Sélection internationale Session 2016
École Normale Supérieure Paris
Épreuve de culture scientifique - **Informatique** Duration : 3 hours

For candidates who chose **computer science** as **primary specialisation**

If you cannot answer a question, you may use it as hypothesis to later questions.
Calculators are not allowed.

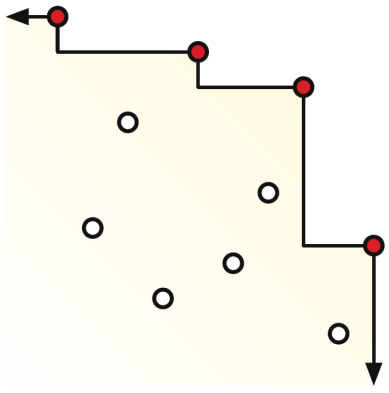
Exercise 1. Suppose we are given an array $A[0 \ldots n+1]$ with fencepost values $A[0] = A[n+1] = -\infty$. We say that an element $A[x]$ is a *local maximum* if it is greater than or equal to its neighbors, or more formally, if $A[x-1] \leq A[x]$ and $A[x] \geq A[x+1]$.
We can obviously find a local maximum in $O(n)$ time by scanning through the array. Describe and analyze an algorithm that returns the index of one local maximum in $O(\log n)$ time.
Hint: With the given boundary conditions, the array must have at least one local maximum. Why?

Exercise 2. We will randomly partition the vertex set of a graph G into two sets S and T. The algorithm picks a bit uniformly at random for each vertex and if the bit is equal to 0, the algorithm puts the vertex in S and if the bit is equal to 1, it puts the vertex in T.

1. Show that the expected number of edges with one endpoint in S and the other endpoint in T is exactly half the edges in G.

2. Now say the edges have weights. What can you say about the sum of the weights of the edges with one endpoint in S and the other endpoint in T?

Exercise 3. Suppose you are given a set P of n points in the plane. A point $p \in P$ is *maximal* in P if no other point in P is both above and to the right of p. Intuitively, the maximal points define a "staircase" with all the other points of P below it.



Describe and analyze an algorithm to compute the number of maximal points in P in $O(n \log n)$ time. Given the ten points shown above, your algorithm should return the integer 4.

Exercise 4.

1. An *extendable array* is a data structure that stores a sequence of items and supports the following operations.

- AddToFront(x) adds x to the beginning of the sequence.
- AddToEnd(x) adds x to the end of the sequence.
- Lookup(k) returns the k-th item in the sequence, or Null if the current length of the sequence is less than k.

Describe a simple data structure that implements an extendable array. Your AddToFront and AddToEnd algorithms should take $O(1)$ amortized time, and your Lookup algorithm should take $O(1)$ worst-case time. The data structure should use $O(n)$ space, where n is the current length of the sequence.

2. A stack is a last-in-first-out data structure. It supports two operations Push and Pop. Push adds a new item to the back of the queue while Pop removes the last item from the back of the queue (the one most recently added).An *ordered stack* is a data structure that stores a sequence of items and supports the following operations:

   - OrderedPush(x) removes all items smaller than x from the beginning of the sequence and then adds x to the beginning of the sequence.
   - Pop deletes and returns the first item in the sequence (or Null if the sequence is empty).

   Suppose we implement an ordered stack with a simple linked list, using the obvious OrderedPush and Pop algorithms. Prove that if we start with an empty data structure, the amortized cost of each OrderedPush or Pop operation is $O(1)$.

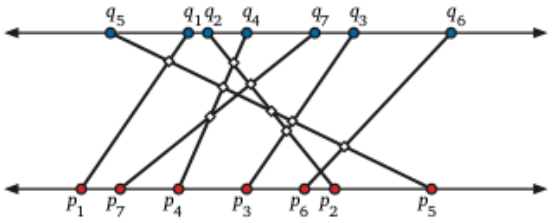
3. A queue is a first-in-first-out data structure. It supports two operations Push and Pop. Push adds a new item to the back of the queue, while Pop removes the first item from the front of the queue. Show how you can simulate a queue by using two stacks. Any sequence of pushes and pops should run in amortized constant time.

Exercise 5.

1. An inversion in an array $A[1 \ldots n]$ is a pair of indices (i, j) such that $i < j$ and $A[i] > A[j]$. The number of inversions in an n-element array is between 0 (if the array is sorted) and $\binom{n}{2}$ (if the array is sorted backward). Describe and analyze a divide-and-conquer algorithm to count the number of inversions in an n-element array in $O(n \log n)$ time. Assume all the elements of the input array are distinct.

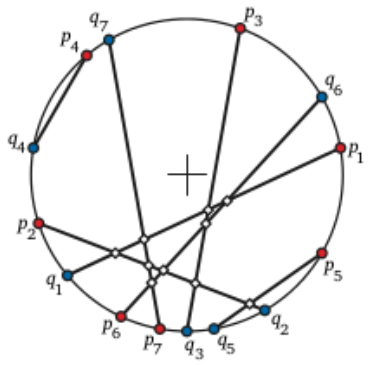
2. Suppose you are given two sets of n points, one set $\{p_1, p_2, \ldots, p_n\}$ on the line $y = 0$ and the other set $\{q_1, q_2, \ldots, q_n\}$ on the line $y = 1$. Create a set of n line segments by connect each point p_i to the corresponding point q_i . Describe and analyze a divide-and-conquer algorithm to determine how many pairs of these line segments intersect, in $O(n \log n)$ time.
   Hint: Use your solution to question 1.

   Assume a reasonable representation for the input points, and assume the x-coordinates of the input points are distinct. For example, for the input shown below, your algorithm should return the number 9.



3. Now suppose you are given two sets $\{p_1, p_2, \ldots, p_n\}$ and $\{q_1, q_2, \ldots, q_n\}$ of n points on the unit circle. Connect each point p_i to the corresponding point q_i . Describe and analyze a divide-and-conquer algorithm to determine how many pairs of these line segments intersect in $O(n \log^2 n)$ time.
Hint: Use your solution to question 2.

   Assume a reasonable representation for the input points, and assume all input points are distinct. For example, for the input shown below, your algorithm should return the number 10.



4. ($\star$) Propose an improved algorithm that runs in $O(n \log n)$ time.