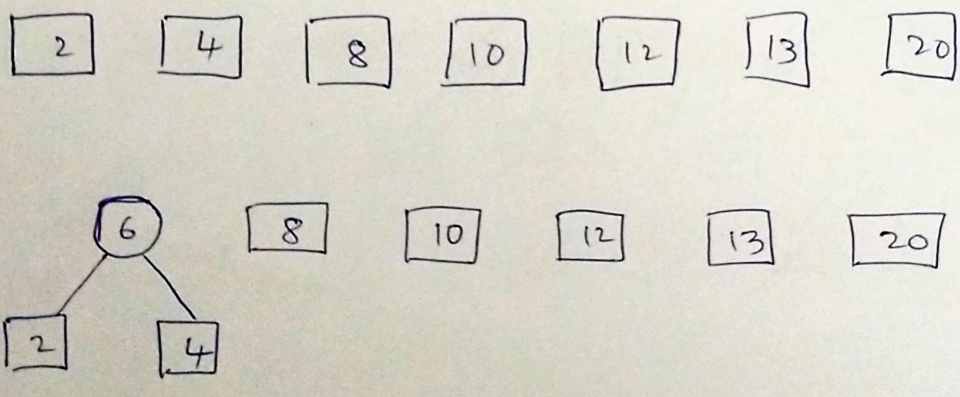
instances of a problem need to be solved (7)
and a brute-force algorithm can solve those instances with acceptable speed.

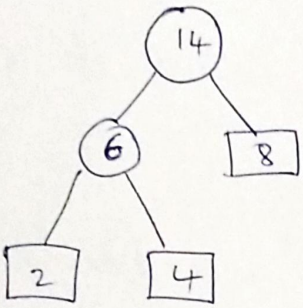
Even if too inefficient in general, a brute-force algorithm can still be useful for solving small-size instances of a problem.

Finally, a brute-force algorithm can serve an important theoretical or educational purpose as a yardstick with which to judge more efficient alternatives for solving a problem.

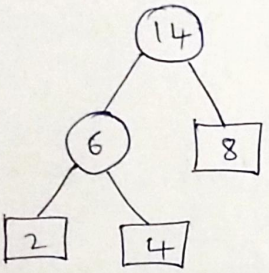
3. Find an optimal binary merge pattern for files whose lengths are: 2, 4, 8, 10, 12, 13 and 20

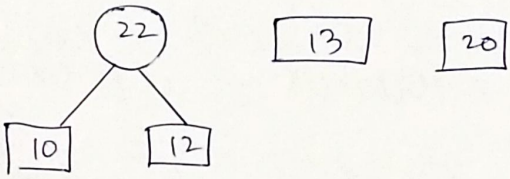
Ans:

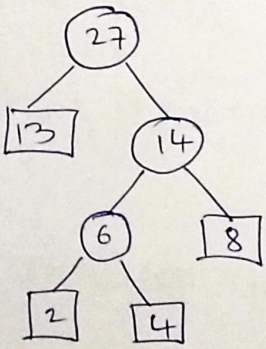


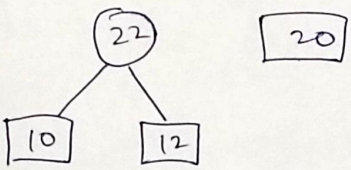

14
6
8
2
4

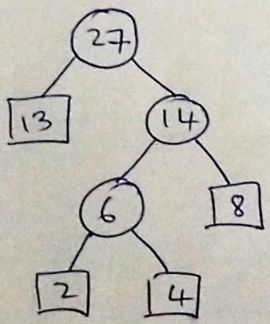

10
12
13
20

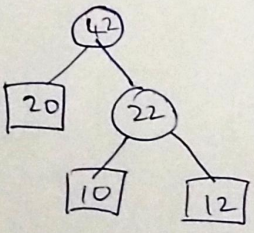

14
6
8
2
4

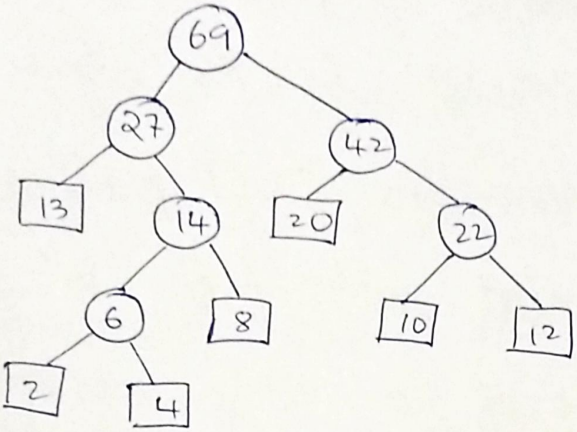

22
10
12
13
20


27
13
14
6
8
2
4


22
10
12
20

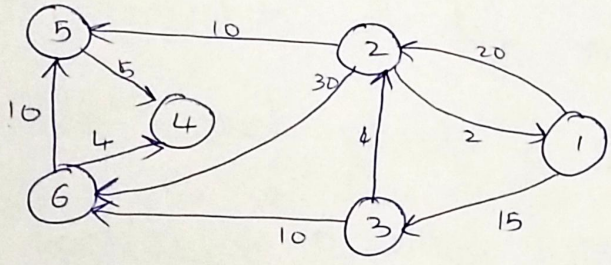

27
13
14
6
8
2
4


42
20
22
10
12



This is the required optimal binary merge pattern

4. Write in detail with an example the "Single Source Shortest Path" / "Dijkstra's algorithm"



Ans: The Single Source Shortest Path are special cases of the path problem. The length of a path is defined to be the sum of the weights of the edges on that path. The starting vertex of the path is referred to as the 'source' and the last vertex the 'destination'. The graphs are digraphs to allow for one-way streets.

Given a directed graph $G = (V, E)$, with a weighting function 'cost' for the edges of G, and a source vertex 'V_0'. The SSSP problem is to determine the shortest paths from 'V_0' to all the remaining vertices of G. It is assumed that all the weights are positive. The shortest path between 'V_0' and some other node 'v' is an ordering among a subset of the edges. Hence, the SSSP problem fits the ordering paradigm

To formulate a greedy-based algorithm to generate the shortest paths, a multistage solution to the problem and an optimization measure should be conceived. One possibility is to build the shortest paths one by one and use the sum of the lengths of all paths so far generated as an optimization measure.

The greedy way to generate the shortest paths

from 'V_0' to the remaining vertices is to generate those paths in non decreasing order of path length. First, a shortest path to the nearest vertex is generated. Then a shortest path to the second nearest vertex is generated, and so on. In order to generate the shortest paths in this order, determine

→ The next vertex to which a shortest path must be generated

→ A shortest path to this vertex

Let 'S' denote the set of vertices to which the shortest paths have been generated. Let 'dist[w]' be the length of the shortest path starting from 'V_0', going through those vertices that are in 'S', and ending at 'w'. It is observed:

- If the next shortest path is to vertex 'u', then the path begins at 'V_0', ends at 'u', going

through the vertices that are in 's'.

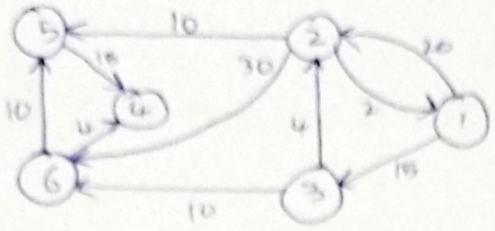
- The destination of the next path generated must be that of vertex 'u', which has the minimum distance, 'dist[u]', among all vertices not in 's'.

- Having selected a vertex 'u' and generated the shortest 'v₀' to 'u' path, vertex 'u' becomes a member of 's'. The length of, the shortest path starting at 'v₀', going through vertices in 's', ending at vertex 'w' may decrease, i.e, the value of 'dist[w]' may change.

The above observations leads to the algorithm for the SSSP problem.

This algorithm is known as 'Dijkstra's Algorithm'. It determines only the lengths of the shortest paths from 'v₀' to all other vertices in G.

Example:



Set S	Vertex Selected	1	2	3	4	5	6
–	–	0	∞	∞	∞	∞	∞
{}	1	0	20	15	∞	∞	∞
{1}	3	0	20	15	17	∞	25
{1,3}	2	0	20	15	17	50	25
{1,3,2}	4	0	20	15	17	50	21
{1,3,2,4}	6	0	20	15	17	31	21
{1,3,2,4,6}	5	0	20	15	17	31	21

Shortest paths from vertex 1 (increasing order)

1 → 3 : 15

1 → 4 : 17

1 → 2 : 20

1 → 6 : 21

1 → 5 : 31

5. For the identifier set (a_1, a_2, a_3, a_4) = (end, goto, print, stop) with $(P_1, P_2, P_3, P_4) = (1/20, 1/5, 1/10, 1/20)$ and $(q_0, q_1, q_2, q_3, q_4) = (1/5, 1/10, 1/5, 1/20, 1/20)$

Construct an OBST

Ans: Multiply values of p's and q's by 20

$p(1) = 1$, $p(2) = 4$, $p(3) = 2$, $p(4) = 1$

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

Initially $c(i,i) = 0$, $r(i,i) = 0$, $0 \le i \le 4$

$\therefore$ $c(0,0) = 0$ $\quad$ $r(0,0) = 0$

$c(1,1) = 0$ $\quad$ $r(1,1) = 0$

$c(2,2) = 0$ $\quad$ $r(2,2) = 0$

$c(3,3) = 0$ $\quad$ $r(3,3) = 0$

$c(4,4) = 0$ $\quad$ $r(4,4) = 0$

$w(i,i) = q(i)$

$w(0,0) = q(0) = 4$

$w(1,1) = q(1) = 2$

$w(2,2) = q(2) = 4$

$w(3,3) = q(3) = 1$

$w(4,4) = q(4) = 1$

	0	1	2	3	4
$j-i=0$	$w(0,0)=4$ $c(0,0)=0$ $r(0,0)=0$	$w(1,1)=2$ $c(1,1)=0$ $r(1,1)=0$	$w(2,2)=4$ $c(2,2)=0$ $r(2,2)=0$	$w(3,3)=1$ $c(3,3)=0$ $r(3,3)=0$	$w(4,4)=1$ $c(4,4)=0$ $r(4,4)=0$
$j-i=1$	$w(0,1)=7$ $c(0,1)=7$ $r(0,1)=1$	$w(1,2)=10$ $c(1,2)=10$ $r(1,2)=2$	$w(2,3)=7$ $c(2,3)=7$ $r(2,3)=3$	$w(3,4)=3$ $c(3,4)=3$ $r(3,4)=4$	
$j-i=2$	$w(0,2)=15$ $c(0,2)=22$ $r(0,2)=2$	$w(1,3)=13$ $c(1,3)=20$ $r(1,3)=2$	$w(2,4)=9$ $c(2,4)=12$ $r(2,4)=13$		
$j-i=3$	$w(0,3)=18$ $c(0,3)=32$ $r(0,3)=2$	$w(1,4)=15$ $c(1,4)=27$ $r(1,4)=2$			
$j-i=4$	$w(0,4)=20$ $c(0,4)=39$ $r(0,4)=2$				

∵ Now

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

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

$$r(i,j) = k \text{ where } i<k\le j$$

for $j-i=1$ $\quad T_{01}, T_{12}, T_{23}, T_{34}$

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

$$= 1+2+4$$

$$= 7$$

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

$$= \min \{0+0\} + 7$$

$$= 7$$

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

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

$$= 4 + 4 + 2$$

$$= 10$$

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

$$= \min \{0+0\} + 10$$

$$= 10$$

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

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

$$= 2 + 1 + 4$$

$$= 7$$

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

$$= \min \{0+0\} + 7$$

$$= 7$$

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

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

$$= 1 + 1 + 1$$

$$= 3$$

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

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

$$= 3$$

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

For $j-i=2$ T_{02} T_{13} T_{24}

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

$$= 4 + 4 + 7$$

$$= 15$$

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

$$+ w(0,2)$$

$$= \min \{ 0+10, 7+0 \} + 15$$

$$= \min \{ 10, 7 \} + 15$$

$$= 7 + 15 = 22$$

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

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

$= 2+1+10$

$= 13$

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

($k=2$ for $c(1,1)+c(2,3)$; $k=3$ for $c(1,2)+c(3,3)$)

$= \min \{0+7, 10+0\} + 13$

$= \min \{7, 10\} + 13$

$= 7+13 = 20$

$r(1,3) = 2$

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

$= 1+1+7$

$= 9$

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

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

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

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

$= 3+9 = 12$

$r(2,4) = 3$

For $j-i=3$ T_{03} T_{14}

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

$$= 2 + 1 + 15$$

$$= 18$$

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

$$= \min \{ 0+20, 7+7, 22+0 \} + 18$$

$$= \min \{ 20, 14, 22 \} + 18$$

$$= 14 + 18 = 32$$

$$r(0,3) = 2$$

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

$$= 1 + 1 + 13$$

$$= 15$$

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

$$= \min \{ 0+12, 10+3, 20+0 \} + 15$$

$$= \min \{ 12, 13, 20 \} + 15$$

$$= 12 + 15 = 27$$

$r(1,4) = 2$

For $j-i = 4$ $\quad T_{04}$

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

$$= 1 + 1 + 18$$

$$= 20$$

$$c(0,4) = \min_{0<k\le 4} \{ \underset{k=1}{c(0,0) + c(1,4)}, \underset{k=2}{c(0,1) + c(2,4)},$$

$$\underset{k=3}{c(0,2) + c(3,4)}, \underset{k=4}{c(0,3) + c(4,4)} \}$$

$$+ w(0,4)$$

$$= \min \{ 0+27, 7+12, 22+3, 32+0 \} + 20$$

$$= \min \{ 27, 19, 25, 32 \} + 20$$

$$= 19 + 20 = 39$$

$r(0,4) = 2$

To build the optimal Binary Search Tree

T_{04}; is the root node is: $r(0,4) = 2$

$\Rightarrow k = 2$

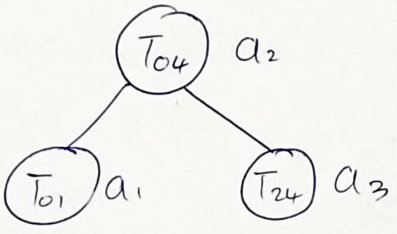
$\therefore a_2$ becomes the root node

$\because T_{ij} = T_{i,k-1} , T_{kj}$

T_{04} is (T_{01}, T_{24}) [$\because T_4$ is divided into 2 parts]

$T_{01} = r(0,1) = 1 \Rightarrow k=1 \Rightarrow a_1$

$T_{24} = r(2,4) = 3 \Rightarrow k=3 \Rightarrow a_3$



T_{01} is divided into T_{00}, T_{11} [$\because k=1$]

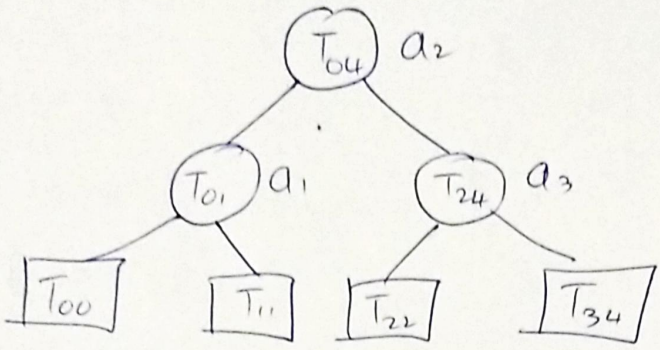
$T_{00} = r(0,0) = 0 \rightarrow$ external node

$T_{11} = r(1,1) = 0 \rightarrow$ external node

T_{24} is divided into T_{22} T_{34} [$\because k=3$]

$T_{22} = r(2,2) = 0 \rightarrow$ external node

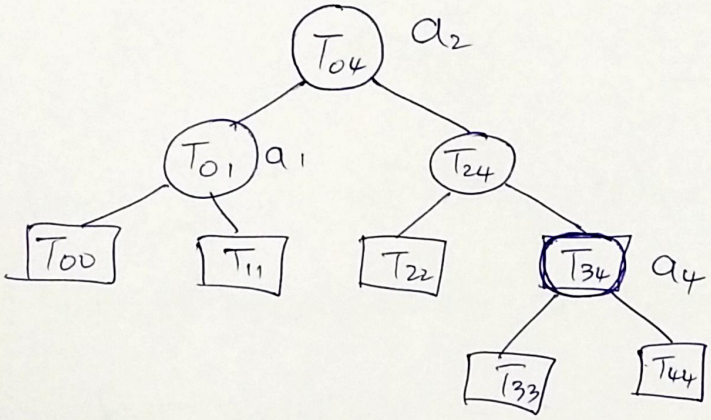
$T_{34} = r(3,4) = 4 \Rightarrow k=4 \Rightarrow a_4$



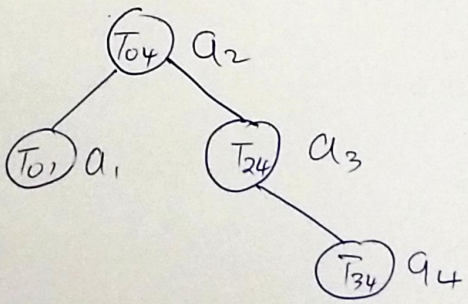
T_{34} is divided into T_{33}, T_{44} $[\because k = 4]$

$T_{33} = r(3,3) = 0 \rightarrow$ external node

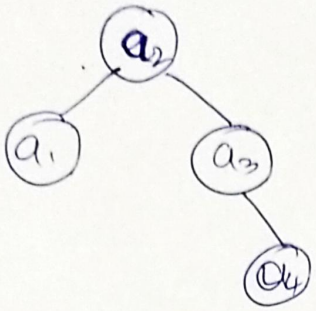
$T_{44} = r(4,4) = 0 \rightarrow$ external node

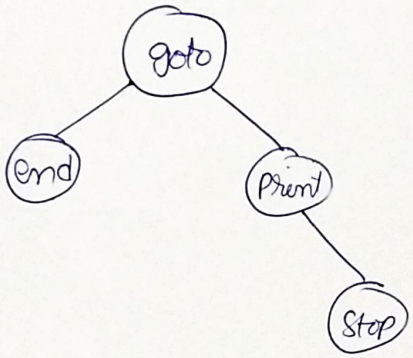


T_{00}, T_{11}, T_{22}, T_{33}, T_{44} are external nodes and can be neglected



Substituting the identifiers the required OBST is :





∵ the identifiers (a_1, a_2, a_3, a_4) = (end, goto, print, stop)

The cost of the OBST is 39