L3-INFO Algorithmique et Modélisation

Université Joseph Fourier GRENOBLE UFR IM²AG

Examen – session 1 – 6 mai 2015 – durée 3 h

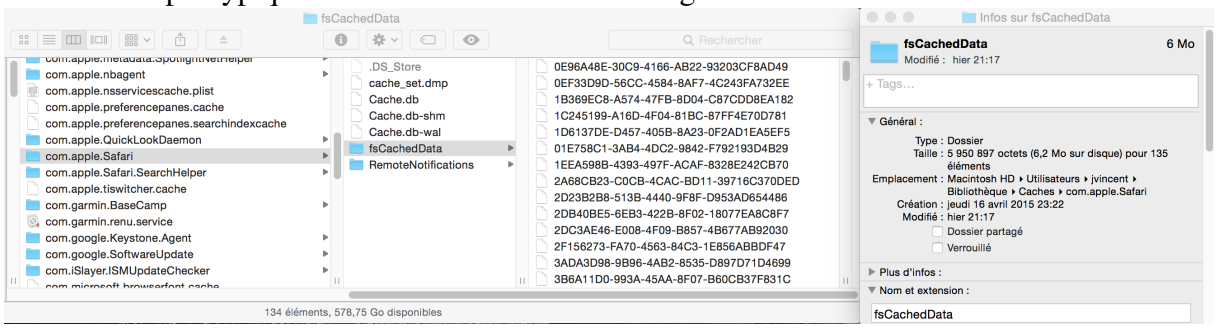
Sont interdits : les documents, les ordinateurs, les téléphones (incluant smartphone, tablettes,... tout ce qui contient un dispositif électronique).
Seuls les dictionnaires papier pour les personnes de langue étrangère sont autorisés.
Il sera tenu compte de la qualité de la rédaction et de la clarté de la présentation.
Le barème indicatif : Exercice 1 : 4pts ; Exercice 2 : 6pts ; Exercice 3 : 10pts.

Exercice 1 : Navigation

Lors de l'observation du comportement de surfers sur le Web on constate que les pages visitées sont visitées plusieurs fois, qu'un même objet peut apparaitre sur plusieurs pages visitées (logos, images, ...). Pour améliorer les performances d'un navigateur internet en évitant de télécharger plusieurs fois un même objet, on implémente un mécanisme de cache qui stocke localement une partie du contenu des pages visitées.

Afin de retrouver rapidement les objets dans le cache, on nomme les objets à partir de leur url ce qui permet un accès direct à la copie locale. Ce nommage est effectué par une fonction de hachage qui prend en argument l'url et qui renvoie une chaîne de caractères de longueur fixe.

Un exemple typique de cache est donné dans la figure ci-dessous :



Pour simplifier l'analyse on suppose que seuls les 8 premiers caractères sont générés par une fonction de hachage (chiffres et lettres A,B,C,D,E,F).

Question 1 : Collision

En supposant que la taille du cache est de 6Mo et que la taille moyenne d'un objet caché est de 50Ko calculer la probabilité d'avoir une collision, c'est à dire la probabilité que 2 objets aient la même valeur de hachage. On donnera juste l'ordre de grandeur de cette probabilité. Quelle serait la taille du cache, en gardant des objets de même taille moyenne, pour que cette probabilité soit de l'ordre de 1% ?

Question 2 : Analyse

Commenter le résultat. Est-il pertinent de tester si l'on a une collision ? Comment s'améliore cette probabilité si on concatène au résultat de la première fonction de hachage, le résultat d'une deuxième fonction de hachage générant une chaîne de taille 12 (fin du nom de fichier dans l'exemple ci-dessus) ?

Exercice 2 : Compter les chemins

L'objectif de cet exercice est de concevoir un algorithme qui compte le nombre de chemins de longueur l allant de i à j dans un graphe orienté $\mathcal{G} = (X, A)$. Les chemins peuvent passer plusieurs fois par le même sommet ou le même arc

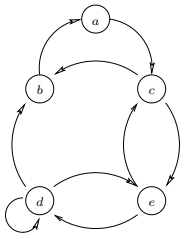
$\mathcal{G} = (X, A)$ avec $|X| = 5$ et $|A| = 9$

$X = \{a, b, c, d, e\}$

$A = \{(a, c), (b, a), (c, b), (c, e), (d, b), (d, d), (d, e), (e, c), (e, d)\}$

$C = (a, c, e, d, e, d, d, b)$

C est un chemin de longueur 7 de a à b



Question 1 : Exemple

Donner la matrice d'adjacence du graphe ci-dessus et le nombre de chemins de a à b de longueur l pour $l = 0, 1, 2, 3, 4, 5$.

On note $n_{i,j}^{(l)}$ le nombre de chemins de i à j de longueur l et $N^{(l)}$ la matrice associée

Question 2 : Équations de récurrence

Écrire les équations de récurrences exprimant les coefficients $n_{i,j}^{(l)}$ en fonction des coefficients $n_{i,k}^{(l-1)}$. Justifier en français ces équations et faire un dessin explicatif.

Question 3 : Algorithme simple puis avancé

En vous inspirant des équations de récurrence écrire un algorithme qui calcule la matrice $N^{(l)}$ en $|X|^3.l$ opérations. Expliquer une méthode pour réduire le coût à moins de $C.|X|^{\log_2 7}.2 \log l$ opérations avec C un facteur constant.

Problème : Gros sous-tableau

On se donne un tableau T de taille n (indicé de 1 à n), contenant des nombres (positifs ou négatifs). On cherche à trouver le sous-tableau contigu $T[i : j]$ de somme maximale. Autrement dit, on cherche les indices i et j qui maximisent $\sum_{k=i}^{j} T[k]$.

Question 1 : Exemples

Calculer (à la main), une solution pour les tableaux suivants (à chaque fois, donner les indices d'un sous-tableau de somme maximale et la somme).

$$T_1 = [1, 4, -2, 3, 3, -4, -2, -2, 2, -1] \quad T_2 = [1, -2, 3, 3, 1, -5, -1, -3, 8, -5]$$

Question 2 : Algorithme naïf

Évaluer le coût d'un algorithme naïf calculant les sommes de tous les sous-tableaux.

Question 3 : Programmation dynamique

Proposer un algorithme en $\mathcal{O}(n^2)$ pour résoudre le problème (indication : on pourra se servir de programmation dynamique pour calculer toutes les valeurs de $\sum_{k=i}^{j} T[k]$).

Question 4 : Diviser pour régner

Pour résoudre ce problème, on utilise une approche diviser pour régner. On découpe le tableau T en deux sous-tableaux T_L et T_H de taille moitié : T_L contient les éléments de 1 à $\frac{n}{2}$ et T contient les éléments de $\frac{n}{2}$ à n, on considèrera dans un premier temps que n est une puissance de 2. Le sous-tableau de somme maximale est un des trois tableaux suivants :

T_L^{max} le sous-tableau de somme maximale de T_L ;
T_H^{max} le sous-tableau de somme maximale de T_H ;
T_{LH} la concaténation du sous-tableau de somme maximale de T_L terminant à $\frac{n}{2}$ et du sous-tableau de somme maximale de T_H commençant à $\frac{n}{2}+1$.

Les sous-tableaux T_L et T_H sont calculés en appelant la fonction récursivement.

1. Faire un dessin explicatif de l'algorithme.
2. Montrer que l'on peut calculer le tableau T_{LH} en temps $\mathcal{O}(n)$.
3. Soit $C(n)$ la complexité de l'algorithme sur un tableau de taille n. Donner une formule de récurrence pour $C(n)$ et donner l'ordre de grandeur de $C(n)$.
4. Comment adapter l'algorithme lorsque n n'est pas une puissance de 2 ?
5. Écrire l'algorithme récursif.

Question 5 : Encore mieux ?

Bart prétend que l'on peut résoudre le problème en utilisant l'algorithme suivant :

Entrées : Un tableau T de taille n
Sorties : La somme des valeurs du sous-tableau de somme maximale
somme_courante $\leftarrow 0$;
meilleure_somme $\leftarrow 0$;
pour $k = 0$ *à* $n-1$ **faire**
  somme_courante $\leftarrow$ somme_courante $+ T[k]$;
  si *somme_courante* $>$ *meilleure_somme* **alors**
    meilleure_somme $\leftarrow$ somme_courante;
  sinon
    si *somme_courante* ≤ 0 **alors**
      somme_courante $\leftarrow 0$
Retourner meilleure_somme

1. Quelle est la complexité de cet algorithme ?
2. Cet algorithme rend-il la bonne somme ? Justifier votre réponse, c'est-à-dire : si oui, démontrer que l'algorithme est correct, si non donner un contre-exemple.
3. Si l'algorithme donne la somme maximale, proposer une modification de l'algorithme pour qu'il retourne également les indices du sous-tableau de somme maximale.

Question 6 : Synthèse

Parmi les quatre algorithmes proposés, lequel recommander, et pourquoi ?