Jacob Ciurca

Dr. Lu

Lab 2

Cosc 320

9/14/2020

**PreLab:** deSelsort function is working properly and ready to be implemented into this week’s lab. Also reviewed the design, implementation, and time complexity of linear and binary search.

**Lab:** Question:   For a random list of integers, what is the maximum number of comparisons required to find a target value by binary search?

Answer: Using binary search it takes a maximum of log2(n) comparisons in order to find a given element in an array. This is because every time you make a comparison you eliminate half of the remaining elements of the array in your search. For example, if you have an array with 1 billion elements it would only take you a maximum of 30 comparisons to find said element in the array which is astronomically better than the maximum comparisons for linear search being equal to the size of the array.

**Binsearch.h:**

#ifndef BINSEARCH\_H\_INCLUDED

#define BINSEARCH\_H\_INCLUDED

#include <iostream>

using namespace std;

int sumFailCom = 0;

int successTotal = 0;

void binSearch(int arr[], int left, int right, int target, int pass = 0){

if(right>=left){

int middle = left+(right-left) / 2;

if(arr[middle] == target){

successTotal = successTotal + 1;

return;

}

if(arr[middle] > target){

return binSearch(arr, left, middle-1, target, pass+1);

}

else{

return binSearch(arr, middle+1, right, target, pass+1);

}

}

sumFailCom = sumFailCom + pass;

}

#endif // BINSEARCH\_H\_INCLUDED

**deSelsort.h:**

#ifndef DESELSORT\_H\_INCLUDED

#define DESELSORT\_H\_INCLUDED

template <class T>

void double\_ended\_selection(T a[], int n){

T min;

T max;

int pass = 1;

for (int i = 0, j = n - 1; i < j; i++, j--) {

min = a[i];

max = a[i];

int min\_i = i, max\_i = i;

for (int k = i; k <= j; k++) {

if(a[k] > max) {

max = a[k];

max\_i = k;

} if(a[k] < min) {

min = a[k];

min\_i = k;

}

}

T temp = a[i];

a[i] = a[min\_i];

a[min\_i] = temp;

if(a[i] != a[min\_i]){

}

if (a[min\_i] == max){

T temp = a[j];

a[j] = a[min\_i];

a[min\_i] = temp;

if(a[i] != a[min\_i]){

}

}

else{

T temp = a[j];

a[j] = a[max\_i];

a[max\_i] = temp;

if(a[i] != a[max\_i]){

}

}

}

}

#endif // DESELSORT\_H\_INCLUDED

**Lab02.cpp:**

#include <iostream>

#include <stdlib.h>

#include <time.h>

#include <algorithm>

#include "binsearch.h"

#include "deSelsort.h"

using namespace std;

int main()

{

int arr[10000];

for(int i=0; i<10000; i++){

arr[i] = rand() % 100000; //set array to number between 0-99999

}

double\_ended\_selection(arr, 10000);

int n = sizeof(arr) / sizeof(arr[0]);

int target;

for(int i = 0; i<10000; i++){

target = rand() % 100000;

binSearch(arr, 0, n-1, target);

}

float averageCase = sumFailCom/(10000-successTotal);

cout << "Total number of successful searches out of 10000: " << successTotal << endl;

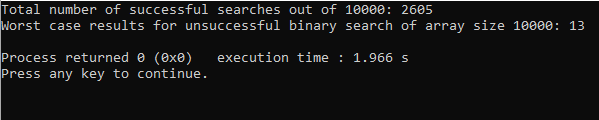
cout << "Worst case results for unsuccessful binary search of array size 10000: ";

cout << averageCase << endl;

return 0;

}

**Sample Output:**



Question: Does your empirical results verify your answer for the maximum number of comparisons required to find a target value by the binary search?

Answer: Yes, the results of the binary search test does verify my answer for the maximum number of comparisons required because log2(10,000) equals 13 which is the result we got as you can see above in the sample output.

**PostLab:** This lab was very good at showing us different search algorithms and showing us the usefulness of binary search when dealing with an array of an extremely large size. This lab took me approximately 2 hours and 15 minutes to complete. I completed this lab by myself with no help from my classmates.