

## Partiel du 3 mars 2008

Aucun document autorisé. Le barème est uniquement indicatif.

Vous pouvez écrire les algorithmes en C ou en pseudo-code. Si cela vous pose trop de difficultés n'hésitez pas à répondre en décrivant un algorithme par des phrases! Vous pouvez faire appel à des fonctions auxiliaires vues en cours (comparaisons, sous-tableau, etc.).


### Première partie

9 points

#### Exercice 1.

Rappeler les définitions utilisées et justifier vos réponses.

1. Est-ce que  $\sum_{i=1}^n i = \Theta(n^2)$ ?
2. Est-ce que  $n^2 = \Omega(2^n)$ ?
3. Est-ce que  $\sum_{i=0}^n \left(\frac{2}{3}\right)^i = O(1)$ ?


 1 pt  
9 min

 1 pt  
9 min

 1 pt  
9 min

#### Exercice 2.

Classer les fonctions de complexité  $n \log n, 2^n, \log n, n^2, n$  par ordre croissant et pour chacune d'elle donner l'exemple d'un algorithme (du cours ou des TD) qui a asymptotiquement cette complexité en pire cas, en temps. Répondre dans un tableau en donnant le nom de l'algorithme ou le nom du problème qu'il résout.

 2 pt  
18 min

#### Exercice 3.

Soit un tableau  $T$  de  $N$  éléments deux à deux comparables. On souhaite partitionner le tableau autour de son premier élément  $T[0]$ , appelé pivot. Après partition, le pivot est à l'indice  $p$ , les éléments plus petits que le pivot sont (dans le désordre) aux indices inférieurs à  $p$  et les éléments strictement plus grands que le pivot sont (dans le désordre) aux indices supérieurs à  $p$ .

Tableau  $T$

Tableau  $T$


*pivot à l'indice  $p$*

Partitionner( $T$ )

Éléments plus petits que le pivot

Éléments plus grands que le pivot

Écrire l'algorithme de partitionnement de manière à ce qu'il effectue  $N - 1$  (ou  $N$ ) comparaisons.


 2 pt  
18 min

**Rappel sur les piles.** Une pile est une structure de données permettant de stocker des éléments et sur laquelle on peut agir à l'aide des opérations primitives suivantes. On peut ajouter,  $\text{Empiler}(P, e)$ , un élément  $e$  quelconque à une pile  $P$ . L'opération  $\text{Dépiler}(P)$  retire un élément et le renvoie. L'élément retiré est toujours le dernier élément ajouté non encore retiré. Sur une pile vide le dépilement n'est pas défini. On dispose également d'un test à vide  $\text{EstVide}(P)$ .

#### Exercice 4 (Déplacement de pile).

On se donne trois piles  $P_1, P_2$  et  $P_3$ . La pile  $P_1$  contient des nombres entiers positifs. Les piles  $P_2$  et  $P_3$  sont initialement vides. En n'utilisant que ces trois piles :

1. Écrire un algorithme pour déplacer les entiers de  $P_1$  dans  $P_2$  de façon à avoir dans  $P_2$  tous les nombres pairs au dessus des nombres impairs.
2. Écrire un algorithme pour copier dans  $P_2$  les nombres pairs contenus dans  $P_1$ . Le contenu de  $P_1$  après exécution de l'algorithme doit être identique à celui avant exécution. Les nombres pairs doivent être dans  $P_2$  dans l'ordre où ils apparaissent dans  $P_1$ .

 1 pt  
9 min

 1 pt  
9 min

## Seconde partie : problèmes

11 points

### Exercice 5.

Le but de cet exercice est trouver un bon algorithme pour la recherche de valeurs médianes. En économie, le revenu médian est le revenu qui partage exactement en deux la population : la moitié de la population dispose d'un revenu plus élevé que le revenu médian, l'autre moitié d'un revenu moins élevé. Plus généralement, dans un tableau l'élément médian (ou valeur médiane) est l'élément qui serait situé au milieu du tableau si le tableau était trié. Lorsque le nombre d'éléments est pair, il n'y a pas d'élément exactement au milieu. Par convention nous prendrons, pour tout  $N$ , comme médian l'élément d'indice  $\lfloor \frac{N}{2} \rfloor$  dans le tableau trié (indexé à partir de 0).

Par exemple, dans le tableau suivant le revenu médian est de 1200 €.

individus	A	B	C	D	E	F	G	H
revenus mensuels	1400	2000	1300	300	700	5000	1200	800

Pour trouver le médian d'un tableau d'éléments deux à deux comparables on peut trier le tableau puis renvoyer la valeur de son élément d'indice  $\lfloor \frac{N}{2} \rfloor$ .

1. Pour cette solution, quelle complexité en moyenne (en temps) peut on obtenir, au mieux ? Quel algorithme de tri choisir ?

.5 pt  
4 min

On cherche un algorithme plus rapide. Il n'est sans doute pas nécessaire de trier tout le tableau pour trouver le médian. Le professeur Hoare suggère d'utiliser l'algorithme de partition de l'exercice 3 pour diviser les éléments à considérer. Après partition le tableau  $T$  est fait de trois parties : la première partie est un tableau  $T'$  des éléments de  $T$  plus petits que le pivot la deuxième partie ne contient que l'élément pivot et la troisième partie est un tableau  $T''$  des éléments plus grands que le pivot.

2. En fonction de l'indice  $p$  du pivot, dans quelle partie chercher le médian ?

.5 pt  
4 min

En général, on ne trouve pas le médian après le premier partitionnement. L'idée de Hoare est de continuer à partitionner la partie dans laquelle on doit chercher le médian, jusqu'à le trouver.

3. Dans quelle partie chercher le médian à chaque étape ? Répondre en donnant l'algorithme complet. Si nécessaire vous pouvez considérer que l'algorithme de partitionnement renvoie un triplet  $(T', p, T'')$  où  $T'$  et  $T''$  sont les deux sous-tableaux de  $T$  évoqués plus haut et  $p$  est l'indice dans  $T$  du pivot.

2 pt  
18 min

On suppose que le partitionnement d'un tableau de  $N$  éléments se fait exactement en  $N$  comparaisons ( $N - 1$  serait également possible). On s'intéresse à la complexité asymptotique de notre algorithme de recherche du médian, exprimée en nombre de comparaisons.

4. Quel est le meilleur cas ? Pour quelle complexité ? Quel est le pire cas ? Pour quelle complexité ?

1 pt  
9 min

Pour estimer si la moyenne est plus proche du meilleur cas ou du pire cas, on fait l'hypothèse que chaque fois que l'on fait une partition sur un tableau  $T$ , le médian se trouve dans un sous-tableau contenant  $\frac{2}{3}$  des éléments de  $T$ .

5. Donner un équivalent asymptotique du nombre de comparaisons faites (on pourra s'aider de l'exercice 1).
6. En supposant que ce résultat représente la complexité moyenne, l'algorithme est-il asymptotiquement optimal en moyenne ?

.5 pt  
4 min

.5 pt  
4 min

### Exercice 6 (Tours de Hanoi : jouer sans ordinateur).


Le jeu des tours de Hanoi se résout très simplement et élégamment de manière récursive au sens où on peut obtenir la liste des actions à effectuer, pour un  $n$  quelconque. Cependant cela ne permet pas à un joueur humain de résoudre le problème s'il ne dispose pas d'ordinateur : la solution suppose de mémoriser et surtout de reproduire un grand nombre d'états du jeu ce qui est hors de portée de notre esprit. Nous proposons ici de trouver la solution optimale sous la forme d'une méthode à appliquer pas à pas pour résoudre le jeu.

tot: 6 pt

Un *état* du jeu est une distribution des disques sur les trois piquets, représentés par des piles,  $a$  (départ),  $b$  (intermédiaire),  $c$  (arrivée). Dans l'état initial la pile  $a$  contient les  $n$  disques, représentés par les entiers de 1 à  $n$ , empilés du plus grand  $n$  (à la base), au plus petit 1 (au sommet). Les deux autres piles sont vides. Il faut déplacer un par un les disques d'une pile vers une autre sans que jamais un disque ne soit posé sur un disque plus petit. L'état final que l'on veut atteindre est celui où la pile d'arrivée contient les  $n$  disques.

L'opération de base est le déplacement d'un disque d'une pile  $p$  vers une autre pile  $q$ .


1. Définir  $\text{Deplacer}(p, q)$  à l'aide des primitives sur les piles.

 .5 pt  
4 min

On sait qu'il y a une unique solution optimale en nombre de déplacements, pour chaque  $n$ . Elle effectue exactement  $2^n - 1$  déplacements.

Ainsi, pour  $n = 2$ , il suffit d'appliquer 3 déplacements :  $\text{Deplacer}(a, b)$ ,  $\text{Deplacer}(a, c)$  et enfin  $\text{Deplacer}(b, c)$ .


2. Donner la suite des déplacements pour  $n = 3$ .


 1 pt  
9 min

Dans un état quelconque, le disque 1 (le plus petit disque) est toujours au sommet d'une pile  $p$ .

3. Si aucune des autres piles n'est vide, combien exactement de déplacements différents sont possibles ? Combien ont pour origine  $p$  ? Combien ont une autre origine ? Et si l'une des deux autres piles que  $p$  est vide ?
4. Si on déplace un disque est-il intéressant de le déplacer à nouveau le coup suivant ?
5. Quel disque est déplacé exactement un coup sur deux ? Combien de fois est-il déplacé en tout, en fonction de  $n$  ?


 1 pt  
9 min


 .5 pt  
4 min


 1 pt  
9 min

On admet la propriété suivante. Lors d'un déplacement du disque 1 celui-ci ne revient jamais sur la pile qu'il occupait avant son déplacement précédent. On en déduit qu'il y a deux possibilités pour les déplacements du disque 1 :

6. soit il occupe tour à tour les piles  $a, b, \dots$  (compléter, sur votre copie) soit il occupe tour à tour les piles  $a, c, \dots$  (compléter).
7. En raisonnant en arithmétique modulo trois, trouver une condition sur  $n$  permettant de déterminer exactement quelle sera la séquence des déplacements du disque 1.
8. Dédurre de ce qui précède les trois déplacements qui suivent celui-ci ( $n$  vaut 6) :

 .5 pt  
4 min

 1 pt  
9 min

 .5 pt  
4 min

$$\begin{array}{|c|c|c|} \hline 5 & 2 & 1 \\ \hline 6 & 3 & 4 \\ \hline a & b & c \\ \hline \end{array} \xrightarrow{\text{Deplacer}(b, a)} \begin{array}{|c|c|c|} \hline 2 & & 1 \\ \hline 5 & & 4 \\ \hline 6 & 3 & \\ \hline a & b & c \\ \hline \end{array}$$

