Institut Galilée — Algorithmique et arbres
Année 2009-2010 — L2

Partiel du 8 mars 2010

Aucun document autorisé. Le barême 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 auxilliaires vues en cours (comparaison, tris, sous-tableau, etc.).

Première partie — 11 points

Exercice 1 (Notation asymptotique)**.**
Rappeler les définitions utilisées et justifier (démontrer) vos réponses à partir de ces définitions.

1. Est-ce que $n \log n = O(n^2)$? — 1 pt, 9 min
2. Est-ce que $\log(n!) = O(n \log n)$? — 1 pt, 9 min
3. Est-ce que $\log(n!) = \Theta(n^2)$? — 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 (Tri par base)**.**
Soit la suite d'entiers décimaux 003, 132, 033, 122, 220. On utilise un tri stable pour trier ces entiers selon leur chiffre le moins significatif (chiffre des unités), puis pour trier la liste obtenue selon le chiffre des dizaines et enfin selon le chiffre des centaines. Écrire les trois listes obtenues. — 1 pt, 9 min

Exercice 4 (Invariant de boucle)**.**
Étant donné un tableau T de $N > 0$ entiers, l'algorithme Maximum renvoie l'indice de l'élément maximum de T. Démontrer le à l'aide d'un invariant de boucle. — 1.5 pt, 13 min

Fonction Maximum(T)

```
m = 0;
pour i ← 1 à Taille(T) − 1 faire
    si T[i] ≥ T[m] alors
        m = i;
retourner m ;
```

Exercice 5 (Arbre de décision, meilleur cas)**.**
Rappel : le tri bulle s'arrête lorsqu'il a fait une passe au cours de laquelle il n'y a eu aucun échange. — tot: 1,5 pt

1. Dessiner l'arbre de décision du tri bulle sur un tableau de trois éléments $[a, b, c]$. — 1 pt, 9 min
2. On note $C(N)$ le nombre de comparaisons faites par le tri bulle dans le meilleur des cas sur un tableau de taille N. Quel tableau en entrée donne le meilleur cas du tri bulle ? Combien vaut $C(N)$ exactement ? (En fonction de N.) — 0,5 pt, 4 min

Exercice 6.
Soit la fonction MaFonction donnée ci-dessous en pseudo-code et en langage C (au choix). — tot: 2 pt

1. En supposant que le tableau passé en entrée est trié par ordre croissant, que renvoie exactement cette fonction (sous quel nom connaissez-vous cet algorithme) ? — 0,5 pt, 4 min

2. Donner une majoration asymptotique, en pire cas, du temps d'exécution de MaFonction en fonction de la taille n du tableau en entrée. Démontrer ce résultat dans les grandes lignes (on pourra se contenter de raisonner sur les cas $n = 2^k - 1$ pour $k \geq 0$ entier).

1.5 pt
13 min

Fonction MaFonction(T, c)

Entrées : Un tableau croissant d'entiers $T[0..n-1]$ de taille Taille(T) $= n$ et un entier c.
Sorties : Une case du tableau T ou bien une case vide.
si Taille(T) > 0 **alors**
  $m = \lfloor \text{Taille}(T)/2 \rfloor$ *(partie entière inférieure)*;
  si $c = T[m]$ **alors**
    retourner $T[m]$;
  sinon
    si $c < T[m]$ **alors**
      T' = le sous tableau $T[0..m-1]$;
      retourner MaFonction(T', c);
    sinon
      T'' = le sous tableau $T[m+1..\text{Taille}(T)-1]$;
      retourner MaFonction(T'', c);
sinon
  retourner case vide;

```
int * mafonction(int T[], int taille, int c) {
  int milieu;
  if (taille > 0) {
    milieu = taille / 2;
    if (c == T[milieu]) {
      return &(T[milieu]);
    }
    else {
      if (c < T[milieu]) {
        return mafonction(T, milieu, c);
      }
      else { /* On a c > T[milieu] */
        return mafonction(T + milieu + 1, taille - milieu - 1, c);
      }
    }
  }
  return NULL;
}
```

Figure 1: `mafonction` en langage C

Seconde partie : problèmes — 9 points

Exercice 7 (Plus grand sous-tableau à somme nulle)**.**
Soit un tableau T de N entiers. On cherche un sous-tableau (contigu) de T, tel que la somme des éléments de ce sous-tableau soit nulle. De plus on souhaite que ce sous-tableau soit le plus grand possible. Un sous-tableau T' non-vide de T est donné par un couple d'indices (i, j) avec $0 \leq i \leq j \leq N-1$, les éléments de T' sont alors $T[i], T[i+1], \ldots, T[j]$.

tot: 3 pt

On peut pour cela calculer toutes les sommes pour tous les sous-tableaux (non-vides) possibles de T et parmi ceux pour lesquels cette somme est nulle en trouver un de longueur maximum.

1. Quel serait le temps d'exécution d'un algorithme fondé sur cette méthode, en notation asymptotique et en fonction de N ? (Il faut expliquer comment vous écririez l'algorithme mais sans nécessairement le détailler). 1 pt 9 min

Supposons maintenant que l'on fabrique un autre tableau S de même taille que T de la manière suivante. Chaque élément $S[i]$ est un couple d'entiers. Au départ, le premier de ces deux entiers, `S[i].somme` en C, est égal à la somme $\Sigma_{j=0}^{j=i} T[j]$. Le second, `S[i].indice`, a simplement pour valeur l'indice i. On tri ensuite S par sommes croissantes à l'aide d'un tri stable.

2. Comment peut on résoudre le problème initial à l'aide de S ? (Décrire l'algorithme sans nécessairement le détailler, en supposant que S trié est fourni). 1 pt 9 min
3. Quel temps d'exécution global peut on obtenir par cette méthode (donner un temps d'exécution pour chaque étape). 1 pt 9 min

Exercice 8.

tot: 6 pt

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 et pour simplifier, 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 du tri rapide, le fameux *quicksort* (comme en cours, la valeur pivot est choisie dans la première case). Après partition le tableau T est fait de trois partie : 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 ? 1 pt 9 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 (mais en considérant que la fonction de partionnement est fournie). Si nécessaire vous pouvez considérer que la fonction de partionnement 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.

1 pt
9 min

6. En supposant que ce résultat représente la complexité moyenne, l'algorithme est-il asymptotiquement optimal en moyenne ?

.5 pt
4 min