

Data Structures and Algorithms

COSC 336 Assignment 3

Instructions.

1. Due date and time: As indicated on Blackboard.
2. This is a team assignment. Work in teams of 3-4 students. Submit on Blackboard one assignment per team, with the names of all students making the team.
3. The exercises will not be graded, but you still need to present your best attempt to solve them. If you do not know how to solve an exercise, say it. This will give me feedback about your understanding of the theoretical concepts.
4. Your programs must be written in Java.
5. Write your programs neatly - imagine yourself grading your program and see if it is easy to read and understand.

Comment your programs reasonably: there is no need to comment lines like "i++" but do include brief comments describing the main purpose of a specific block of lines.

6. You will submit on **Blackboard** 32 files.

The **1-st file** is a pdf file (produced ideally with latex and Overleaf) and it will contain the following:

- (a) The solution to the Exercises (see the remark above).
- (b) A short description of your algorithm for the Programming Task, where you explain the dynamic programming approach. Focus on how you have modified MERGE.
- (c) A table with the results your program gives for the data sets indicated for the programming task.
- (d) The java code (so that the grader can make observations) of the program.

The **2-nd file** is the .java file containing the java source code for Programming Task.

Exercise 1. Analyze the following recurrences using the method that is indicated. In case you use the Master Theorem, state what the corresponding values of a , b , and $f(n)$ are and how you determined which case of the theorem applies.

- $T(n) = 3T(\frac{n}{4}) + 3$. Use the Master Theorem to find a $\Theta()$ evaluation, or say "Master Theorem cannot be used", if this is the case.
- $T(n) = 2T(\frac{n}{2}) + 3n$. Use the Master Theorem to find a $\Theta()$ evaluation, or say "Master Theorem cannot be used", if this is the case.
- $T(n) = 9T(\frac{n}{3}) + n^2 \log n$. Use the Master Theorem to find a $\Theta()$ evaluation, or say "Master Theorem cannot be used", if this is the case.

Exercise 2.

- $T(n) = 2T(n-1) + 1$, $T(0) = 1$. Use the iteration method to find a $\Theta()$ evaluation for $T(n)$.
- $T(n) = T(n-1) + 1$, $T(0) = 1$. Use the iteration method to find a $\Theta()$ evaluation for $T(n)$.
- Give a $\Theta(\cdot)$ evaluation for the runtime of the following code:

```
i= n
while(i >=1) {
    for (j=1; j <=n; j++)
        x=x+1
    i = i/2
}
```

- Give a $\Theta(\cdot)$ evaluation for the runtime of the following code:

```
i= n
while(i >=1) {
    for (j=1; j <=i; j++)
        x=x+1
    i = i/2
}
```

Programming Task.

The input is an array a_1, a_2, \dots, a_n of numbers. A *UP-pair* is a pair (a_i, a_j) so that $1 \leq i < j \leq n$ and $a_i < a_j$. The task is to count the number of UP-pairs in the array.

For example, for $n = 5$ and input sequence 7, 3, 8, 1, 5, we have the following UP-pairs: (7, 8), (3, 8), (3, 5), (1, 5). So, there are 4 UP-pairs. Design an $O(n \log n)$ algorithm which computes the number of UP-pairs for a given input sequence and implement your program in Java.

It's very easy to come up with an algorithm with run time $\Theta(n^2)$ (just compare all pairs), but such an algorithm will not get any credit.

The idea of the $O(n \log n)$ algorithm is to modify MERGE-SORT (from page 34 in the textbook) so that in addition to sorting the array it also counts the number of UP-pairs. Thus, your modified MERGE-SORT (A, p, r) will sort the segment of the array $A[p..r]$ (like in the textbook), but in addition to that will return the number of UP-pairs in this segment. The main modification is in the MERGE procedure, see textbook page 31. In the final for loop, you will also count the UP-pairs in which the first component is in L , and the second component is in R . Keep in mind that L and R are both sorted and therefore if $L[i] < R[j]$, then all pairs $(L[i], R[j]), (L[i], R[j+1]), \dots, (L[i], R[r])$ are UP-pairs. So with a single comparison you can add the number of these pairs to the counter of UP-pairs. Using this observation the modified MERGE can be implemented in $O(n)$ (the same runtime as the standard MERGE).

Test your program and report in a table the results for the following data sets.

Data set 1: 7,3,8,1,5

Data set 2: The numbers from the file input-3.4, available on Blackboard. The first line of the line has the number of elements (which is 1000), and the next line has the elements.

Data set 3: The numbers from the file input-3.5, available on Blackboard. The first line of the line has the number of elements (which is 10000), and the next line has the elements.

NOTE 2: You can find an example of what I mean by “Describe an algorithm ...” at <https://www.geeksforgeeks.org/find-the-element-that-appears-once-in-a-sorted-array/>