Option Informatique en Spé MP et MP*
Structure secondaire de l'ARN de transfert : le corrigé

1 Dénombrement des modèles de structures secondaires

Question 1 • Soit $j \in [\![i, s(i)]\!]$, montrons que $s(j) \in [\![i, s(i)]\!]$. C'est clair si $j = i$, et aussi si $j = s(i)$ puisque $s\big(s(i)\big) = i$. Si $i < j < s(i)$, alors $i < s(j) < s(i)$ sinon les liaisons $(i, s(i))$ et $(j, s(j))$ seraient enchevêtrées.

Question 2 • Un modèle de longueur 4 comporte au plus une liaison secondaire ; il en existe donc quatre : $(1, 2, 3, 4)$; $(3, 2, 1, 4)$; $(4, 2, 3, 1)$ et $(1, 4, 3, 2)$.

• Il existe huit modèles de longueur 5 :

- le modèle banal $(1, 2, 3, 4, 5)$;
- cinq modèles comportant une liaison secondaire : $(3, 2, 1, 4, 5)$; $(4, 2, 3, 1, 5)$; $(5, 2, 3, 4, 1)$; $(1, 4, 3, 2, 5)$ et $(1, 2, 5, 4, 3)$;
- un modèle comportant deux liaisons secondaires : $(5, 4, 3, 2, 1)$.

Question 3 • Le sens direct est clair : l'intervalle $[\![i, s(i)]\!]$ est stable, donc l'intervalle $[\![i+1, s(i)-1]\!]$ l'est aussi. Soit $j \in [\![s(i)+1, n]\!]$: on ne peut avoir $s(j) \in [\![i, s(i)]\!]$ sinon il y aurait enchevêtrement ; on ne peut non plus avoir $s(j) \in [\![1, i-1]\!]$ puisque chaque élément de ce dernier intervalle est invariant par s. Donc $[\![s(i)+1, n]\!]$ est lui aussi stable par s ; il est clair que les propriétés **(1)** et **(2)** sont vérifiées par les applications induites sur les intervalles $[\![i+1, s(i)-1]\!]$ et $[\![s(i)+1, n]\!]$.

• Réciproquement, supposons que s induit un modèle sur chacun des intervalles $[\![i+1, s(i)-1]\!]$ et $[\![s(i)+1, n]\!]$ et que $s(i) > i+1$. La propriété **(1)** est clairement satisfaite, grâce à cette dernière condition ; et on constate qu'il n'y a pas d'enchevêtrement, si bien que la condition **(2)** est elle aussi vérifiée.

Question 4 • N'oublions pas que l'indexation des vecteurs commence à 0 en Caml, alors que la numérotation de l'énoncé commence à 1. La fonction `modèle` détermine récursivement (en appliquant le résultat de la question précédente) si l'application décrite par le vecteur s induit un modèle sur l'intervalle discret $[\![p, q]\!]$ considéré.

```
let rec modèle s p q =
 let sp = s.(p-1) in
  (sp = p & (p = q or modèle s (p+1) q)) or
  (sp = q & q > p+1 & modèle s (p+1) (q-1)) or
  (sp > p+1 & sp < q & modèle s (p+1) (sp-1)
   & modèle s (sp+1) q);;

let est_modèle s = modèle s 1 (vect_length s);;
```

Question 5 • $S(1) = S(2) = 1$ (il n'y a que le modèle banal) ; $S(3) = 2$ car, en plus du modèle banal, il y a le modèle $(3, 2, 1)$; enfin, $S(4) = 4$ et $S(5) = 8$ d'après la question 2.

Question 6 • Classifions les $S(n+2)$ modèles de longueur $n+2$ selon la valeur de $s(n+2)$:

- un modèle vérifiant $s(n+2) = 1$ vérifie aussi $s(1) = n+2$, il est donc parfaitement défini par le modèle qu'il induit sur l'intervalle $[\![2, n+1]\!]$; par suite, il y en a $S(n)$;
- soit $j \in [\![2, n]\!]$ et $i = j-1$: un modèle vérifiant $s(n+2) = j$ vérifie aussi $s(j) = n+2$; un élément de l'intervalle $[\![1, j-1]\!]$ ne peut avoir son image dans $[\![j, n+2]\!]$; le modèle est donc parfaitement défini par les modèles qu'il induit sur les intervalles $[\![1, j-1]\!]$ (de longueur $j-1 = i$) et $[\![j+1, n+1]\!]$ (de longueur $n+1-j = n-i$) ; il existe ainsi $S(i)S(n-i)$ modèles vérifiant $s(n+2) = j$;
- aucun modèle ne vérifie $s(n+2) = n+1$;
- un modèle vérifiant $s(n+2) = n+2$ est parfaitement défini par le modèle qu'il induit sur l'intervalle $[\![1, n+1]\!]$; il y en a donc $S(n+1)$.

Au total, on aura : $S(n+2) = S(n+1) + S(n) + \sum_{1 \leqslant i < n} S(i)S(n-i)$.

Question 7 • La solution raisonnable consiste à calculer de proche en proche les valeurs de $S(k)$ pour $k \in [\![1, n]\!]$, en les stockant dans un vecteur au fur et à mesure. Noter que la case `v.(0)` n'est pas utilisée.

```
let S(n) =
 let v = make_vect (n+1) 1 in
 for k = 3 to n do
  v.(k) <- v.(k-1) + v.(k-2);
  for i = 1 to k-3 do v.(k) <- v.(k) + v.(i) * v.(k-2-i) done;
 done;
 v.(n);;
```

Voici un tableau présentant les premières valeurs de la suite S :

n	1	2	3	4	5	6	7	8	9	10	11	12
$S(n)$	1	1	2	4	8	17	37	82	185	423	978	2283

Le coût $c(n)$ de l'évaluation de $S(n)$ (exprimé en nombre de lectures et écritures d'éléments du vecteur `v`) est défini, pour $n \geqslant 3$, par :

$$\begin{aligned} c(n) &= 1 + \sum_{3\leqslant k\leqslant n} \left(3 + \sum_{1\leqslant i\leqslant k-3} 4\right) = 1 + 3(n-2) + 4 \sum_{3\leqslant k\leqslant n} (k-3) \\ &= 3n - 5 + 4 \sum_{1\leqslant k\leqslant n-3} k = 3n - 5 + 2(n-3)(n-2) = 2n^2 - 7n + 7 \end{aligned}$$

Et, bien entendu : $c(1) = c(2) = 1$.

Question 8 • L'option `remember` permet d'obtenir un coût polynomial, comme à la question précédente :

```
S := proc(n) option remember; local i;
 S(n-1)+S(n-2)+add(S(i)*S(n-2-i),i=1..n-3) end;
S(1) := 1;S(2) := 1;
```

En l'absence de cette option, le coût $\gamma(n)$ de l'évaluation de $S(n)$ vérifie

$$\gamma(n) = 1 + \gamma(n-1) + \gamma(n-2) + 2 \sum_{1\leqslant i\leqslant n-3} \gamma(i)$$

Par différence, il vient $\gamma(n) = 2\gamma(n-1) + \gamma(n-3)$, qui est équivalent quand n tend vers l'infini à r^n, où r est la racine réelle du polynôme $X^3 - 2X^2 - 1$; $r > 2$ est clair. Numériquement, $r \approx 2.205569431$.

Question 9 • On a $S(n+2) \geqslant S(n+1) + S(n)$; notant $(F_n)_{n\in\mathbb{N}}$ la suite de Fibonacci, on constate que $S(1) = F_1$ et $S(2) = F_2$. On en déduit $S(n) \geqslant F_n$. On peut même faire un peu mieux : comme $S(n) \geqslant 1$ pour tout n, on a certainement $S(n+2) \geqslant \sum_{1\leqslant k\leqslant n+1} S(k)$. Une récurrence immédiate nous donne $S(n) \geqslant 2^{n-1}$ pour $n \geqslant 1$.

• On peut aussi prouver que $S(n)$ est majoré par le nombre de Catalan C_n. Rappelons que la famille des nombres de Catalan est définie par $C_0 = 1$ et la relation de récurrence $C_{n+1} = \sum_{0\leqslant k\leqslant n} C_k C_{n-k}$ pour tout $n \in \mathbb{N}$, et que la série entière $\sum_{n\in\mathbb{N}} C_n z^n$ a pour rayon de convergence $1/4$.

• On a clairement $S(1) = 1 = C_1$ et $S(2) = 1 < C_2 = 2$. Supposons la majoration $S(k) \leqslant C_k$ acquise pour tout $k \in [\![1, n+1]\!]$; alors :

$$\begin{aligned} S(n+2) &= S(n+1) + S(n) + \sum_{1\leqslant k\leqslant n-1} S(k)S(n-k) \\ &\leqslant C_{n+1} + C_n + \sum_{1\leqslant k\leqslant n-1} C_k C_{n-k} = C_{n+1} + \sum_{0\leqslant k\leqslant n-1} C_k C_{n-k} \\ &< C_{n+1} + \sum_{0\leqslant k\leqslant n} C_k C_{n-k} = C_{n+1} + C_{n+1} = C_0 C_{n+1} + C_{n+1} C_0 \\ &\leqslant \sum_{0\leqslant k\leqslant n+1} C_k C_{n+1-k} = C_{n+2} \end{aligned}$$

Question 10 • Procédons par récurrence sur $n \geqslant 1$; une expérimentation rapide avec Maple montre que la majoration est acquise pour tout $n \in [\![1, 100]\!]$. Supposons la majoration acquise jusqu'au rang $n_0 + 1$ inclus (où n_0 sera fixé ultérieurement). Alors, pour $n \geqslant n_0$:

$$\begin{aligned} S(n+2) &= S(n+1) + S(n) + \sum_{1 \leqslant k \leqslant n-1} S(k)S(n-k) \\ &\leqslant \frac{3^{n+1}}{(n+1)(n+2)} + \frac{3^n}{n(n+1)} \sum_{1 \leqslant k \leqslant n-1} \frac{3^k}{k(k+1)} \times \frac{3^{n-k}}{(n-k)(n-k+1)} \\ &= \frac{3^{n+2}}{(n+2)(n+3)} F(n) \end{aligned}$$

avec :

$$\begin{aligned} F(n) &= \frac{n+3}{3(n+1)} + \frac{(n+2)(n+3)}{9n(n+1)} + \frac{1}{9} \sum_{1 \leqslant k \leqslant n-1} r(n,k) \\ r(n,k) &= \frac{1}{k(k+1)(n-k)(n-k+1)} \end{aligned}$$

Une décomposition en éléments simples nous donne :

$$r(n,k) = \frac{1}{n(n+1)} \Big(\frac{1}{k} + \frac{1}{n-k} \Big) - \frac{1}{(n+1)(n+2)} \Big(\frac{1}{k+1} + \frac{1}{n-k+1} \Big)$$

Donc, notant classiquement $H_n = \displaystyle\sum_{1 \leqslant k \leqslant n} \frac{1}{k}$:

$$\begin{aligned} \sum_{1 \leqslant k \leqslant n} r(n,k) &= \frac{2H_{n-1}}{(n+1)(n+2)} - \frac{2(H_n - 1)}{(n+1)(n+2)} \\ &= \frac{4H_n}{n(n+1)(n+2)} + \frac{2(n^2 - n - 2)}{n^2(n+1)(n+2)} \end{aligned}$$

En regroupant, il vient :

$$F(n) = \frac{2(2n^4 + 11n^3 + 18n^2 + 5n - 2)}{9n^2(n+1)(n+2)} + \frac{4H_n}{9n(n+1)(n+2)}$$

La suite de terme général $F(n)$ converge vers $4/9$, ce qui est rassurant. Maple indique que $F(1) = 4/3$, et $F(n) < 1$ pour $n \in [\![2, 100]\!]$. Nous allons montrer que la suite de terme général $F(n)$ est décroissante, ce qui terminera la preuve, avec $n_0 = 1$. Notons $g(n) = \dfrac{2n^4 + 11n^3 + 18n^2 + 5n - 2}{n^2(n+1)(n+2)}$; on a :

$$g(n+1) - g(n) = \frac{-5n^4 - 28n^3 - 38n^2 - 7n + 6}{n^2(n+1)^2(n+2)(n+3)}$$

qui est clairement négatif pour tout $n \geqslant 1$.

Par ailleurs, notant $h(n) = \dfrac{H_n}{n(n+1)(n+2)}$, on a :

$$\begin{aligned} h(n+1) - h(n) &= \frac{H_{n+1}}{(n+1)(n+2)(n+3)} - \frac{H_n}{n(n+1)(n+2)} \\ &= \frac{nH_{n+1} - (n+3)H_n}{n(n+1)(n+2)(n+3)} \\ &= \frac{\frac{n}{n+1} - 2H_n}{n(n+1)(n+2)(n+3)} < -\frac{1}{n(n+1)(n+2)(n+3)} < 0 \end{aligned}$$

ceci car $\frac{n}{n+1} < 1$ et $H_n \geqslant 1$. Ainsi, les suites de termes généraux respectifs $g(n)$ et $h(n)$ sont décroissantes ; il en est de même de leur somme, de terme général $F(n)$. **OUF !**

Question 11 • D'après la question précédente, le rayon de convergence est au moins égal à $1/3$; et il résulte de la question 9 que ce rayon est au plus égal à $1/2$.

Question 12 • Multiplions les deux membres de l'égalité établie à la question 6 par z^{n+2}, et sommons pour $n \geqslant 1$, il vient :

$$\begin{aligned}\sum_{n\geqslant 1} S(n+2)z^{n+2} &= z\sum_{n\geqslant 1} S(n+1)z^{n+1} + z^2\sum_{n\geqslant 1} S(n)z^n \\ &\quad + z^2\sum_{n\geqslant 1}\Bigg(\sum_{1\leqslant i\leqslant n-i} S(i)z^i S(n-i)z^{n-i}\Bigg)\end{aligned}$$

soit encore : $\displaystyle\sum_{n\geqslant 3} S(n)z^n = z\sum_{n\geqslant 2} S(n)z^n + z^2\sum_{n\geqslant 1} S(n)z^n + z^2\Bigg(\sum_{n\geqslant 1} S(n)z^n\Bigg)^2$
qui s'écrit : $\varphi(z) - z - z^2 = z\big(\varphi(z) - z\big) + z^2\varphi(z) + z^2\big(\varphi(z)\big)^2$
ou enfin : $z^2\big(\varphi(z)\big)^2 + (z^2+z-1)\varphi(z) + z = 0$.
Ainsi, φ est solution de l'équation algébrique $z^2X^2 + (z^2+z-1)X + z = 0$.

Question 13 • Il y a deux boucles internes : $(3, 11)$ et $(27, 32)$, une boucle finale : $(18, 24)$ et un bulbe : $(12, 17)$.

Question 14 • L'ensemble $\big\{i \in [\![1,n]\!] : i < s(i)\big\}$ n'est pas vide puisque s comporte au moins une liaison. Notons j son plus grand élément. Soient $\mathcal{I}$ l'intervalle $[\![j+1, s(j)-1]\!]$ et $k \in \mathcal{I}$; $s(k)$ est également dans cet intervalle, sinon les liaisons $(j, s(j))$ et $(k, s(k))$ seraient enchevêtrées ; mais alors, de par la définition de j, on ne peut avoir ni $k < s(k)$ si $s(k) < k$; donc tout point de $\mathcal{I}$ est invariant par s, si bien que $(j, s(j))$ est une boucle finale.

• Voici une autre ligne directrice, proposée par un étudiant : considérer une liaison $\big(i, s(i)\big)$ rendant minimale la quantité $\big|i - s(i)\big|$.

Question 15 • Notons E_n le nombre d'épingles à cheveux de longueur n. On a $E_1 = 0$. Soient s une épingle à cheveux de longueur $n+1$ et $j = s(n+1)$:

- si $j = n+1$, alors s est parfaitement définie par l'épingle à cheveux induite sur l'intervalle $[\![1, n]\!]$;
- sinon, $j \neq n$, donc $j \in [\![1, n-1]\!]$; le modèle t induit par s sur l'intervalle $[\![j, n+1]\!]$ comporte la liaison $(j, n+1)$ donc il présente au moins une boucle finale ; comme s n'en comporte qu'une, celle-ci apparaît dans t, et y est unique. Du coup, le modèle induit par s sur l'intervalle $[1, j-1]$ ne comporte aucune liaison (sinon il comporterait une boucle finale), ce qui revient à dire que $s(i) = i$ pour tout $i \in [\![1, j-1]\!]$. Ainsi, s est parfaitement définie par le modèle t' induit sur l'intervalle $[\![j+1, n]\!]$, qui est, soit le modèle banal, soit une épingle à cheveux.

On en déduit immédiatement la formule $E_{n+1} = E_n + \displaystyle\sum_{1\leqslant i<n} (E_i + 1)$. Par suite, $E_{n+2} - E_{n+1} = E_{n+1} + 1$, soit $E_{n+2} = 2E_{n+1} + 1$, que l'on écrit plutôt :

$$E_{n+2} + 1 = 2(E_{n+1} + 1)$$

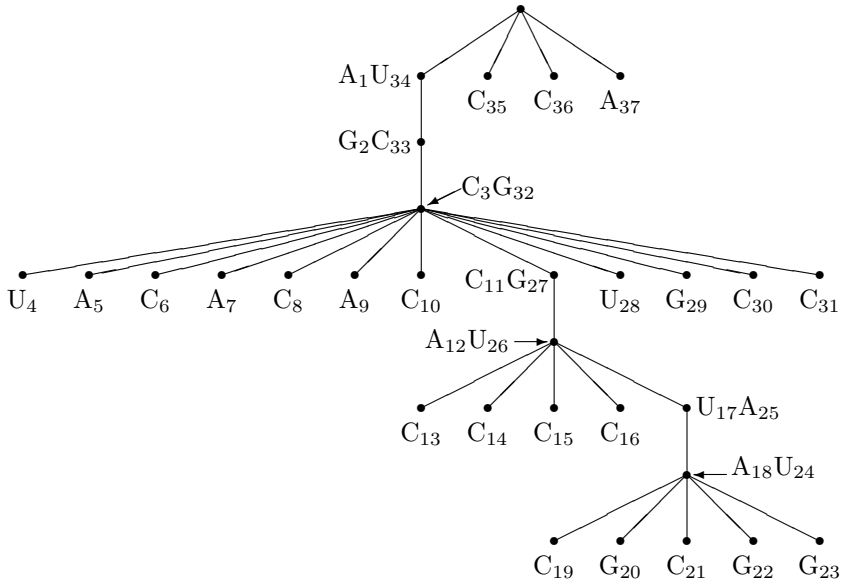
Avec une récurrence immédiate, il vient $E_n + 1 = 2^{n-2}(E_2 + 1)$ soit :

$$\boxed{E_n = 2^{n-2} - 1} \text{ pour } n \geqslant 2$$

Par ailleurs $E_1 = 0$.
Notons que $(3, 2, 1, 6, 5, 4)$ est le plus petit modèle présentant plusieurs boucles finales.

2 Représentation d'un modèle par un arbre

Question 16 • Nous allons raisonner par récurrence sur la longueur n de s. Le cas $n = 1$ est clair : on lui associe l'arbre $\bullet$. Supposons que $\tau(s)$ soit défini pour tout modèle s de longueur au plus égale à n, et considérons un modèle u de longueur $n+1$. Si u n'a aucune liaison secondaire, alors $\tau(u)$ est un arbre de hauteur 1, à la racine duquel sont attachées $n+1$ feuilles. Sinon, chaque liaison secondaire visible induit un modèle de longueur $n-1$ au plus, dont l'image par τ est définie grâce à l'hypothèse de récurrence, si bien que $\tau(u)$ est lui-même défini.



Question 17 Voir la figure 17.

Question 18 • Le nombre de feuilles est égal au nombre de points invariants de s, soit $|s| - 2\ell(s)$; le nombre de nœuds autres que la racine est égal au nombre de liaisons, donc le nombre total de nœuds est $1 + \ell(s)$. Profitons-en pour écrire le calcul en Caml de ℓ :

```
let nls v =
 list_length (filtre (fun (x,y) -> x<y)
  (combine (list_of_vect v,intervalle 1 (vect_length v))));;
```

Question 19 • Notons d'abord que si deux modèles s et t ont même image par τ, alors $|s| - 2\ell(s) = |t| - 2\ell(t)$ et $1 + \ell(s) = 1 + \ell(t)$. On en déduit $|s| = |t|$ et $\ell(s) = \ell(t)$. Il nous suffit donc de prouver que l'assertion $\mathcal{A}_n$ suivante :

$$\text{«si } |s| = |t| = n \text{ et } \tau(s) = \tau(t), \text{ alors } s = t\text{»}$$

est vraie pour tout $n \geqslant 1$, ce que nous allons faire par récurrence sur n.

• Le cas $n = 1$ est immédiat car il n'y a qu'un modèle de longueur 1. Supposons $\mathcal{A}(k)$ acquise pour tout $k \leqslant n$, et considérons deux modèles s et t de même longueur $n + 1$, ayant même image x par τ. Distinguons deux cas de figure, selon la nature du fils f de la racine de x situé le plus à droite. S'il s'agit d'une feuille, c'est que $s(n + 1) = t(n + 1) = n + 1$. Notons alors s' et t' les modèles induits par s et t sur l'intervalle discret $[\![1, n]\!]$; notons x' l'arbre déduit de x par suppression de la feuille f. On a $\tau(s') = \tau(t') = x'$; l'hypothèse de récurrence permet d'affirmer que $s' = t'$ et donc $s = t$.

• Si f n'est pas une feuille, c'est la racine d'un arbre y. Notons $p = s(n + 1)$ et $q = t(n + 1)$; y représente le modèle s' induit par s sur $[\![p, n + 1]\!]$ ainsi que le modèle t' induit par t sur $[\![q, n + 1]\!]$. La remarque faite en début de question implique $p = q$, donc $s' = t'$. Si $p = q = 1$, alors $s = t$. Sinon, notons z l'arbre déduit de x par suppression du sous-arbre y ; notons s'' et t'' les modèles induits respectivement par s et t sur $[\![1, p - 1]\!]$; on aura $\tau(s'') = \tau(t'') = z$, donc $s'' = t''$ grâce à l'hypothèse de récurrence. En «recollant les morceaux» on en déduit $s = t$ et la preuve est terminée.

▶ On peut sans grande difficulté montrer que τ est surjective. Le détail de la preuve est laissé au lecteur !

Question 20 • On procède par lecture du modèle de droite à gauche. La fonction `aux` appliquée à un «accumulateur» et à des indices de début p et de fin q calcule l'arbre associé au modèle induit sur l'intervalle $[\![p, q]\!]$.

```
let tau v =
 let rec aux accu p q =
  if p > q then accu else let r = v.(q-1) in
  if r = q then aux (F::accu) p (q-1)
           else aux (N(aux [] (r+1) (q-1))::accu) p (r-1)
 in N(aux [] 1 (vect_length v));;
```

Pour mémoire, voici le résultat de l'application de `tau` au modèle `v1` sous-jacent à la figure 1 :

```
N[N[N[N[F;F;F;F;F;F;F;N[N[F;F;F;F;N[N[F;F;F;F;F]]]];
 F;F;F;F]]];F;F;F]
```

Question 21 • La fonction `aux` calcule récursivement la ≪portion≫ de modèle qui correspond à la liste placée en un nœud d'un arbre donné ; lorsque la liste est vide ou commence par une feuille, le traitement est clair ; sinon, l'appel récursif de `aux` reçoit la position de départ et rend la position d'arrivée. Le résultat est un couple (position, liste) dont la deuxième composante doit être retournée et convertie en un vecteur.

```
let inv_tau (N t) =
 let rec aux accu p = function
  | [] -> (p,accu)
  | F::q -> aux (p::accu) (p+1) q
  | (N t')::q -> let (p',accu') = aux [] (p+1) t'
    in aux (p::accu'@[p']@accu) (p'+1) q
 in vect_of_list(rev(snd(aux [] 1 t)));;
```

Noter que le filtrage réalisé par `inv_tau` n'est pas exhaustif. On vérifie que `inv_tau(tau v1)=v1`.

3 Représentation linéaire d'un modèle

Question 22 • La preuve est identique à celle de la question 16, et ne sera donc pas détaillée.

Question 23 • $\left|\psi(s)\right|_x$ est le nombre de points fixes de s, soit $|s| - 2\ell(s)$; $\left|\psi(s)\right|_g = \left|\psi(s)\right|_d = \ell(s)$.

Question 24 • À la main ou, mieux, en utilisant la fonction de la question suivante, on trouve :

$$gggxxxxxxggxxxxggxxxxxddddxxxxdddxxx$$

Ce résultat se lit plus aisément sous la forme $g^3x^7g^2x^4g^2x^5d^4x^4d^3x^3$.

Question 25 • Caml détecte un filtrage non exhaustif dans la fonction `psi` :

```
let psi(N t) =
 let rec aux = function
  | [] -> ""
  | F::q -> "x" ^ (aux q)
  | (N t')::q -> "g" ^ (aux t') ^ "d" ^ (aux q)
 in aux t;;
```

Question 26 • Il suffit de reprendre la démarche de la question 19 : le détail de la preuve ne sera donc pas explicité ici. Notons que cette fois, on n'a pas la surjectivité : par exemple, la chaîne de caractères dg n'a pas d'antécédent ; pourtant, elle vérifie les conditions établies à la question 23.

Question 27 • La fonction `sch` extrait d'une chaîne s la sous-chaîne qui commence à la position p et se termine juste avant le caractère `d` associé au `g` en position $p-1$.

```
let rec sch s p =
 let rec aux pp nivo = match s.[pp] with
  | 'x' -> aux (pp+1) nivo
  | 'g' -> aux (pp+1) (nivo+1)
  | 'd' when nivo = 0 -> (pp+1,sub_string s p (pp-p))
  | 'd' -> aux (pp+1) (nivo-1)
  | _ -> failwith "rencontre d'un caractère illégal"
 in aux p 0;;
```

On peut alors parcourir la chaîne s de gauche à droite pour construire récursivement l'arbre associé :

- si s_p est le caractère `x`, on ajoute une feuille à la liste associée au nœud courant ;
- si s_p est le caractère `g`, on extrait avec `sch` la sous-chaîne qui commence en position $p+1$ et se termine juste avant le `d` associé ; on calcule l'arbre associé à cette sous-chaîne, et on l'ajoute comme nœud à la liste courante.

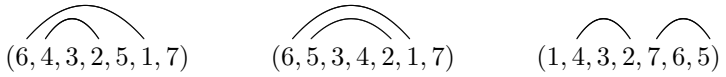
Comme on parcourt la chaîne de gauche à droite, et que l'on ajoute les éléments en tête de liste, l'arbre obtenu doit être «retourné» entièrement, avec la fonction `revtree`.

```
let rec revtree = function
 | F -> F
 | N t -> N(map revtree (rev t));;

let inv_psi s =
 let rec aux ss p accu = let lss = string_length ss in
  if p = lss then accu else match ss.[p] with
   | 'x' -> aux ss (p+1) (F::accu)
   | 'g' -> let (p',ss') = sch ss (p+1)
     in let r = N(aux ss' 0 []):accu in aux ss p' r
 in try revtree(N(aux s 0 []))
  with _ -> failwith "chaîne incorrecte";;
```

4 Optimisation d'une structure secondaire

Question 28 • Les liaisons possibles sont $(1,6)$; $(3,6)$; $(2,4)$; $(2,5)$; $(4,7)$ et $(5,7)$. On en déduit immédiatement six structures à une seule liaison ; on met en évidence trois structures à deux liaisons :



Il n'existe pas de structure à trois liaisons ou plus. Mais il ne faut pas oublier la structure banale, sans aucune liaison !

Question 29 • Soit s une structure secondaire sur l'intervalle $[\![i, j{+}1]\!]$. Si $s(j{+}1) = j{+}1$, alors $\ell(i, j{+}1) = \ell(i, j)$ et $\sigma(i, j) = 0$ puisque $\rho(k, j + 1) = 0$ pour tout $k \in [\![1, j - 1]\!]$. Sinon, notons $k = s(j + 1)$; $k \in [\![i, j - 1]\!]$; le nombre de liaisons de s est au plus égal à la somme de trois termes :

- $\ell(i, k - 1)$: contribution de la structure induite sur l'intervalle $[\![i, j - 1]\!]$;
- $\ell(k + 1, j)$: contribution de la structure induite sur l'intervalle $[\![k + 1, j]\!]$;
- 1 : contribution de la liaison entre $j + 1$ et k.

Ceci donne déjà $\ell(i, j + 1) \leqslant \sigma(i, j)$. L'égalité résulte de la constatation suivante : si s réalise le maximum, alors chacune des structures induites sur $[\![i, k - 1]\!]$ et $[\![k + 1, j]\!]$ réalise aussi un maximum.

Question 30 • On remplit les cases d'un tableau carré t d'ordre n, situées au-dessus de la diagonale, avec les valeurs de $\ell(i, j)$ pour $1 \leqslant i \leqslant j \leqslant n$. On procède par diagonales successives, en partant de la diagonale principale (garnie de zéros). En fin de remplissage, la valeur de $\lambda(m)$ se trouve dans la case en haut et à droite, contenant $\ell(1, n)$. Cette technique (reposant sur la mémorisation de valeurs intermédiaires) est connue sous le nom de *programmation dynamique*.

Question 31 • Le coût se déduit directement de l'application de la formule : le calcul de $\sigma(i, j)$ coûte $2(j - i)$ consultations (deux par valeur de $k \in [\![i, j - 1]\!]$), donc le coût total est :

$$\begin{aligned}\sum_{1\leqslant i\leqslant j<n}(2j - 2i + 1) &= \sum_{i=1}^{n-1}\sum_{j=i}^{n-1}(2j - 2i + 1) = \sum_{i=1}^{n-1}\big((n - i)^2\big)\\ &= \frac{2n^3 - 3n^2 + n}{6} = \mathcal{O}(n^3)\end{aligned}$$

On a négligé les coûts d'initalisation de t (quadratiques, en tout état de cause).

Question 32 • On applique l'algorithme décrit plus haut. La fonction `rho` détermine si deux bases peuvent être appariées ; la fonction `calcule_l` construit la matrice ℓ du texte ; elle sera réutilisée dans la question 33.

```
let rho c1 c2 = match (c1,c2) with
 | ('A','U') | ('U','A') | ('C','G') | ('G','C') -> true
 | _ -> false;;

let calcule_l s =
 let n = string_length s in
 let t = make_matrix (n+1) (n+1) 0 in
  for d = 1 to n-1 do
   for i = 1 to n-d do (* j+1=i+d *)
    t.(i).(i+d) <- t.(i).(i+d-1);
    for k = i to i+d-2 do
     if (rho s.[k-1] s.[i+d-1]) then
      let v = t.(i).(k-1) and w = t.(k+1).(i+d-1)
       in t.(i).(i+d) <- max t.(i).(i+d) (v+1+w)
    done
   done
  done;
 t;;

let lambda s = if s = "" then 0
 else (calcule_l s).(1).(string_length s);;
```

Appliquée à l'exemple de la figure 1, cette fonction rend la valeur 11.

Question 33 • Soit s la chaîne de longueur n dont on veut calculer une structure secondaire optimale. Si $\lambda(s)$ est nul, il n'y a rien à faire. Sinon, soit p le plus petit indice tel que $\ell(1,p) = \lambda(s)$; il existe une structure secondaire optimale dans laquelle p est l'image d'un certain $i \in [\![1, p-2]\!]$, et chaque $j \in [\![p+1, n]\!]$ est invariant.

Pour chaque position $j \in [\![1, p-2]\!]$ dont la base peut être associée à celle qui est en position p, on va calculer $\varphi(j) = \lambda(s[1..j-1]) + \lambda(s[j+1..p-1])$; puis on choisit une position i qui maximise $\varphi(j)$. Il existe une structure secondaire optimale constituée de la liaison (i,p), d'une structure secondaire optimale sur $s_1 = s[1..i-1]$ et d'une structure secondaire optimale sur $s_2 = s[i+1..p]$. Il ne reste plus qu'à calculer (récursivement) ces deux dernières. Nous avons besoin d'une panoplie de fonctions auxiliaires :

- `last_p` détermine la position p
- `bonnes_positions` dresse la liste des associés possibles de p
- `max_of_list` calcule le maximum d'une liste
- `nth_elt` extrait le n-ième élément d'une liste
- `découpe` découpe une chaîne selon les positions i et p comme expliqué plus haut

```
let last_p s =
 let l = (calcule_l s).(1) in
 index l.(string_length s) (list_of_vect l);;

let bonnes_positions s p =
 filtre (fun i -> rho s.[i-1] s.[p-1]) (intervalle 1 (p-2));;

let rec max_of_list = function
 | [] -> failwith "max_of_list illégal"
 | t::q -> it_list max t q;;

let rec nth_elt n = function
 | t::_ when n = 0 -> t
 | _::q -> nth_elt (n-1) q
 | _ -> failwith "appel nth_elt incorrect";;

let découpe s i p =
 (sub_string s 0 (i-1),sub_string s i (p-i-1));;
```

On peut maintenant dresser une liste de liaisons, de longueur maximale. Le fait de manipuler des chaînes (au lieu de listes de caractères) oblige à quelques acrobaties (qui se manifestent par la présence de `delta`).

```
let rec liste_liaisons une_chaîne =
 let rec aux delta = function
 | "" -> []
 | s -> match last_p s with
  | 0 -> []
  | p -> let d i = découpe s i p in
    let bss = map d (bonnes_positions s p)
    and f(u,v) =  (lambda u) + (lambda v) in
    let lss = map f bss in
    let (u,v) = nth_elt (index (max_of_list lss) lss) bss in
    let pss = string_length u + 1 in
     (delta+pss,delta+p)::(aux (delta+pss) v)@(aux delta u)
 in aux 0 une_chaîne;;
```

Il ne reste plus qu'à déduire, de cette liste des transpositions, le modèle réalisant une structure optimale :

```
let optimise s =
 let n = string_length s in
 let v = make_vect n 0 in
 for i = 0 to n-1 do v.(i) <- i+1 done;
  let f(x,y) = v.(x-1) <- y; v.(y-1) <- x in
  do_list f (liste_liaisons s);
 v;;
```

Appliquée à l'exemple de la figure 1, cette fonction rend la liste :

```
[|34;33;32;7;5;6;4;29;28;27;26;23;22;20;15;16;17;18;19;
  14;21;13;12;24;25;11;10;9;8;30;31;3;2;1;35;36;37|]
```

Références bibliographiques

▶ Les idées de base de ce texte ont été trouvées dans le chapitre 13 du livre *Introduction to Computational Biology : Sequences, Maps and Genomes*, de Michael Waterman (éd. Chapman Hall, 1995).

▶ Le *Journal of Computational Biology* publie quatre numéros par an, depuis 1994. Allez visiter sa page Web :
`http://www.cs.sandia.gov/jcb/`

▶ Tous les ans, depuis 1990, se tient le congrès *Combinatorial Pattern Matching*. À partir de 1992, ses actes ont été publiés dans les *Lecture Notes in Computer Science* (éd. Springer). Une large place y est accordée aux articles traitant de l'algorithmique du génome.

▶ Vous trouverez des pointeurs vers les pages personnelles de nombreux chercheurs s'intéressant à la bio-informatique à l'URL :
`http://www.cs.purdue.edu/homes/stelo/pattern.html`

▶ Signalons encore le *deambulum* de l'Université René Descartes, à l'URL :
`http://www.infobiogen.fr/services/deambulum/fr/index.html`
C'est un bon point de départ pour découvrir les aspects algorithmiques ou informatiques de l'étude des molécules de la vie et du génome.

▶ Ce sujet a été traité par les étudiants au cours des vacances de la Toussaint 1998.

▶ Un grand merci à Hubert Fauque, Michel Quercia et Alain Schauber qui m'ont aidé à venir à bout de la majoration de $S(n)\ldots$

FIN