Informatique — MPSI/PCSI, 2023–2024

TP n° 10 – Terminaison et correction des algorithmes

1 Logarithme en base 10

Soit x un nombre réel strictement strictement positif. On rappelle que le logarithme décimal de x, note $\log_{10}(x)$, est l'unique réel y tel que $10^y = x$. On cherche ici à déterminer la partie entière de ce logarithme décimal, noté $\lfloor \log_{10}(x) \rfloor$.

Un élève propose le programme suivant.

```
def entlog10(x):
    n=0
    while 10**n<=x:
        n+=1
    return n-1
```

Exercice 10.1

1. Ce programme termine-t-il systématiquement ?
2. Est-il correct ?
3. Proposer une spécification pour ce programme et le compléter pour qu'il vérifie cette spécification.

2 Valeur d'un polynôme en un réel

Dans tous ce suite, P sera un polynôme à coefficients dans $\mathbb{R}$. Les polynômes seront codés par des listes de nombres flottants, un polynôme de degré n $P = a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$ étant associé à la liste $[a_0, a_1, a_{n-1}]$. Un polynôme sera souvent confondu avec la liste associée.

On désire écrire un algorithme qui, prenant en entrée un polynôme P et un réel x, calcul la valeur $P(x)$.

Un premier algorithme naïf consiste à calculer puis ajouter successivement la valeur des monômes $a_i x^i$ pour i variant de 0 à $n-1$. On propose pour cela le programme Python ci-dessous.

```
def poly(P,x):
    l=len(P) # Longueur de la liste, égale à degré de p+1
    s=0 # Élément neutre de la somme
    for k in range(l):
       s+=P[k]*x**k
    return s
```

Exercice 10.2

1. Ce programme termine-t-il systématiquement ? Est-il correct ?
2. Proposer des pré-conditions et post-conditions pour cet algorithme et montrer que le programme satisfait cette spécification en utilisant un invariant de boucle.

Une autre possibilité est d'utiliser l'algorithme de Horner. Cet algorithme est basé sur l'écriture suivante du polynôme P :

$$\begin{aligned} P(x) &= a_n x^n + \ldots + a_1 x + a_0 \\ P(x) &= (a_n x^{n-1} + \ldots + a_2 x + a_1)x + a_0 \\ P(x) &= ((\ldots(a_n x + a_{n-1})x + a_{n-2})x + \ldots a_2)x + a_1)x + a_0 \end{aligned}$$

On peut alors décrire l'algorithme `horner(P,x)` récursivement de la manière suivante : si la longueur de P est égale à 1, l'algorithme retourne a_0. Sinon, il retourne $a_0 + x * horner([a_1, \ldots, a_{p-1}], x)$.

Exercice 10.3

1. Écrire une fonction Python prenant en entrée une liste P et un réel x et calculant à l'aide de l'algorithme de Horner de façon récursive $P(x)$.
2. Proposer des pré-conditions et post-conditions pour cet algorithme et montrer que le programme termine et est correct.
3. Préciser le nombre de multiplications et d'additions faite par l'algorithme de Horner en fonction de n et comparer avec la version naïve.

3 Le problème du drapeau hollandais

Le problème du drapeau hollandais est un problème de programmation proposé par Edsger Dijkstra [1] et peut se présenter de la manière suivante : étant donné une liste d'entiers L et un entier p, peut-on réordonner L de telle façon à ce que celle-ci contiennent d'abord les éléments strictement plus petit que p, puis les éléments égaux à p, puis les éléments strictement supérieur à p ?

L'algorithme proposé ci-dessous permet de résoudre le problème en faisant un minimum d'échanges. Il est basé sur l'idée consistant à définir trois entiers i, j et k tel qu'à tout moment dans l'exécution :

— les éléments du tableau d'indices compris entre 0 et $i-1$ sont strictement inférieur à p ;
— les éléments du tableau d'indices compris entre i et $j-1$ sont égaux à p ;
— les éléments du tableau d'indices compris entre j et $k-1$ ne sont pas encore classés ;
— les éléments du tableau d'indices compris entre k et $n-1$ sont strictement supérieur à à p.

Au cours de l'algorithme, si on colorie en bleu les petits éléments (strictement inférieurs à p), en blanc les moyens (égaux à p), en rouge les grands (strictement supérieurs à p) et en gris les non classés, le liste est donc ordonnée comme indiqué sur la figure 1.

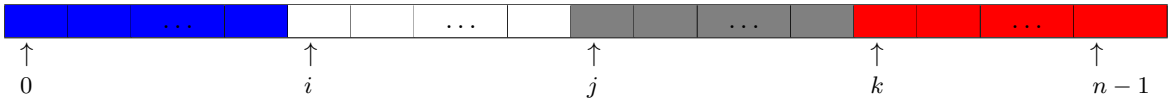

FIGURE 1 – État de la liste au cours de l'exécution de l'algorithme.

Au début de l'exécution, on affecte à i et j la valeur 0 et à k la valeur n. Ensuite, tant que j est strictement inférieur à k, on compare l'élément $L[j]$ à p et on effectue une des trois actions suivantes :

— si $L[j] < p$ (l'élément doit être classé dans la première partie de la liste), on échange $L[j]$ et $L[i]$ et on augmente les entier i et j de 1 ;
— si $L[j] = p$ (l'élément doit être classé dans la deuxième partie), on le laisse en place et on augmente j de 1 ;
— si $L[j] > p$ (l'élément doit être classé dans la dernière partie), on échange $L[j]$ et $L[k-1]$ et on diminue k de 1.

Exercice 10.4

1. Programmer l'algorithme en Python.
2. Montrer que le programme termine à l'aide d'un variant de boucle.
3. Proposer une post-condition pour le programme et montrer que celui-ci satisfait cette condition au moyen d'un invariant de boucle.
4. Déterminer le nombre de tests faits par le programme et le nombre d'échanges faits dans le pire des cas.

1. Edsger Dijkstra (1930-2002), informaticien hollandais. Le problème du drapeau hollandais est appelé ainsi car le résultat final se présente sous la forme d'une liste à trois bandes que l'on peut colorier en bleu, blanc et rouge.

4 La fin des haricots

Le problème des haricots de Gries[2] peut de décrire comme suit : on dispose d'un boite contenant n haricots noirs et b haricots blancs. On tire au hasard et sans regarder deux haricots de la boite. Si les deux haricots sont de la même couleur, on les jette et on remet dans la boite un haricot noir. S'ils sont de couleurs différentes, on jette le noir et on remet le blanc dans la boite, et ainsi de suite jusqu'à ce qu'il ne reste qu'un seul haricot. Que peut-on dire de la couleur du haricot restant en fonction de la répartition initiale des haricots noirs et blancs ?

Exercice 10.5

1. Préciser les préconditions du programme.
2. Programmer l'algorithme en Python. (On pourra utiliser la fonction `random.randrange(n)` du package `random`).
3. Justifier que l'algorithme termine en utilisant un variant de boucle.
4. Conjecturer un résultat sur la couleur du dernier haricot en fonction des valeurs initiales de n et b.
5. Démontrer la conjecture précédente.

* *
*

2. David Gries (né en 1939), informaticien américain