Title→ Optimal binary search tree:

Name: saikrishna kovuru

→ Problem statement:

Finding the optimal solution (most searched video) while we search online(YouTube)
Let's consider few scenarios

Scenario-1:

Let us assume there are 10 people and 6 people searched on YouTube for a same video and rest of the 4 people searched some other videos. So, the most searched or streamed video has to be on top order in the youtube's database if not the most frequently streamed video is on the topmost index in the database the searching cost goes high as well as the search time.
So we have to make most searched video to be optimal. So, that as soon as someone look for the video which has higher frequency, it has to be searched with less cost and time.

Scenario-2:

In this scenario we will look into the common problem with indexing in the databases, Same as the above scenario 1, databases also need indexing while searching or retrieving any of the records. If a commonly or most frequently searched record is given with high frequency, then every time the people search with that most frequent record will be retrieved sooner than the rest of the records because the indexing has been changed as priority for that most searched record.

The problem is optimizing the cost and time while searching.

→ Algorithm description:

As discussed in the problem, we should be able to retrieve any of the most searched stream video in YouTube/record in database. This needed indexing or giving frequency to the one which has the higher rate of visit, so that that particular video/record will be set with higher frequency.
Items with higher frequency has to be moved upside similarly the items with lesser frequencies will be moved downside.
We use Binary Tree structure to set the structure up in the tree form. Decoding the above line again moving the items upside with has the higher frequencies that means the items which has the higher frequencies will be moved upside in the tree. In a particular item has the highest frequency than the rest of other items then that can be

taken as a root node. Since the tree traversal starts from the node, we have the chance of fetching that particular node in the very first search so that we can save the cost of the search. The whole above thing is called the optimal binary search tree. This means we are optimizing the tree based on the frequencies.

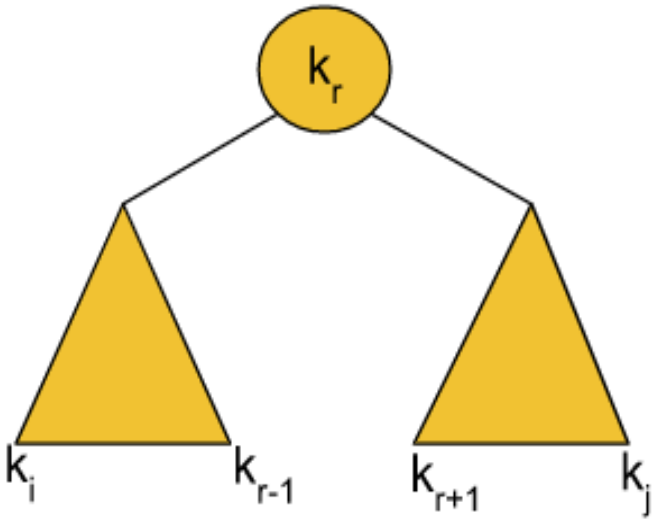
Algorithm:

- One of the keys in k_i, ……,k_j, say k_r, where $i \leq r \leq j$, must be the root of an optimal subtree for these keys.
- Left subtree of k_r contains k_i, …,k_{r-1}.
- Right subtree of k_r contains k_r+1, …..,k_j.

To find an optimal BST :

- Examine all candidate roots k_r ,for $i \leq r \leq j$
- Determine all optimal BST containing k_i, …,k_{r-1} and containing k_r+1, …..,k_j

Below diagram describes how the optimal tree structure is formed.



Recursive solution:

- Find optimal BST for k_i,…..,k_j, where $i \geq 1$, $j \leq n$, $j \geq i-1$. When $j = i-1$, the tree is empty.
- Define

A=[i, j]=expected search cost of optimal BST for k_i, …..,k_j.

- If $j = i-1$, then $A[i, j] = 0$.
- If $j \geq i$,

- Find optimal BST for k_i,…..,k_j, where i ≥ 1, j ≤ n, j ≥ i-1. When j = i-1, the tree is empty.
- Define

  A=[i, j]=expected search cost of optimal BST for k_i, …..,k_j.
- If j = i-1, then A[i, j] = 0.
- If j ≥ i,

  - Select a root k_r, for some $i \leq r \leq j$.
  - Recursively make an optimal BSTs
  - for k_{r+1}, …,k_j as the right subtree.
  - for k_i, …,k_{r-1} as the left subtree, and

Expected search cost:

- A=[i, j]=expected search cost of optimal BST for k_i, …..,k_j.

  Formulae followed:

$$A[i][j] = \min_{i \leq k \leq j}(A[i][k-1] + A[k+1][j]) + \sum_{m=i}^{j} P_m \qquad (i<j)$$

→ Results and discussion:

The input we give is for the binary tree input along the frequency of the each node respectively.
I throwed an exception saying bstInput and the frequency has to the same number because there should be same number of frequencies for the same number of the nodes.

Later running the programme I first displayed the binary tree format how it looks like the binary tree based on the input

Like in the next page

Printing the tree structure

```
****.......................................................****
                              10
               --                                  12
        --              --              --              20
****.......................................................****
```

```
[java]
[java]  Printing the tree structure
[java]
[java] ****.......................................................****
[java]                                         10
[java]                      --                                   12
[java]           --                   --                   --                  20
[java] ****.......................................................****
```

Here is how the tree looks like,

To confirm with the input we gave I also displayed the inputs which are BSTInput and frequency, they appear like below

list of BST values given as input:
[10, 12, 20]

```
[java]
[java]  list of BST values given as input:
[java] [10, 12, 20]
[java]
```

list of Frequencies given as input:
[34, 8, 50]

```
[java]
[java]  list of Frwquencies given as input:
[java] [34, 8, 50]
[java]
```

At last the total cost of the BST will be displayed.

```
[java]
[java]
[java]  The total cost of the optimal BST based on frequencies is: 142
```

Here is how the overall o/p looks like while we run.

```
run:
     [java]
     [java]
     [java]  list of BST values given as input:
     [java] [10, 12, 20]
     [java]
     [java]
     [java]  Printing the tree structure
     [java]
     [java] ****.....................................................****
     [java]                                  10
     [java]                    --                                  12
     [java]           --                 --                 --              20
     [java] ****.....................................................****
     [java]
     [java]
     [java]  list of Frequencies given as input:
     [java] [34, 8, 50]
     [java]
     [java]
     [java]  The total cost of the optimal BST based on frequencies is: 142
```

Time complexity

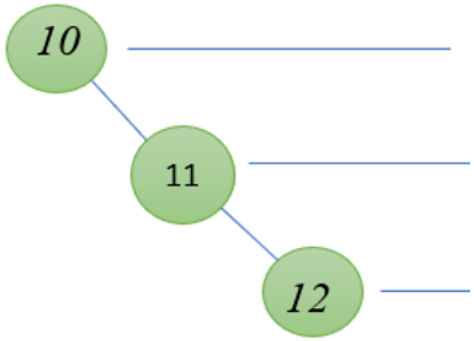
Best case time complexity:

This occurs when tree is balanced binary search tree which is O(logn).

Worst case time complexity:

This occurs when the binary search tree is a Skewed binary search tree which is O(n).

Here is how the Skewed binary search tree looks like



NOTE: I hope that I completed the justification of optimal binary search tree and I noticed no bugs so far

→ References:

GeeksforGeeks and stackOverFlow.

Implementation:

I implemented the BST in java language and please look into readme file to run and compile the program. Used Ant build tool and described the commands to run the program.