Ryan Ellis
COSC 320 --- 001 Lab 2 2/11/2025

Lab report: In this lab, the binary search algorithm was implemented along with the previously created double-ended selection sorting algorithm from Lab 1, to ascertain the worst-case timing complexity for the binary search by finding the maximum number of comparisons needed for searching values in an array. This lab took me about an hour to complete, cumulatively.

Pre-lab: Review and understand the implementation and worst-case time complexity of each of the following algorithms:

- Linear Search
- Binary Search

Lab 1.1

Exercise 1:

Header File

```
#ifndef BINSEARCH_H
#define BINSEARCH_H

#include <utility>
#include <vector>
#include <iostream>

using namespace std;

template <class T>
void deSelsort( T arr[], int size);
template <class T>
void printArray(T arr[], int size);
template <class T>
T binSearch(const T arr[], int size, T val, T &p);      //pass in tracking variable by ref?

#endif
template <class T>
void printArray(T arr[], int size ){        //function to print array
   for(int i = 0; i < size; i++){
         cout<< arr[i]<< " ";
   }
   cout<<endl;
}
template <class T>
```

```
T binSearch(const T arr[], int size,  T val , T &p){
  //int pass = 0;    inital thought to have variable to hold passes, instead use parameter by
ref
  int first = 0;
  int last = size - 1;
  int middle = 0;

  while (first <= last) {
    middle = (first + last) / 2;
    if (arr[middle] == val) {
      return middle;
    } else if (arr[middle] > val)
      last = middle - 1;
    else
      first = middle + 1;
    p++;
  }
  return -1;
}

template <class T>
void deSelsort(T arr[], int size){          //double ended selection sort

    int minIndex, maxIndex;             // min and max index, right side set at end of array
    int right = size -1;

    for(int left = 0; left < right; left++, right--){       //for loop to end once left and right
meet, increase left, decrease right

        minIndex = left;
        maxIndex = right;          //set min to left, max to right

        for(int index = left; index <= right; index++){     // nested for loop with index set to
left, less than or equal to right

            if( arr[index] < arr[minIndex]){            //conditional statements to set index to
min, max dependent on greater/less than
                minIndex = index;
            }
            if(arr[index] > arr[maxIndex]){
                maxIndex = index;
            }
        }

        swap(arr[left], arr[minIndex]);         //swap values
```

```
        if(maxIndex == left){
           maxIndex = minIndex;          //set max index to min if it was at left before swap
        }

        swap(arr[right], arr[maxIndex]);

        //cout<< "Pass " << left + 1<< ": ";      //print array at pass
        //printArray(arr,size );

     }

}
```

Implementation File (main program)

```
#include "binSearch.h"
#include <iostream>
#include <ctime>

using namespace std;

void div();     //prototypes
void setArray(int arr[], int size);

const int ARRSIZE = 10000;          //global constants
const int RANDOMLIMIT = 99999;
const int RANDOMVALUES = 10000;

int main(){

   srand(time(0));     //set random seed

   int array[ARRSIZE];    //array and counter variables
   int sumFailCom = 0;
   int sumSucCom = 0;
   int successTotal = 0;
   int count = 0;
   int flag = 0;          //variable flag for -1 value

   setArray(array,ARRSIZE);
   deSelsort(array,ARRSIZE);  // set and sort array

   while (count < RANDOMVALUES){        //while loop to run through search runs
      flag = 0;                  //set flag to 0
      int passes = 0;
```

```
        flag = binSearch(array, ARRSIZE, rand()%RANDOMLIMIT, passes);
//search for random value and set flag

        if(flag == -1)      //conditional statements for flag value
           sumFailCom += passes;   //add unsuccessful passses
        if(flag > 0){
           sumSucCom += passes;    //add unsuccessful passes
           successTotal++;         // increment success total

        }
        count++;           //increase count
        //passes = 0;     //reset passes
    }

    cout<<"Sum of Failed Comparisons : "<<sumFailCom<<endl;
    cout<<"Sum of Successful Comparisons : "<<sumSucCom<<endl;
    cout<<"Successful Searches Total : "<<successTotal;

    div();

    cout<<"RESULTS";
    div();
    cout<<"Worst-case comparison for unsuccessful binary search:
"<<sumFailCom/(RANDOMVALUES-successTotal);
    div();
    cout<<"Worst-case comparison for successful binary search:
"<<sumSucCom/(successTotal);
    div();



    return 0;

}
void div(){
    cout<<"\n================================"<<endl;     //function to print
divide line
}

void setArray(int arr[], int size){
    int val = 0;
    for(int i = 0; i < size; i++){
        val = rand()%RANDOMLIMIT;
        arr[i] = val;
    }
}
```

```
 cd '/home/ryan/Documents/COSC 320/Labs/Lab 2/Lab 2'
ryan@ryan-MacBookPro:~$  cd '/home/ryan/Documents/COSC 320/Labs/Lab 2/Lab 2'
ryan@ryan-MacBookPro:~/Documents/COSC 320/Labs/Lab 2/Lab 2$ make
make: 'prog' is up to date.
ryan@ryan-MacBookPro:~/Documents/COSC 320/Labs/Lab 2/Lab 2$ ./prog
Sum of Failed Comparisons : 121329
Sum of Successful Comparisons : 10447
Successful Searches Total : 917
==============================
RESULTS
==============================
Worst-case comparison for unsuccessful binary search: 13
==============================
Worst-case comparison for successful binary search: 11
==============================
```

```
ryan@ryan-MacBookPro:~/Documents/COSC 320/Labs/Lab 2/Lab 2$ ./prog
Sum of Failed Comparisons : 120403
Sum of Successful Comparisons : 10949
Successful Searches Total : 970
==============================
RESULTS
==============================
Worst-case comparison for unsuccessful binary search: 13
==============================
Worst-case comparison for successful binary search: 11
==============================
```

```
ryan@ryan-MacBookPro:~/Documents/COSC 320/Labs/Lab 2/Lab 2$ ./prog
Sum of Failed Comparisons : 121291
Sum of Successful Comparisons : 10374
Successful Searches Total : 925
==============================
RESULTS
==============================
Worst-case comparison for unsuccessful binary search: 13
==============================
Worst-case comparison for successful binary search: 11
==============================
ryan@ryan-MacBookPro:~/Documents/COSC 320/Labs/Lab 2/Lab 2$
```

Lab Questions:

For a random list of integers, what is the maximum number of comparisons required to find a target value by binary search? Please elaborate your answer.

Because the time complexity of binary search is O(log(n)) we can expect that the maximum number of comparisons required to find a target value would be equivalent to log(n) where n is the number of elements in the array to be searched.

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

The results conclude in tandem with the hypothesis that the number of maximum comparisons would be equivalent to $\log_2(n)$ where, in this case n = 10,000, which is equal to approximately 13.29 and the worst-case comparison for unsuccessful comparisons was 13, rounded.