Dynamic Programming

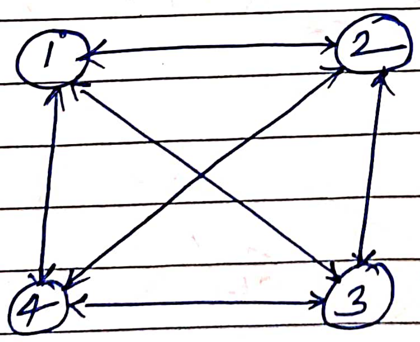
Travelling Salesman Problem (TSP)

Travelling salesman problem find the minimum cost on the given graph G(V,E) that visit each nodes exactly once and return to the starting vertex. This refers to find the Hamiltonian cycle. The problem is easy to explain but very difficult to solve the same problem, due its complexity of the problem. The complexity of the problem executes |n times to find the solution. TSP is known as a non-deterministic problem.

Applications Areas

- A news paper distributed in an ~~ar~~ area visited each house exactly once and return to starting point.
-
- Postal van pickup mail from different mail boxes located at n different sites and return to Head-office where it started.

Suppose we consider one of the graph G with its adjacency ~~cost~~ cost matrix.

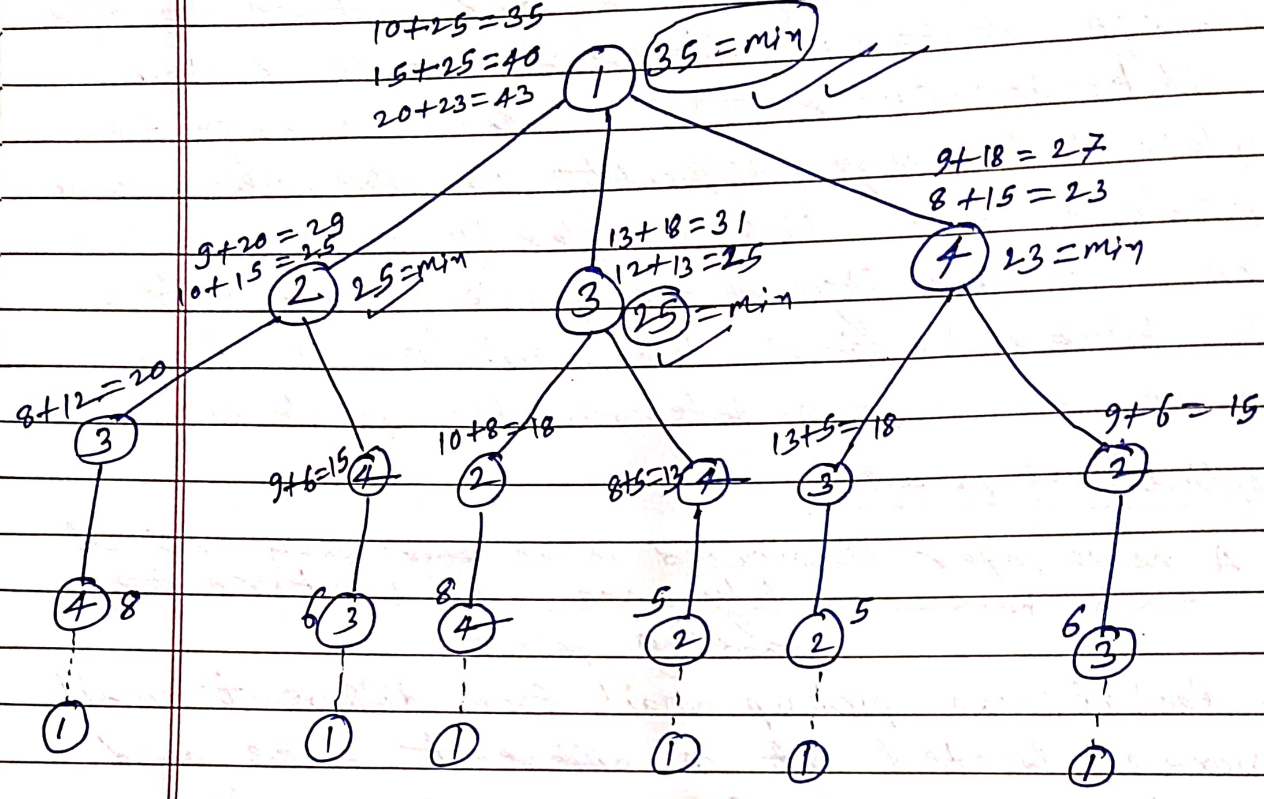


$$\begin{bmatrix} 0 & 10 & 15 & 20 \\ 5 & 0 & 9 & 10 \\ 6 & 13 & 0 & 12 \\ 8 & 8 & 9 & 0 \end{bmatrix}$$

Sol^n:- 1. First create adjacency matrix

2. Create a state-space tree or apply brute force to find all possible solutions.
3. Assume one of the vertex as starting vertex.

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



How the formula is generalized?

$$G(1, \{2,3,4\}) = \min_{K \in \{2,3,4\}} \{ C_{1K} + G(K, \{2,3,4\} - \{K\}) \}$$

General formula

$$G(i, S) = \min_{K \in S} \{ C_{iK} + G(K, S - \{K\}) \}$$

How Dynamic Programming solves it?
start from bottom vertex, either from 2, 3, 4 and return to 1 vertex

C_{i1}

$G(2, \phi) = 5$

$G(3, \phi) = 6$

$G(4, \phi) = 8$

Consider No vertex

Consider ~~&~~ first vertex ~~as remaining~~ vertex

$G(2, \{3\}) = 15$

$G(2, \{4\}) = 18$

$G(3, \{2\}) = 18$

$G(3, \{4\}) = 20$

$G(4, \{2\}) = 13$

$G(4, \{3\}) = 15$

Consider Next vertex

$G(2, \{3,4\}) = 25$

$G(3, \{2,4\}) = 25$

$G(4, \{2,3\}) = 23$

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

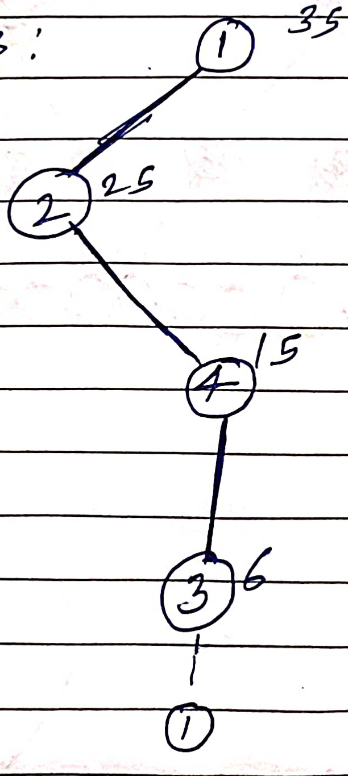
$$= \min \{ 10+25,\ 15+25,\ 20+23 \}$$

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

$$= 35$$

The path is:

29V25



This ~~is~~ path is considered where from we get the minimum.

Dynamic Programming

It is a technique for solving a problems in minimum or maximum finding for the particular problem. It applies bottom-up approach, means it solves a smaller sub-problem, store its results, and reuse its previous results for solving a larger sub-problem. The nature of the problem is frequently used in solving the problem in large.

Applications

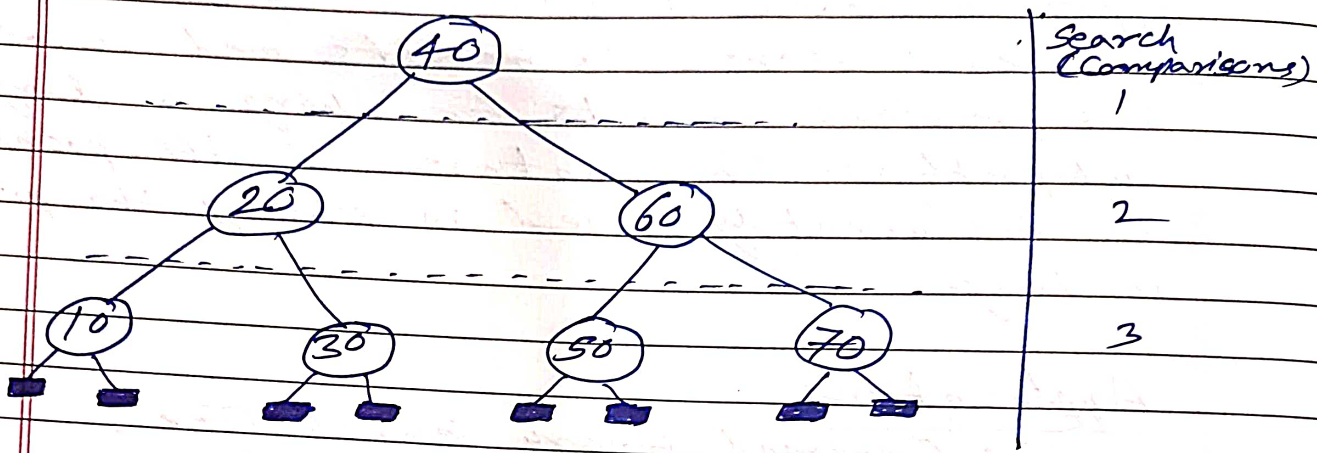
- Optimal Binary Search Tree
- 0/1 Knapsack Problem
- All-pairs shortest Path Problem
- Reliability Design
- Travelling Salesman Problem
- Matrix Chain multiplication
- Longest Common Subsequence
- Mathematical Optimization.

Optimal Binary Search Tree (OBST)

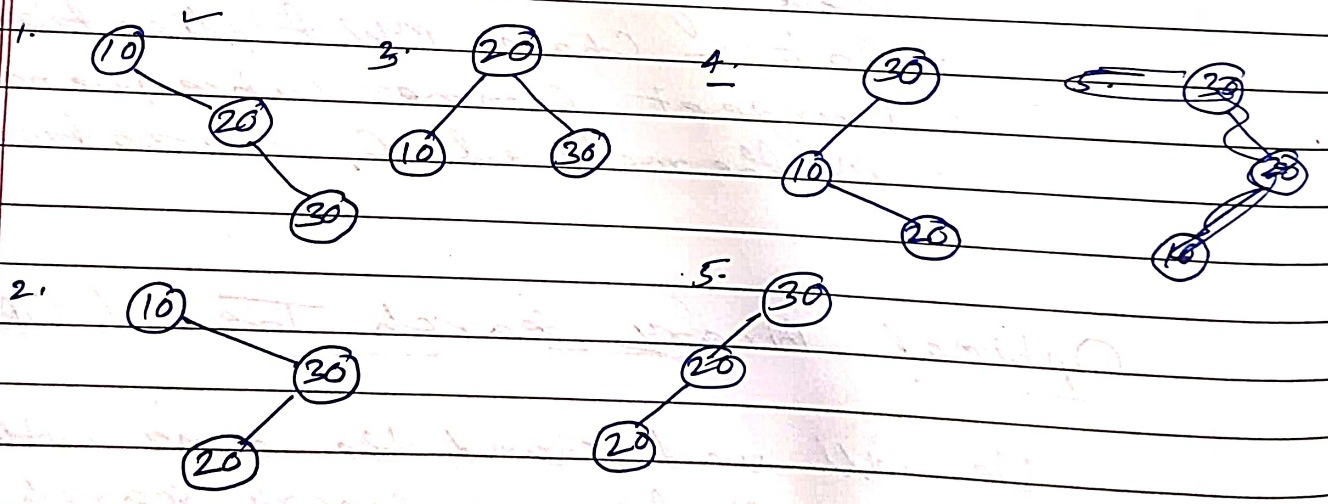
We will understand the same with:

- What is a Binary search Tree
- Cost of searching
- Probability of search/What is OBST
- Dynamic Programming Approach
- Formula / Deriving
- Solving a Problem
- Constructing a OBST

Binary search Tree (BST) or Binary sorted Tree is one the tree where left ~~dat~~ childs of roots are smaller and rights childs of ~~roots~~ are greater than of roots.



Suppose we have keys 10, 20, 30, there are following ways BST can be created.



The cost of searchings for node in ①, ②, ④ & ⑤ needs 3 comparisons where in ③ cost of searching is 2 comparisons. One can see this one is best, here.

The search depends upon height of the tree, if the height is less, comparisons will be less.

How many possible ways to create a tree that depends upon, this formula

$$\frac{^{2n}C_n}{n+1}, \text{ where } n = \text{no. of nodes/elements}$$

In previous example, $n = 3$

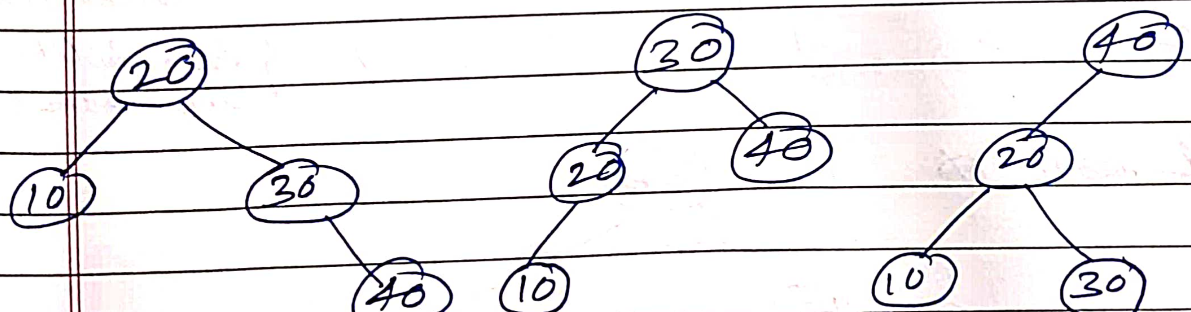
$$\frac{^{2\times 3}C_3}{3+1} = \frac{^{6}C_3}{4} = \frac{\frac{6!}{3!\,(6-3)!}}{4} = \frac{\cancel{6}\times 5\times 4\times \cancel{3}}{\cancel{3}\times \cancel{3}\times 2} = 5$$

Another example

Suppose keys are 10, 20, 30, 40, then how many tree can be possible

$$= \frac{^{2n}C_n}{n+1} = \frac{^{2\times 4}C_4}{4+1} = \frac{\frac{8!}{4!\,4!}}{5} = \cancel{\frac{8\times 7\times 6\times 5}{1}}$$

$$= \frac{8\times 7\times \overset{2}{\cancel{6}}\times 5\times \cancel{4!}}{\cancel{4!}\times \cancel{4}\times \cancel{3}\times \cancel{2}\times 1}\times\frac{1}{\cancel{5}} = 14$$



Real-Life Example

1. We in general visit markets for buying vegetables, fruits or grocessory's.

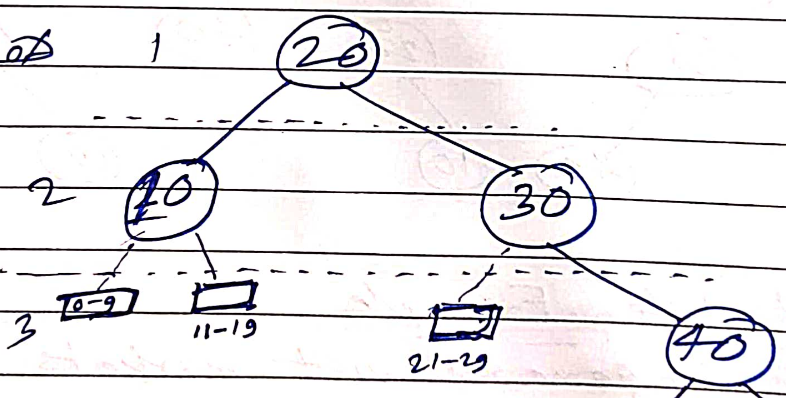
I give one of own example, how I efficiently purchase vegetables & fruits, so should to carry less weight during purchasing.

We create a tree, according to items we ~~purc~~ purchase or we have in list (our). How Items are pharchasing that depends upon the frequency of item, how I am getting the same. If the item is present in the availability, ~~depends upon~~ it is known by success but if the item doesn't it is ~~na~~ named by unsuccessful search.

One example

		10	20	30	40	
p_i		2	3	1	1	{Probability of successful}
q_i	2	3	1	1	1	{Probability of unsuccessful}

~~Total cost of~~



$(1\times 2 + 2\times 2 + 2\times 1 + 3\times 1)$ + $\underline{2\times 2 + 2\times 3 + 2\times 1 + 3\times 1 + 3\times 1}$

Success search cost

Unsuccessful search Cost

for j^{th} item

$$\text{cost}[0,j] = \sum_{0\le i<n} p_i * \text{level of Keys}(a_i) + \sum_{0\le i<n} q_i * \text{level of External Node}(E_{i-1})$$

What is Optimal BST (OBST)

An OBST is a problem in which we have to generate all BST whose cost is minimum. Creating all BST's and evaluating its costs are time consuming for finding that contains the minimum cost.

Without generating all BST's & finding the minimum cost, Dynamic ~~problem~~ Programming is helpful in it. We will try to generate all tree, with costs, but not directly we will do the same. In the result minimum cost finding, that will be beauty of the Dynamic Programming.

Three formula's helpus:

1. $w[i,j] = w[i, j-1] + \underbrace{p[j]}_{\text{Successful Probability}} + \underbrace{q[j]}_{\text{Unsuccessful Probability}}$

2.

2. $c[i,j] = \min\limits_{i<k\leq j} \{c[i, k-1] + c[k, j]\} + w[i,j]$

3. $r[i,j] = k$

$w[i,j] = w[0,-1] + p(0) + q(0)$

$\approx q[j]$

Initial Assumptions

If $i==j$ $\quad w[i,j] = q[j]$

$c[i,j] = 0$

$r[i,j] = 0$

$i==j$, if

i,j	
0	0
1	1
2	2
3	3

Suppose we want to find minimum cost for given nodes, such as

	0	1	2	3	4
Key		10	20	30	40
Probability of Success ← p_i		3	3	1	1
Probability of Unsuccess q_i	2	3	1	1	1

j \ i →	0	1	2	3	4
4	w[0,4] = 16 c[0,4] = 32 r[0,4] = 2	w[1,4] = 11 c[1,4] = 19 r[1,4] = 2	w[2,4] = 5 c[2,4] = 8 r[2,4] = 3	w[3,4] = 3 c[3,4] = 3 r[3,4] = 4	w[4,4] = 1 c[4,4] = 0 r[4,4] = 0
3	w[0,3] = 14 c[0,3] = 25 r[0,3] = 2	w[1,3] = 9 c[1,3] = 12 r[1,3] = 2	w[2,3] = 3 [2,3] = 3 [2,3] = 3	w[3,3] = 1 c[3,3] = 0 r[3,3] = 0	
2	w[0,2] = 12 c[0,2] = 19 r[0,2] = 1	w[1,2] = 7 c[1,2] = 7 r[1,2] = 2	w[2,2] = 1 c[2,2] = 0 r[2,2] = 0		
1	w[0,1] = 8 c[0,1] = 8 r[0,1] = 1	w[1,1] = 3 c[1,1] = 0 r[1,1] = 0			
0	w[0,0] = 2 c[0,0] = 0 r[0,0] = 0				

Initial Assumption ~~w[0,0]=2~~ for i == j or for i, j = 0 Difference

$w[0,0] = q_0 = 2,\ w[1,1] = q_1 = 3,\ w[2,2] = q_2 = 1,$

$w[3,3] = q_3 = 1,\ w[4,4] = q_4 = 1$

$c[0,0] = c[1,1] = c[2,2] = c[3,3] = c[4,4] = 0$

$r[0,0] = r[1,1] = r[2,2] = r[3,3] = r[4,4] = 0$

For i, j = 1 Difference [0,1], [1,2], [2,3], [3,4]

1) $w[0,1] = w[0,0] + p(j) + q(j) = 2 + 3 + 3 = 8$

$c[0,1] = \min_{0<k\le 1} \{c[0,0] + c[1,1]\} + w[0,1] = 0 + 0 + 8$

$k = 1$

$r[0,1] = 1$

(2) $w[1,2] = w[1,1] + p(2) + q(2) = 3 + 3 + 1 = 7$

$c[1,2] = \min_{1<k\le 2} \{c[1,1] + c[2,2]\} + w[1,2] = 0 + 0 + 7 = 7$

$k = 2$

$r[1,2] = 2$

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

$c[2,3] = \min_{2<k\le 3} \{c[2,2] + c[3,3]\} + w[2,3] = 0 + 0 + 3 = 3$

$k = 3$

$r[2,3] = 3$

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

$c[3,4] = \min_{3<k\le 4} \{c[3,3] + c[4,4]\} + w[3,4] = 0 + 0 + 3$

$= 3$

$k = 4$

$r[3,4] = 4$

For $i, j = 2$ Difference

$[0,2], [1,3], [2,4]$

i. $w[0,2] = w[0,1] + p(2) + q(2) = 8 + 3 + 1 = 12$

$c[0,2] = \min_{0<k\le 2} \{c[0,0] + c[1,2]\} + w[0,2]$

$k = 1$ $= 0 + 7 + 12 = 19$

$k = 2$ $\{c[0,1] + c[2,2]\} + w[0,2] = 8 + 0 + 12 = 20$

$r[0,2] = 1$

ii. $w[1,3] = w[1,2] + p(3) + q(3) = 7 + 1 + 1 = 9$

$c[1,3] = \min_{1<k\le 3} \{c[1,1] + c[2,3]\} + w[1,3] = 0 + 3 + 9$

$= 12$

$k = 2$

$\min_{1<k\le 3} \{c[1,2] + c[3,3]\} + w[1,3] = 7 + 0 + 9$

$k = 3$ $= 16$

$r[1,3] = 2$ $\{9, 12, 2\}$ (w c r)

iii. $w[2,4] = w[2,3] + p(4) + q(4) = 3 + 1 + 1 = 5$

$c[2,4] = \min_{2<k\le 4} \{c[2,2] + c[3,4]\} + w[2,4] = 0 + 3 + 5 = 8$

$k = 3$

$k = 4$ $\{c[2,3] + c[4,4]\} + w[2,4] = 3 + 0 + 5 = 8$

$r[2,4] = 3$

For $i,j = 3$ Difference

$[0,3]$ & $[1,4]$

$[0,3]$

$w[0,3] = w[0,2] + p(3) + q(3) = 12+1+1 = 14$

$c[0,3] = \min_{0<k\leq 3} \{ c[i,k-1] + c[k,j] \} + w[i,j]$

$k=1 \quad \{ c[0,0] + c[1,3] \} + 14 = 0+12+14 = 26$

$k=2 \quad \{ c[0,1] + c[2,3] \} + 14 = 8+3+14 = 25$ ✓

$k=3 \quad \{ c[0,2] + c[3,3] \} + 14 = 19+0+14 = 33$

$r[0,3] = 2$

$[1,4]$

$w[1,4] = w[1,3] + p(4) + q(4) = 9+1+1 = 11$

$c[1,4] = \min_{1<k\leq 4} \{ c[1,1] + c[2,4] \} + w[1,4] = 0+8+11 = 19$

$k=2$ ✓

$k=3 \quad \{ c[1,2] + c[3,4] \} + w[1,4] = 7+3+11 = 20$

$k=4 \quad \{ c[1,3] + c[4,4] \} + w[1,4] = 12+0+11 = 23$

$r[1,4] = 2$

For $i,j = 4$ Difference

$[0,4]$ Only

$w[0,4] = w[0,3] + p(4) + q(4) = 14+1+1 = 16$

$c[0,4] = \min_{0<k<4} \{ c[0,0] + c[1,4] \} + w[0,4] = 0+19+16 = 35$

$k=1$

$k=2 \quad \{ c[0,1] + c[2,4] \} + w[0,4] = 8+8+16 = 32$

$k=3 \quad \{ c[0,2] + c[3,4] \} + w[0,4] = 19+3+16 = 38$

$k=4 \quad \{ c[0,3] + c[4,4] \} + w[0,4] = 25+0+16 = 41$

$r[0,4] = 2$

1 search item

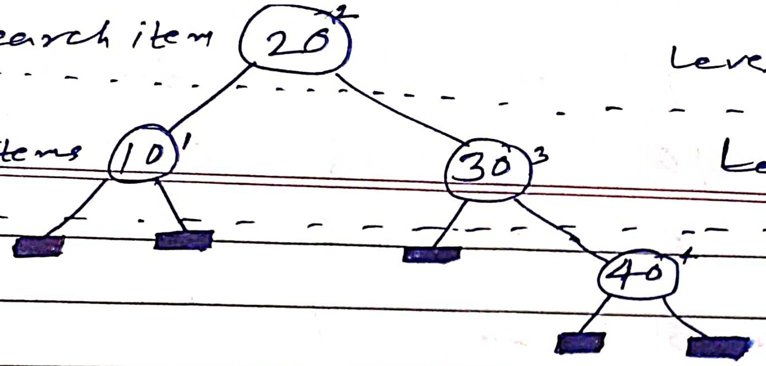
Level-1

2 search items

Level-2

3 search items.

Level-3



No. of search(s)

No. of Unsuccessful search

$\cancel{1\times 3}$ 8 $1\times 3 + 2\times 3 + 2\times 1 + 3\times 1 + 2\times 2 + 2\times 3 + 2\times 1 + 3\times 1 + 3\times 1$

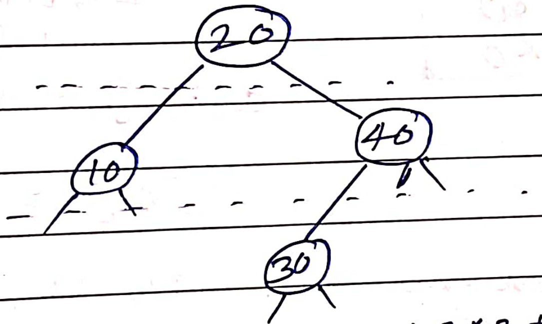
Search success possibilities (Frequency)

Unsuccessful possibilities (Frequency)

$\boxed{= 32}$

Total cost

<u>or</u>



$(1\times 3 + 2\times 3 + 2\times 1 + 3\times 1) + (2\times 2 + 2\times 3 + 3\times 1 + 3\times 1 + 2\times 1)$

Cost will be ~~with~~ same in both ~~these~~ tree