Concours CPGE EPITA-IPSA-ESME 2023
Option Sciences du Numérique
Filière MPI
Corrigé

I. Algorithmique sur les arbres et programmation en langage Ocaml : Mots de Dyck et arbres binaires

Question 1. $\varepsilon, abab$ et $aaabbb$ sont de Dyck.

Question 2.

```
let verifie_dyck m =
  let rec aux compteur reste = match reste with
    | [] -> compteur = 0
    | A::r -> aux (compteur+1) r
    | B::r -> compteur >= 1 && aux (compteur - 1) r
  in aux 0 m
;;
```

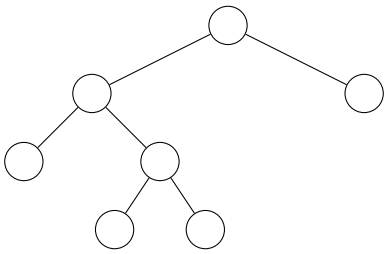
Question 3. Voici les décompositions :

	ab	$aababb$	$ababab$	$aabbab$
u	ε	$ababa$	ε	ab
v	ε	ε	$ababa$	ab

Question 4.

```
let decompo_dyck m =
  let rec dec compteur u reste = match reste with
    | [] -> failwith "impossible"
    | A::r ->  dec (compteur+1) (A::u) r
    | B::r when compteur = 1 -> List.rev u, r
    | B::r -> dec (compteur-1) (B::u) r
  in dec 1 [] (List.tl m)
;;
```

Question 5. L'arbre associé au mot de Dyck $m = aababb$ est le suivant.



Question 6.

```
let rec mot_a_arbre m=
  if m = [] then
    F
  else
    let u,v=decompo_dyck m in N(mot_a_arbre u, mot_a_arbre v)
;;
```

Question 7.

```
let rec arbre_a_mot a=match a with
  | F -> []
  | N(g,d) -> A::arbre_a_mot g @ [B] @ arbre_a_mot d
;;
```

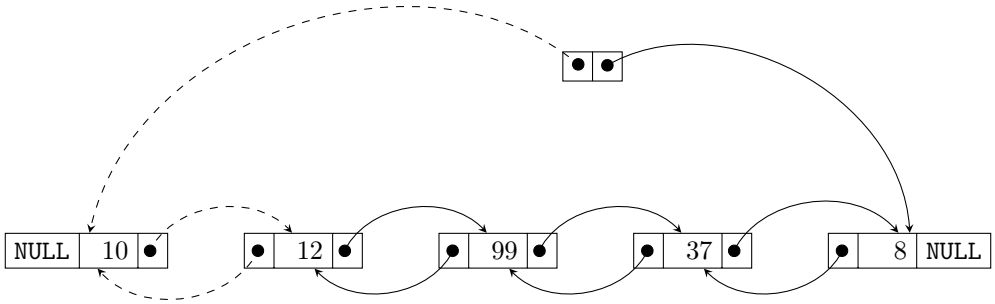
II. Algorithmique et programmation en langage C : structure de liste doublement chaînée et applications

II.1. Structure de liste doublement chaînée

Question 8. Voici la fonction d'en tête `int elem_g(dbliste* q)` :

```
int elem_g(dbliste* q)
{
  return q->gauche->valeur ;
}
```

Question 9. Pour l'ajout à gauche de v, dans le cas d'une liste non vide, il suffit de créer une nouvelle cellule c contenant v, sur qui on fera pointer `gauche`. Le pointeur `prec` de c est `NULL`, le pointeur `suiv` est celui de la cellule précédemment situé tout à gauche. Enfin, il faut faire pointer le pointeur `prec` de l'ancienne cellule située à gauche vers c. Le schéma suivant montre l'ajout de 10 dans la liste proposée dans le sujet.



Dans le cas où la liste doublement chaînée est initialement vide, il faut faire pointer les deux pointeurs `gauche` et `droit` vers la nouvelle cellule, et les pointeurs `prec` et `suiv` de cette cellule sont `NULL`.

Question 10. Voici la fonction d'en tête `void push_g(dbliste* q, int v)`.

```
void push_g (dbliste* q, int v)
{
  cellule* c = malloc(sizeof(cellule)) ;
  c->valeur = v ;
  c->suiv = q->gauche ;
  c->prec = NULL ;
  if (q->droite == NULL)
    q->droite = c;
  else
    q->gauche->prec = c ;
  q->gauche =c ;
  return ;
}
```

II.2. Élement majoritaire dans une liste

Question 11.

```
int occurences(int t[], int taille, int x)
{
  int i=0, nb=0 ;
  for (i=0; i<taille; i++)
    {
      if (t[i]==x) nb++ ;
    }
  return nb;
}
```

Question 12.

```
bool majoritaire(int t[], int taille)
{
```

```
  int i=0;
  for (i=0; i<taille; i++)
    {
      if (occurences(t, taille, t[i])>taille/2)
        {
          return true;
        }
    }
  return false;
}
```

Question 13. La complexité est en $O(n^2)$, avec n la taille du tableau.

Question 14. Notons P la propriété « toutes les balles éventuellement présentes dans la corbeille ont même couleur que celle au sommet du tube ».

- la propriété est vraie avant la boucle, car la corbeille est vide ;
- supposons cette propriété vérifiée, et effectuons un tour de boucle :
  - si la balle piochée est de la même couleur que celle au sommet du tube, elle est placée à la corbeille et la propriété est conservée ;
  - sinon, on la place au sommet du tube, et si la corbeille est vide la propriété est conservée, sinon on place une balle de la corbeille au sommet, qui a alors la même couleur que celles présentes dans la corbeille, s'il en reste. La propriété est également conservée dans ce cas.

P est donc bien un invariant de boucle.

Question 15. On procède de même en notant Q la propriété « deux balles situées côte à côte (ou l'une en dessous de l'autre, plutôt) dans le tube ont des couleurs différentes ».

- initialement, il n'y a qu'une balle dans le tube : Q est donc triviale.
- supposons Q vérifiée et effectuons un tour de boucle. Si la balle piochée est de la même couleur que celle au sommet du tube, celui-ci est inchangé est Q toujours vérifiée. Sinon, on place la balle au sommet du tube (Q est encore vérifiée), et on y met éventuellement une balle de la corbeille, qui est de la même couleur que l'ancienne balle au sommet, donc Q est toujours vérifiée.

Par suite, Q est un invariant de boucle.

Question 16. Notons $t \leq n$ le nombre de balles du tube et c une couleur qui n'est pas celle de la balle au sommet. Sans compter la balle au sommet, il y a $t-1$ balles dans le tube, et au plus $\left\lceil \frac{t-1}{2} \right\rceil \leq \frac{t}{2} \leq \frac{n}{2}$ ont la couleur c. Comme d'après P les balles dans la corbeille n'ont pas la couleur c, c apparaissait au plus $\frac{n}{2}$ fois dans le seau.

Question 17. Il suffit de se restreindre à l'un des côtés : par exemple le gauche. Le sommet de la pile sera donné par `elem_g`, et les fonctions `push_g` et `take_g` permettent d'empiler et de dépiler.

Question 18. La corbeille est simplement représentée par le nombre de balles qu'elle contient, la couleur est inutile puisque c'est celle au sommet du tube !

```
bool majoritaire_eff(int t[], int taille)
{
  dbliste p ;
  init (&p);
  int corbeille = 0 ;
  int i=0;
  int x=0;
  push_g(&p, t[0]) ;
  for (i=1; i<taille; i++)
    {
      x=elem_g(&p) ;
      if (x==t[i])
        corbeille ++ ;
      else
      {
          push_g(&p, t[i]) ;
          if (corbeille != 0)
            push_g(&p, x) ;
        }
    }
  return occurences(t, taille, elem_g(&p))>taille/2 ;
}
```

La complexité de l'approche est maintenant $O(n)$.

II.3. Parcours en largeur dans un graphe $0/1$

Question 19.

```
void parcours(int graph[TAILLE][TAILLE], int larg[TAILLE], int s)
{
  dbliste f ;
  init(&f) ;
  int i ;
  int u ;
  for (i=0;  i<TAILLE; i++)
    larg[i]=-1 ;
  larg[s]=0 ;
  push_g(&f, s) ;
  while (!vide(&f))
    {
      u = take_g(&f) ;
      for (i=0; i<TAILLE; i++)
        if (graph[u][i] != -1 && larg[i] == -1)
          if (graph[u][i]==0)
            {
              larg[i]=larg[u] ;
              push_g(&f, i) ;
            }
          else
            {
              larg[i]=larg[u]+1 ;
              push_d(&f, i);
            }
    }
}
```

III. Lemme d'Arden et langage reconnu par un automate

III.1. Lemme d'Arden

Dans cette sous-partie, on s'intéresse à l'équation $L = (A \cdot L) \cup B$, où l'inconnue est le langage L, A et B étant deux langages fixés.

Question 20. Soit $L = A^* \cdot B$. On a alors $A \cdot L = A \cdot \left(\cup_{n=0}^{+\infty}\right) = \cup_{n=1}^{+\infty} A^n B$. Puisque $A^0 = \{\varepsilon\}$, on a $A \cdot L \cup B = \cup_{n=0}^{+\infty}(A^n \cdot B) = A^* \cdot B = L$. Donc L est solution de l'équation.

Question 21. Montrons par récurrence sur n que si L est solution, alors $A^n \cdot B \subset L$:

- c'est vrai pour $n = 0$, car $B \subset A \cdot L \cup B = L$.
- si la propriété est vraie au rang n, elle l'est au rang $n+1$ car $A^{n+1} \cdot B = A \cdot (A^n \cdot B) \subset A \cdot L \subset A \cdot L \cup B = L$.

Par principe de récurrence, la propriété est démontrée. Ainsi $A^* \cdot B = \cup_{n=0}^{+\infty} A^n \cdot B \subset L$

Question 22. Suivons l'indication : on suppose L solution de l'équation, et montrons par récurrence forte la propriété $P(n)$: « Si m est un mot de L de longueur n, alors $m \in A^* \cdot B$ ».

- si $\varepsilon \in L = A \cdot L \cup B$, alors $\varepsilon \in B$ car $\varepsilon \notin A \cdot L$ puisque $\varepsilon \notin A$. $P(0)$ est donc vérifiée.
- soit $n \geq 1$ et supposons la propriété démontrée pour les entiers précédents. Soit m de longueur n dans $L = A \cdot L \cup B$.
  - si $m \in B$, alors $m \in A^* \cdot B$.
  - sinon, $m \in A \cdot L$. On peut donc écrire $m = uv$ avec $u \in A$ et $v \in B$. Puisque $\varepsilon \notin A$, la longueur de v est strictement inférieure à m. Par hypothèse de récurrence $v \in A^* B$, donc $m \in A \cdot (A^* \cdot B) \subset A^* \cdot B$.

Par principe de récurrence, la propriété est démontrée, et $L = A^* \cdot B$.

III.2 Un exemple d'application

Question 23. Soit m un mot non vide de L_2. Si m s'écrit am', puisque $\delta(q_2, a) = q_0$, on a $m' \in L_0$. Réciproquement, un tel mot am' avec $m' \in L_0$ est bien dans L_2. De même, si m s'écrit bm' alors $m' \in L_2$ et la réciproque est vraie. Comme $q_2 \in F$, on a $\varepsilon \in L_2$. Par suite, $L_2 = \{\varepsilon\} \cup bL_2 \cup aL_0$.

Question 24. Puisque $\varepsilon \notin \{b\}$, on a d'après le lemme d'Arden, $L_2 = b^*(\varepsilon \cup aL_0)$. (On confond ici expression rationnelle et langage dénoté, ce qui n'est pas très rigoureux mais plus court à écrire).

Question 25. On a $L_0 = aL_1$ et $L_1 = aL_0 \cup bL_2$.

Question 26. $L_1 = aL_0 \cup bL_2 = bb^* \cup (bb^* \cup \varepsilon)aL_0 = bb^* \cup b^*aL_0$. Puisque $L_0 = aL_1$, on a $L_0 = abb^* \cup ab^*aL_0$. En appliquant le lemme d'Arden, on trouve $L_0 = (ab^*a)^*abb^*$: cette expression rationnelle dénote bien L_0.

III.3 Cas général

Question 27. Il suffit comme dans l'exemple de la question 23 de discuter suivant la première lettre d'un mot non vide de L_i. Avec

$$B_i = \begin{cases} \{\varepsilon\} & \text{si } q_i \in F \\ \emptyset & \text{sinon} \end{cases} \qquad \text{et} \qquad A_{i,j} = \{a \in \Sigma \mid \delta(q_i, a) = q_j\}$$

on obtient précisément les équations voulues.

Question 28. 1. Puisque $\varepsilon \notin A_{n-1,n-1} \subset \Sigma$, on peut appliquer le lemme d'Arden :

$$L_{n-1} = A_{n-1,n-1}^* \left(\left(\bigcup_{j=0}^{n-2} A_{n-1,j} L_j \right) \cup B_{n-1} \right)$$

2. Il suffit de reporter l'expression précédente de L_{n-1} dans les autres équations, pour obtenir

$$\forall\, i \in [\![0, n-2]\!] \quad L_i = \left(\bigcup_{j=0}^{n-2} A'_{i,j} L_j \right) \bigcup B'_i$$

avec $A'_{i,j} = A_{i,j} \cup A_{i,n-1}A_{n-1,n-1}^* A_{i,n-1}$ et $B'_i = B_i \cup A_{i,n-1}A_{n-1,n-1}^* B_{n-1}$. On remarque que ces langages sont tous rationnels, et que ε n'appartient pas à $A'_{i,j}$ car il n'appartient ni à $A_{i,j}$ ni à $A_{i,n-1}$.

Question 29. Considérons pour $n \geq 1$ la propriété $P(n)$: « Considérons un système de n équations à n inconnues $L_0, \ldots, L_{n-1}$ de la forme

$$\forall\, i \in [\![0, n-1]\!], \quad L_i = \left(\bigcup_{j=0}^{n-1} A_{i,j} L_j \right) \bigcup B_i$$

où tous les langages $(A_{i,j})$ et (B_i) sont rationnels, les $A_{i,j}$ ne contenant pas ε. Alors ce système possède une unique solution $(L_0, \ldots, L_{n-1})$, avec chacun des L_i un langage rationnel ».

— Pour $n = 1$, cette propriété est conséquence du lemme d'Arden ;

— Pour $n > 1$, la question précédente montre qu'on peut ramener le système à un système en les $n-1$ inconnues $(L_0, \ldots, L_{n-2})$ vérifiant les mêmes propriétés. Par hypothèse de récurrence, il admet une unique solution, dont les composantes sont rationnelles. De plus, $L_{n-1} = A_{n-1,n-1}^* \left(\left(\bigcup_{j=0}^{n-2} A_{n-1,j} L_j \right) \cup B_{n-1} \right)$ est également rationnel.

On en déduit la propriété par principe de récurrence. En particulier, L_0 est rationnel.