PA02 Report on Timing Analysis of Search in a BST

Trends in the average time to search for N codes in a BST (N is defined by the size of the Dataset)
This table reports statistics about the ***average time to search in a BST*** for different data sets.

Dataset	Number of runs (W)	Average (Microseconds)
20 Ordered	50	5620
20 Random	50	1120
100 Ordered	50	28560
100 Random	50	2880
1000 Ordered	50	333480
1000 Random	50	2540

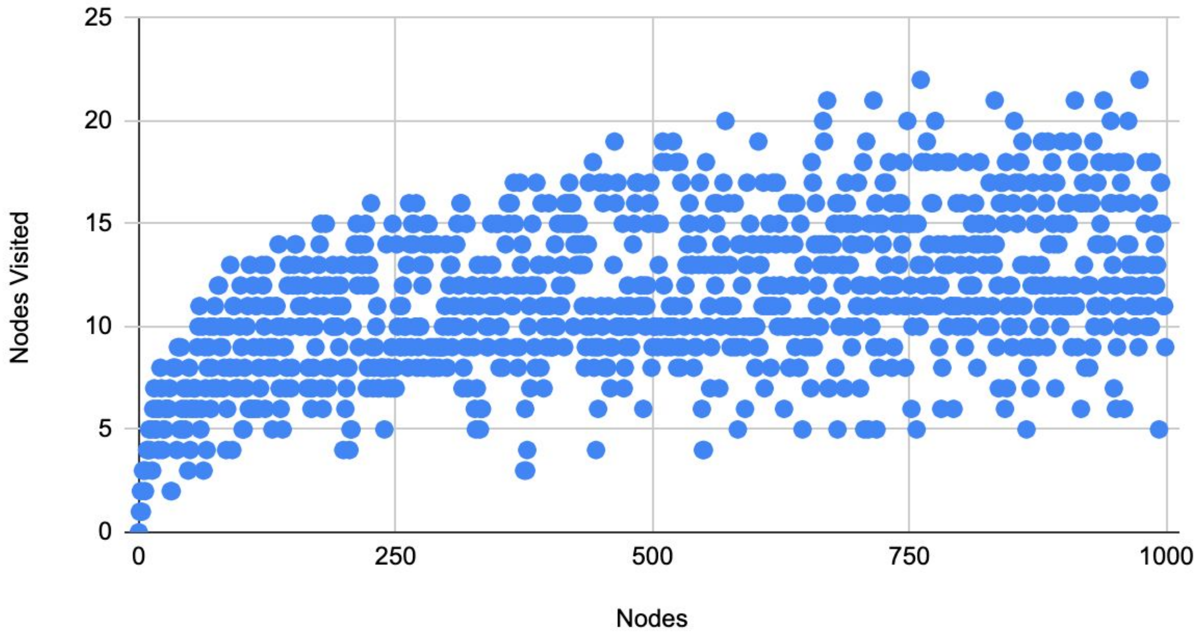
(1) Explain the trends that you observe in the above table.

For the ordered input files, the average time per search was much larger than for the random files. The largest random file is faster on average than the smallest ordered file.

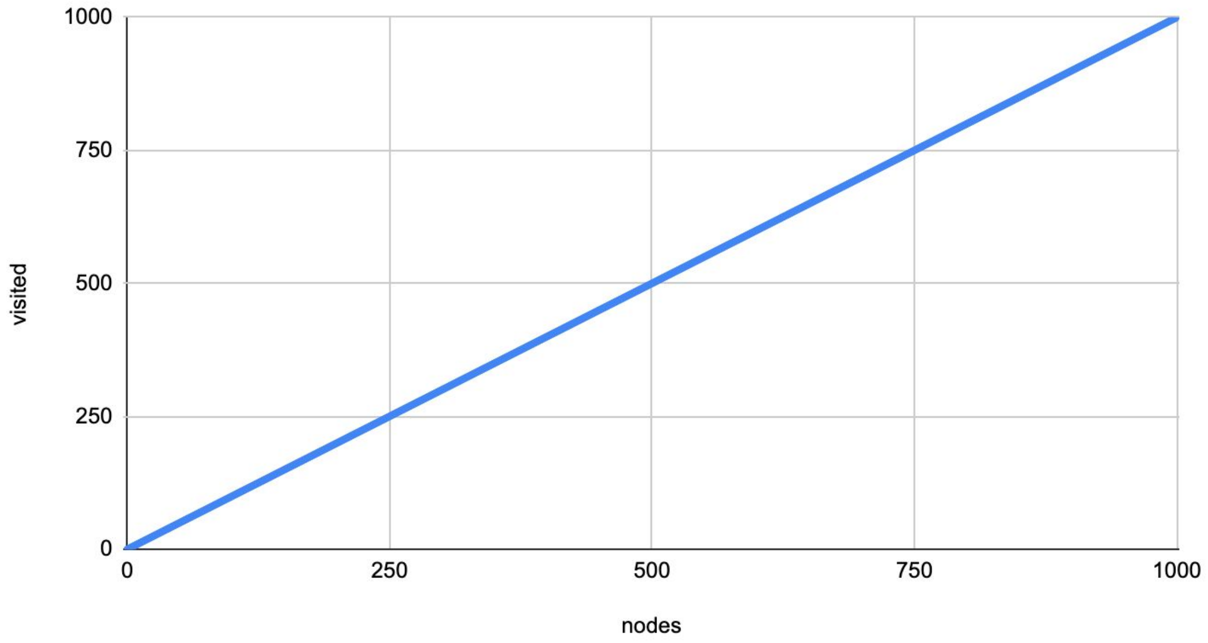
(2) Are the trends as you expect? Why or why not?

This makes sense since the ordered file gets organized into a BST with only right subtrees, effectively making it a linked list. Therefore, it must go through all the nodes rather than half the remainder.

Random



Ordered



(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?

When they are inserted in alphabetical order, the number of nodes visited is equal to the number of nodes. When they are inserted in random order, the number of nodes visited increases logarithmically compared to the total number of nodes

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

```
bool MovieBST::insert(string value) {
   // handle special case of empty tree first
         string mName;
         double mRating;
         int comma;

         comma = value.rfind(",");
         mName = value.substr(0,comma);
         mName.erase(remove(mName.begin(), mName.end(), '"'), mName.end());
         mRating = stod(value.substr(comma+1));

   if (!root) {
                  root = new Node(mName, mRating, 0);
                  return true;
   }
   // otherwise use recursive helper
   return insert(mName, mRating, root, 1);
}

// recursive helper for insert (assumes n is never 0)
bool MovieBST::insert(const string& mName, const double& mRating, Node *n, int mDepth) {
   if (mName == n->name)
                  return false;
   if (mName < n->name) {
                  if (n->left)
                  return insert(mName, mRating, n->left, mDepth+1);
                  else {
                     n->left = new Node(mName, mRating, mDepth);
                     n->left->parent = n;
                     return true;
                  }
```

```
    }
    else {
                    if (n->right)
                        return insert(mName, mRating, n->right, mDepth + 1);
                    else {
                        n->right = new Node(mName, mRating, mDepth);
                        n->right->parent = n;
                        return true;
                    }
    }
}
```

This is a Olog(n) function on average, but it is a O(n) function in the worse case (alphabetical)

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

These trends are expected since inserting into a BST is supposed to be Olog(n) on average and O(n) in the worst case.