CPSC 4100 HW #2 – Programming Assignment, Fall 2022

Divide-and-Conquer

26 points

Date assigned: Wednesday, 09/28/2022

Due: 11:59 pm, Wednesday, 10/05/2022

This programming assignment is focused on Divide-and-Conquer algorithm design techniques. Unless otherwise specified, you need to tackle the problems using Divide-and-Conquer. You will lose at least 50% of points if you do not use Divide-and-Conquer for any problem in this assignment.

1. Problem: Counting Transactions Yielding a Capital Gain

Rabbit Jerry is interested in a stock RABJ and he downloads a list of N ($2 \le N \le 10^5$) stock prices in the order of trading dates. The stock prices are integer numbers which could be a negative integer in the rabbit stock market though. If he buys RABJ on a trading day and then sells it later on another trading day (buying and selling cannot happen on the same trade date), can you help him count the number of transactions that yields a capital gain. A transaction includes a pair of the purchase and sale prices.

The yield of a transaction is calculated by subtracting the purchase price from the sale price. A positive result means a transaction has a capital gain while a negative result means it has a loss.

Output the total number of transactions that yields a capital gain.

INPUT FORMAT (input arrives from the terminal / stdin):

First line contains N.
The next line contains N space-separated integers denoting the input sequence. The price ranges from -1,000,000 to 1,000,000.

OUTPUT FORMAT (print output to the terminal / stdout):

A single line containing the total number of transactions that yield a capital gain. Note that the output can be very large and cause integer overflow. So, you should use C++ **long long**.

C++ PROGRAM NAME:

The C++ program name must be hw2_prob1.cpp

SAMPLE INPUT:

```
7
100  7  19  34  1000  45 78
```

SAMPLE OUTPUT:

```
14
```

SCORING:

- Test cases 1-5 satisfy N⩽1000.
- Test cases 6-10 satisfy N⩽100,000. An $O(N^2)$ or worse algorithm is regarded to be failures of these test cases, even though the program produces the correct results. We consider them timeouts.

2. Problem: Maximum Capital Gain

Rabbit Jerry wonders if he can only make one transaction given the N ($2 \le N \le 10^5$) stock prices of RABJ as descried in problem 1, what is the maximum capital gain?

Output the maximum capital gain for that transaction.

INPUT FORMAT (input arrives from the terminal / stdin):

First line contains N.
The next line contains N space-separated integers denoting the input sequence. The price ranges from -1,000,000 to 1,000,000.

OUTPUT FORMAT (print output to the terminal / stdout):

A single line containing the maximum capital gain.

C++ PROGRAM NAME:

The C++ program name must be hw2_prob2.cpp

SAMPLE INPUT:

```
7
100  7  19  34  1000  45 78
```

SAMPLE OUTPUT:

```
993
```

SCORING:

- Test cases 1-5 satisfy N⩽1000.
- Test cases 6-10 satisfy N⩽100,000. An O(N^2) or worse algorithm is regarded to be failures of these test cases, even though the program produces the correct results. We consider them timeouts.

3. Problem: Maximum Subarray Sum

Rabbit Jerry downloads the list of N ($2 \le N \le 10^5$) stock prices of RABJ and stores them into a circular buffer/array in a clockwise manner (i.e., the first stock price follows the N-th stock price on the circular buffer) starting from index position 0. It is worth pointing out that the stock prices can be negative numbers in the rabbit stock market. He is curious about the maximum subarray sum over this circular buffer. Certainly, the maximum subarray can possibly straddle over both index positions 0 and N-1.

Output the maximum subarray sum. The maximum subarray must contain at least one element.

INPUT FORMAT (input arrives from the terminal / stdin):

First line contains N.
The next line contains N space-separated integers denoting the input sequence. The price ranges from -1,000,000 to 1,000,000.

OUTPUT FORMAT (print output to the terminal / stdout):

A single line containing the maximum subarray sum. Note that the output can cause integer overflow. So you should use C++ **long long**.

C++ PROGRAM NAME:

The C++ program name must be hw2_prob3.cpp

SAMPLE INPUT:

```
7
100  -7  19  -34  1000  -45 78
```

SAMPLE OUTPUT:

```
1156
```

SCORING:

- Test cases 1-5 satisfy $N \leqslant 1000$.
- Test cases 6-10 satisfy $N \leqslant 100{,}000$. An $O(N^2)$ or worse algorithm is regarded to be failures of these test cases, even though the program produces the correct results. We consider them timeouts.

4. Test Case Files and Program Execution

For each problem, you are provided with a list of 10 test cases. Each test case comes with two separate files, one contains the inputs and the other contains the outputs.

For example, the package named prob1_tests.tar contains 10 input and output files for Problem 1. You need to unpack it on cs1 by running the following command:

```
tar  -xvf  prob1_tests.tar
```

The input files are: 1.in, 2.in, …, 10.in.
The corresponding output files are: 1.out, 2.out, …, 10.out.

How to run your program?
Assume your executable for problem 1 is **hw2_prob1**. In order to use the test cases to test your program. You need to run the program as follows:

```
cat  1.in  | ./hw2_prob1 > my.out
```

If you have taken CPSC 3500, you should be familiar with the above symbols | and >.

Then, you need to check whether the program output matches the provided counterpart. Runt the following command:

```
diff    1.out   my.out
```

If nothing is displayed on the terminal, it means that your program produces the correct output for this test case.

5. What to Submit?

- C++ program source files: hw2_prob1.cpp, hw2_prob2.cpp, and hw2_prob3.cpp.
- Makefile: one single makefile produces three executables hw2_prob1, hw2_prob2, and hw2_prob3, corresponding to the above three source files.
- README: a text file containing the following:
    - Name