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

Pour les candidats ayant choisi **l'informatique** comme **discipline principale**

Si vous ne parvenez pas à répondre à une question, vous pouvez cependant l'utiliser comme hypothèse pour les questions suivantes.

Calculatrices interdites.

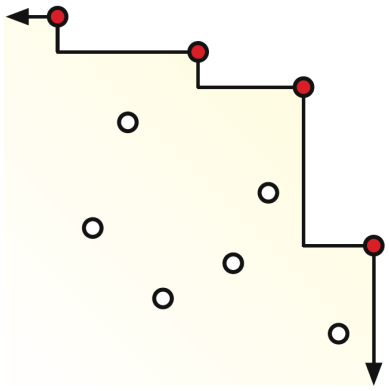
Exercice 1. Supposons donné un tableau $A[0 \dots n+1]$ avec les valeurs extrêmes $A[0] = A[n+1] = -\infty$. Un élément $A[x]$ est un *maximum local* si il est supérieur ou égal à ses voisins, ou de façon plus formelle si $A[x-1] \leq A[x]$ et $A[x] \geq A[x+1]$.
Nous pouvons de façon évidente trouver un maximum local en temps $O(n)$ en parcourant le tableau. Décrire et analyser un algorithme qui retourne l'indice d'un maximum local en temps $O(\log n)$.
Indication: Avec les conditions aux bords, le tableau doit contenir un maximum local. Pourquoi ?

Exercice 2. Nous allons partitionner l'ensemble des sommets d'un graphe G en deux ensembles S et T. L'algorithme tire un bit uniformément au hasard pour chaque sommet et si le bit vaut 0, l'algorithme place le sommet dans S et si le bit vaut 1, il le place dans T.

1. Montrer que le nombre espéré d'arêtes du graphe avec une extrémité dans S et une extrémité dans T est égal à la moitié du nombre d'arêtes de G.

2. Supposons désormais que les arêtes sont pondérées. Que peut-on dire sur la somme des poids des arêtes du graphe avec une extrémité dans S et une extrémité dans T ?

Exercice 3. Supposons donné un ensemble P de n points dans le plan. Un point $p \in P$ est *maximal* dans P si aucun autre point de P n'est à la fois au-dessus et à droite de p. Intuitivement, les points maximaux définissent un "escalier" avec tous tous les autres points de P en dessous.



Décrire and analyser un algorithme qui calcule le nombre de points maximaux de P en temps $O(n \log n)$. Étant donné les dix points de l'exemeple ci-dessus, l'algorithme doit retourner l'entier 4.

Exercice 4.

1. Un *tableau extensible* est une structure de données qui stocke une suite d'éléments et permet les opérations suivantes:

   - AddToFront(x) ajoute x au début de la suite.
   - AddToEnd(x) ajoute x à la fin de la suite.
   - Lookup(k) retourne le k-ième élément de la suite ou Null si la longueur de la suire ne dépasse pas k.

   Décrire une structure de données simple qui implante un tableau extensible. Les algorithmes AddToFront et AddToEnd doivent s'exécuter en temps amorti $O(1)$. L'algorithme Lookup doit s'exécuter en temps $O(1)$ dans le pire des cas. La structure de données doit utilise un espace $O(n)$, où n est la longueur courrante de la suite.

2. Une pile est une structure de données dernier-entré premier-sorti. Elle permet les opérations Push et Pop. Push ajoute un élément au sommet de la pile et Pop retire et retourne l'élément du sommet de la pile (l'élément ajouté le plus récemment). Une *pile ordonnée* est une structure de donnée qui stocke une suite d'éléments et permet les opérations suivantes:

   - OrderedPush(x) retire tous les éléments inférieurs à x à partir du sommet de la pile et ajoute x au sommet de la pile.
   - Pop etire et retourne l'élément du sommet de la pile (ou Null si la pile est vide).

   Supposons que nous implantons une pile ordonnée avec une liste simplement chaînée en utilisant les algorithmes OrderedPush et Pop évidents. Montrer que si l'on démarre d'une structure de donnée vide, le coût amorti des opérations OrderedPush et Pop est $O(1)$.

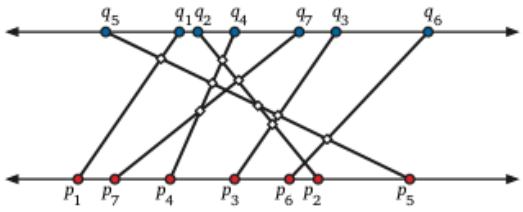
3. Une file est une structure de données dernier-entré dernier-sorti. Elle permet les opérations Push et Pop. Push ajoute un élément au début de la file et Pop retire et retourne l'élément à la fin de la file. Montrer comment simuler une file avec deux piles. Une séquence de Push et Pop devra s'éxecuter en temps amorti constant.

Exercice 5.

1. Une inversion dans un tableau $A[1 \dots n]$ est un couple d'indice (i, j) tel que $i < j$ et $A[i] > A[j]$. Le nombre d'inversions d'un tableau de longeur n est entre 0 (si le tableau est trié) et $\binom{n}{2}$ (si le tableau est trié dans le sens inverse). Décrire et analyser un algorithme de type "diviser-pour-régner" qui retourne le nombre d'inversions dans un tableau de longueur n en temps $O(n \log n)$. Nous supposons que tous les éléments du tableau d'entrée sont différents.

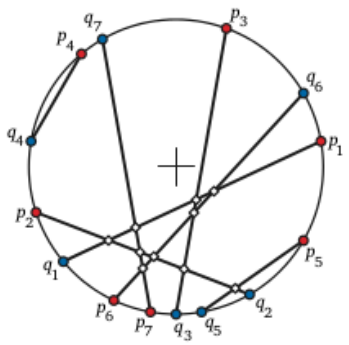
2. Étant donnés deux ensembles de n points, un ensemble $\{p_1, p_2, \dots, p_n\}$ sur la droite $y = 0$ et un autre ensemble $\{q_1, q_2, \dots, q_n\}$ sur la droite $y = 1$, nous associons un ensemble de n segments de droites en reliant chaque point p_i au point correspondant q_i . Décrire et analyser un algorithme de type "diviser-pour-régner" qui retourne, en temps $O(n \log n)$, le nombre de couples de segments qui s'intersectent.
   Indication: Utiliser la solution de la question 1.

   Nous supposons une représentation "raisonnable" des points de l'entrée et nous supposons que les coordonnées x de ces points sont deux à deux distinctes. Par exemple, sur l'entrée ci-dessous, l'algorithme doit retourner le nombre 9.



3. Étant donnés deux ensembles de n points $\{p_1, p_2, \ldots, p_n\}$ et $\{q_1, q_2, \ldots, q_n\}$ sur le cercle unité, nous associons un ensemble de n segments de droites en reliant chaque point p_i au point correspondant q_i . Décrire et analyser un algorithme de type "diviser-pour-régner" qui retourne, en temps $O(n \log^2 n)$, le nombre de couples de segments qui s'intersectent. **Indication:** Utiliser la solution de la question 2.

   Nous supposons une représentation "raisonnable" des points de l'entrée et nous supposons que les points sont deux à deux distincts. Par exemple, sur l'entrée ci-dessous, l'algorithme doit retourner le nombre 10.



4. ($\star$) Proposer un algorithme améliorer en temps $O(n \log n)$.