Bohan Liu 915355219

Program 1 Measurement Analysis

The goal of this program is to compare the complexity between linear search and binary search. Before conducting any measurement, the task is estimated by the Big-O theory. Assume that there are j lines of data in newbooks.dat and i lines of data in request.dat. The linear search function is formed by two for-loop, according to the Big-O theory, the complexity of linear search is O(i*j) if the worst case. The binary search is constituted by an for-loop and a while-loop which runs half of j in the worst case, so the complexity of binary search should be O(i*log j). Then compare these two algorithm, i*j > i*log j when i > 0 and j > 0, which means the growth of linear search should always grater than binary search.

Therefore, I hypothesize that the running time of linear search is longer than binary search.

```
int linearSearch(newBooks, requiredBooks){
    int count = 0;
    for(int i = 0; i < requiredBooks.size(); i++){
        for(int j = 0; j < newBooks.size(); j++){
            … …
                count++;
        }
    }
    return count;
}
```

```
int binarySearch(newBooks_vector, reqBooks_vector){
    int count = 0;
    for(int i = 0; i < reqBooks.size(); i++){

        int low, high;

        while(low <= high){
            int mid = (low + high) / 2;
                if(reqBooks_vector[i] >
newBooks_vector[mid])
                        low = mid + 1;
                else if(reqBooks_vector[i] <
newBooks_vector[mid])
                high = mid – 1;
                else:
                        count++
    return count;
}
```

Then I test my program by four groups of data and each test three times. The first group has just a few of data, and the result is that the binary search running longer time than linear search.

create_testData 5 3			
Linear Search	0 matched 29.394 microseconds	0 matched 22.132 microseconds	0 matched 12.696 microseconds
Binary Search	0 matched 31.069 microseconds	0 matched 24.768 microseconds	0 matched 27.332 microseconds

There are more data added into the second group. The result is still the binary search running longer time than linear search. But the difference ratio are getting smaller.

create_testData 50 30			
Linear Search	0 matched 85.274 microseconds	2 matched 70.079 microseconds	1 matched 137.847 microseconds
Binary Search	0 105.785 microseconds	2 144.32 microseconds	1 191.325 microseconds

Group three given a big amount of data, and the result shows the binary search runs less time than linear search.

create_testData 500 300			
Linear Search	10 matched 2521.08 microseconds	7 matched 6398.74 microseconds	15 matched 8595.59 microseconds
Binary Search	10 matched 1817.23 microseconds	7 matched 1788.57 microseconds	15 matched 1102.01 microseconds

Finally, in group of huge data, it makes a huge difference between these two searching methods. And we see that the Binary search is absolutely faster.

create_testData 5000 3000			
Linear Search	104 matched 243055 microseconds	98 matched 243695 microseconds	83 matched 243634 microseconds
Binary Search	104 matched 16129.3 microseconds	98 matched 11896 microseconds	83 matched 17047.4 microseconds

Afterward, it is not clear about why linear search is a little faster than binary search when there is a few data, which is opposite to the hypothesis. One things I noticed is that before binary search, the data should be sorted by function std::stort() which has a complexity of O(j*log j). Overall, the complexity of binary search should be O(i*log j) + O(j*log j), which may haul the speed of binary search.

Also, it does not mean that linear search is useless. When there are a bunch of data that can not be sorted or difficult to sort, it is better and more convenient to use linear search.

On the other hand, if there is just a few data, it is also recommend to use linear search, since the test result shows linear search faster than binary search when there is just a few data.