Assignment 4

Mouna Kacimi, Werner Nutt, Camilo Thorne

Intersection of Arrays, Matching Pairs, and a Mystery Algorithm

Instructions: Each student shall write down and submit his/her solutions separately. If you worked together with other students, please list their names.
You can write up your answers by hand (provided your handwriting is legible) or use a word processing system like Latex or Word.
For a programming task, your solution must contain (i) an explanation of your solution to the problem, (ii) the Java code, in a form that we can run it, (iii) instructions how to run it. Also put the source code into your solution document. For all programming tasks, it is not allowed to use any external libraries ("import") if not stated otherwise.
Please, include name, matriculation number and email address in your submission.

1. Intersecting Arrays

Develop an (efficient!) algorithm that takes as input two arrays of integers A and B and outputs a third array C that contains exactly those numbers that occur in both A and B. Each number should occur only once in C, even if it occurs more than once in A or B.

1. Write the algorithm in pseudo-code and explain briefly why it solves the problem.

2. Find out the asymptotic worst-case running time of the algorithm and briefly explain your answer.

3. Implement the algorithm in Java.

4. Test the algorithm for extreme inputs: identify five extreme cases and ensure that your algorithm performs correctly on them. Document the cases in your report.

(10 Points)

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 is to check, given s, whether A has a matching pair for s.

1. Develop an algorithm in pseudocode that solves the matching pair problem as fast as possible.

   Hint: An algorithm that you know from the lecture may be useful.

2. What is the asymptotic worst-case complexity of your algorithm? Briefly explain your answer.

3. Prove that your algorithm is correct. That is, show that whenever there are two values in A that add up to s, your algorithm returns *Yes*, and that whenever there are no two such values, your algorithm returns *No*.

   Hint: Choose the right loop invariant.

(10 Points)

3. A Mystery

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

```
procedure MYSTERY($A, l, r$)
    $range := r - l + 1$
    $subrange := \lceil 2 \cdot range/3 \rceil$
    if $range = 2$ and $A[l] > A[r]$ then
        swap $A[l] \leftrightarrow A[r]$
    else if $range \geq 3$ then
        MYSTERY($A, l, l + subrange - 1$)
        MYSTERY($A, r - (subrange - 1), r$)
        MYSTERY($A, l, l + subrange - 1$)
    end if
end procedure
```

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? Give a proof for your claim.

   Hint: Since MYSTERY is a recursive procedure, your proof should use induction. Note that if MYSTERY is called with range from l to r, then the recursive calls of MYSTERY apply to a smaller range. Therefore, in your proof you assume that after the recursive calls the subranges of those calls have the claimed property. From this, conclude that the entire range from l to r has the property.

2. What is the asymptotic running time of MYSTERY? Provide an argument for your answer.

(10 Points)

Submission: Until Mon, 7 April 2014, 8:30 am, to

```
dsa-submissions AT inf DOT unibz DOT it.
```