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

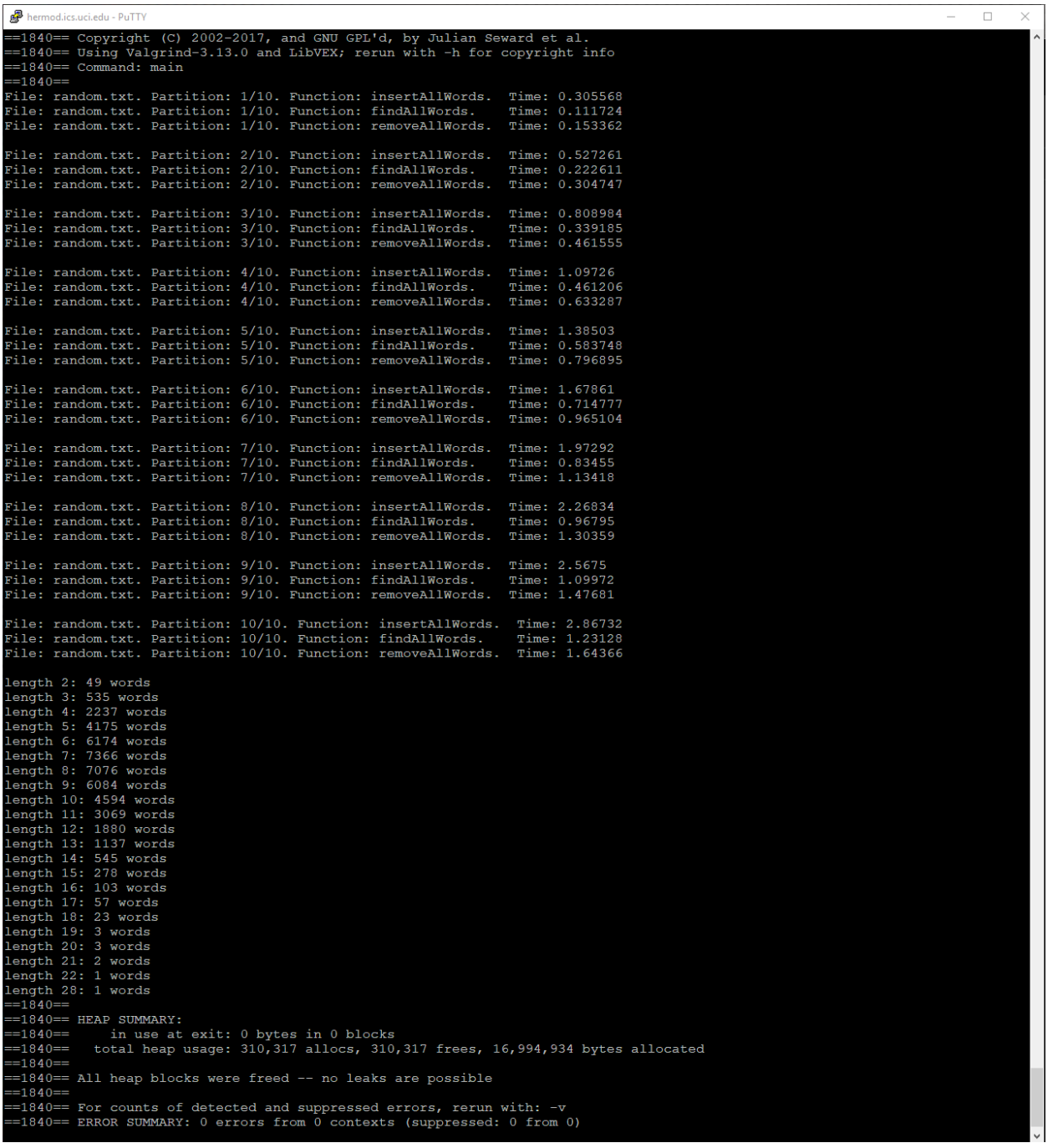
```
static TreeNode* find(KeyType key, TreeNode* t)
//T(N) = 6N + 2
//Big-O = O(N)
{
        TreeNode* temp = t;//1
        while(temp!=nullptr)//N
        {
                if(key < temp->key)//N
                {
                        temp = temp->left;//N
                }
                else if(key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

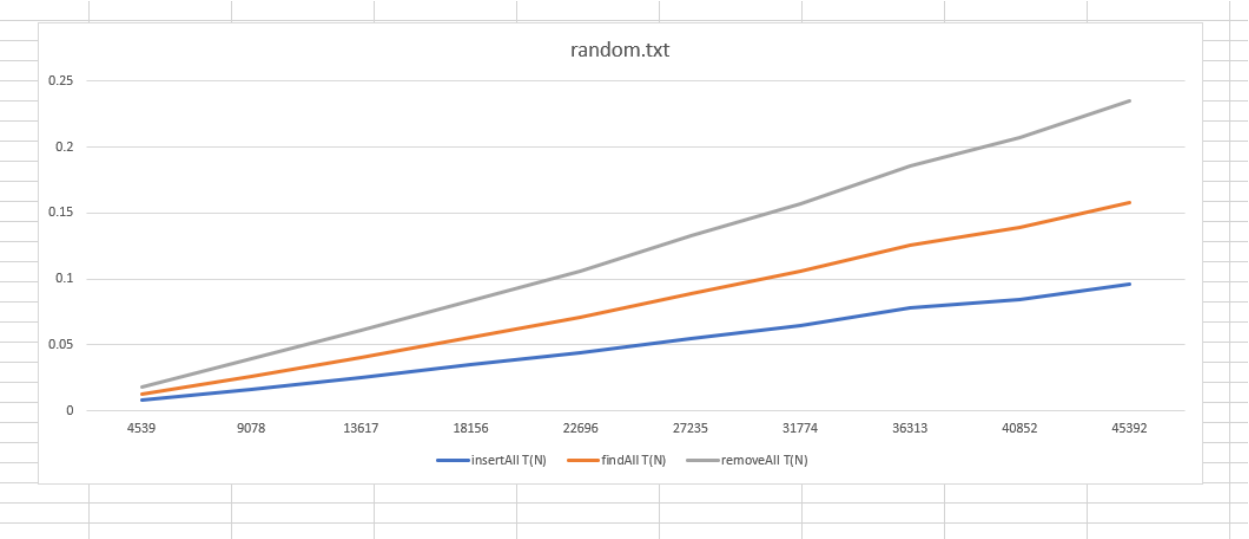
```
static TreeNode* remove(KeyType key, TreeNode* t)
//Big-O = O(N)
{
        TreeNode* toRemove = find(key, t);
        if(toRemove == nullptr)
        {
                return nullptr;
        }
        else if(toRemove->left == nullptr)
        {
                if(toRemove->right == nullptr)
                {
                        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%

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.



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.