ICS4U Module 6: Note & Exercise 2a

Binary Search

- The main reason to sort arrays is to perform a more efficient search!

- The *Linear Search* algorithm was introduced in Module 5 (Note 1c – Searching an Array)
    - Also called a *sequential search*
    - Much less efficient than a binary search
    - Does not require a sorted list (as binary search does)

Binary Search

- Like the mergesort algorithm, the binary search algorithm uses a divide-and-conquer approach
- The Binary Search requires an array sorted low to high

Binary Search Algorithm

- Examine the middle item of an array
- Determine if this is the item sought
    - If not, is it above or below the middle term
    - If below: a new binary search is applied to the lower half
    - If above: a new binary search is applied to the top half
- Continue until item is reached or there are no values left to be considered
- Very efficient
    - An array of 100 elements checks no more than 8 elements in a search
    - An array of 1 000 000 elements checks no more than 20 elements in a search
    - An array of the population of the world would check no more than 40 times to find any one person!!!

Binary Search Pseudocode

```
if (goal == items[mid]) {
      return(mid)
} else if (goal < items[mid]) {
      return(binarySearch(lowerhalf))
} else {
      return(binarySearch(upperhalf))
}
```

ICS4U Module 6: Note & Exercise 2a

Binary Search Implementation

The Searches class implements a Binary Search:

```
/*
 * Searches.java
 * A class that implements searching algorithms.
 * Lawrenceville Press
 * June 10, 2005
 */

 public class Searches{

      /**
       * Searches items array for goal
       * pre: items is sorted from low to high
       * post: Position of goal has been returned,
       * or -1 has been returned if goal not found.
       */
      public static int binarySearch(int[] items, int start, int end, int
goal) {

            if (start > end) {
                  return(-1);
            } else {
                  int mid = (start + end) / 2;
                  if (goal == items[mid]) {
                        return(mid);
                  } else if (goal < items[mid]) {
                        return(binarySearch(items, start, mid-1, goal));
                  } else {
                        return(binarySearch(items, mid+1, end, goal));
                  }
            }
      }
}
```

The TestSorts application has been modified to sort an array of integers and then prompt the user for a number to search for:

```
 /**
  * Sort algorithms are tested.
  */

 import java.util.Scanner;

 public class TestSorts {

      public static void displayArray(int[] array) {
            for (int i = 0; i < array.length; i++) {
                  System.out.print(array[i] + "  ");
            }
            System.out.println("\n");
      }

      public static void sortIntArray() {
            Scanner input = new Scanner(System.in);
            int numItems, searchNum, location;
            int[] test;

            System.out.print("Enter number of elements: ");
            numItems = input.nextInt();

            /* populate array */
            test = new int[numItems];
            for (int i = 0; i < test.length; i++) {
                  test[i] = (int)(101 * Math.random());
            }
            System.out.println("Unsorted:");
            displayArray(test);

            //Sorts.selectionSort(test);
            //Sorts.insertionSort(test);
            Sorts.mergesort(test, 0, test.length - 1);

            System.out.println("Sorted:");
            displayArray(test);

            /* search for number in sorted array */
            System.out.print("Enter a number to search for: ");
            searchNum = input.nextInt();
            while (searchNum != -1){
                  location = Searches.binarySearch(test, 0, test.length-1,
searchNum);
                  System.out.println("Number at position: " + location);
                  System.out.print("Enter a number to search for: ");
                  searchNum = input.nextInt();
            }
      }


public static void main(String[] args) {
            sortIntArray();
      }
```

Output similar to:

```
Enter number of elements: 15

Sorted:
1  4  4  10  14  15  16  27  37  46  47  56  58  59  95

Enter a number to search for: 47
Number at position: 10
Enter a number to search for: 4
Number at position: 1
Enter a number to search for: 14
Number at position: 4
Enter a number to search for: -1
```

Programming Exercise:

Create a BinaryLocater application that displays the positions examined during a binary search. The application output should look similar to:

```
Enter number of elements: 100
Unsorted:
37  44  72  22  17  95  31  79  3  16  42  65  61  5  37  74  54  60  37  83
54  93  14  68  26  61  78  56  20  25  99  95  85  35  12  96  81  47  64
46  47  80  2  88  14  51  21  58  11  79  46  97  46  96  86  61  62  2  9
33  97  72  25  67  63  17  63  4  81  93  56  70  79  87  28  89  40  62  57
61  82  42  90  5  48  43  19  5  38  51  33  89  26  39  46  36  74  100  5
60

Sorted:
2  2  3  4  5  5  5  5  9  11  12  14  14  16  17  17  19  20  21  22  25  25
26  26  28  31  33  33  35  36  37  37  37  38  39  40  42  42  43  44  46
46  46  46  47  47  48  51  51  54  54  56  56  57  58  60  60  61  61  61
61  62  62  63  63  64  65  67  68  70  72  72  74  74  78  79  79  79  80
81  81  82  83  85  86  87  88  89  89  90  93  93  95  95  96  96  97  97
99  100

Enter a number to search for: 39
Examining 49
Examining 24
Examining 36
Examining 30
Examining 33
Examining 34
Number at position: 34

Enter a number to search for:
```

Submit your source code to the Google Doc “ICS4U – Activity Submission Form”