Do not turn this page until you are instructed to do so!

Instructions: Please answer the following questions to the best of your ability. If you are asked to develop an algorithm, you must give a plain text description of the algorithm, show that it is correct and prove that it has the required complexity. When writing proofs, please strive for clarity and brevity (in that order). Your work can be either in English or in French.

The exam is closed-book, closed-note, closed-internet, etc. However, you may use one A4-sized sheet (front and back) of notes as reference.

The exam has six questions, numbered 1 to 6, for a total of 53 points. You can solve them in any order. Please write your solution to the first three exercises on a different sheet of paper than the solution to the last three exercises, as they will be graded separately.

You have 180 minutes to complete the exam. Good luck!

Q. 1 *(6 points)* Given an integer array A of length n, where each entry has a color, and an integer $1 \leq k \leq n$, develop a $O(n)$-time algorithm to compute the maximum of the minimum values in all intervals of A containing cells of at least k different colors.

Hint: Use a queue that supports the minimum operation. You must explain how to implement it, for example, using two stacks.

Q. 2 *(6 points)* In the problem of pattern matching with wildcards, you are given a text T of length n and a pattern of length m over an alphabet of integers. Both T and P may contain wildcards "?", special characters that match any other character of the alphabet. Your task is to develop an efficient algorithm to find all positions where P matches T.

Example: There are four positions where a pattern $P = 2\,?\,2$ matches the text $T = 2\,1\,2\,2\,2\,?\,2$: they are $3, 5, 6, 7$.

(a) *(2 points)* Briefly explain the idea of the algorithm for fast multiplication of polynomials. State the complexity of the algorithm and explain how one can use the Master theorem to derive it.

(b) *(2 points)* Given two integer vectors X, Y of lengths n and m, $n \geq m$, their convolution is defined as a vector C of length $n - m$, where $C[i] = \sum_{j=1}^{m} X[i + m - j]Y[j]$. Show an $O(n \log m)$-time algorithm for computing the convolution.

Hint: Consider two polynomials, $\sum_{i=1}^{n} X[i]z^i$ and $\sum_{j=1}^{m} Y[j]z^j$. What are the coefficients of their product?

(c) *(1 point)* Assume that P and T contain only positive (> 0) integers and wildcards. Let $T'[i] = T[i]$ if $T[i]$ is an integer and $T'[i] = 0$ if $T'[i] =?$. Define P' analogously. Show that there is an occurrence of P that ends at the position i of T if and only if $\sum_{j=1}^{m} P'[j]T'[i - m + j](P'[j] - T'[i - m + j])^2 = 0$.

(d) *(1 point)* Show that the values $\sum_{j=1}^{m} P'[j]T'[i - m + j](P'[j] - T'[i - m + j])^2$, $j = m, \ldots, n$ can be computed in $O(n \log m)$ time. Hence, show an $O(n \log m)$ time algorithm for pattern matching with wildcards.

Hint: Expand $P'[j]T'[i - m + j](P'[j] - T'[i - m + j])^2$ and then apply your algorithm for computing the convolution two times.

Q. 3 *(15 points)* In this question your task is to build a data structure called the suffix array, and show that it can be used for pattern matching queries.

(a) *(2 points)* Give the definition of the suffix tree of a string of length n.

(b) *(1 point)* Draw the suffix tree of $T = a\,b\,a\,a\,a$.

(c) *(1 point)* Assuming a constant-size alphabet, what is the space complexity of the suffix tree? Explain your answer.

(d) *(2 points)* Explain how one can use the suffix tree to find all occurrences of a pattern P in T. What is the time complexity of the algorithm?

(e) *(2 points)* Give a simple $O(n^2)$-time algorithm to construct the suffix tree. What is the best construction algorithm? Explain why it is optimal.

Definition: The suffix array SA of a string T has length n, and $SA[i]$ is defined to be the starting position of the i-th suffix of T in the lexicographic order.

Example: For $T = a\,b\,a\,a\,a$, its suffixes are a, $a\,a$, $a\,a\,a$, $a\,b\,a\,a\,a$, and $b\,a\,a\,a$ (in the lexicographic order), and hence its suffix array is $\begin{bmatrix}5 & 4 & 3 & 1 & 2\end{bmatrix}$.

(f) *(2 points)* Show that if there is a $T(n)$-time construction algorithm for the suffix tree, then the suffix array can be constructed in $T(n) + O(n)$ time.

Hint: Consider the leaves of the suffix tree in the left-to-right order.

(g) *(2 points)* Show how to find all occurrences of a pattern P in T in $O(|P| \log |T|)$ time using the suffix array of T. *Hint:* Modify the binary search.

(h) *(2 points)* Improve the time complexity of your algorithm to $O(|P| + \log |T|)$.

Please write your solution to the following three exercises on a different sheet of paper than the solution to the previous three exercises, as they will be graded separately.

Q. 4 *(6 points)* An independent set is a set of vertices such that none of them are connected by an edge. An independent set is called maximum if it has maximum size among all independent sets. Prove that a maximum independent set can be found in polynomial time in a bipartite graph.

Hint: A matching is a set of disjoint edges. Recall that the Ford–Fulkerson algorithm can be used to find a maximum matching in a bipartite graph and prove that if $G = (V, E)$ is a bipartite graph, then:

$$|V| = |\text{maximum matching of } G| + |\text{maximum independent set of } G|$$

Q. 5 *(6 points)* Consider the following sorting algorithm for an array T of size n:

- If $n = 2$ and $T[0] > T[1]$, swap $T[0]$ and $T[1]$
- If $n \geq 3$:
  - Recursively sort the first $\lceil \frac{2}{3}n \rceil$ elements of T
  - Recursively sort the last $\lceil \frac{2}{3}n \rceil$ elements of T
  - Recursively sort the first $\lceil \frac{2}{3}n \rceil$ elements of T

(a) *(3 points)* Show that the algorithm is a correct sorting algorithm.

(b) *(2 points)* Compute the worst-case complexity of the algorithm.

(c) *(1 point)* Comment on the complexity of this algorithm with respect to other sorting algorithms.

Q. 6 *(14 points)* A *semiring* $(\mathbb{K}, \oplus, \otimes, \mathbb{0}, \mathbb{1})$ is a set $\mathbb{K}$ together with two binary operations $\oplus : \mathbb{K} \times \mathbb{K} \to \mathbb{K}$, $\otimes : \mathbb{K} \times \mathbb{K} \to \mathbb{K}$ and two distinguished elements $\mathbb{0}, \mathbb{1} \in \mathbb{K}$, such that:

- $(\mathbb{K}, \oplus, \mathbb{0})$ is a commutative monoid (i.e., $\oplus$ is commutative, associative, and $\mathbb{0}$ is a neutral element for $\oplus$);
- $(\mathbb{K}, \otimes, \mathbb{1})$ is a monoid (i.e., $\otimes$ is associative and $\mathbb{1}$ is a neutral element for $\otimes$);
- $\otimes$ is distributive over $\oplus$, i.e., $\forall x, y, z \in \mathbb{K}$, $x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z)$ and $(y \oplus z) \otimes x = (y \otimes x) \oplus (z \otimes x)$;
- $\mathbb{0}$ is annihilator for $\otimes$, i.e., $\forall x \in \mathbb{K}$, $x \otimes \mathbb{0} = \mathbb{0} \otimes x = \mathbb{0}$.

Such a semiring is *bounded* if for all $x \in \mathbb{K}$, $\mathbb{1} \oplus x = \mathbb{1}$.

Given a semiring $(\mathbb{K}, \oplus, \otimes, \mathbb{0}, \mathbb{1})$, a $\mathbb{K}$-*annotated graph* (V, E, λ) is a directed graph (V, E) together with an annotation function $\lambda : E \to \mathbb{K}$.

Let $(\mathbb{K}, \oplus, \otimes, \mathbb{0}, \mathbb{1})$ be a semiring and $G = (V, E, \lambda)$ a $\mathbb{K}$-annotated graph. For a path $P = (x_1, \ldots, x_n)$ in G with $x_1 \ldots x_n \in V$, we note $\lambda(P) = \lambda(x_1, x_2) \otimes \lambda(x_2, x_3) \cdots \otimes \lambda(x_{n-1}, x_n)$. For $s, t \in V$, the $\mathbb{K}$-*distance from* s *to* t is defined (abstractly) as:

$$\bigoplus_{\text{every path } P \text{ from } s \text{ to } t} \lambda(P) \tag{1}$$

Note that, in all generality, this is an infinite sum, and therefore not well-defined.

(a) *(2 points)* Show that, if $\mathbb{K}$ is bounded, Equation (1) is well defined as it amounts to a finite sum.

(b) *(1 point)* Give an example of a bounded semiring $\mathbb{K}_0$ and of a classical algorithm that can be used to compute the $\mathbb{K}_0$-distance from a node s to a node t in $\mathbb{K}_0$-annotated graphs.

(c) *(2 points)* We define the *natural order over* $\mathbb{K}$, $\leq_{\mathbb{K}}$ as $(a \leq_{\mathbb{K}} b) \iff \exists x \in \mathbb{K}, (a \oplus x = b)$. Shows that if $\mathbb{K}$ is bounded, $\leq_{\mathbb{K}}$ is a partial order over $\mathbb{K}$.

(d) *(5 points)* Assume that $\mathbb{K}$ is bounded and that $\leq_{\mathbb{K}}$ is a *total* order. Give an (efficient) algorithm to compute the $\mathbb{K}$-distance from a node s to a node t in a $\mathbb{K}$-annotated graph. Justify correctness of the algorithm. What is its complexity in terms of $|V|$, $|E|$, and the times $\mathrm{T}_\oplus$ and $\mathrm{T}_\otimes$ necessary to evaluate one $\oplus$ or $\otimes$ operation?

(e) *(5 points)* Provide an example of a semiring $\mathbb{K}$ that is bounded but where $\leq_{\mathbb{K}}$ is not total. Does the algorithm from the previous question still work? Justify your answer.