Programming Assignment 2

CSCD300 Data Structure

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

Due: 11:59pm, January 16, 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 best trading strategy

Imagine you have successfully finished your computer science degree at Eastern and have started working in the IT department of a finance company at Wallstreet. On your first day with that company, your tough boss gave you the following computational challenge and asked you to present an efficient solution to him.

Computational Challenge. The company has a massive text file, which can be loaded into the main memory that you will be working on. The file has n lines of integer numbers, recording the prices of a particular stock over the past n days. The integer number on the kth line, denoted as $p(k)$, represents the price of the stock on the kth day, $0 \leq k \leq n-1$. Let's assume all the numbers in the file are **DISTINCT**. Your boss wants you to find two numbers in and out, $0 \leq in \leq out \leq n-1$, such that

$$p(out) - p(in) = \max\{p(j) - p(i) \mid 0 \leq i \leq j \leq n-1\}$$

That is, you want to find two days to have the stock purchased and sold such that the profit from trading this stock is maximized. Note that (1) there might be multiple such pairs of in and out, but you only need to find one of them in the case there are multiple choices. (2) in and out could even be the same, for example, in the case that the prices are going down all the way over those n days.

An inefficient idea

Before programming, you first need to figure out a time-efficient computational strategy to solve the challenge. The following trivial idea, which examines every pair of two days and returns the pair that gives the highest profit, will not work well when the number of lines in the file is large.

```
Read in the numbers from the file and save them in an array p[0...n-1];
in = 0; out = 0; max_profit = 0;
for(i = 0; i < n; i++)
   for(j = i; j < n; j++)
      if(p[j] - p[i] > max_profit)
         in = i;
         out = j;
         max_profit = p[j] - p[i];
return (in, out, max_profit);
```

Obviously, the time complexity of the above code is $O(n^2)$, since it has two nested `for` loops and each has size bounded by n. Therefore, the code will be very slow when the value of n is large.

Your goal

Your goal is to find an interesting **recursion-based** computational strategy, such that the overall time cost of your solution is $O(n \log n)$, which is extremely faster than the trivial idea above.

Specification of your program

1. Your program should be named as: `Test_BestTrading.java`

2. The input and output of your program.

   The input to your program is a text file, where each line is an integer number. All the numbers in the text file are guaranteed to be DISTINCT. The file name should be supplied to your program as a command line parameter. For example, if we supply the file named `data.txt` that has the following content

   ```
   9
   1
   5
   ```

   Then, the command line you should type would be:

   ```
   $java Test_BestTrading data.txt
   ```

   and the output should be:

   ```
   $[1,2,4]
   ```

   telling us that we should have bought the stock on day 1 and sold it on day 2, so that the profit is maximized as 4. Note that we use the 0-based indexing here.

3. The specification of the function that implements your algorithm is

```
public static Trade BestTrade(int[] p)
```

where (1) the array p[ ] stores the data that the program reads from the given input data file, and (2) `Trade` is a class that you can define as follows in order for you to return three integers by the above function [1].

```
class Trade{
   public int in;
   public int out;
   public int profit;
}
```

Submission

- All your work files must be saved in one folder, named: **firstname_lastname_EWUID_cscd300_prog2**
  (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. Explain why your algorithm's time cost is $O(n \log n)$.
- 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 three integers.