

Option Informatique en Spé MP et MP*

Devoir à rendre après les vacances de Noël 2008 : le corrigé

Polyominos

► Commençons par quelques outils :

```
let flat = it_list (prefix @) [] ;;

let list_of_matrix m = map list_of_vect (list_of_vect m) ;;

let sum = it_list (prefix +) 0 ;;

let transpose m =
  let nl = vect_length m and nc = vect_length m.(0) in
  let t = make_matrix nc nl m.(0).(0) in
  for i = 0 to nl-1 do
    for j = 0 to nc-1 do
      t.(j).(i) <- m.(i).(j)
    done
  done ;
  t ;;
```

Question 1 Facile, avec la préparation d'artillerie précédente :

```
let aire m = sum(flat(list_of_matrix m)) ;;
```

Question 2 Écriture particulièrement lourde. Notez l'utilisation astucieuse des exceptions. En bordant la matrice m par des zéros, on simplifierait nettement le travail.

```
let périmètre m =
  let nl = vect_length m and nc = vect_length m.(0) in
  let k = make_matrix nl nc 0 in
  for i = 0 to nl - 1 do
    for j = 0 to nc - 2 do
      if m.(i).(j) = 1 && m.(i).(j+1) = 0 then k.(i).(j) <- k.(i).(j) + 1
    done
  done ;
  for i = 0 to nl - 1 do
    for j = 1 to nc - 1 do
      if m.(i).(j) = 1 && m.(i).(j-1) = 0 then k.(i).(j) <- k.(i).(j) + 1
    done
  done ;
  for j = 0 to nc - 1 do
    for i = 0 to nl - 2 do
      if m.(i).(j) = 1 && m.(i+1).(j) = 0 then k.(i).(j) <- k.(i).(j) + 1
    done
  done ;
  for j = 0 to nc - 1 do
    for i = 1 to nl - 1 do
      if m.(i).(j) = 1 && m.(i-1).(j) = 0 then k.(i).(j) <- k.(i).(j) + 1
    done
  done ;
  for i = 0 to nl - 1 do
    if m.(i).(0) = 1 then k.(i).(0) <- k.(i).(0) + 1 ;
  done ;
  for i = 0 to nl - 1 do
```


Chemins de Dyck

Question 8 Si l'on est dans le premier cas, alors la portion de chemin codée par v ne passe que par des points d'abscisse strictement positive; en faisant «descendre» d'un cran cette portion, on obtient un chemin de DYCK menant du point $(1, 0)$ au point $(2n - 1, 0)$; donc v est un mot de DYCK;

Réciproquement, supposons $s = uv$. Il est clair que $\sum_{1 \leq k \leq 2n} s_k = 0$; pour $j \in \llbracket 1, 2n-1 \rrbracket$, $\sum_{1 \leq k \leq j} s_k = 1 + \sum_{2 \leq k \leq j} s_k > 0$ puisque la dernière somme est associée à un préfixe de v , lequel est un mot de DYCK. Donc s est bien un mot de DYCK.

Question 9 Si q est impair, alors $\sum_{1 \leq k \leq q} s_k$ est la somme d'un nombre impair d'entiers relatifs impairs; il ne peut donc pas être nul. Tout préfixe v de $s[1..q]$ est préfixe de s , donc $\sum_{1 \leq k \leq |v|} v_k \geq 0$; de plus, de par le choix

de q , $\sum_{1 \leq k \leq q} s_k = 0$; donc s est un mot de DYCK. Notons $w = s[q+1..2n]$; par différence, $\sum_{q+1 \leq k \leq 2n} s_k = 0$; et, pour $j \in \llbracket 0, 2n - q \rrbracket$, nous aurons :

$$\sum_{1 \leq k \leq j} w_k = \sum_{q+1 \leq k \leq q+j} s_k = \sum_{1 \leq k \leq q} s_k + \sum_{q+1 \leq k \leq q+j} s_k = \sum_{1 \leq k \leq q+j} s_k \geq 0$$

Ceci montre que w est un mot de DYCK.

Question 10 L'inclusion de droite à gauche est banale. Pour l'inclusion inverse, il suffit de noter qu'un mot de DYCK est, soit le mot vide, soit (d'après la question précédente) de la forme $uvdw$, où v et w sont des mots de DYCK.

Question 11 Soit s un mot de DYCK de longueur $2(n+1)$. D'après la question 8, ou bien $s = uv$, où v est un mot de DYCK; ou bien $s = vw$, où v est le plus court mot de DYCK qui est préfixe non vide de s , et w un mot de DYCK. Notons $2k$ la longueur de v ; alors, avec $C_0 = 1$, il vient :

$$C_{n+1} = C_n + \sum_{1 \leq k \leq n} C_k C_{n-k} = C_0 C_n + \sum_{1 \leq k \leq n} C_k C_{n-k} = \sum_{0 \leq k \leq n} C_k C_{n-k}$$

Question 12 À RÉDIGER !

Codage d'un mot de Dyck par ses pics et ses creux

Question 13 Soient $n \geq 1$ et j un entier impair compris entre 1 et $n-1$ inclus. Notons $q = \lfloor \frac{j-1}{2} \rfloor$. Nous montrons que $(r_i)_{1 \leq i \leq j}$ est la liste des pics et des creux d'un mot de DYCK de longueur $2n$ ssi elle vérifie les conditions suivantes, où $r_0 = r_{j+1} = 0$:

1. $r_{2i-1} > r_{2i-2}$ et $r_{2i-1} > r_{2i}$ pour $1 \leq i \leq q$;
2. pour $s \in \llbracket 0, q-1 \rrbracket$, la quantité $\sum_{1 \leq i \leq q} (r_{2i-1} - r_{2i-2}) - \sum_{1 \leq i \leq q} (r_{2i-1} - r_{2i})$ est positive ou nulle, et elle est nulle pour $s = q$.

Dans la condition 2, la première quantité est la somme des longueurs des q premières montées, et la deuxième est la somme des longueurs des q premières descentes.

Que ces conditions soient nécessaires est une banalité. Prouvons que ces conditions sont suffisantes. Notons $s = u^{r_1-r_0} d^{r_1-r_2} \dots u^{r_{2q-1}-r_{2q-2}} d^{r_{2q-1}-r_{2q}}$. Alors :

- s est un mot de DYCK, grâce à la condition 2: en effet, cette condition revient à dire que le chemin se termine à l'ordonnée nulle, mais ne passe pas sous l'axe des abscisses;
- la suite $(r_i)_{1 \leq i \leq j}$ est bien la liste des pics et des creux de ce mot, grâce à la condition 1.

Question 14 Soient s et t deux mots de DYCK distincts. S'ils n'ont pas la même longueur, ou le même nombre de pics, alors leurs images par λ sont distinctes. Sinon, nous aurons $s = vux$ et $t = vdy$, avec $|v| > 0$ et $|x| = |y| > 0$. Si v se termine par u , alors t présente un pic à l'abscisse $|v|$, mais pas s ; si par contre v se termine par d , alors s présente un creux à l'abscisse $|v|$, mais pas t . Dans tous les cas, $\lambda(s) \neq \lambda(t)$.

Question 15 Aucune difficulté particulière.

```
type lettre = U | D ;;

let pics_et_creux w =
  let rec aux = function
    | ([],_,r) -> r
    | ([_],_,r) -> r
    | (U::D::q,j,r) -> aux(D::q,j+1,j::r)
    | (D::U::q,j,r) -> aux(U::q,j+1,j::r)
    | (_::q,j,r) -> aux(q,j+1,r)
  in rev(aux(w,1,[])) ;;
```

Codage des polyominos parallélogrammes

Question 16 Pas de difficultés particulières.

```
let s m = map intrv (list_of_matrix (transpose m)) ;;

let codage_poly_par m =
  let rec aux = function
    | (b1,h1)::(b2,h2)::q ->
      let d = min (h1+b1) (h2+b2) - max b1 b2 in
      h1::d::aux((b2,h2)::q)
    | [(_,h)] -> [h]
    | [] -> failwith "erreur Q16"
  in aux(s m) ;;
```

Question 17 Il est nécessaire de découper le travail en plusieurs étapes :

- vérifier la validité des arguments h et d ;
- calculer la hauteur du polyomino ;
- remplir la matrice, en procédant par colonnes successives.

```
let hauteur h d =
  let rec aux = function
    | ([],[],r) -> r
    | (a::x,b::y,r) -> aux (x,y,r+a-b)
    | _ -> failwith "erreur Q17a"
  in aux (h,(0::d),0) ;;

let fill m k b t =
  for j = b to t do
    try
      m.(j).(k) <- 1
      with _ -> pr "k=" k ; pr " j=" j ; failwith " ???"
    done ;;

let remplir m h d =
  let rec aux = function
    | ([a],[],k,v) -> fill m k v (v+a-1) ; m
    | (a::x,b::y,k,v) -> fill m k v (v+a-1) ; aux(x,y,k+1,v+a-b)
    | _ -> failwith "Q17b"
  in aux(h,d,0,0) ;;

let décodage_poly_par (h,d) =
  let p = list_length h and p' = list_length d in
```

```

if p < 1 or p <> p+1 then failwith "erreur Q17" else
let q = hauteur h d in
let m = make_matrix p q 0 in
remplir m h d ; m ;;

```

Question 18 L'unicité est une conséquence du résultat établi à la question 14. Notons $\delta_0 = \delta_p = 1$. Remarquons que δ_{k-1} et δ_k sont tous deux majorés par h_k , pour $k \in \llbracket 1, p \rrbracket$. Alors, pour $j \in \llbracket 1, p-1 \rrbracket$, la quantité $\sum_{1 \leq k \leq j} (h_k - \delta_{k-1} + 1) - \sum_{1 \leq k \leq j} (h_k - \delta_k + 1)$ est strictement positive; et la quantité $\sum_{1 \leq k \leq p} (h_k - \delta_k + 1) - \sum_{1 \leq k \leq p} (h_k - \delta_{k-1} + 1)$ est nulle. Le résultat de la question 13 nous assure que $(h_1, \delta_k - 1, \dots, h_{p-1}, \delta_p - 1, h_p)$ est la liste entrelacée des pics et des creux d'un mot de DYCK. Le mot $u^{h_1 - \delta_0 + 1} d^{h_1 - \delta_1 + 1} u^{h_2 - \delta_1 + 1} d^{h_2 - \delta_2 + 1} \dots u^{h_p - \delta_{p-1} + 1} d^{h_p - \delta_p + 1}$ répond à la question.

Chemins et mots de Motzkin

Question 19 Un chemin de MOTZKIN de longueur n et comportant $n - 2k$ pas h (où $k \in \llbracket 0, n \rrbracket$) est obtenu en choisissant k occurrences de la lettre h, ce qui peut se faire de $\binom{n}{n-2k} = \binom{n}{2k}$ parmi les n du mot; puis en «distribuant» les $2k$ lettres d'un mot de DYCK dans les $2k$ emplacements libres, ce qui peut se faire de C_k façons.

Question 20 hud, uhd, udh, hhh.

Question 21 Si le facteur du apparaît à une abscisse paire $2k$, alors le mot possède un creux à l'abscisse $2k+1$, laquelle est impaire.

Question 22 Soit $\mathbf{m} \in \Delta_n$: ce mot se décompose en n facteurs de longueur 2, qui sont de la forme uu, dd ou ud. Codons ces facteurs par les lettres u, d et h respectivement. Notons F le codage ainsi défini. Il est clair que F est une fonction injective de Δ_n sur l'ensemble \mathcal{M}_n des mots de MOTZKIN de longueur n . Il reste à établir la surjectivité: soit $x = x_1 x_2 \dots x_n$ un mot de MOTZKIN de longueur n ; remplaçons chaque occurrence de u (resp. d, h) par uu (resp. dd, ud); nous obtenons un mot y de longueur $2n$ sur l'alphabet $\{u, d\}$. Pour $j \in \llbracket 0, n \rrbracket$: l'ordonnée finale du préfixe de longueur $2j$ de x est $2|x|_u - 2|x|_d \geq 0$, en particulier l'ordonnée finale de y est 0. Pour $j \in \llbracket 0, n-1 \rrbracket$, l'ordonnée finale du préfixe de longueur $2j+1$ est $2|x|_u - 2|x|_d + 1 \geq 0$ (car $x_{2j+1} = u$). Ceci montre que y est un mot de DYCK; et, par construction, le facteur du ne peut pas apparaître à une abscisse paire dans ce mot. Donc y appartient bien à Δ_n .

Chemins et mots de Motzkin colorés

Question 23 Commençons par montrer que $\Phi(\mathbf{m})$ est un mot de DYCK. Le chemin associé commence et se termine à la hauteur 0; il reste à établir qu'il ne passe pas sous l'axe des abscisses. Notons h_k la hauteur atteinte après k pas; $h_0 = 0, h_1 = 1$; ensuite, le nombre de facteurs uu reste au moins égal au nombre de facteurs dd qui le suivent, donc leur contribution ne permet pas de descendre sous l'ordonnée 1; les facteurs ud ne changent pas ceci; et les facteurs du peuvent au pire faire atteindre l'axe des abscisses.

L'injectivité de Φ est évidente. La surjectivité également: un mot de DYCK \mathbf{s} de longueur $2n+2$ peut s'écrire $ut_1 t_2 \dots t_n d$, où les t_k sont des mots de longueur 2 sur l'alphabet $\{u, d\}$; son antécédent est $\varphi^{-1}(t_1)\varphi^{-1}(t_2)\dots\varphi^{-1}(t_n)$.

Question 24 C_{n+1} est le nombre de mots de DYCK de longueur $2n+2$. Comptons les chemins de MOTZKIN colorés de longueur n : nous fixons d'abord le nombre k de pas u; nous choisissons les $n-2k$ pas colorés; pour ce k , le nombre total de mots est $2^{n-2k} C_k \binom{n}{2k}$. En sommant pour k allant de 0 à $\lfloor n/2 \rfloor$, nous obtenons la formule de TOUCHARD.

Question 25 La fonction \mathcal{K} est clairement injective. Notons $\mathcal{E}_n n$ son image: c'est l'ensemble des couples (s, t) de mots de DYCK qui codent deux chemins menant, l'un de $(0, 1)$ à $(p-1, q)$, l'autre de $(1, 0)$ à $(p, q-1)$, et soumis à la contrainte suivante: ces deux chemins n'ont aucun point en commun, autre que $(0, 0)$ et (p, q) .

Le codage proposé est injectif, et il associe à tout couple (s, t) un mot de MOTZKIN coloré \mathbf{m} de longueur n . Réciproquement, on peut décoder \mathbf{m} pour obtenir un couple (s, t) vérifiant la contrainte exposée plus haut; on en déduit le couple (S, I) sans difficulté.

Question 26 L'écriture de la fonction suivre pourrait certainement être allégée.

```

let suivre(s,t,n) =
  let rec aux = function
    | (j,_,_,_,_) when j=n -> true
    | (_,ax,ay,bx,by) when (ax,ay) = (bx,by) -> false
    | (j,ax,ay,bx,by) when s.[j] = 'v' && t.[j] = 'h'
      -> aux(j+1,ax,ay+1,bx+1,by)
    | (j,ax,ay,bx,by) when s.[j] = 'h' && t.[j] = 'v'
      -> aux(j+1,ax+1,ay,bx,by+1)
    | (j,ax,ay,bx,by) when s.[j] = 'v' && t.[j] = 'v'
      -> aux(j+1,ax,ay+1,bx,by+1)
    | (j,ax,ay,bx,by) when s.[j] = 'h' && t.[j] = 'h'
      -> aux(j+1,ax+1,ay,bx+1,by)
  in aux(1,0,1,1,0) ;;

let vérifie_chemins (s,t) =
  let n = string_length s and n' = string_length t in
  n = n' &&
  s.[0] = 'v' && s.[n-1] = 'h' && t.[0] = 'h' && t.[n-1] = 'v' &&
  suivre(s,t,n) ;;

```

Question 27 La présence de chaînes de caractères autorise l'emploi du for.

```

let codage_SI_vers_motzkin (s,t) =
  let n = string_length s in
  let r = make_string (n-2) '*' in
  for i = 0 to n-3 do
    r.[i] <- match (s.[i+1],t.[i+1]) with
      | ('v','h') -> 'u'
      | ('h','n') -> 'd'
      | ('v','v') -> 'b'
      | ('h','h') -> 'r'
      | (_,_) -> failwith "erreur Q27"
  done;
  r ;;

```

Question 28 Même remarque; il y avait une coquille dans l'énoncé.

```

let codage_motzkin_vers_SI r =
  let n = string_length r in
  let (s,t) = (make_string (n+2) '*',make_string (n+2) '*') in
  s.[0] <- 'v' ; s.[n+1] <- 'h' ; t.[0] <- 'h' ; t.[n+1] <- 'v' ;
  for i = 0 to n-1 do
    match r.[i] with
      | 'u' -> (s.[i+1] <- 'v' ; t.[i+1] <- 'h')
      | 'd' -> (s.[i+1] <- 'h' ; t.[i+1] <- 'v')
      | 'b' -> (s.[i+1] <- 'v' ; t.[i+1] <- 'v')
      | 'r' -> (s.[i+1] <- 'h' ; t.[i+1] <- 'h')
  done; (s,t) ;;

```

Sources

- le chapitre 9 du livre *Applied Combinatorics on Words*
- le cours en ligne de Sophie GIRE
- bricoles diverses glanées sur le Web (en particulier sur l'OEIS)

FIN