**n = number of integers in all the text files read**

**BST \*CreateTree(BST &tree);**

Series of statements.

Runtime: f(n) = a constant number, k

All cases are constant.

**bool IsEmpty(BST \*root);**

Series of statements.

Runtime: f(n) = a constant number, k

All cases are constant.

**BST \*CreateNode(const int & itemInsert);**

Series of statements.

Runtime: f(n) = a constant number, k

All cases are constant.

**void InsertNode(BST \*&root, BST \* &temp, BST & tree);**

Calls FindNode which has an efficiency of f(n) = n.

Calls IsEmpty which has an efficiency of f(n) = a constant number, k

InsertNode has a while loop resulting in f(n) = n.

f(n) = n + n = n

Runtime: f(n) = n

**BST \*FindNode(BST \*root, const int & findItem, bool & found);**

While loop resulting in f(n) = n.

Runtime: f(n) = n

**BST \*DeleteNode(BST \*root, int deleteItem);**

Series of statements with recursion.   
  
f(n) = a constant number, k -> + some number of steps = f(n + 1) = n.

Runtime: f(n) = n

**BST\* FindMin(BST\* root);**

While loop resulting in f(n) = n.

Runtime: f(n) = n

**void InOrderDisplay(BST \*ptr, int & count);**

Series of statements with two recursion calls.

f(n) = a constant number, k -> + some number of steps = f(n + 1) = n.

k + k = 2\*k steps per f(n + 1) = 2n

Runtime: f(n) = 2n

**void FreeNodes(BST \*tree);**Series of statements with two recursion calls.

f(n) = a constant number, k -> + some number of steps = f(n + 1) = n.

k + k = 2\*k steps per f(n + 1) = 2n

Runtime: f(n) = 2n

**void DestroyTree(BST &tree);**

Series of statements.

Runtime: f(n) = a constant number, k

All cases are constant.

**bool emptyFile(const string &fileName);**

Series of statements.

Runtime: f(n) = a constant number, k

All cases are constant.

**void readFile(BST \*&root, BST & tree, const string &fileName);**

Calls emptyFile = f(n) = a constant number, k

Calls FindNode = f(n) = n

Calls CreateNode = f(n) = a constant number, k

Calls InsertNode = f(n) = n

f(n) = n + n = n

Runtime: f(n) = n

**int main( )**

Calls CreateTree = f(n) = a constant number, k

Do While loop = f(n) = n

Calls readFile = f(n) = n

Do While Loop = f(n) = n

Calls IsEmpty = f(n) = a constant number, k

Calls InOrderDisplay = f(n) = 2n

Calls FindNode = f(n) = n

Calls CreateNode = f(n) = a constant number, k

Calls InsertNode = f(n) = n

Calls Is Empty = f(n) = a constant number, k

Calls FindNode = f(n) = n

Calls DeleteNode = f(n) = n

Calls FindNode = f(n) = n

Calls IsEmpty = f(n) = a constant number, k

Calls FindNode = f(n) = n

Calls InOrderDisplay = f(n) = 2n

Calls FreeNodes = f(n) = 2n

Calls DestroyTree = f(n) = a constant number, k

9(n) + 2n + 2n + 6 (constant)

So the runtime for the entire program is: f(n) = 2n