Algorithmique avancée

Examen du 29 janvier 2002, 8h00-11h00

Jean-Michel Dischler et Frédéric Vivien

Aucun document n'est autorisé

Remarques et commentaires :
- Commencez par lire le sujet dans son intégralité.
- Les deux problèmes sont indépendants les uns des autres.
- Écrivez lisiblement et en français, car les copies seront lues (anonymat oblige) !
- Facilitez la lecture et la compréhension des algorithmes proposés.
- Ce sujet est **infaisable en trois heures** : sa longueur excessive vous permet de choisir à quelles questions vous souhaitez répondre, et vous n'avez aucunement besoin de tout faire pour décrocher une très bonne note. Le barème, sur 30 points, est indicatif.

Rappels

Théorème 1 (Résolution des récurrences « diviser pour régner »).
Soient $a \geq 1$ et $b > 1$ deux constantes, soit $f(n)$ une fonction et soit $T(n)$ une fonction définie pour les entiers positifs par la récurrence :

$$T(n) = aT(n/b) + f(n),$$

où l'on interprète n/b soit comme $\lfloor n/b \rfloor$, soit comme $\lceil n/b \rceil$.

$T(n)$ peut alors être bornée asymptotiquement comme suit :

1. *Si $f(n) = O(n^{(\log_b a)-\epsilon})$ pour une certaine constante $\epsilon > 0$, alors $T(n) = \Theta(n^{\log_b a})$.*
2. *Si $f(n) = \Theta(n^{\log_b a})$, alors $T(n) = \Theta(n^{\log_b a} \log n)$.*
3. *Si $f(n) = \Omega(n^{(\log_b a)+\epsilon})$ pour une certaine constante $\epsilon > 0$, et si $af(n/b) \leq cf(n)$ pour une constante $c < 1$ et n suffisamment grand, alors $T(n) = \Theta(f(n))$.*

1 Enregistrement d'un CD sur cassette (21 points)

Nous avons à notre disposition une cassette de 60 minutes : sur chacune de ses faces nous pouvons donc enregistrer 30 minutes de musique. Nous souhaitons enregistrer sur cette cassette des morceaux provenant d'un CD contenant n morceaux pour une durée totale de 78 minutes : nous allons donc devoir choisir quels morceaux mettre sur la cassette et lesquels ne pas enregistrer, sachant que nous nous imposons qu'un morceau du CD soit enregistré au plus une fois sur la cassette.

Notations. Pour faciliter l'écriture des algorithmes, nous supposons que tous les morceaux durent un nombre entier de minutes et qu'ils sont tous de durée non nulle ! Nous notons n le nombre de morceaux et, afin d'être général, d la durée maximale enregistrable sur une *face* de la cassette (dans l'exemple ci-dessus, $d = 30$). Finalement, nous notons d_i la durée du i^{e} morceau (pour $1 \leq i \leq n$).

Le choix des morceaux à enregistré est déterminé par la politique d'enregistrement. Plusieurs politiques sont envisageables.

1.1 Un maximum de musique, quitte à couper des morceaux (2 points)

Dans cette partie nous cherchons à maximiser la durée totale de musique enregistrée sur la cassette, sachant que l'on s'autorise à couper un morceau (et ce autant de fois que nécessaire).

1. Proposez un algorithme glouton de résolution du problème. (1 point)

   Proposition de notation : si le morceau i n'est pas entièrement enregistré sur une face de la cassette mais seulement un fragment de longueur f, on pourra utiliser la notation : $i(\frac{f}{d_i})$.

2. Quelle est sa complexité ? (1 point)

1.2 Un maximum de musique mais sans couper de morceaux (19 points)

Dans cette partie nous cherchons à maximiser la durée totale de musique enregistrée sur la cassette, sachant que l'on *interdit* de couper un morceau : un morceau figure en entier sur une face ou n'est pas enregistré du tout.

1. **Relation de récurrence.** On note $durée(i, t_1, t_2)$ la durée maximale de musique que l'on peut enregistrer sur la cassette, sachant que l'on peut mettre au maximum t_1 minutes de musique sur la première face et t_2 sur la deuxième, et sachant que l'on n'enregistre que des morceaux parmi les i premiers. $durée(n, d, d)$ nous donnera donc la solution de notre problème.

   Proposez une formule de récurrence définissant $durée(i, t_1, t_2)$. (3 points)

   Indication : dans une solution optimale, soit le i^{e} morceau est enregistré sur la première face, soit il est enregistré sur la deuxième, soit il n'est enregistré sur aucune des deux faces.

   Note sur la correction. Toutes les questions suivantes dépendent de la formule établie à cette question ci. Ces questions seront corrigées d'après **votre** réponse à cette question : par exemple, si l'algorithme que vous proposerez à la question 3a est bien la transcription de votre formule de récurrence sous la forme d'un algorithme de programmation dynamique, vous obtiendrez le maximum de points à cette question, même si votre formule de récurrence est fausse.

2. **Résolution récursive.** (3 points)
   (a) Proposez un algorithme de résolution récursive du problème calculant simplement la récurrence établie à la question précédente. (1 point)
   (b) Montrez que cet algorithme est de complexité super-polynomiale dans le pire cas. (2 points)
   Indication : vous pourrez supposer ici que tous les morceaux durent une minute.

3. **Résolution par programmation dynamique « pure ».** (4 points)
   (a) Proposez un algorithme basé sur le principe de la programmation dynamique et résolvant notre problème. Cet algorithme utilisera bien évidemment la relation de récurrence précédemment établie. (3 points)
   (b) Quelle est la complexité de cet algorithme ? (1 point)

4. **Résolution par recensement.** (5 points)
   (a) Proposez un algorithme basé sur le principe de recensement (variante de la programmation dynamique) et résolvant notre problème. (3 points)
   (b) Quelle est la complexité de cet algorithme ? (1 point)
   (c) La solution par programmation dynamique « pure » et celle par recensement sont-elles de complexités comparables ? Un des deux algorithmes effectue-t-il plus de calculs que l'autre (sans tenir compte des initialisations) ? (1 point)

5. **Sélection des morceaux.** Jusqu'à présent, tout ce que l'on a fait, c'est de calculer la durée totale maximale de musique enregistrable sous les conditions de l'énoncé. Nous n'avons pas encore explicité comment sélectionner les morceaux qui permettent d'atteindre ce maximum. Modifiez un de vos algorithmes pour qu'il génère une telle liste des morceaux (une liste par face). (4 points)

2 Élections présidentielles aux États-Unis (9 points)

Le décompte des voix après une élection présidentielle aux États-Unis est un problème excessivement sensible et qui doit être résolu le plus rapidement possible. Ce problème se trouve compliqué par le fait que n'importe qui peut recevoir des voix et être élu, les électeurs pouvant tout simplement écrire sur leur bulletin la personne pour laquelle ils votent [1]. Dans le pire des cas le nombre de personnes ayant reçu au moins une voix est égal au nombre de bulletins !

Notations. Nous notons n le nombre des votes. Ces votes sont stockés dans le tableau *Votes*. Pour faciliter l'écriture des algorithmes, nous supposerons que n est une puissance de deux (il existe donc un entier p tel que $n = 2^p$). Nous supposons également que la seule opération à notre disposition nous permet de vérifier si deux éléments sont ou non égaux.

Problème. Nous nous intéresserons ici uniquement au problème de savoir si quelqu'un a obtenu la majorité absolue des voix, c'est-à-dire si quelqu'un a obtenu strictement plus de $n/2$ voix.

1. **Algorithme naïf.** (3 points)
   (a) Écrivez un algorithme qui calcule le nombre de voix reçues par le citoyen x parmi les votes stockés entre les indices i et j du tableau *Votes*. (0,5 point)
   (b) Quelle est la complexité de cet algorithme ? (1 point)
   (c) Au moyen de l'algorithme précédent, écrivez un algorithme MAJORITÉ-ABSOLUE qui vérifie si, parmi les citoyens qui ont reçu des voix, il en existe un qui a remporté la majorité absolue des votes enregistrés dans le tableau *Votes*. (0,5 point)
   (d) Quelle est la complexité de cet algorithme ? (1 point)
2. **Algorithme « diviser pour régner ».** (6 points)
   (a) Proposez un algorithme MAJORITÉ-ABSOLUE-DIV construit suivant le paradigme « diviser pour régner ». Cet algorithme divisera en deux le tableau *Votes* sur lequel il travaille. Cet algorithme renverra le couple (VRAI, x) s'il existe un citoyen (noté x) ayant reçu la majorité absolue des votes stockés dans le tableau *Votes* et renverra le couple (FAUX, 0) sinon. (4 points)
   (b) Quelle est la complexité de cet algorithme ? (2 points)

[1] Cette information, aussi loufoque qu'elle paraisse, est véridique.