Exercices - Algorithmes : corrigé

Cette feuille d'exercice recense quelques algorithmes trouvés aux détours d'un problème.

Écrire des algorithmes

Exercice 1 - Renversant ! - *L1/Math Sup/Master Enseignement* - ⋆⋆

L'idée est que si l'on prend le reste de a dans la division par 10, on récupère le dernier chiffre, et si on prend le quotient, on récupère a privé de son dernier chiffre. Il suffit d'itérer le procédé, en décalant à chaque fois le résultat provisoire vers la gauche. Une solution est donc :

```
b=0
Tant que a>0 faire
      b=10b+reste(a,10)
      a=quotient(a,10)
Retourner b
```

Exercice 2 - Temps d'arrêt - *L1/Master Enseignement* - ⋆

Nous donnons deux algorithmes.

Algorithme 1 :

```
n=0
H=0
Tant que (H<a) faire
      n=n+1
      H=H+1/n
Fin tant que
Afficher n.
```

Algorithme 2 :

```
n=1
H=0
Tant que (H<a) faire
      H=H+1/n
      n=n+1
Fin tant que
Afficher n-1.
```

Il faut bien remarquer la gestion différente de l'indice n entre le premier et le deuxième algorithme, et notamment le fait que l'on doit retourner $n-1$ dans le deuxième algorithme.

Exercice 3 - Équation de Pell-Fermat - *Master Enseignement* - ⋆

On notera Ent(x) la partie entière de x.

```
Pour y allant de 0 à 100
      z=sqrt(1+2y*y)
      Si z=Ent(z) afficher(sqrt(z),y)
Fin pour.
```

Exercice 4 - En base 2 - *L1/Math Sup/Master Enseignement* - ⋆⋆

1. Cette question est évidemment destinée à comprendre comment proposer un algorithme pour la question suivante. L'idée est de partir d'abord du dernier chiffre. En effet, si 21

s'écrit $b_k \dots b_1$ en base 2, alors on a

$$21 = b_k 2^k + b_{k-1} 2^{k-1} + \dots + 2b_2 + b_1,$$

et donc b_1 est le reste dans la division par 2 de 17. On a donc

$$21 = 2 \times 10 + 1.$$

On continue avec 10 :

$$10 = 2 \times 5 + 0,$$

soit

$$21 = 2^2 \times 5 + 2^1 \times 0 + 1.$$

On reprend avec 5 :

$$5 = 2 \times 2 + 1,$$

soit

$$21 = 2^3 \times 2 + 2^2 \times 1 + 2^1 \times 0 + 1,$$

soit finalement

$$21 = 2^4 \times 1 + 2^3 \times 0 + 2^2 \times 1 + 2^1 \times 0 + 1.$$

Ainsi, 21 s'écrit en base 2 : 10101.

2. Voici un algorithme possible. Il met les chiffres successifs dans une liste. Puis on lit la liste en sens contraire...

```
Lire n
i=0
tant que (n>0) faire
      l[i]=reste(n,2)
      n=(n-l[i])/2
      i=i+1
Fin tant que
Pour j allant de i-1 jusque 0 faire
      Afficher l[j]
Fin tant que.
```

Exercice 5 - Encadrement d'intégrale - *L1/Math Sup/Master Enseignement* - ⋆⋆

1. D'après la relation de Chasles, on sait que

$$\int_0^1 f(t)dt = \sum_{k=0}^{n-1} \int_{k/n}^{(k+1)/n} f(t)dt.$$

Mais, puisque f est croissante, pour tout $t \in \left[\frac{k}{n}, \frac{k+1}{n}\right]$, on a

$$f(t) \geq f(k/n) \implies \int_{k/n}^{(k+1)/n} f(t)dt \geq \int_{k/n}^{(k+1)/n} f(k/n)dt \geq \frac{1}{n} f(k/n).$$

Il suffit de sommer ces relations pour k allant de 1 à n pour trouver que $U_n \leq \int_0^1 f(t)dt$. La preuve est similaire pour V_n.

2. On sait que
$$0 \leq \int_0^1 f(t)dt - U_n \leq V_n - U_n.$$
Pour que U_n soit une valeur approchée (par défaut) à 10^{-3} près de $\int_0^1 f(t)dt$, il suffit que $V_n - U_n \leq 10^{-3}$. Notre algorithme va donc calculer les valeurs de U_n et V_n jusqu'à ce que cette condition soit remplie.

```
n=1
Répète
     U=0
     V=0
     Pour k allant de 0 à n-1 faire
          U=U+exp(-(k*k)/(n*n))
          V=V+exp(-((k+1)*(k+1))/(n*n))
     n=n+1
     Fin pour
Jusqu'à ((V-U)<0.001)
Afficher U
```

Analyser des algorithmes

Exercice 6 - pgcd - *Master Enseignement* - ⋆

L'algorithme ne fonctionne pas. En effet, imaginons qu'on entre dans la boucle Tant que avec $a = 25$ et $b = 12$. Après la première ligne du Tant que, on a $a = 12$ et $b = 12$, puis, dès la deuxième ligne, $b = 0$. Dans tous les cas, l'algorithme va renvoyer la valeur initiale de b. Pour le corriger, il faudrait introduire une troisième variable r, et écrire le Tant que sous la forme suivante :

```
Tant que (b non nul) Faire
      r=reste de a par b
      a=b
      b=r
Fin Tant que.
```

Exercice 7 - Marche aléatoire - *Master Enseignement* - ⋆

L'algorithme ne fonctionne pas, car on utilise, dans la même itération de la boucle Pour, trois tirages aléatoires différents. Rien ne dit que l'une des trois conditions sera remplie. Au contraire, les trois conditions pourraient être remplies. Un algorithme correct est :

```
x=0
Pour t allant de 1 à n Faire
      r=random()
      Si (r<1/3) alors x=x+1
      Si (r>=1/3) et (r<2/3) alors x=x+1
      Si (r>=2/3) alors x=x-3
Fin Pour.
Afficher x.
```

Exercice 8 - Triplets pythagoriciens - *Master Enseignement* - ⋆⋆

Analysons les algorithmes un par un. Celui de l'étudiant 1 est parfaitement correct. En effet, les boucles permettent de parcourir tous les triplets (a,b,c) où a, b et c sont compris entre 0 et 10000, et le test permet de n'afficher que les triplets qui correspondent aux conditions imposées.
Dans l'algorithme de l'étudiant 2, a et b n'évoluent pas indépendamment. Quand b vaut 0, a vaut 0, quand b vaut 1, a vaut 1. L'algorithme ne donnera que les triplets pythagoriciens pour lesquels a=b.
Dans l'algorithme de l'étudiant 3, on ne teste pas si $\sqrt{a^2+b^2}$ est un entier... L'algorithme 4 est correct et plus efficace que le premier.
Le problème de l'algorithme de l'étudiant 5 est plus subtil. En effet, lorsqu'on sort du Tant Que "imbriqué", on a $a^2+b^2+\sqrt{a^2+b^2} > 10000$. On ne change pas la valeur de b et on fait $a = a+1$. Ainsi, on a toujours $a^2+b^2+\sqrt{a^2+b^2} > 10000$, et on sort aussi du Tant Que "externe" immédiatement. On ne teste donc que les valeurs de a pour lesquelles $a = 0$.
Enfin, l'algorithme de l'exercice 6 comporte une boucle infinie. En effet, lorsque la condition $a^2+b^2 = c^2$ est remplie, alors on ne fait pas varier c, et donc on boucle indéfiniment dans le Tant que $a+b+c \leq 10000$ (cette condition d'ailleurs ne testerait pas que $a+b+c \leq 10000$.