Algorithmique — M1

Examen du 11/1/11

Université Paris Diderot

Documents autorisés : Deux feuilles de papier format A4 Durée : 3h (de 15h30 à 18h30)

On applique un algorithme de cours

Exercice 1 – *Routage*

Le serveur S est connecté à la machine T par un réseau avec les noeuds A, B, C, D, les capacités de connexions entre les noeuds sont (en Mbit/s) :

	A	B	C	D	T
S	2	6	1		
A		3		7	
B				3	5
C		2		6	
D	3				4

L'utilisateur de la machine T télécharge un très grand fichier du serveur S. On veut trouver le routage qui maximise le débit.

1. Quel problème algorithmique de cours correspond à ce problème de routage ? Comment s'appelle l'algorithme du cours ?
2. Appliquez cet algorithme. Donnez toutes les étapes de son application.
3. Quel est le débit maximal, quel routage assure ce débit ?

Un algorithme facile

Exercice 2 – *Valeur encadrée*

Étant donné un tableau trié d'entiers $A[s..f]$ et deux entiers ("bornes") $a \leqslant b$, on cherche s'il existe un élément $A[i]$ du tableau tel que $a \leqslant A[i] \leqslant b$ (s'il y en a plusieurs trouvez un)

Exemple Soit le tableau $A[1..5] = [3, 7, 8, 43, 556]$ et les bornes $a = 40, b = 50$. Dans ce cas-là la valeur encadrée existe : c'est $A[4] = 43$.

1. Donnez (en pseudocode) un algorithme "diviser-pour-régner" qui résout ce problème. Expliquez briévement.
2. Analysez sa complexité.

Un jeu - deux puzzles - deux algorithmes

Un *Domino* est un rectangle qui contient 2 lettres, par exemple | A | H | (il ne peut pas être retourné).

Règle : Une séquence de dominos est une *chaîne*, si les lettres voisines coïncident sur chaque paire des dominos consécutifs, par exemple | A | H | | H | K | | K | A | | A | Z | | Z | V | .

Donnée initiale pour les deux puzzles : n pièces de dominos : $D_1 =$ | x_1 | y_1 | ,..., $D_n =$ | x_n | y_n |

Puzzle 1 (permutation) : Il faut trouver une permutation de tous ces dominos qui forme une chaîne (selon la règle ci-dessus), si celle-là existe.

Puzzle 2 (sous-séquence) : Il faut trouver la plus longue sous-séquence de tous ces dominos qui forme une chaîne (selon la règle ci-dessus).

Exemple : Pour les pièces | A | H | , | A | Z | , | K | A | , | H | K | , | Z | V | :

- la solution du puzzle 1 est | A | H | | H | K | | K | A | | A | Z | | Z | V |;
- la solution du puzzle 2 est | A | H | | H | K | , ou bien | A | Z | | Z | V | .

Exercice 3 – *Premier puzzle - backtracking*

. On cherche à développer un algorithme de type backtracking qui résout le puzzle 1.

1. Écrivez l'algorithme (en pseudocode)
2. Expliquez cet algorithme (vous pouvez vous inspirer de l'indication)
3. Estimez le nombre d'opérations nécessaire
4. Montrez le déroulement de votre algorithme pour les pièces
   | A | H | , | A | Z | , | K | A | , | H | K | , | Z | V | .

   Indication : Essayez de répondre aux questions suivantes pour parvenir à un tel algorithme.
   - Comment tester que deux dominos peuvent être adjacents ?
   - Comment définir une solution partielle ?
   - Quelle est une solution partielle de taille 0 ?
   - Comment passer d'une solution partielle à une solution plus grande ?
   - Quand s'arrêter ?
   - Comment représenter une solution partielle par une structure de données ? Comment représenter les dominos déjà utilisés en chaîne et les dominos encore disponibles ?

Exercice 4 – *Deuxième puzzle - programmation dynamique*

On cherche à développer un algorithme de type programmation dynamique qui résout le puzzle 2.

1. Soit $c(i)$ une fonction entière qui donne la longueur de la chaîne la plus longue qui est une sous-séquence de $D_1, D_2, \ldots D_i$ et qui se termine par la pièce D_i. Écrivez les équations de récurrence pour cette fonction sans oublier les cas de base.
2. Écrivez un algorithme efficace (récursif avec "marquage" ou itératif) pour calculer c .
3. En sachant calculer la fonction choisie c, comment trouver la longueur de la sous-chaîne la plus longue parmi $D_1, D_2, \ldots D_i$.
4. Analysez la complexité de votre algorithme.
5. Montrez le déroulement de votre algorithme pour les pièces
   | A | H | , | A | Z | , | K | A | , | H | K | , | Z | V | .
6. Comment modifier votre algorithme pour qu'il trouve la sous-chaîne la plus longue (et non seulement sa longueur).