Option Informatique en Spé MP et MP*

Devoir surveillé du mercredi 3 avril 2002

Résumé

Le premier problème propose l'étude d'un algorithme efficace pour construire un arbre binaire de recherche optimal, lorsque l'on connaît pour chaque clé la probabilité qu'elle fasse l'objet d'une recherche. Il est alors avantageux de placer les clés les plus fréquemment demandées près de la racine. Nous retrouvons ici l'idée du magasinier économe de ses gestes, qui placera près de lui les articles qui ont la faveur de sa clientèle, et reléguera les rossignols au fond du magasin !

Nous verrons que cette question entretient une forte parenté avec la recherche d'une triangulation optimale d'un polygone convexe et avec l'optimisation de l'évaluation d'un produit de plusieurs matrices.

Dans le deuxième problème, nous nous intéressons à une relation sur des mots de même longueur : deux mots sont liés si l'on peut transformer le premier en le deuxième par retournement d'un facteur. Cette question présente un intérêt dans l'étude du génôme : des séquences appelées *transposons* peuvent venir s'insérer dans un brin d'ADN, dans l'un ou l'autre des deux sens de lecture.

Veuillez rédiger chaque partie sur une copie séparée.

Table des matières

1 Arbres binaires de recherche optimaux

1.1 Préliminaires Caml

▶ Nous disposons de la fonction `intervalle`, définie comme suit :

```
let rec intervalle i j = if i>j then [] else i::(intervalle (i+1) j) ;;
```

Question 1 • Rédigez en Caml une fonction de signature

```
min_of_list : int list -> int
```

spécifiée comme suit : `min_of_list q` rend le plus petit élément de la liste d'entiers q, ou lève une exception si cette liste est vide. L'emploi de références ou de structures de données mutables (vecteurs) est interdit.

Question 2 • Déterminez le type de la fonction suivante, et ce qu'elle calcule :

```
let somme f i j =
 let q = map (fun k -> f.(k)) (intervalle i j) in
 it_list (prefix +) 0 q ;;
```

1.2 Arbres binaires de recherche

▶ Un arbre binaire t non réduit à une feuille sera noté (g, r, d) où g (resp. d) est le sous-arbre gauche (resp. droit) et r la racine. Soit t un arbre binaire dont les nœuds sont étiquetés par les éléments d'un ensemble totalement ordonné $(E, \leqslant)$; nous dirons que t est un arbre binaire de recherche (en abrégé : un a.b.r.) s'il vérifie la propriété suivante :

- ou bien t est réduit à une feuille ;
- ou bien $t = (g, r, d)$ auquel cas r majore (resp. minore) les étiquettes des nœuds de g (resp. d), et g et d sont eux aussi des a.b.r.

▶ Soit $t = \big((g_g, r_g, d_g), r, d\big)$ un arbre binaire non réduit à une feuille, et dont le sous-arbre gauche n'est pas réduit à une feuille. La *rotation à droite* transforme t en l'arbre $t' = \big(g_g, r_g, (d_g, r, d)\big)$. Définissons de même la rotation à gauche. La figure 1 présente deux arbres a et a' : il est clair que a' se déduit de a par une rotation droite autour de la racine de a.

▶ Nous dirons que deux arbres sont *équivalents* si l'on peut passer de l'un à l'autre au moyen d'une suite de rotations. Il est clair que nous définissons ainsi une relation d'équivalence sur l'ensemble des arbres binaires, et que deux arbres équivalents ont même nombre de nœuds et mêmes étiquettes.

Question 3 • Montrez que l'image d'un a.b.r. par une rotation est encore un a.b.r.

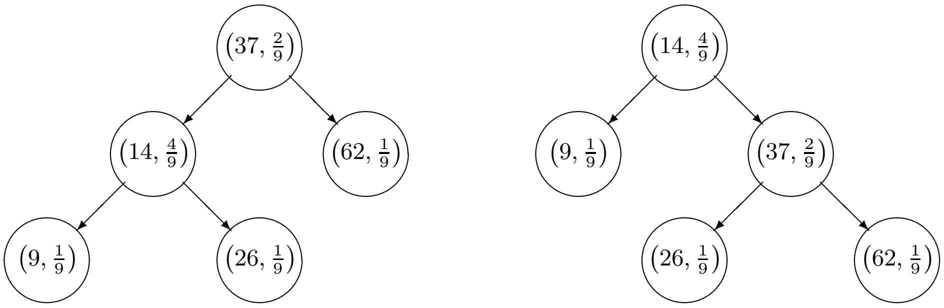
Question 4 • Montrez que deux a.b.r. ayant mêmes étiquettes sont équivalents.

1.3 Arbres binaires de recherche avec nœuds pondérés

▶ Nous nous intéressons à des arbres binaires de recherche avec nœuds pondérés. Ce sont des arbres binaires dont les nœuds sont étiquetés par des couples (Z_i, f_i) où les clés Z_i sont des éléments d'un ensemble totalement ordonné $(E, \leqslant)$ et les *poids* f_i sont des réels positifs. Les feuilles ne portent aucune information. La clé Z_i de chaque nœud majore (resp. minore) les clés du sous-arbre gauche (resp. droit).

▶ La figure 1 présente deux tels arbres. Les clés sont des naturels ; par souci de clarté, les feuilles n'apparaissent pas.

▶ Soit t un arbre binaire de recherche avec nœuds pondérés. En pratique, le réel f_i est la probabilité qu'une recherche fructueuse porte sur la clé Z_i ; dans ce cas, la somme des f_i est égale à 1. Par exemple, les clés peuvent être les mots d'un dictionnaire, classés par ordre lexicographique. Des études statistiques effectuées sur un ensemble homogène de textes montreraient que le mot *menteur* est plus fréquent que le mot *abracadabrantesque*. Il est souhaitable que les mots les plus fréquents soient plus proches de la racine



L'arbre a L'arbre a'

Figure 1: deux exemples d'arbres binaires de recherche avec nœuds pondérés

de t. Associons à t son *coût*, noté $\pi(t)$ et égal par définition à la somme des produits $(1+p_i)\times f_i$ où p_i est la profondeur du nœud (Z_i, f_i). Par exemple, pour les arbres a et a' de la figure 1 nous trouvons :

$$\begin{aligned}
\pi(a) &= 1\times\frac{2}{9}+2\times\frac{4}{9}+2\times\frac{1}{9}+3\times\frac{1}{9}+3\times\frac{1}{9}=2\\
\pi(a') &= 1\times\frac{4}{9}+2\times\frac{1}{9}+2\times\frac{2}{9}+3\times\frac{1}{9}+3\times\frac{1}{9}=\frac{16}{9}
\end{aligned}$$

Ceci nous amènerait à préférer l'arbre a'. Nous dirons qu'un arbre t est *optimal* si tout arbre t' équivalent à t vérifie $\pi(t') \geqslant \pi(t)$. Il est clair que a' est optimal.

Question 5 • Lorsque les f_i ont toutes la même valeur, que pouvez-vous dire d'un arbre optimal ?

Question 6 • Construisez un arbre optimal filiforme possédant n nœuds.

► L'algorithme glouton consiste à choisir comme racine le nœud dont le poids est le plus élevé, puis à recommencer avec les sous-arbres gauche et droit. Par exemple, l'arbre a' serait obtenu en appliquant l'algorithme glouton.

Question 7 • Donnez un exemple montrant que l'algorithme glouton ne donne pas nécessairement un arbre optimal.

► Nous allons décrire un algorithme efficace pour construire un arbre optimal, à partir de la liste $(Z_i, f_i)_{1\leqslant i\leqslant n}$ des étiquettes, classées par ordre croissant des clés : $Z_i \leqslant Z_{i+1}$ pour $i \in [\![1, n-1]\!]$. Soient i et j deux indices vérifiant $1 \leqslant i \leqslant j \leqslant n$; nous noterons $\mu(i,j)$ le coût d'un arbre optimal dont les étiquettes sont celles d'indice compris entre i et j inclus. Par exemple, aux arbres a et a' de la figure 1 est associée la même liste $\big((9,\frac{1}{9});(14,\frac{4}{9});(26,\frac{1}{9});(37,\frac{2}{9});(62,\frac{1}{9})\big)$ et nous avons $\mu(2,4)=\frac{11}{9}$. Nous conviendrons que $\mu(i,j)=0$ si $i=j+1$.

Question 8 • Combien vaut $\mu(i,i)$?

Question 9 • Justifiez l'affirmation suivante : tout sous-arbre d'un arbre optimal est lui-même optimal.

Question 10 • Pour $1 \leqslant i < j \leqslant n$, justifiez la formule :

$$\mu(i,j)=\left(\sum_{i\leqslant k\leqslant j} f_k\right)+\min_{i\leqslant k\leqslant j}\big(\mu(i,k-1)+\mu(k+1,j)\big)$$

► L'étudiant Charles-Édouard Dugratin a transcrit en Caml la formule précédente :

```
let rec mu i j = somme f i j +
 (if i>=j then 0 else
 let v = make_vect (j-i+1) 0 in
 for k = i to j do
  v.(k-i) <- mu i (k-1) + mu (k+1) j
 done;  min_of_vect v) ;;
```

Question 11 • Proposez une définition de `min_of_vect`.

Question 12 • Que pensez-vous des performances du programme proposé par notre ami Charles-Édouard ?

Question 13 • Expliquez comment ordonnancer les calculs, pour construire une matrice M carrée d'ordre n, telle $M_{i,j} = \mu(i,j)$ pour $1 \leqslant i \leqslant j \leqslant n$. Comptez les opérations (comparaisons et additions) que vous effectuez.

Question 14 • Mettez ceci en œuvre, en rédigeant une fonction :

```
construire_matrice : int vect -> int vect vect
```

spécifiée comme suit : `construire_matrice f` rend la matrice M, carrée d'ordre n, associée au vecteur des fréquences $f = (f_k)_{1 \leqslant k \leqslant n}$.

Question 15 • Montrez que le remplissage de la matrice M nous donne aussi les informations pour construire un arbre binaire optimal.

1.4 Triangulations optimales

▶ Soit $\mathcal{P} = (A_k)_{1 \leqslant k \leqslant n}$ un polygone convexe à n sommets. Convenons de noter $A_0 = A_n$. Les côtés de $\mathcal{P}$ sont les n segments $[A_k, A_{k+1}]$, où $k \in [\![0, n-1]\!]$. Les *diagonales* de $\mathcal{P}$ sont les segments $[A_i, A_j]$ reliant deux sommets distincts non adjacents ; ceci revient à dire que $|i-j|$ est au moins égal à 2. Une *triangulation* de $\mathcal{P}$ est une famille maximale de diagonales n'ayant deux à deux aucun point commun à l'intérieur de $\mathcal{P}$. La figure 2 donne un exemple de triangulation d'un octogone régulier.

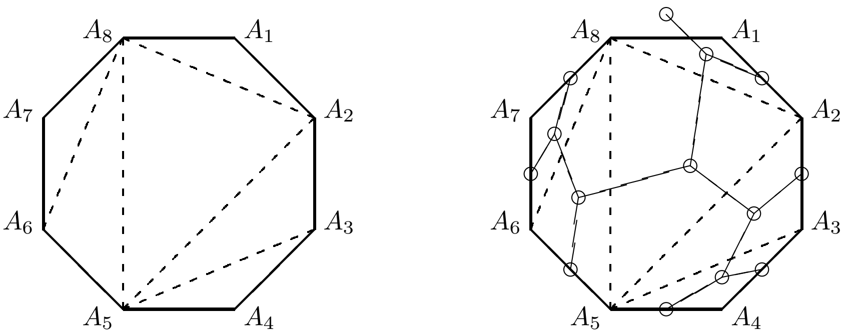


Figure 2: un octogone triangulé (à gauche) et l'arbre associé (à droite)

▶ À chaque triangulation, nous pouvons associer un arbre binaire dont la racine est le côté A_0A_1, et dont les nœuds sont les triangles et les autres côtés du polygone. La figure 2 montre l'arbre associé à la triangulation qui a été choisie.

Question 16 • Combien une triangulation compte-t-elle de diagonales et de triangles ?

▶ Soit γ une fonction qui, à un triangle t, associe un coût $\gamma(t)$. Une triangulation est *optimale* si elle minimise la somme des coûts des triangles qu'elle met en jeu.

Question 17 • Montrez que la recherche d'un arbre binaire de recherche optimal équivaut à la recherche d'une triangulation optimale ; vous préciserez le polygone et la fonction γ.

▶ Soit $(M_i)_{1 \leqslant i \leqslant n}$ une famille de matrices telle que le produit $P = M_1 \times M_2 \times \cdots \times M_n$ soit défini. Le produit de $A \in \mathcal{M}_{n,p}(\mathbb{K})$ par $B \in \mathcal{M}_{p,q}(\mathbb{K})$ a pour coût npq. Il est clair que le coût total du calcul de P dépend de l'ordre dans lequel sont faits les calculs.

Question 18 • Déterminez le coût minimal lorsque $n = 4$, $M_1 \in \mathcal{M}_{15,8}(\mathbb{K})$, $M_2 \in \mathcal{M}_{8,12}(\mathbb{K})$, $M_3 \in \mathcal{M}_{12,20}(\mathbb{K})$ et $M_4 \in \mathcal{M}_{20,10}(\mathbb{K})$.

Question 19 • Montrez que la détermination d'un ordre de calcul réalisant le coût minimal revient à la recherche d'une triangulation optimale ; vous préciserez le polygone et la fonction γ.

Question 20 • Le professeur Jean-Marcel Chaudron de Gamelle subodore que, lorsque $\gamma(t)$ est l'aire du triangle t, il existe un algorithme efficace de détermination d'une triangulation optimale. Qu'en pensez-vous ?

2 D'après une question de l'oral de Cachan

► Soit Σ un alphabet fini, contenant aux moins deux lettres. Soit $u \in \Sigma^*$, de longueur $n \geqslant 1$; pour $u \in [\![1, n]\!]$, nous noterons u_i la i-ième lettre de u. Pour $1 \leqslant i \leqslant j \leqslant n$, nous noterons $u[i..j]$ le facteur $u_i u_{i+1} \dots u_{j-1} u_j$ de u.

► Rappelons que le *miroir* $\widetilde{u}$ du mot u est défini par $\widetilde{\varepsilon} = \varepsilon$, et $\widetilde{u} = u_n u_{n-1} \dots u_1$ si $u = u_1 u_2 \dots u_n$.

► Le vecteur de PARIKH du mot u est la famille $P_u = \big(|u|_x\big)_{x \in \Sigma}$, où $|u|_x$ désigne le nombre d'occurrences de la lettre x dans u.

► Soient u et v deux mots sur Σ ; nous noterons $u \to v$ lorsque v se déduit de u par «miroir d'un facteur». Plus rigoureusement : $u \to v$ ssi il existe des mots x, y et z tel que $u = xyz$ et $v = x\widetilde{y}z$. Clairement, cette relation est réflexive (prendre $y = \varepsilon$) et symétrique.

Question 1 • La relation $\to$ est-elle transitive ?

Question 2 • Soient u et v deux mots. Montrez que $u \to v$ ssi $\widetilde{u} \to \widetilde{v}$.

Question 3 • Soient u, v et t trois mots. Montrez que $u \to v$ ssi $tu \to tv$.

Question 4 • Proposez un algorithme testant si deux mots u et v de même longueur n vérifient $u \to v$. Cet algorithme devra effectuer $\mathcal{O}(n)$ comparaisons de lettres.

► Une opération «Cons» consiste à construire une liste `t::q` à partir de sa tête et de sa queue, ou à décomposer une liste en sa tête et sa queue, avec le motif `t::q`.

Question 5 • Rédigez en Caml une fonction de signature

```
fleche : 'a list -> 'a list -> bool
```

spécifiée comme suit : `fleche u v` décide si les mots u et v (représentés de façon naturelle par les listes `u` et `v`) vérifient $u \to v$. Votre fonction devra effectuer $\mathcal{O}\big(|u| + |v|\big)$ opérations «Cons». Vous pourrez utiliser la fonction `rev`, qui construit la liste $\widetilde{\ell}$ miroir d'une liste ℓ donnée, pour un coût $\mathcal{O}\big(|\ell|\big)$ opérations «Cons». L'emploi de références ou de structures de données mutables (vecteurs) est interdit.

► Soit L un langage quelconque sur l'alphabet Σ. Notons $\Phi(L)$ l'ensemble des mots $u \in \Sigma^*$ vérifiant $u \to v$ pour au moins un mot v de L.

Question 6 • Soit L un langage rationnel. $\Phi(L)$ est-il rationnel ?

Question 7 • Donnez un exemple de langage L non rationnel tel que $\Phi(L)$ soit rationnel.

Question 8 • Donnez un exemple de langage L non rationnel tel que $\Phi(L)$ ne soit pas rationnel.

► Notons $\overset{*}{\to}$ la clôture transitive de $\to$, définie comme suit : $u \overset{*}{\to} v$ ssi $u = v$, ou il existe une suite $(w_i)_{0 \leqslant i \leqslant n}$ de mots telle que $w_0 = u$, $w_i \to w_{i+1}$ pour tout $i \in [\![0, n-1]\!]$, et $w_n = v$. Il est clair que $\overset{*}{\to}$ est une relation d'équivalence sur Σ^*.

Question 9 • Donnez une CNS pour que $u \overset{*}{\to} v$, faisant intervenir des vecteurs de PARIKH.

Question 10 • Dans cette question uniquement, nous supposons $\Sigma = \{a,b,c\}$. Combien existe-t-il de classes modulo $\overset{*}{\to}$ de mots de longueur n ?

► Soit L un langage quelconque sur l'alphabet Σ. Notons $\Phi^*(L)$ l'ensemble des mots $v \in \Sigma^*$ vérifiant $u \overset{*}{\to} v$ pour au moins un mot u de L. Ainsi $\Phi^*(L)$ est la réunion des classes modulo $\overset{*}{\to}$ des mots de L.

Question 11 • Donnez un exemple de langage L non rationnel vérifiant les deux conditions suivantes : $\Phi(L)$ n'est pas rationnel ; $\Phi^*(L)$ est rationnel.

FIN