Assignment 3

Julien Corman, Tiziano Dalmonte, Werner Nutt

Divide and Conquer and Other Array Algorithms

Instructions: This coursework comprises three regular questions, which contribute to the 45% of the final mark determined by the coursework, and a last question, which you may find more difficult than the other ones in this assignment.
For the additional question you will receive extra credit: it will contribute to the same extent as a regular 40 Points question.
The first question is conceptual only, while each of the other three requires a conceptual answer and a piece of Java code. The conceptual questions must be answered in a PDF document.
Use only data structures and methods known from the course.

1. A Mysterious Procedure

Consider the following procedure that takes as input an array and two indices.

```
1: procedure MYSTERY(int[] A, int l, int r)
2:     range := r − l + 1
3:     subrange := ⌈2 · range/3⌉
4:     if range = 2 and A[l] > A[r] then
5:         SWAP(A, l, r)
6:     else if range ≥ 3 then
7:         MYSTERY(A, l, l + subrange − 1)
8:         MYSTERY(A, r − (subrange − 1), r)
9:         MYSTERY(A, l, l + subrange − 1)
```

Note that division in line 3 is division of real numbers and recall that the ceiling function $\lceil x \rceil$ returns the least integer that is greater or equal to x.

1. What effect does the call MYSTERY$(A, 1, A.\text{length})$ have on an array A?
   Explain your answer.

   Hint: Since MYSTERY is a recursive procedure, you need to provide an inductive argument, which may have the following shape.

   You want to show that the array segment $A[l..r]$ has a certain property P after MYSTERY(A, l, r) is executed. Your argument may consist of two parts:

   - **Base case**: show that a segment having length 1 or 2 has this property P after MYSTERY is executed.
   - **Inductive case:** assume that the array segment in line 7 has property P after MYSTERY is executed, and similarly for lines 8 and 9. And show that under this assumption, the segment $A[l..r]$ must have property P at the end of the execution of the method.

2. What is the asymptotic running time of MYSTERY? Explain your answer.
   Hint: use an appropriate technique from the lecture.

(Weight: 30% of this CW)

2. Matching Pairs

Let s be an integer and A an array of integers. Then A has a *matching pair* for s if there are two distinct positions i, j such that $A[i] + A[j] = s$. The matching pair problem consists in checking, given A and s, whether A has a matching pair for s. Your goal is to develop an *efficient* algorithm that solves the matching pair problem.

1. Describe in words your idea for an algorithm for this task.
2. Write this algorithm in pseudocode or Java.
   Hint: an algorithm that you know from the lecture may be useful.
3. Evaluate the asymptotic worst-case running time of your algorithm (and explain your answer).
4. Explain why your algorithm is correct. That is, show that whenever your algorithm returns `true`, there are two values in A that add up to s, and that if your algorithm returns `false`, there are no such values.
   Hint: choose the right loop invariant.
5. Implement the method `hasMatchingPair` of the class `Assignment3` in the Codeboard project for this assignment.

(Weight: 30% of this CW)

3. Stable Production

Consider an array A of positive integers storing the number of units that a certain machine can produce each hour, for a certain period of time.
For instance, the array

$$A = [3, 2, 4, 1, 7, 4, 5, 1]$$

indicates that the machine can produce at most 3 units during the first hour, at most 2 units during the second hour, etc.
We are interested in the largest cumulated number of units that the machine can produce, while maintaining a stable production rate per hour. For instance, if the machine is only active during the first 3 hours, then the largest stable production rate that it can maintain is 2 units per hour (since $2 < 3$ and $2 < 4$). So over this time interval, the largest *cumulated* number of units that can be produced at a stable rate is $2 \cdot 3 = 6$.
Over the segment $A[6..7]$, this number is instead $\min(7, 4) \cdot 2 = 8$.
Over the whole array, the segment that maximizes this value is $A[5..7]$, with $\min(7, 4, 5) \cdot 3 = 12$ units produced. We call this value the *maximal stable production* for A.

1. Write in pseudocode a brute-force algorithm that takes an array A of positive integers as input, and returns the maximal stable production for A.
   What is its worst-case asymptotic running time?

2. Develop a more efficient approach, using the divide-and-conquer paradigm:

   (a) write your algorithm in Java or pseudocode, and explain why it is correct,

   (b) evaluate its worst-case asymptotic running time (explain your answer),

   (c) implement this algorithm as the method `stableProduction` of Class `Assignment3` in the Codeboard project for this assignment.

   Hint: for the divide-and-conquer approach, you may take inspiration from the maximal segment sum problem seen during Lab 5.

(Weight: 40% of this CW)

4. Disorder*

This is a starred exercise, meaning that it can be more challenging than the others. You may consider it optional: if you submit it, then the percentage of the final mark determined by this assignments will be increased accordingly.

Given an array A of distinct integers, we want to determine how many pairs of elements violate the natural order over integers, i.e. the number of pairs of indices i, j such that $i < j$ but $A[i] > A[j]$.
For instance, given as input the array

$$A = [2, 3, 6, 1, 5],$$

the algorithm should return the value 4, because the array contains 4 such violations, namely

$$A[1] > A[4], \qquad A[2] > A[4], \qquad A[3] > A[4], \qquad \text{and} \qquad A[3] > A[5].$$

1. Write in pseudocode a brute-force algorithm for this problem, and evaluate its wort-case asymptotic running time.

2. Develop a more efficient approach, using the divide-and-conquer paradigm:

   (a) write your algorithm in Java or pseudocode, and explain why it is correct,

   (b) evaluate its worst-case asymptotic running time (explain your answer),

   (c) implement this algorithm as the method `violations` of Class `Assignment3` in the Codeboard project for this assignment.

   Hint: for the divide-and-conquer approach, one solution consists in extending the classical MERGESORT algorithm.

(Weight: $+40\%$ for this CW)

Deliverables. Submit two copies of your code:

- one via Codeboard (instructions are available here),
- one via OLE (together with the other deliverables).

The other questions must be answered in a PDF document.
Combine all deliverables into one zip file, which you submit via the OLE submission page for this assignment. Please include name, student ID and email address in your submission.

Submission until

Monday, 28 November 2022, 23:55,

to Codeboard and OLE.