Option Informatique en Spé MP et MP*
DS du mercredi 3 avril 2002 : le corrigé

Arbres binaires de recherche optimaux

Question 1 • Le filtrage doit envisager trois cas : liste réduite à un élément (le résultat est cet élément) ; liste `t::q` de longueur au moins égale à 2 (on prend le plus petit de `t` et du min de `q`) ; liste vide (on lève une exception).

```
let rec min_of_list = function
 | [t] -> t
 | t::q -> min t (min_of_list q)
 | [] -> failwith "liste vide" ;;
```

Nous pouvons facilement donner une formulation récursive terminale, en notant que le plus petit élément d'une liste de longueur au moins égale à 2 est le plus petit élément de la liste obtenue en remplaçant les deux premiers éléments par leur min. Avantages supplémentaires de cette présentation : l'exhaustivité du filtrage est évidente, tout comme la terminaison.

```
let rec min_of_list = function
 | [t] -> t
 | t::u::q -> min_of_list ((min t u)::q)
 | [] -> failwith "liste vide" ;;
```

Question 2 • `i` et `j` sont de type `int` puisque ce sont les arguments de `intervalle`. La présence de `f.(k)` fait que `f` est un vecteur ; pour déterminer son type de base, observons que `it_list` est appliqué dans l'ordre à `prefix +` (de type `int -> int -> int`), à `0` (de type `int`, ce qui est heureux) et à `q`, qui doit donc être de type `int list` ; le résultat est de type `int`. L'évaluation de `map (fun k -> f.(k)) (intervalle i j)` construit la liste des images par f des éléments de l'intervalle $[\![i, j]\!]$; ces images devant être de type `int`, il en résulta que `f` est de type `int vect`. Conclusion : `somme` est de type `int vect -> int -> int -> int`, et calcule la somme des éléments de f d'indice compris entre i et j inclus.

Question 3 • Montrons que tout a.b.r. est équivalent à un peigne à gauche et à un peigne à droite. Raisonnons par récurrence sur le nombre n de nœuds : l'affirmation est évidente pour $n = 1$; supposons-la acquise jusqu'au rang n inclus, et considérons un arbre binaire $t = (g, r, d)$ à $n + 1$ nœuds. Alors g (resp. d) est équivalent à un peigne gauche g' (resp. un peigne droit d'), donc t est équivalent à l'arbre $t' = (g', r, d')$. Par une suite de rotations gauches (resp. droites) autour de la racine, on peut transformer t' en un peigne gauche γ (resp. un peigne droit δ).

Question 4 • Donnons la preuve pour une rotation droite. Soient $t = \big((g_g, r_g, d_g), r, d\big)$ et $t' = \big(g_g, r_g, (d_g, r, d)\big)$ son image par la rotation droite autour de la racine. g_g, d_g et d ne sont pas modifiés, donc ce sont toujours des a.b.r. Soient n un nœud de g_g, n' un nœud de d_g et n'' un nœud de d ; comme t est un a.b.r., nous avons $n < r_g < n' < r < n''$. Ces inégalités montrent que t' est un a.b.r.

Question 5 • Si les f_i ont tous la même valeur, un arbre est optimal ssi tous ses niveaux sont remplis, à l'exception éventuelle du dernier.

Question 6 • Prenons $f_1 = 2^{-n}$ et $f_i = 2^{i-1-n}$ pour $i \geqslant 2$; le nœud numéro n doit être la racine, puis, par récurrence, l'arbre est filiforme.

Question 7 • Prenons $n = 3$, $f_1 = 3/7$, $f_2 = f_3 = 2/7$. L'algorithme glouton mène à un arbre de poids $13/7$, dont la racine est le nœud numéro 1. Or, en prenant le nœud numéro 2 comme racine, nous obtenons un arbre de poids $12/7$.

Question 8 • $\mu(i, i) = f_i$ car l'accès réussi à la racine coûte exactement une comparaison.

Question 9 • Si un sous-arbre s de t n'est pas optimal, on peut au moyen de rotations transformer s en un arbre s' équivalent vérifiant $\pi(s') < \pi(s)$; on obtient alors un arbre t' équivalent à t, de poids $\pi'(t') < \pi(t)$, contredisant le caractère optimal de t.

Question 10 • Considérons un arbre binaire optimal dont les étiquettes sont celles d'indice appartenant à $[\![i, j]\!]$. Soit $k \in [\![1, n]\!]$ tel que la racine soit étiquetée (Z_k, f_k). Le coût de cet arbre est la somme des trois quantités suivantes :

- le coût du sous-arbre gauche, soit $\mu(i, k-1) + \sum\limits_{i \leqslant \ell < k} f_\ell$ puisque ce sous-arbre est optimal ;

- le coût du sous-arbre droit, soit $\mu(k+1,j) + \sum_{k<\ell\leqslant j} f_\ell$ puisque ce sous-arbre est optimal ;

- le coût associé à la racine, soit f_k.

En rassemblant les fréquences, on obtient $\mu(i,k-1) + \mu(k+1,j) + \sum_{i\leqslant k\leqslant j} f_k$. L'arbre optimal est donc obtenu en minimisant la quantité $\mu(i,k-1) + \mu(k+1,j)$.

Question 11 • Il suffit de composer la fonction `min_of_list` (construite à la question 1) avec la fonction `list_of_vect` de la bibliothèque Caml :

```
let min_of_vect v = min_of_list (list_of_vect v);;
```

Question 12 • Notons $\gamma(i,j)$ le coût du calcul de $\mu(i,j)$ avec la méthode de Charles- Édouard, exprimé en nombre d'appels de la fonction μ. La formule montre que $\gamma(i,j) \geqslant \gamma(i,j-1) + \gamma(i-1,j)$; comme $\gamma(i,i) = 1$, nous pouvons affirmer que $\gamma(i,j) \geqslant 2^{j-i}$. Le coût de cette méthode est donc exponentiel.

Question 13 • Définissons une matrice auxiliaire F par la formule $F_{i,j} = \sum_{i\leqslant k\leqslant j} f_k$ pour $1 \leqslant i \leqslant j \leqslant n$; le triangle inférieur ne nous intéresse pas. Cette matrice peut être initialisée en n'effectuant qu'une opération par coefficient, avec la formule $F_{i,j+1} = F_{i,j} + f_j$; le coût est donc de $\frac{n(n-1)}{2}$. On peut ensuite remplir les diagonales successives de M : d prenant successivement les valeurs 1 à $n-1$, on peut calculer chaque $M_{i,i+d}$ pour $1 \leqslant i \leqslant n-d$ en effectuant $d+1$ comparaisons et $d+2$ additions. Le coût total est donc :

$$\begin{aligned}
\sum_{d=1}^{n-1}\sum_{i=1}^{n-d}(2d+3) &= \sum_{d=1}^{n-1}(2d+3)(n-d) = \sum_{d=1}^{n-1}(3n+(2n-3)d-2d^2) \\
&= 3n(n-1) + \frac{(2n-3)(n-1)n}{2} - \frac{(n-1)n(2n-1)}{3} \\
&= \frac{n(n-1)}{6} \times \big(18 + 3(2n-3) - 2(2n-1)\big) = \frac{n(n-1)(2n+11)}{6}
\end{aligned}$$

Question 14 • La rédaction est peu intéressante. Pour éviter des jongleries liées au changement d'indextion, nous utiliserons des matrices carrées d'ordre $n+1$.

```
let construire_matrice f =
 let n = vect_length f in
 let F = make_matrix (n+1) (n+1) 0 in
 for i=1 to n do
  F.(i).(i) <- f.(i-1);
  for j=i+1 to n do
   F.(i).(j) <- F.(i).(j-1) + f.(j-1)
  done
 done;
 let M = make_matrix (n+1) (n+1) 0 in
 let mini i j =
  let phi k = (if k>1 then M.(i).(k-1) else 0)
    + (if k<n then M.(k+1).(j) else 0)
  in map phi (intervalle i j) in
 for i=1 to n do M.(i).(i) <- 1 done;
 for d=1 to n-1 do
  for i=1 to n-d do
   let j=i+d in
    M.(i).(j) <- F.(i).(j) + min_of_list (mini i j)
  done
 done; M
;;
```

Question 15 • Il suffit de construire une matrice R donnant, pour $1 \leqslant j \leqslant n$, l'indice k pour lequel le minimum de $\mu(i,k-1) + \mu(k+1,j)$ est atteint. $R_{1,n}$ est la racine de l'a.b.r. optimal, puis on remonte dans la matrice R pour obtenir les racines des sous-arbres gauche et droit et ainsi de suite.

Question 16 • Une triangulation d'un polygone à n côtés compte $n-3$ diagonales et $n-2$ triangles. Prouvons ceci par récurrence. Le résultat est clair pour $n = 3$. Supposons-le acquis jusqu'au rang $n \geqslant 3$ inclus, et considérons un polygone $\mathcal{P}$ à $n+1$ côtés. Choisissons arbitrairement une diagonale $d = A_1A_k$, avec $3 \leqslant k \leqslant n$; elle découpe $\mathcal{P}$ en deux polygones $\mathcal{P}_d$ et $\mathcal{P}_g$. $\mathcal{P}_d$ possède k côtés (les $k-1$ premiers côtés de $\mathcal{P}$, plus d) donc une triangulation de $\mathcal{P}_d$ possède $k-3$ diagonales et $k-2$ triangles. $\mathcal{P}_g$ possède $n-k+3$ côtés (les $n-k+2$ derniers côtés de $\mathcal{P}$, plus d) donc une triangulation de $\mathcal{P}_g$ possède $n-k+2$ diagonales et $n-k+1$ triangles. Dressons le bilan : une triangulation de $\mathcal{P}$ comporte alors $(k-3)+(n-k-2)+1 = n-2 = (n+1)-3$ diagonales et $(k-2)+(n-k+1) = n-1 = (n+1)-2$ triangles. Ceci établit le résultat au rang $n+1$.

Question 17 • Il est clair que le mécanisme décrit dans le texte met en bijection les arbres binaires de recherche possédant $2n-2$ nœuds et les triangulations d'un polygone à n sommets. $\gamma(t)$ est la somme des fréquences du nœud associé à t, et des nœuds placés sous t dans l'arbre.

Question 18 • Le coût minimal est 4560, il est obtenu avec l'ordonnancement $M_1 \times (M_2 \times (M3 \times M4)))$.

Question 19 • Soit n le nombre de matrices. Nous noterons ℓ_k (resp. c_k) le nombre de lignes (resp. de colonnes) de la matrice M_k. Le sommet A_1 est étiqueté ℓ_1 ; pour $k \in [\![2,n]\!]$, le sommet A_k est étiqueté $c_{k-1} = \ell_k$; le sommet A_{n+1} est étiqueté c_n. $\gamma(t)$ est le produit des étiquettes des sommets de t. Étiquetons les côtés du polygone et les diagonales de la triangulation : le côté A_kA_{k+1} est étiquetée M_k, pour $1 \leqslant k \leqslant n$; le côté $A_{n+1}A_1$ n'est pas étiqueté. On peut ensuite étiqueter de proche chaque diagonale par un produit matriciel, parenthésé pour indiquer l'ordre de calcul. Si nous reprenons l'octogone de l'énoncé et sa triangulation, nous aurons l'étiquetage suivant des côtés et sommets :

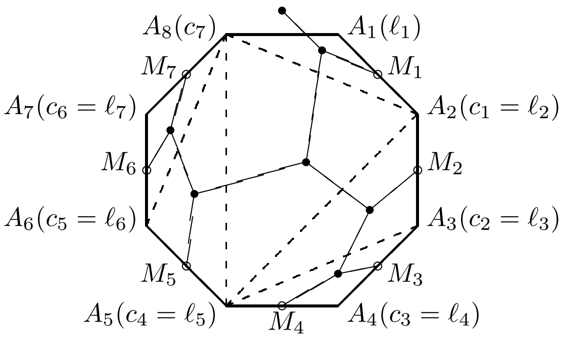

Figure 1: octogone et produit matriciel

La diagonale A_3A_5 est étiquetée $M_3 \times M4$; la diagonale A_2A_5 est étiquetée $M_2 \times (M_3 \times M4)$; la diagonale A_6A_8 est étiquetée $M_6 \times M7$; la diagonale A_5A_8 est étiquetée $M_5 \times (M_6 \times M7)$ et ainsi de suite. Au final, le côté A_1A_8 sera étiqueté par une expression parenthésée représentant un calcul ordonnancement du calcul du produit des huit matrices menant à un coût minimal.

Question 20 • Le coût total est l'aire du polygone, laquelle est indépendants de la triangulation choisie. On peut donc prendre la triangulation formée des $n-3$ diagonales A_1A_k, où $3 \leqslant k \leqslant n-2$. Bravo pour votre flair, professeur Chaudron de Gamelle !

D'après une question de l'oral de Cachan

Question 1 • La relation $\to$ n'est pas transitive : prenons $\Sigma = \{a,b\}$, $u = ababb$, $v = abbab$ et $w = babba$. Clairement, $u = ab(ab)b \to vab(\widetilde{ab})b$ et $v = abbab \to w = \widetilde{v}$; pourtant, on n'a pas $u \to w$: aucune des trois transformations amenant un b en tête ne convient.

Question 2 • Soient x, y et z tel que $u = xyz$ et $v = x\widetilde{y}z$. Notant $x' = \widetilde{x}$, $y' = \widetilde{y}$ et $z' = \widetilde{z}$ nous aurons $\widetilde{u} = z'y'x'$ et $\widetilde{v} = z'yx' = z'\widetilde{y'}x'$ donc $\widetilde{u} \to \widetilde{v}$.

Question 3 • Le sens direct est clair, et il suffit d'établir la réciproque lorsque $t \in \Sigma$. Soient donc x, y et z tels que $tu = xyz$ et $tv = x\widetilde{y}z$. Si $x \neq \varepsilon$, alors $x = tx'$ avec $|x'| = |x|-1$, puis $tu = tx'yz$ et $tv = tx'\widetilde{y}z$ donc $u = x'yz$ et $v = x'\widetilde{y}z$, si bien que $u \to v$. Si $x = \varepsilon$, alors $tu = yz$ et $tv = \widetilde{y}z$. Ainsi, t est la première lettre de y et de $\widetilde{y}$; c'est donc aussi la dernière lettre de y. Si $|y| = 1$, alors $y = t$, puis $u = v = z$ donc $u \to v$. Sinon, $y = ty't$ puis $tu = ty'tz$ et $tv = \widetilde{ty't}z = t\widetilde{y'}tz$; nous en déduisons $u = y'tz$ et $v = \widetilde{y'}tz$ ce qui montre que $u \to v$.

Question 4 • En appliquant répétitivement le résultat précédent, nous pouvons nous ramener au cas où u et v n'ont ni préfixe commun, ni suffixe commun. Dans ces conditions, $u \to v$ ssi $u = \widetilde{v}$. Ceci peut être effectué pour un coût $\mathcal{O}(n)$: notant p (resp. q) la longueur du plus long préfixe (resp. suffixe) commun à u et v, on effectue $p+q+2$ comparaisons de caractères pour les éliminer ; puis $n-p-q$ comparaisons pour décider si $u[p+1..n-q]$ et $v[p+1..n-q]$ sont miroirs l'un de l'autre.

Question 5 • Comme nous travaillons sur des listes, il nous faut les retourner pour éliminer leur plus long suffixe commun. Le coût total sera donc de $2n$ opérations ≪Cons≫ au lieu de n.

```
let rec efface_prefixe_commun = function
 | (t::x,u::y) when t=u -> efface_prefixe_commun (x,y)
 | c -> c ;;

let equivalents u v =
 let (u1,v1) = efface_prefixe_commun (u,v) in
 let (u2,v2) = (rev u1,rev v1) in
 let (u3,v3) = efface_prefixe_commun (u2,v2) in
  rev(u3) = v3 ;;
```

Question 6 • La réponse est positive. Soit $\mathcal{A} = (Q, \delta, I, F)$ un automate fini (pas forcément déterministe) reconnaissant L. Soient q et q' deux états de $\mathcal{A}$; nous noterons $L_{q,q'}$ l'ensemble des mots u tels que $q \xrightarrow{u} q'$: il est clair que ce langage est rationnel, et son miroir $\widetilde{L_{q,q'}}$ l'est donc également.

• Nous allons montrer que $\Phi(L)$ est la réunion des langages $L_{i,q} \cdot \widetilde{L_{q,'q}} \cdot L_{q',f}$ où $i \in I$, $f \in F$ et $q, q' \in Q$. Ces langages sont en nombre fini, plus précisément n^4 ; chacun d'eux est rationnel en tant que produit de rationnels). Donc $\Phi(L)$ sera rationnel.

• Soit $v \in \Phi(L)$: il existe $x, y, z \in \Sigma^*$ tels que $xyz \in L$ et $v = x\widetilde{y}z$. Observons un calcul réussi de $\mathcal{A}$ d'étiquette xyz ; il est de la forme $i \xrightarrow{x} q \xrightarrow{y} q' \xrightarrow{z} f$ avec $i \in I$ et $f \in F$. Alors $v = x\widetilde{y}z$ appartient au produit $L_{i,q} \cdot \widetilde{L_{q,'q}} \cdot L_{q',f}$.

• Réciproquement, considérons un mot v appartenant à la réunion décrite plus haut. Il existe donc $i \in I$, $f \in F$ et $q, q' \in Q$ tels que $v \in L_{i,q} \cdot \widetilde{L_{q,'q}} \cdot L_{q',f}$. Donc $v = xyz$ avec $x \in L_{i,q}$, $y \in \widetilde{L_{q,'q}}$ et $z \in L_{q',f}$. Notons $u = x\widetilde{y}z$: ce mot est l'étiquette d'un calcul réussi de $\mathcal{A}$, à savoir $i \xrightarrow{x} q \xrightarrow{\widetilde{y}} q' \xrightarrow{z} f$. Donc u appartient à L, et finalement v appartient à $\Phi(L)$.

Question 7 • Prenons comme alphabet $\Sigma = \{a,b\}$. Alors $L = \{a^n b a^n \mid n \in \mathbb{N}\}$ répond à la question. Il est clair qu'il n'est pas rationnel ; et $\Phi(L) = \{a^p b a^q \mid p+q \text{ est pair}\}$ est rationnel.

Question 8 • Il suffit de choisir un langage non rationnel L tel que $\Phi(L) = L$. Ainsi, avec $\Sigma = \{a,b\}$ le langage $L = \{u \in \Sigma^* \mid |u|_a = |u|_b\}$ convient.

Question 9 • Il est clair que, si $u \to v$, alors $|u|_x = |v|_x$ pour toute lettre $x \in \Sigma$; il en est donc de même lorsque $u \xrightarrow{*} v$. Montrons que cette condition est suffisante : il suffit de prouver que, si $\Sigma = \{x_1, \dots, x_p\}$, alors le mot $u = u_1 \dots u_n$ est équivalent au mot ${x_1}^{|u|_{x_1}} \dots {x_p}^{|u|_{x_p}}$. Prouvons-le par récurrence sur n ; le résultat est clair si $n = 0$ ou $n = 1$. Supposons-le acquis au rang n, et considérons un mot u de longueur $n+1$: $u = x_j v$ avec $x_j \in \Sigma$ et $|v| = n$. L'hypothèse de récurrence s'applique à v : $v \xrightarrow{*} {x_1}^{|v|_{x_1}} \dots {x_p}^{|v|_{x_p}}$. En utilisant les relations $|u|_{x_i} = |v|_{x_i} + \delta_{i,j}$ et $xy^p \to y^p x$, nous aurons clairement :

$$
\begin{aligned}
u &= x_j {x_1}^{|v|_{x_1}} \dots {x_p}^{|v|_{x_p}} \to {x_1}^{|v|_{x_1}} x_j {x_2}^{|v|_{x_2}} \dots {x_p}^{|v|_{x_p}} \xrightarrow{*} {x_1}^{|v|_{x_1}} x_j {x_{j-1}}^{|v|_{x_{j-1}}} {x_j}^{|v|_{x_j}} \dots {x_p}^{|v|_{x_p}} \\
&\to {x_1}^{|v|_{x_1}} {x_{j-1}}^{|v|_{x_{j-1}}} x_j {x_j}^{|v|_{x_j}} \dots {x_p}^{|v|_{x_p}} = {x_1}^{|v|_{x_1}} {x_{j-1}}^{|v|_{x_{j-1}}} {x_j}^{1+|v|_{x_j}} \dots {x_p}^{|v|_{x_p}} \\
&= {x_1}^{|u|_{x_1}} {x_{j-1}}^{|u|_{x_{j-1}}} {x_j}^{|u|_{x_j}} \dots {x_p}^{|u|_{x_p}}
\end{aligned}
$$

Question 10 • D'après l'analyse effectuée à la question précédente, il existe autant de classes de mots de longueur n modulo $\xrightarrow{*}$ que de triplets (i, j, k) de naturels vérifiant $i + j + k = n$; ce nombre est :

$$\sum_{0 \leqslant i \leqslant n} \sum_{0 \leqslant j \leqslant n-i} 1 = \sum_{0 \leqslant i \leqslant n} (n-i+1) = \sum_{1 \leqslant i \leqslant n+1} i = \frac{(n+1)(n+2)}{2}$$

Question 11 • Montrons que le langage $L = \{a^n b a^n b a^n b a^n \mid n \in \mathbb{N}\}$ répond à la question.

• L n'est pas rationnel : s'il l'était, le lemme de l'étoile assurerait l'existence d'une constante N telle que tout mot u de L de longueur au moins égale à N possède une factorisation $u = xyz$ vérifiant $|xy| \leqslant N$, $y \neq \varepsilon$ et $xy^k z \in L$ pour tout $k \in \mathbb{N}$. Choisissons alors $u = a^N b a^N b a^N b a^N$; xy est préfixe de a^N, donc $x = a^i$ et $y = a^j$ avec $j \neq 0$, donc $z = a^{N-i-j} b a^N b a^N b a^N$. Mais alors le mot $xz = a^{N-j} b a^N b a^N b a^N$ n'appartient pas à L car $N - j < N$.

• Montrons que $\Phi(L)$ n'est pas rationnel. L'idée est que tout mot de ce langage est de la forme $a^\alpha ba^\beta ba^\gamma ba^\delta$, deux des exposants α, β, γ et δ devant être égaux. Supposons $\Phi(L)$ rationnel, et notons N la constante fournie par le lemme de l'étoile. Soit $v = a^N ba^N ba^N ba^N$, qui appartient à L. La factorisation donnée par le lemme de l'étoile est de la forme xyz avec $x = a^i$, $y = a^j$ et $z = a^{N-i-j} ba^N ba^N ba^N$, avec $j \neq 0$. Notons $v = xz = a^{N-j} ba^N ba^N ba^N$ et prouvons que ce mot n'appartient pas à $\Phi(L)$. C'est clair si j n'est pas multiple de 4, puisque Φ conserve le nombre d'occurrences de a dans un mot. Soient donc $p = N - j/4$, $w = a^p ba^p ba^p ba^p$ et w' un mot tel que $w \to w'$: il existe des mots r, s et t tels que $w = rst$ et $w' = r\widetilde{s}t$. Si $|s|_b = 0$, ou si $s = a^k ba^k$ ou $s = a^k bs' ba^k$, alors $w' = w$ donc $w' \neq v$. Il reste six cas à envisager selon les b qui apparaissent dans s. Examinons un de ces cas : si s contient le premier b de $'$, alors $r = a^{p-j}$, $s = a^j ba^k$ et $t = a^{p-k} ba^p ba^p$ avec $j \neq k$; par suite $w' = a^{p-j+k} na^{p-k+j} ba^p ba^p$ donc $w' \neq v$. On se convaincra aisément que la situation est la même dans les cinq autres cas : w' est de la forme $a^p ba^q ba^r ba^s$

• Il résulte de la question 9 que $\Phi^*(L)$ est l'ensemble des mots $u \in \Sigma^*$ tels que $|u|_b = 3$ et $|u|_a$ est multiple de 4 ; il est clair que ce langage est rationnel.

Références bibliographiques

▶ Pour le premier problème : le livre *Computer Algorithms*, de Sara Baase et Allen Van Gelder (édité par Addison-Wesley) ; ainsi que l'*Introduction to Algorithms*, de Thomas H. Cormen, Charles E. Leiserson et Ronald L. Rivest (édité par The MIT Press).

▶ D. Hirschberg, L. Larmore et M. Molodovitch se sont intéressés au problème suivant : soit t' un sous-arbre d'un a.b.r. optimal t ; notant d la profondeur dans t de la racine de t', encadrer le rapport $\pi(t')/\pi(t)$ par des fonctions de d. Leur article *Subtree weight ratios for optimal binary search trees* est disponible sur le Web.

▶ Pour le deuxième problème : l'énoncé original (huit lignes) provient de la *Revue de Mathématiques Spéciales*, année 2001-2002, numéro 2 ; il demandait simplement au candidat de décrire un algorithme décidant si $u \xrightarrow{v}$, puis de dire si Φ conservait le caractère rationnel d'un langage.

▶ Une permutation de l'intervalle discret $[\![1, n]\!]$ peut être vue comme un mot (très particulier) sur l'alphabet constitué par cet intervalle discret. Sous-jacente au texte apparaît la notion de *permutation-miroir* : il existe deux indices i et j tels que $s(k) = k$ si $k < i$ ou $k > j$, et $s(k) = i + j - k$ si $i \leqslant j \leqslant k$. Toute permutation se décompose en produit de permutations-miroirs : mathématiquement, ceci signifie que les $n - 1$ transposition $\tau_{i,i+1}$, où $i \in [\![1, n-1]\!]$, engendrent le groupe des permutations $[\![1, n]\!]$. Informatiquement, c'est le tri à bulle.

▶ Définissons la distance entre deux permutations de $[\![1, n]\!]$ comme le plus petit nombre de permutations-miroirs permettant de passer de l'une à l'autre. Le problème du calcul de cette distance intervient dans l'analyse du génôme ; Caprara a montré en 1997 que ce problème est NP-complet.

▶ Définissons maintenant la notion de *permutation signée* : à chaque élément de $[\![1, n]\!]$ est associé un signe $+$ ou $-$; lorsque l'on applique une permutation-miroir, le signe des éléments «miroités» est inversé. Dans ce cadre, le calcul de la distance a un coût polynomial. Anne Bergeron a écrit sur ce thème un article très accessible, intitulé *A very elementary presentation of the Hannenhalli-Pevzner theory* (disponible sur le Web).

FIN