EC330 Applied Algorithms and Data Structures for Engineers
Spring 2020

Homework 4

Out: March 5, 2020
Due: March 21, 2020

This homework has a written part and a programming part. Both are due at 11:59 pm on March 21. You should submit both parts on Gradescope.
For the written part, you should submit a single PDF file containing either typeset answers or scanned copies of hand-written answers. Make sure you write your answers clearly.
For the programming part, you should make sure your code compile and run on Gradescope. You should also try to create your own test cases to test and debug your code. Make sure you write your name and BU ID in a comment at the top of the program, and use clear comments to explain the key steps in your code.

1. "Does it sort?" [10 pt]

For the following algorithm:

- **i.** Provide the result when the algorithm is run on the array [8, 7, …, 3, 2, 1].
- **ii.** Does the algorithm correctly sort all arrays A of size *n* containing only positive integers, for $n \geq 8$?
- **iii.** If the answer to the previous question is "yes", provide a short explanation and a worst-case (asymptotic) running time. If "no" give a small counterexample.

```
sortA(Array A[0..n-1])
   for i = 0 to n/2
      for j = n/2+1 to n-1
         if (A[i] <= A[j])
            swap(A[i], A[j])
   return A;
```

2. Programming [90 pt]

NOTE: You should implement your solution in the provided hw4.cpp file. You only need to submit this file on Gradescope.

a) Given an *unsorted* vector of integers, find the integer that occurs the most often in the array (also known as the *mode*). **[10 pt]**
The fastest solution (needs to be substantially faster than the rest for large inputs) will receive **5 bonus points**.

b) Given a *sorted* vector of integers, find the *mode* in this array. **[10 pt]**
The fastest solution (needs to be substantially faster than the rest for large inputs) will receive **5 bonus points**.

c) Given an *unsorted* vector of integers, find the pair of integers that are the closest in value. If there are multiple such pairs, you can output any of them. **[10 pt]**
The fastest solution (needs to be substantially faster than the rest for large inputs) will receive **5 bonus points**.

d) Given an *unsorted* vector of 2D coordinates (x_i, y_i), find the pair that are the closest in terms of Euclidean distance. **[15 pt]**
The fastest solution (needs to be substantially faster than the rest for large inputs) will receive **10 bonus points**.

e) Given an *unsorted* vector of integers, sort using *only* the following `flip(i, j)` operation, which picks two indices `i` and `j`, and reverse the elements between these two indices (inclusive). **[15 pt]**
The fastest solution (needs to be substantially faster than the rest for large inputs) will receive **10 bonus points**.

f) Given an *unsorted* vector of *n* distinct integers, sort it so that each integer is either greater than both integers immediately precedes and comes after it or less than both integers immediately precedes and comes after it. **[10 pt]**

Example:
Input: [4, 1, 2, 3]
Output: [1, 3, 2, 4] or [1, 4, 2, 3] (other outputs are possible)

g) In this problem, we will sort a vector of alphanumeric strings (each string may contain characters a – z, A – Z, 0 – 9). Sort the strings in *increasing order first by length, and then by alphanumerical order in their ASCII representations* (i.e. in the order of 0 – 9, A – Z and then a – z). NOTE: You may not use any built-in sort functions for this part. **[20 pt]**

Example:
Input: [“EC330”, “homework4”, “EC327”, “033CE”]
Output: [“033CE”, “EC327”, “EC330”, “homework4”]