DS 221 (AUG) 3:1 Introduction to Scalable Systems

Assignment 1

Q2. Tree Search Problem

Time complexity of In-order Traversal is O(n), here n is number of nodes. This is because each node is visited only once in In-order Traversal.

Time complexity of Level-order Traversal is O(~2n), here n is number of nodes. This is because each node is visited sometimes thrice and sometimes only once (Leaf Nodes) in Level-order Traversal. Sometimes it is also represented as O(n+v), here n is number of nodes and v in number of vertices in the Binary Tree.

Time Complexity of Linear Search is in worst case for searching one key is O(n) and for searching all the elements in the array is $O(n^2)$ if we are searching these keys in random order. And, if the keys are not repeated the Time Complexity for searching becomes $O((n^2+n)/2)$.

Time Complexity of Binary Search is O(log n) because after each iteration size of sample reduces to half. The best case will be O(1) when the middle value will directly match the desired value. The worst-case scenario can be if the desired value is at either extreme point of the array or if the value is not in the array.

1. So in In-Order search time complexity becomes $(O(n^2))$ in worst case if for all elements if we had to travel whole tree. Again, if we are calling our function recursively for In-order search overhead tasks (e.g. memory access) increases the execution time.

2. And for Level-order search time complexity becomes $O((n^2+n)/2)$ as we travel along data file the data we check is same order as it is stored in the tree so 1^{st} element will require O(1) time 2^{nd} will require O(2) time. Thus, for n elements total time required will be in $O((n^2+n)/2)$.

By observing these two complexities we can conclude that for In-order search time required will be more. As $O(n^2)$ will be always more than $O((n^2+n)/2)$ and will be same only when the elements in the query file are not repeated. And, recursion in the In-order traversal again takes more time.

In the code I tried for 6 different queries with different number of elements and calculated time required for them:

1) 5 elements
2) 68 elements
3) 277 elements
4) 500 elements
5) 1000 elements
6) 1500 elements

Observation:

1. As the number of elements in query file increases time difference between In-order traversal search and Level-order traversal search increases with In-order taking more time than level-order.
2. We can observe in the Fig 1. the time required for Binary Search was less than both the above algorithm as its time complexity is O(log n).

```
sameer@LAPTOP-K7M6TAOK:/mnt/d/ds221-2020/a1/q2$
A2a:5,6900,5100
A2c:3700
sameer@LAPTOP-K7M6TAOK:/mnt/d/ds221-2020/a1/q2$
A2a:68,29000,43000
A2c:11500
sameer@LAPTOP-K7M6TAOK:/mnt/d/ds221-2020/a1/q2$
A2a:277,376600,217400
A2c:159200
sameer@LAPTOP-K7M6TAOK:/mnt/d/ds221-2020/a1/q2$
A2a:500,1077200,448300
A2c:94100
sameer@LAPTOP-K7M6TAOK:/mnt/d/ds221-2020/a1/q2$
A2a:1000,6505500,1657400
A2c:644500
sameer@LAPTOP-K7M6TAOK:/mnt/d/ds221-2020/a1/q2$
A2a:2000,19158900,3844400
A2c:1586300
```

Fig 1. Output for query files with different number of elements

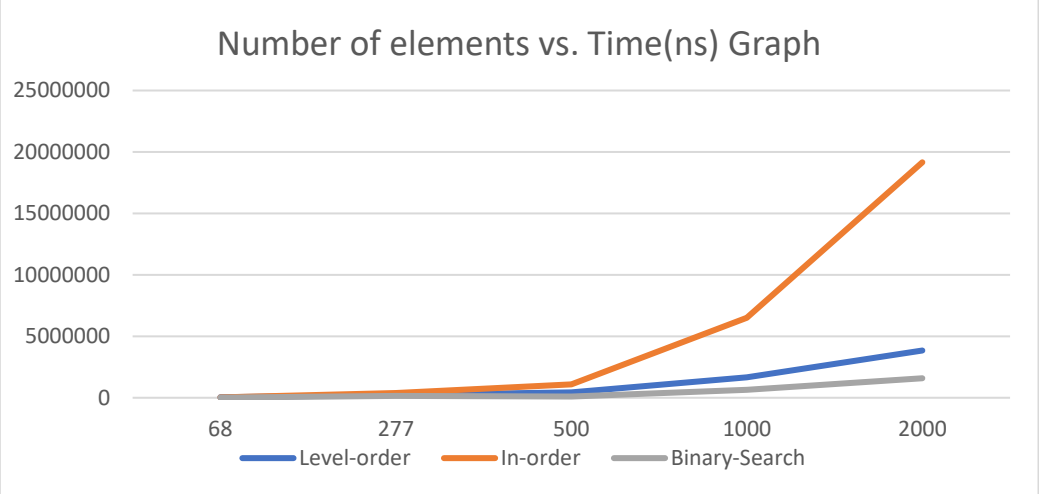

Fig 2. Graph for the Time Vs. Number of elements

Steps I followed in the code:

1. Took two input files as command line argument.
2. For first file I made it as a array which can be implemented as a binary tree.
3. For second file I made a binary tree by inserting elements one by one.
4. Then searched keys in this tree by in-order traversal and level-order traversal.

The array size required was same as number of elements in the number file and I made this array using vectors.

Submitted By:
Name: Khadatkar Sameer Raju
Sr. No.: 06-18-01-10-51-20-1-17830
Email Id: sameerraju@iisc.ac.in