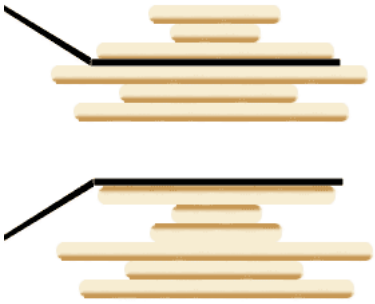
Sélection internationale Session 2017
École Normale Supérieure Paris
Épreuve de culture scientifique - **Informatique** Duration : 2 hours

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

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

Exercice 1. Supposons que nous avons une pile de n crêpes de tailles différentes. Nous voulons trier les crêpes de sorte que chaque crêpe est tourjous au dessus d'une crêpe de diamètre supérieure. La seule opération autorisé pour le tri est un retournement qui consiste à insérer une spatule sous les k crêpes du dessus (for k compris entre 1 et n) et à les retourner toutes.



1. Donner une borne inférieure sur le nombre de retournements nécessaires pour un trier un ensemble arbitraire de n crêpes.
2. Décrire un algorithme efficace pour trier un ensemble arbitraire de n crêpes. Combien de retournements effectue exactement votre algorithme dans le pire des cas ?
3. Supposons maintenant que une face de chaque crêpe est brulée. Combien de retournements doit-on effectuer pour trier les crêpes de sorte que les faces brulées de toute la crêpe soient en dessous ?

Exercice 2.

1. Supposons que nous avons deux tableaux triés $A[1 \dots m]$ et $B[1 \dots n]$ et un entier k. Décrire un algorithme pour trouver le k-ième plus petit élément de l'union de A et de B en temps $O(\log(m+n))$.

   Par exemple, sur l'entrée

   $$A[1 \dots 8] = [0, 1, 6, 9, 12, 13, 18, 20],\ B[1 \dots 5] = [2, 5, 8, 17, 19] \text{ et } k = 6$$

   votre algorithme doit retourner 8. Vous pouvez supposer que les tableaux ne contiennent pas de doublon.
2. Supposons que nous avons deux tableaux triés $A[1 \dots n]$ et $B[1 \dots n]$. Donner un algorithme en temps$O(\log^2 n)$ pour trouver le n-ième plus petit des $2n$ éléments.
3. Supposons que nous avons k tableaux triés, chacun de n éléments et que nous voulons les combiner en un seul tableau trié de kn éléments.

(a) Considérons la stratégie suivante : en utilisant la procédure de fusion (du tri fusion), fusionner les deux premiers tableaux, puis le résultat avec le troisième, puis le résultats avec le quatrième, et ainsi de suite. Quelle est la complexité en temps de cet algorithme en fonction de k et de n ?

(b) Donner une solution plus efficace à ce problème, en utilisant une stratégie diviser-pour-régner. Quelle est la complexité en temps de cet algorithme en fonction de k et de n ?

4. Soit $M[1 \dots n][1 \dots n]$ une matrice $n \times n$ où chaque ligne et chaque colonne est triée. Un telle matrice est dite *totalement mononote* et nous supposons que tous ses éléments sont distincts.

   (a) Décrire et analyser un algorithme pour résoudre le problème suivant en temps $O(n)$: étant donnés des indices i, j, i', j', calculer le nombre d'entiers de M plus petit que $M[i][j]$ et plus grand que $M[i'][j']$.

   (b) Décrire et analyser un algorithme pour résoudre le problème suivant en temps $O(n)$: étant donnés des indices i, j, i', j', retourner un élément de M tiré uniformément aléatoirement parmi les éléments plus petits que $M[i][j]$ et plus grand que $M[i'][j']$. On supposera que l'ensemble demandé est toujours non vide.

   (c) Décrire et analyser un algorithme probabiliste pour calculer l'élément médian de M en temps espéré $O(n \log n)$.

5. Soit $A[1 \dots n]$ un tableau dont les $n\sqrt{n}$ premiers éléments sont triés (mais sur lequel on ne sait rien pour les éléments restants). Écrire un algorithme qui trie A en un temps substantiellement meilleure que $O(n \log n)$ opérations.

Exercice 3. Soit $k, n \in \mathbb{N}$. Le problème (k, n)-œuf est le suivant : considérons un immeuble de n étages, nous savons qu'il existe un étage f tel que si un œuf est laché du f-ième étage, il se casse alors que si il est jeté du $(f-1)$-ième étage, il ne se casse pas (il est possible que $f = 1$ et dans ce cas, l'œuf se casse toujours ou que $f = n + 1$ et dans ce cas, l'œuf ne se casse jamais).

Si les œufs se cassent lorsqu'ils sont lachés d'un certain étage, alors ils cassent aussi si ils sont lachés d'un étage supérieur. Si les œufs ne cassent pas lorsqu'ils sont lachés d'un certain étage, alors ils ne se cassent pas non plus si ils sont lachés d'un étage inférieur.

Le but de l'exercice est, étant donnés k œufs, de trouver f. La seule opération autorisée est de lancer un œuf à partir d'un étage et de voir ce qu'il se passe. Si un œuf est cassé, il ne peut pas être réutilisé. Le but est de lacher des œufs le moins de fois possible. Soit $E(k, n)$ le nombre minimal de lacher à effectuer pour toujours déterminer f.

1. Montrer que $E(1, n) = n$.

2. Montrer que $E(k, n) = \Theta(n^{1/k})$

3. Trouver une relation de récurrence pour $E(k, n)$. Écrire un programme dynamique pour trouver $E(k, n)$. Quelle est sa complexité en temps ?

4. Écrire un programme dynamque qui peut être utilisée pour trouver la stratégie optimale pour trouver l'étage f. Quelle est sa complexité en temps ?