Programming Assignment 1

CSCD300 Data Structure

Instructor: Dr. Bojian Xu
Eastern Washington University, Cheney, Washington

Due: 11:59pm, January 9, 2015 (Friday)

Please follow these rules strictly:

1. Verbal discussions with classmates are encouraged, but each student must independently write his/her own work, without referring to anybody else's solution.
2. No one should give his/her code to anyone else.
3. The deadline is sharp. Late submissions will **NOT** be accepted (it is set on the Canvas system). Send in whatever you have by the deadline.
4. Every source code file must have the author's name on the top.
5. All source code should be commented reasonably well.
6. Sharing any content of this assignment and its keys in any way with anyone who is not in this class of this quarter is NOT permitted.

Note: we use 0-based indexing.

Project: Finding the number of duplicates in a sorted array

In the class, we have discussed the binary search algorithm that can quickly find the location of a given number x in a sorted array A of numbers, if x exists in the A. Otherwise, the binary search algorithm can also quickly detect the inexistence of x. Note that it is possible that there can be duplicates of any particular value in the given array A. For example, the given array A could be like the following.

$$A[\,] = \{2, 5, 5, 5, 7, 9, 9, 10, 15, 15, 17, 19\}$$

where each of the values 5, 9, and 15 occurs more than once. In this case, if we use the binary search to find the location of a given number x which occurs more than once in the array, we will ONLY be able to find ONE location of x.

This programming assignment aims to modify the basic binary search algorithm, such that the modified binary search algorithm can find all the duplicates of the given number x in the given array A, by reporting the locations of the leftmost and the rightmost occurrences of the given number x in the given array A. For example, suppose we are given the above array A, then:

- If $x = 5$, the modified binary search should print $[1, 3]$, indicating 5 occurs at $A[1...3]$.
- If $x = 10$, the modified binary search should print $[7, 7]$, indicating 10 occurs at $A[7...7]$.
- If $x = 6$, the modified binary search should print $[-1, -1]$, indicating 6 does not occur in A.

An inefficient idea

Before you start programming, you first need to figure out how to modify the binary search algorithm for the given task. Note that your new algorithm must still have a time complexity of $O(\log n)$, where n is the size of the given array A. The following idea for modifying the lectured binary search will not work well in the worst case and thus should not be adopted:

```
You first use the basic binary search algorithm to find one location
of $x$ in the array $A$. Say that location is $i$.  Then you just walk
to the left and to the right within the array $A$, starting from the
location $i$. Your walk will stop in either direction whenever you
observe the array element that is different from $x$ or you have
reached the boundary of the array $A$. By doing such walking, you will
be able to find the location of the leftmost and the rightmost
occurences of $x$ in $A$.
```

However, the above idea does not work well in the worst case. Suppose x occupies the majority portion of A, then in order to finish your walking, you would have to walk through the majority of the array A by spending time linear of the length of A, which is order of magnitudes slower than $O(\log n)$, where n is the length of the array A. So, you need to figure out a better idea for modifying the basic binary search.

Specification of your program

1. Your program should be named as: `Test_BinarySearchDup.java`
2. The input and output of your program.

   There are two inputs of your program. One is a text file, where each line is an integer number. All the numbers in the text file are already sorted in the ascending order. The file name should be supplied to your program as a command line parameter. The other input is the integer value for the given number x, which should also be supplied to your program as a command line parameter. For example, if we supply $x = 10$ and the given file named `data.txt` has the following content

   ```
   2
   2
   5
   6
   8
   10
   10
   11
   21
   25
   26
   ```

   Then, the command line you type would be:

   ```
   $java Test_BinarySearchDup data.txt 10
   ```

   and the output should be:

   ```
   $[5,6]
   ```

3. The specification of the function that implements the modified binary search algorithm is:

```
public static Pair BinarySearchDup(int[] A, int x)
```

where (1) the array A stores the data that the program reads from the given input data file, (2) x stores the value that is supplied by the command line, and (3) `Pair` is a class that you can define as follows in order for you to return two integers by the above function [1]. The returned value of the above function is the pair of the indexes of the leftmost and the rightmost occurrences of x in A, if x does exists; otherwise, the returned value of the above function will be a pair of -1.

```
class Pair{
   public int left;
   public int right;
}
```

Submission

- All your work files must be saved in one folder, named: **firstname_lastname_EWUID_cscd300_prog1**
  (1) We use the underline '_' not the dash '-'.
  (2) All letters are in the lower case including your name's initial letters.
  (3) If you have middle name(s), you don't have to put them into the submission's filename.
  (4) If your name contains the dash symbol '-', you can keep them.
- You need to include a pure ascii text file in the above folder, which contains the description of the algorithm that you use for modifying the binary search.
- You then compress the above whole folder into a .zip file.
- Submit .zip file onto the Canvas system by the deadline.

[1] You can of course use any other way that you like to return two integers.