

Raul Cervantes

RVCERVA 77825705

Report

Start of Report

```

hermod.ics.uci.edu - PuTTY
using namespace std;
#include <iostream>
#include <string>
#include <map>
#include <iterator>
template <typename KeyType, typename ElementType>

class SearchTree
{
    struct TreeNode{
        KeyType key;
        ElementType info;
        TreeNode* left, * right;
        //int nodesCreated = 0;
        TreeNode(KeyType new_key, ElementType new_info, TreeNode* new_left, TreeNode* new_right)
            :key(new_key), info(new_info), left(new_left), right(new_right) {}

        TreeNode(const TreeNode& tn) = delete;

        TreeNode& operator = (const TreeNode& tn) = delete;

        static int count_nodes(TreeNode* t)
        {
            return t==nullptr?0:1+count_nodes(t->left)+count_nodes(t->right);
        }

        static TreeNode* newNode(KeyType k, ElementType e, TreeNode* l, TreeNode* r)
        //T(N) = 1001
        //Big-O = O(1)
        {
            // ++nodesCreated;
            return new TreeNode(k,e,l,r);
        }

        static TreeNode* insert(KeyType key, ElementType info, TreeNode* RootT)
        //T(N) = 2014N + 1005
        //Big-O = O(N)
        {
            if(!RootT)
            {
                RootT = newNode(key, info, nullptr, nullptr); //1002
                return RootT; //1
            }
            TreeNode* t = RootT; //1
            while(t->key != key) //N
            {
                if(key < t->key) //N
                {
                    if(!t->left) //N
                    {
                        t->left = newNode(key, info, nullptr, nullptr); //1002N
                        return RootT; //N
                    }
                    t = t->left; //N
                }
                else if(key > t->key) //N
                {
                    if(!t->right) //N
                    {
                        t->right = newNode(key, info, nullptr, nullptr); //1002N
                        return RootT; //N
                    }
                    t = t->right; //N
                }
                t->info = info; //N
            }
            return RootT; //1
        }
    }
};

```

```
hermod.ics.uci.edu - PuTTY
static TreeNode* find(KeyType key, TreeNode* t)
//T(N) = 6N + 2
//Big-O = O(N)
{
    TreeNode* temp = t; //1
    while(temp != nullptr) //N
    {
        if(key < temp->key) //N
        {
            temp = temp->left; //N
        }
        else if(key > temp->key) //N
        {
            temp = temp->right; //N
        }
        else
        {
            return temp; //N
        }
    }
    return nullptr; //1
}

static TreeNode* findPred(TreeNode* t)
//T(N) = 2N + 3
//Big-O = O(N)
{
    TreeNode* tn = t; //1
    tn = tn->left; //1
    while(tn->right != nullptr) //N
    {
        tn = tn->right; //N
    }
    return tn; //1
}

static TreeNode* findSucc(TreeNode* t)
//T(N) = 2N + 3
//Big-O = O(N)
{
    TreeNode* tn = t; //1
    tn = tn->right; //1
    while(tn->left != nullptr) //N
    {
        tn = tn->left; //N
    }
    return tn; //1
}

static void swapKeyAndInfo(TreeNode* predNode, TreeNode* toRemove)
//T(N) = 6
//Big-O = O(1)
{
    KeyType tempkey;
    ElementType tempinfo;

    tempkey = predNode->key;
    tempinfo = predNode->info;

    predNode->key = toRemove->key;
    predNode->info = toRemove->info;

    toRemove->key = tempkey;
    toRemove->info = tempinfo;
}
```

```
hermod.ics.uci.edu - PuTTY
static TreeNode* remove(KeyType key, TreeNode* t)
//Big-O = O(N)
{
    TreeNode* toRemove = find(key, t);
    if(toRemove == nullptr)
    {
        return nullptr;
    }
    else if(toRemove->left == nullptr)
    {
        if(toRemove->right == nullptr)
        {
            return t;
        }
        else
        {
            TreeNode* succ = findSucc(toRemove);
            swap(succ->info, toRemove->info);
            swap(succ->key, toRemove->key);
            return t;
        }
    }
    else if(toRemove->right == nullptr)
    {
        TreeNode* pred = findPred(toRemove);
        swapKeyAndInfo(pred, toRemove);
        return t;
    }
    else
    {
        TreeNode* predecessorNode = findPred(toRemove);
        swapKeyAndInfo(predecessorNode, toRemove);
        toRemove = remove(key, toRemove);
        return t;
    }
    return nullptr;
}

static void print(ostream& out, TreeNode* t)
//Big-O = O(N)
{
    if(t)
    {
        out << '[';
        print(out, t->left);
        out << '(' << t->key << ',' << t->info << ')';
        print(out, t->right);
        out << ']';
    }
    else
    {
        out << "nullptr";
    }
}

static void deleteNode(TreeNode* t)
//T(N) = 1
//Big-O = O(1)
{
    // --nodesCreated;
    delete t;
}

static void deleteTree(TreeNode* t)
//Big-O = O(1)
{
    if(t)
    {
        deleteTree(t->left);
        deleteTree(t->right);
        deleteNode(t);
    }
}

142,2-16 63%
```

```
hermod.ics.uci.edu - PuTTY
public
    TreeNode* root;
    :
    SearchTree()
        : root(nullptr)
    {
    }

    map<int, int> countLength;

    SearchTree(const SearchTree& st) = delete;

    SearchTree& operator = (const SearchTree& st) = delete;

    int& operator[] (KeyType key)
    //T(N) = 8N + 3
    //Big-O = O(N)
    {
        TreeNode* temp = TreeNode::find(key, root); //6N + 2
        if(temp != nullptr) //N
        {
            return temp->info; //N
        }
        throw; //1
    }

    void insert(KeyType s, ElementType c)
    //T(N) = 3015N + 2005
    //Big-O = O(N)
    {
        countLength[s.size()]++; //1001N + 1000
        root = TreeNode::insert(s, c, root); //2014N + 1005
    }

    bool find(KeyType s)
    //T(N) = 6N + 4
    //Big-O = 2
    {
        TreeNode* temp;
        temp = TreeNode::find(s, root); //6N + 2

        if(temp)
        {
            return 1; //1
        }
        return 0; //1
    }

    void countLengths()
    //T(N) = 1001N + 1000
    //Big-O = O(N)
    {
        map<int, int>::iterator it; //1000
        for(auto const& c : countLength) //N
        {
            cout<<"length "<<c.first<<": "<<c.second<<" words"<<endl; //1000N
        }
    }

    void remove(KeyType s)
    //Big-O = O(N)
    {
        root = TreeNode::remove(s, root);
    }

    void count()
    {
        cout << TreeNode::count_nodes(root) << endl;
    }

    ~SearchTree()
    {
        TreeNode::deleteTree(root);
    }
}
```

291,1-8 99%

The 4 above is my both my TreeNode and Binary Search Tree implementations with T(N) and Big-O notations. The entire class is already templated.

```
hermod.ics.uci.edu - PuTTY
==1840== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==1840== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==1840== Command: main
==1840==
File: random.txt. Partition: 1/10. Function: insertAllWords. Time: 0.305568
File: random.txt. Partition: 1/10. Function: findAllWords. Time: 0.111724
File: random.txt. Partition: 1/10. Function: removeAllWords. Time: 0.153362

File: random.txt. Partition: 2/10. Function: insertAllWords. Time: 0.527261
File: random.txt. Partition: 2/10. Function: findAllWords. Time: 0.222611
File: random.txt. Partition: 2/10. Function: removeAllWords. Time: 0.304747

File: random.txt. Partition: 3/10. Function: insertAllWords. Time: 0.808984
File: random.txt. Partition: 3/10. Function: findAllWords. Time: 0.339185
File: random.txt. Partition: 3/10. Function: removeAllWords. Time: 0.461555

File: random.txt. Partition: 4/10. Function: insertAllWords. Time: 1.09726
File: random.txt. Partition: 4/10. Function: findAllWords. Time: 0.461206
File: random.txt. Partition: 4/10. Function: removeAllWords. Time: 0.633287

File: random.txt. Partition: 5/10. Function: insertAllWords. Time: 1.38503
File: random.txt. Partition: 5/10. Function: findAllWords. Time: 0.583748
File: random.txt. Partition: 5/10. Function: removeAllWords. Time: 0.796895

File: random.txt. Partition: 6/10. Function: insertAllWords. Time: 1.67861
File: random.txt. Partition: 6/10. Function: findAllWords. Time: 0.714777
File: random.txt. Partition: 6/10. Function: removeAllWords. Time: 0.965104

File: random.txt. Partition: 7/10. Function: insertAllWords. Time: 1.97292
File: random.txt. Partition: 7/10. Function: findAllWords. Time: 0.83455
File: random.txt. Partition: 7/10. Function: removeAllWords. Time: 1.13418

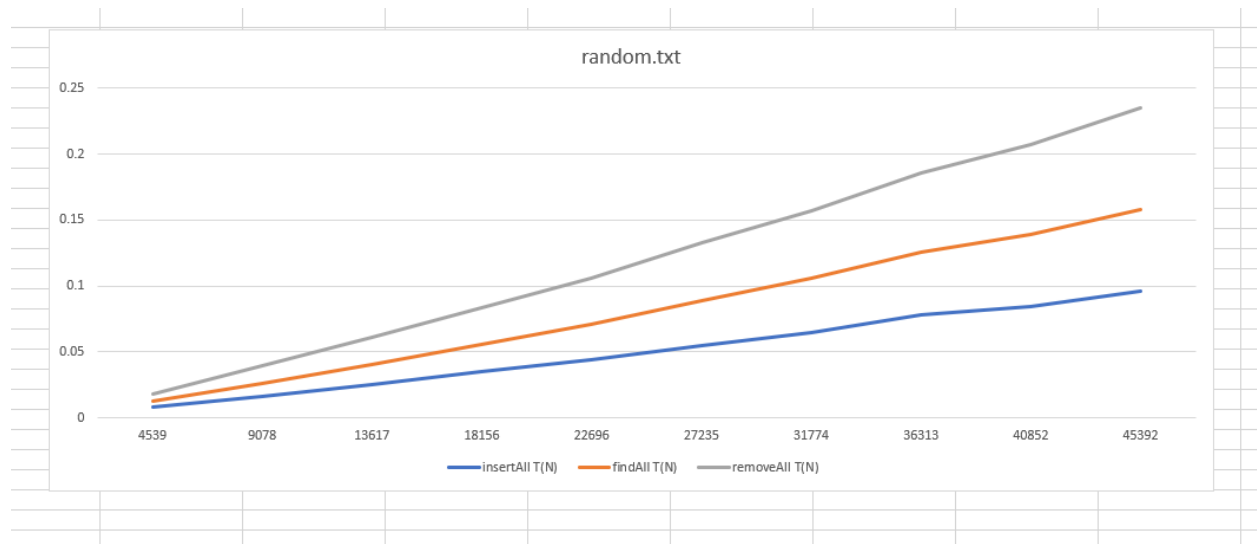
File: random.txt. Partition: 8/10. Function: insertAllWords. Time: 2.26834
File: random.txt. Partition: 8/10. Function: findAllWords. Time: 0.96795
File: random.txt. Partition: 8/10. Function: removeAllWords. Time: 1.30359

File: random.txt. Partition: 9/10. Function: insertAllWords. Time: 2.5675
File: random.txt. Partition: 9/10. Function: findAllWords. Time: 1.09972
File: random.txt. Partition: 9/10. Function: removeAllWords. Time: 1.47681

File: random.txt. Partition: 10/10. Function: insertAllWords. Time: 2.86732
File: random.txt. Partition: 10/10. Function: findAllWords. Time: 1.23128
File: random.txt. Partition: 10/10. Function: removeAllWords. Time: 1.64366

length 2: 49 words
length 3: 535 words
length 4: 2237 words
length 5: 4175 words
length 6: 6174 words
length 7: 7366 words
length 8: 7076 words
length 9: 6084 words
length 10: 4594 words
length 11: 3069 words
length 12: 1880 words
length 13: 1137 words
length 14: 545 words
length 15: 278 words
length 16: 103 words
length 17: 57 words
length 18: 23 words
length 19: 3 words
length 20: 3 words
length 21: 2 words
length 22: 1 words
length 28: 1 words
==1840==
==1840== HEAP SUMMARY:
==1840==   in use at exit: 0 bytes in 0 blocks
==1840==   total heap usage: 310,317 allocs, 310,317 frees, 16,994,934 bytes allocated
==1840==
==1840== All heap blocks were freed -- no leaks are possible
==1840==
==1840== For counts of detected and suppressed errors, rerun with: -v
==1840== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Above is my program executed with valgrind. It includes the times and the countLengths() function results. The classes are already templated.



Above is a picture of the graph.

```
hermod.ics.uci.edu - PuTTY
using namespace std;
#include <string>
#include <iostream>
#include <fstream>
#include "TreeNode.cpp"
#include "Timer.h"
#define NPartitions 10
#define NSamples 45392

void insertAll(ifstream& file, SearchTree<string, int>& S, int NWords)
{
    string word;
    for(int i = 0; i < NWords; ++i)
    {
        file >> word;
        S.insert(word, i);
    }
    // S.count();
}

void findAll(ifstream& file, SearchTree<string, int>& S, int NWords)
{
    string word;
    for(int i = 0; i < NWords; ++i)
    {
        file >> word;
        S.find(word);
    }
}

void removeAll(ifstream& file, SearchTree<string, int>& S, int NWords)
{
    string word;
    for(int i = 0; i < NWords; ++i)
    {
        file >> word;
        S.remove(word);
    }
    //S.countLengths();
}
```



```

int measure_TreeNode(string inputFileName, int NWords, int part)
{
    ifstream in(inputFileName);
    SearchTree<string, int> ST;
    Timer t1, t2, t3;
    double t1Time, t2Time, t3Time;

    t1.start();
    insertAll(in, ST, NWords);
    t1.elapsedUserTime(t1Time);

    in.clear();
    in.seekg(0, ios::beg);

    t2.start();
    findAll(in, ST, NWords);
    t2.elapsedUserTime(t2Time);

    in.clear();
    in.seekg(0, ios::beg);

    t3.start();
    removeAll(in, ST, NWords);
    t3.elapsedUserTime(t3Time);

    in.close();

    cout << "File: " << inputFileName << ". Partition: " << part << "/10. Function: insertAllWords. Time: "<< t1Time <<
endl;
    cout << "File: " << inputFileName << ". Partition: " << part << "/10. Function: findAllWords. Time: "<< t2Time <<
endl;
    cout << "File: " << inputFileName << ". Partition: " << part << "/10. Function: removeAllWords. Time: "<< t3Time <<
endl;
    cout << endl;

    return 1;
}

void printAllWords(string inputFileName)
{
    ifstream in(inputFileName);
    SearchTree<string, int> ST;

    insertAll(in, ST, NSamples);
    ST.countLengths();
}

int partitions(string inputFileName)
{
    for(int i = 1; i <= NPartitions; ++i)
    {
        //cout << i*NSamples/NPartitions << endl;
        measure_TreeNode(inputFileName, i*NSamples/NPartitions, i);
    }

    return 0;
}

int main()
{
    //cout << "Testing TreeNode" << endl;
    string file = "random.txt";
    partitions(file);
    printAllWords(file);

    return 0;
}

```

106,1

Bot

Above is my main.cpp with all the tests utilizing Timer.h to time each test function for the class and all templated as well.