WESTERN NEW ENGLAND UNIVERSITY
COLLEGE OF ARTS AND SCIENCES
CS 366 Design and Analysis of Algorithms September 29, 2023

Programming Assignment #1
Due Date: October 16, 2023

Objectives:

- write a program that solves a problem in two different ways (1) using brute force and (2) using divide-and-conquer
- compare the theoretical and actual running times for the two methods of solving the problem

Directions: Suppose you work for an investment company that carefully studies past data for the price of a particular stock over an n-day period. The days are numbered 0, 1, 2, …, $n - 1$ and $p(i)$ represents the price per share for the stock on day i. If a broker buys on day j and then sells on day k, where $j < k$, then the profit or loss is $p(k) - p(j)$.

Example: Consider the following data for $n = 10$:

Day	0	1	2	3	4	5	6	7	8	9
Price per share	$40	$41	$48	$53	$56	$49	$42	$36	$39	$43

If a broker buys on day 1 and sells on day 4, the profit is $56 – $41 = $15. If a broker buys on day 3 and sells on day 6, the loss is $42 – $53 = – $11. You must determine which days j and k (where $0 \le j < k \le n - 1$) a broker should buy and sell, respectively, to maximize the profit. To get one unique answer, if there is more than one pair of numbers j, k with the same maximum profit, report the pair with the earliest day j, and (if there is still more than one pair) the earliest day k.

1. **Input**: The data must be read from an input file. The first line of the data file will be a single integer, `n`, representing the number of days of data. This will be followed by `n` lines of data, one value per line, representing the stock prices. You may assume that the stock prices are of type `int`. Store the stock prices in a primitive array of type int. Do NOT use ArrayList as there are hidden costs involved!

2. **Method of Solving and Output**: You must solve this problems in two ways. You may NOT use any built-in library functions such as `add`, `remove`, `min`, or `max`.

   1) **Brute force**: calculate the profit/loss for each pair of days j and k, $0 \le j < k \le n - 1$, then output the days j and k, with the maximum profit. Use the system time function `System.nanoTime()` to determine the actual elapsed time in nanoseconds for this method of solving the problem. Output the elapsed time for this method.

   2) **Divide-and-conquer**: use the divide-and-conquer design technique to solve this problem recursively. Output the days j and k, with the maximum profit.
   Hint: Split the array into two halves. The optimal solution (days j and k that give the maximum profit) will be the best of three possible options:
      - the largest profit from the left half (both j and k fall within the left half),
      - the largest profit from the right half (both j and k fall within the right half), or

- the largest profit splits across the boundary: (the index *j* is in the left half and *k* is in the right half; for this case, think how you can easily determine what *j* and *k* should be without having to calculate and compare any profits)

Use the system time function `System.nanoTime()` to determine the actual elapsed time in nanoseconds for the divide-and-conquer method of solving the problem. Output the elapsed time for this method.

3. **Analysis (big theta)**: Analyze the theoretical running times.
   a. Find a theta notation for the brute force method.
   b. Write a recurrence for the divide-and-conquer method. Use the Master Theorem to solve the recurrence.
   c. In theory, which method should be faster?

4. **Analysis (elapsed clock time)**: Report the actual running times (elapsed time for each using system time function).
   a. Run your program and experiment with various input sizes. (Testing suggestion: You can create data sets of various sizes by writing another program to randomly generate `n` integers and write them to a file.)
   b. Make a table (in Word or Excel) that shows the results (values of n, j, k, maximum profit and elapsed times) for each of the two methods.
   c. For which input sizes does it appear that the brute force method is faster? For which input sizes does it appear that the divide-and-conquer method is faster? Explain why you think this might be the case.

5. **Documentation:** Comment your source code! Include a header that contains your name, date, title and description of assignment. Include comments for each method that describe the purpose, parameters and return values. (Consider using style similar to Javadoc).

6. **Submission Instructions:** To submit your assignment, create a folder named ONLY with **your** last name (e.g., if your last name is Smith, create a folder named **Smith**). Inside the folder, include your source code (.java or .cpp file). Include file(s) that contain your answers to parts 3 and 4 of the assignment. Zip the file and upload it to the Kodiak Assignments folder for Program 1.

Sample input:

```
8
12
19
10
30
35
80
20
43
```

Sample Output:

```
Brute force:
Buy on day:  2
Sell on day: 5
Max profit:  70
Time elapsed: 1900 nanoseconds

Divide-and-conquer:
Buy on day:  2
Sell on day: 5
Max profit:  70
Time elapsed: 8900 nanoseconds
Press any key to continue . . .
```