CSE202 2019-2020 – FINAL EXAM

The 6 exercises are independent and can be treated in arbitrary order. Within an exercise, the answer to a question can be used in the next ones even if a proof has not been found.

The number of points indicated in front of each question is an indication of the relative difficulties of the questions. It is not necessary to solve all the questions to obtain the maximal possible grade.

Your descriptions of algorithms can be written in Python or in pseudo-code.

Exercise 1. Given two arrays A and B of integers, an inversion between A and B is a pair of indices i and j such that $A[i] > B[j]$.

(1) (2pts) Describe an algorithm that counts the number of inversions between two arrays A and B of at most n elements, each of them being sorted. Your algorithm should use $O(n)$ comparisons.

An inversion in an array A of integers is a pair of indices (i, j) such that $i < j$ and $A[i] > A[j]$. For instance, there are 4 inversions in the array $[6, 4, 9, 2]$, found at the indices $(1, 2), (1, 4), (2, 4), (3, 4)$.

(2) (4pts) Describe and analyze an algorithm to count the number of inversions in an n-element array in $O(n \log n)$ comparisons. A solution in $O(n \log^2 n)$ comparisons gives you only half of the points.

Exercise 2. (2pts) In a binary search tree built from n random keys, what is the probability that the successful search cost for one of these keys is 2?

Exercise 3. If $P \in \mathbb{K}[X]$ is a polynomial of degree n with coefficients in a field $\mathbb{K} \subset \mathbb{C}$, and $\alpha_1, \ldots, \alpha_n$ are its complex roots, the *Newton sums* s_k for $k \in \mathbb{N}$ are the sums of the kth powers of the α_i: $s_k = \alpha_1^k + \cdots + \alpha_n^k \in \mathbb{K}$. The aim of this exercise is to prove that $(s_0, \ldots, s_N)$ can be computed in $O(\mathrm{Mul}(N))$ arithmetic operations in $\mathbb{K}$, without knowing the α_i's. Here $\mathrm{Mul}(N)$ is a bound on the number of coefficients operations needed to multiply two polynomials of degree at most N.

(1) (1pt) By considering the Taylor expansion of $1/(1-aX)$, give an algorithm for the case when the degree n of P is 1;
(2) (3pts) Let $Q(X) = X^n P(1/X)$ with roots $1/\alpha_i$, $i = 1, \ldots, n$ and consider the partial fraction expansion of Q'/Q to design an efficient algorithm for arbitrary n. (Assume for simplicity that the roots α_i are distinct.) [It may help to consider first the case of a quadratic polynomial $(X - \alpha_1)(X - \alpha_2)$.]

Exercise 4. (1) (1pt) Give the KMP automaton for the pattern 313131, assuming a 4-character alphabet 0, 1, 2, and 3.
(2) (1pt) What is the sequence of transitions performed by the KMP algorithm to determine whether the text 1232033313230313131 contains this pattern?
(3) (1pt) How many comparisons of characters are needed for this?

Exercise 5. (2pts) Consider the following sorting algorithm: it takes as input a set S of n integers; initializes an empty binary search tree T; inserts the elements of S into T in random order; outputs the elements found by a depth-first search traversal of T, visiting recursively the left subtree, then the node, then the right subtree. Show that the expected number of operations performed by this algorithm is $O(n \log n)$.

Exercise 6. The partition problem is the following: given a list L of positive integers, partition the list into two lists A and B such that $\max(\sum_{a\in A} a, \sum_{b\in B} b)$ is minimal.

(1) (1pt) Show that the partition problem is NP-hard.
(2) (1pt) Consider the following algorithm:

```
def partition(X):
    suma = sumb = 0
    for i in sorted(X,reverse=True):
        if suma<sumb: suma += i
        else: sumb += i
    return max(suma,sumb)
```

Show that its complexity is polynomial in the size of the input.
(3) (2pts) Show that if the loop is run with `reverse=False` (ie, in increasing order), then the approximation ratio is at least 3/2.
(4) (4pts) Show that the algorithm with `reverse=True` has approximation ratio at most 4/3. [Hint: a possible proof is by contradiction.]