Option Informatique en Spé MP et MP*
DS du 28 mars 2001 : le corrigé (succinct et incomplet)

Question 1 • Banalité de chez banal.

Question 2 • $a_{k+1} = \frac{2(2k+1)}{k+2} a_k$.

Question 3 • Avec le changement d'indice $k \to n-k$, il vient $2T_n = nS_n$.

Question 4 • Alors $T_{n+1} + S_{n+1} = \sum_{0 \leqslant k \leqslant n+1} (k+1) a_k a_{n+1-k} = a_0 a_{n+1} + 2 \sum_{1 \leqslant k \leqslant n+1} (2k-1) a_{k-1} a_{n-(k-1)} = a_{n+1} + 2 \sum_{0 \leqslant k \leqslant n} (2k+1) a_k a_{n-k} = S_n + 4T_n + 2S_n = (2n+3)S_n$.

Question 5 • $T_{n+1} = \frac{n+1}{2} S_{n+1}$, donc $\frac{n+3}{2} S_{n+1} = (2n+3)S_n = (2n+3)a_{n+1}$ soit $S_{n+1} = \frac{2(2n+3)}{n+3} a_{n+1}$ ou enfin $S_{n+1} = a_{n+2}$.

Question 6 • *Banalitas banalitatis, omnia banalitatum.*

Question 7 • On utilise un compteur, incrémenté par les `a`, décrémenté par les `b` :

```
let bon_mot u =
 let rec aux accu = function
  | [] -> accu = 0
  | a::q -> aux (accu + 1) q
  | b::_ when accu = 0 -> false
  | b::q -> aux (accu - 1) q
 in aux 0 u ;;
```

Question 8 • L_n est fini, donc rationnel.

Question 9 • Supposons L rationnel. Le lemme de l'étoile affirme l'existence d'un entier N tel que tout mot de L de longueur au moins N se factorise en $u = xyz$ avec $|xy| \leqslant N$, $y \neq \varepsilon$ et $xy^*z \subset L$. Prenons $u = a^N b^N$ (qui est clairement un mot de L) ; alors $x = a^j$, $y = a^k$ (avec $k > 0$) et $z = a^{N-j-k} b^N$. Mais $xz = a^{N-k} b^N \notin L$: d'où la contradiction. Donc L n'est pas rationnel.

Question 10 • Banalité : C_n est majoré par le nombre de mots de longueur $2n$ contenant exactement n lettres a, soit justement $\binom{2n}{n}$.

Question 11 • Existence : si $u = abz$, alors $z \in L$, si bien que $v = \varepsilon$ et $w = z$ conviennent. Sinon, $u = aaz$. Notons $\varphi : i \in [\![0, 2n]\!] \mapsto \big|u[1..i]\big|_a - \big|u[1..i]\big|_b$. $\varphi(0) = \varphi(2n) = 0$; et $\varphi(i+1) - \varphi(i) = \pm 1$ pour $0 \leqslant i < 2n$. Considérons le plus petit i tel que $\varphi(i) = 0$: notons $v = u[2..i-1]$ et $w = u[i+1..2n]$, étant entendu que $w = \varepsilon$ si $i = 2n$. On a $u = avbw$, et on vérifie que v et w sont dans L.

• Unicité : supposons que u possède deux décompositions distinctes : $u = avbw = av'bw'$. Alors $|v| \neq |v'|$. Supposons $|v| < |v'|$ pour fixer les idées. On met en évidence une contradiction en observant le préfixe z de v' de longueur $|v| + 1$: il vérifie $|z|_b = |z|_a + 1$.

Question 12 • Le résultat de la question précédente affirme l'existence d'une bijection entre L_n et la réunion (disjointe) des $L_k \times L_{n-1-k}$, pour $0 \leqslant k < n$. On se place au rang suivant, et on passe aux cardinaux.

Question 13 • Banalité : $C_n = S_n$ puisque les deux suites sont définies par les mêmes règles. Donc $C_n = \frac{\binom{2n}{n}}{n+1}$.

Question 14 • Il suffit d'interpréter les a comme des pas vers la droite, et les b comme des pas vers le haut.

Question 15 • $\mathrm{Cout}(C_0) = 1$, et $\mathrm{Cout}(C_n) \geqslant 2\,\mathrm{Cout}(C_{n-1})$ pour $n \geqslant 2$. Donc $\mathrm{Cout}(C_n) \geqslant 2^n$.

Question 16 • Il suffit de mémoriser les valeurs de C_k au fur et à mesure qu'on les calcule. Le calcul de C_k se ramène au calcul de C_1 à C_{k-1}, suivi de $\mathcal{O}(k)$ opérations (k multiplications et $k-1$ additions ; un argument de symétrie permettrait facilement de diviser par 2 ce coût).

```
let sum = it_list (prefix +) 0 ;;

let c n =
 let v = make_vect (n+1) 1 in
 for k = 1 to n do
  v.(k) <- sum (map (fun j -> v.(j) * v.(k-j)) (intervalle 0 (k-1))
 done; v.(n) ;;
```

Question 17 • Infantile.

Question 18 • Banalité de première espèce.

Question 19 • Le calcul de $(AB)C$ coûte $3 \times 10 \times 5 + 3 \times 5 \times 8 = 150 + 120 = 270$; celui de $A(BC)$ coûte $10 \times 5 \times 8 + 3 \times 10 \times 8 = 400 + 240 = 640$.

Question 20 • Le calcul de $(AB)C$ coûte $npq+nqr = nq(p+r)$, et celui de $A(BC)$ coûte $pqr+npr = pr(n+q)$; le rapport est donc $\dfrac{nq(p+r)}{pr(n+q)}$. Prenons $n = q = \tau$ et $p = r = 1$, on obtient le résultat demandé.

Question 21 • Banal : $K_4 = 5$.

Question 22 • Même argument qu'à la question 12.

Question 23 • Chacune des valeurs qui participent à la prise de min dans le membre de droite correspond à l'une des façons de mener le calcul. D'où l'égalité.

Question 24 • $\gamma_n = (n-1)\gamma_{n-1}$ et $\gamma_2 = 1$. Donc $\gamma_n = (n-1)!$: cette méthode n'est donc pas envisageable !

Question 25 • Non, bien entendu : il suffit de calculer les produits $\ell_i \ell_{k+1} \ell_j$.

Question 26 • On procède comme à la question 16 : on mémorise les résultats, et on procède par valeurs croissantes de $j - i$. On ne remplit que les cases utiles (la moitié du tableau, en gros). En bref : on commence par mettre la valeur 0 dans les cases diagonales $M_{i,i}$. Ensuite, pour d allant de 1 à $n-1$, on remplit les cases $M_{i,j}$ avec $j = i + d$: on procède donc par ≪bandes≫ parallèles à la diagonale.

Question 27 • Sans intérêt.

Question 28 • La fonction `min_of_list` calcule le min d'une liste (non vide) d'entiers.

```
let min_of_list l = it_list min (hd l) l ;;

let cout_minimal l =
 let n = (list_length l - 1) and v = vect_of_list l in
 let mu = make_matrix n n (-1) in
 for i = 1 to n do mu.(i-1).(i-1) <- 0 done;
 for d = 1 to n-1 do
  for i = 1 to n - d do
   let j = i + d in
   let f k = mu.(i-1).(k-1) + mu.(k).(j-1) + v.(i-1) * v.(k) * v.(j) in
   let m' = map f (intervalle i (j-1)) in
   mu.(i-1).(j-1) <- min_of_list m'
  done
 done; mu.(0).(n-1) ;;
```

Question 29 • Il suffit de noter un indice k pour lequel le min est atteint : ensuite, en partant de la case $M_{n,n}$, on remonte dans le tableau.

Question 30 • On réécrit la fonction `min_of_list`.

```
let min_of_list2 l =
 let compare x y = if (fst x) <= (fst y) then x else y
 in it_list compare (hd l) l ;;

let ordonnancement l =
 let n = (list_length l - 1) and v = vect_of_list l in
 let mu = make_matrix n n (-1,-1) in
 for i = 1 to n do mu.(i-1).(i-1) <- (0,0) done;
 for d = 1 to n - 1 do
  for i = 1 to n - d do
   let j = i + d in
   let f k = (fst mu.(i-1).(k-1) + fst mu.(k).(j-1)
     + v.(i-1) * v.(k) * v.(j) , k-1 ) in
   let m' = map f (intervalle i (j-1)) in
   mu.(i-1).(j-1) <- min_of_list2 m'
```

```
 done
done; (* il reste à faire la remontée *) ;;
```

Question 31 • Les nœuds et feuilles sont étiquetés par des intervalles discrets $[\![i,j]\!]$ avec $1 \leqslant i \leqslant j \leqslant n$; la relation d'ordre est $[\![i,j]\!] \preceq [\![\ell,k]\!]$ ssi $i \leqslant \ell \leqslant j \leqslant k$. Les feuilles sont étiquetées par des intervalles d'amplitude 1 ; la racine est étiquetée $[\![1,n]\!]$. Enfin, les fils du nœud $[\![i,j]\!]$ sont $[\![i,k]\!]$ et $[\![k+1,j]\!]$ pour la valeur de k correspondant au choix des matrices A_k et A_{k+1} que l'on multiplie.

Question 32 • Ce sont les arbres filiformes, c'est-à-dire ceux dans lesquels chaque nœud a au moins une feuille parmi ses deux fils.

Question 33 • Notons $h(t)$ le nombre minimal d'emplacements de mémoire requis pour le calcul associé à l'arbre t. On aura $h(t) = 1$ pour un arbre réduit à une feuille ; puis, pour un arbre non réduit à une feuille, notant g et d les fils gauche et droit de la racine, on peut : soit évaluer d'abord g, puis d, ce qui mène à un coût $\max\big(2 + h(g), h(d)\big)$; soit évaluer d'abord g puis d, ce qui mène à un coût $\max\big(2 + h(d), h(g)\big)$. Le plus petit de ces deux coûts nous indique le meilleur ordre de calcul. Ceci est l'algorithme d'Ershov ; on retrouve cette problématique dans le le sujet d'informatique proposé en 1997 par les E.N.S. de Lyon et Paris. Ce type d'idée a été développé, indépendamment, par des géographes, pour caractériser le bassin d'un fleuve (combinatoire de Horton et Strahler).

Références bibliographiques

▶ Les $(C_n)_{n\in\mathbb{N}}$ sont les nombres de Catalan. Le combinatoriste Richard Stanley donne 66 interprétations différentes de ces nombres, dans un article disponible à l'url

`http://www-math.mit.edu/~rstan/ec/catalan.ps.gz`

Munissez-vous de `gunzip` et `ghostview` pour en profiter pleinement.

FIN