PA02 Report: Time Analysis of Search in a BST
Authors: Anika Arora and Gabriel Fei

I. Trends in the average running time to search for N keys in a BST (N is based on the size of the dataset)

Dataset	Number of runs (W)	Minimum Time (micro seconds)	Maximum time (micro seconds)	Median (micro seconds)
20 Ordered	50	1000	5650	2950
20 Random	50	1050	4550	2800
100 Ordered	50	2410	3520	2880
100 Random	50	2010	3090	2560
1000 Ordered	50	2709	4075	3571.5
1000 Random	50	3013	4612	3297

Table 1: Stats on the Average Time to Search in a BST for different data sets

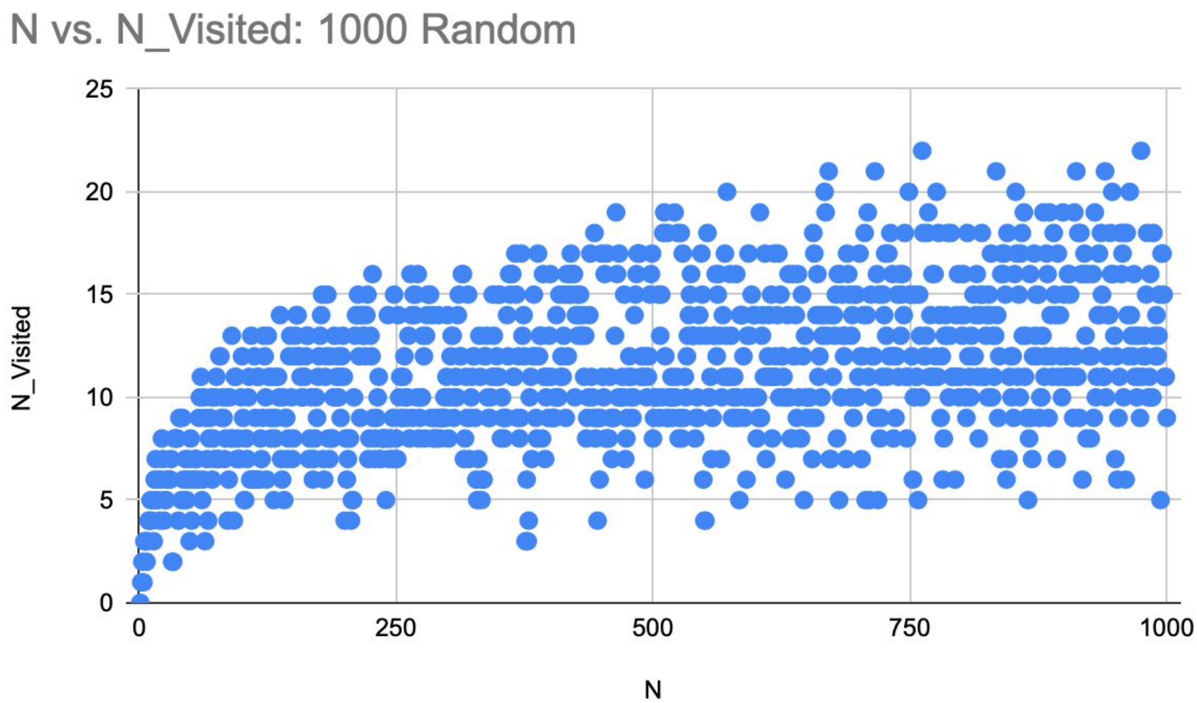
(1) Explain the trends you observe in the table above.

As the number of movies in the dataset increases, the minimum time generally increases. However, for the 100 data sets, the minimum time for ordered was bigger than for random whereas for the 20 and 1000 data sets, the minimum time for ordered was smaller than for random. Similarly, the maximum time for 20 and 100 ordered were both higher than the randomized data sets, but for 1000, the maximum time was higher for the random set. The median was the only one that was consistent throughout the three quantities of data. All of the ordered sets had a higher median than the random sets.

(2) Are the trends as you expect? Why or why not?

Most of the trends were not as we expected. We were expecting more uniform trends throughout the data sets. One trend that was expected was that the minimum time would generally increase as the amount of data increased, however the results of 100 ordered vs 100 random challenged our initial thoughts. This could be due to the execution machine itself or the fact that even though we might assume an increase in time given a certain number of inputs, we don’t know for sure how time will necessarily scale with said amount of inputs.

(3) (Optional) Add any other related plots that you think are relevant to this question.



II. Trends in the number of nodes visited when inserting a new key into the BST [for random and ordered data sets with 1000 entries]

(1) How does the number of operations for insert vary with the number of the current nodes present in the tree when (a) nodes are inserted into the BST in alphabetical order (b) nodes are inserted in random order?
    (a) The number of operations increases linearly as nodes are inserted in alphabetical order.
    (b) The number of operations also increases as nodes are inserted in random order.

(2) Include the code for your insert function, do a Big O analysis and use it to explain the trends you observed?

```
bool BST::insert(string a, double b) {
    if(!root) {
        root = new Node(a, b);
        return true;
    }
    insertHelper(a, b, root);
}

   bool BST::insertHelper(string a, double b , Node *n) {
       if(a == n->name) {
           return false;
       }
       if(a < n->name) {
           if(n->left) {
               return insertHelper(a, b, n->left);
           }
           else {
               n->left = new Node(a, b);
               n->left->parent = n;
               return true;
           }
       }
       else {
           if(n->right) {
               return insertHelper(a, b, n->right);
           }
           else {
               n->right = new Node(a, b);
```

```
                n->right->parent = n;
                return true;
            }
        }
    }

```

Our Big-O Analysis on our insert function gives us O(logN) for the average case and O(N) for the worst case. In terms of trends, the trend in the 1000 random graph matches a O(logN) graph and the trend in the 1000 ordered graph matches a O(N) graph.

(3) Are the trends you observed as you expect? Why or Why not?

Yes. We expected there to be more nodes visited as we inputted more nodes into the BST. This trend is seen in both graphs (ordered and random). This is because as you input more nodes, you must traverse through more nodes and more operations have to be done to reach the desired location.