Algorithmique : Terminaison et correction de boucle

Yves Josse

Lycée Chateaubriand – PCSI 3

Année 2024-2025

Objectifs et rappels

Objectifs

- Connaître le moyen de sortir d'une boucle sans la parcourir entièrement.
- Démontrer qu'une boucle se termine.
- Justifier qu'une boucle produit l'effet attendu en utilisant un invariant de boucle.

Boucle conditionnelle while

```
while condition:
    bloc_d_instructions
```

Boucle inconditionnelle for

```
for variable in iterable:
    bloc_d_instructions
```

Instruction break

exemple : trouver le plus grand diviseur d'un nombre.

```
n=#affectation de n
k=n-1
sol=0
while k!=0:
    if n%k==0:
        sol=k
        break
    k=k-1
print(sol)
```

```
n=#affectation de n
sol=0
for k in range(n-1,0,-1):
    if n%k==0:
        sol=k
        break
print(sol)
```

Remarque

- La commande `break` interrompt la boucle. Dans une fonction la commande `return` est équivalente.

Exercice

Pour montrer qu'un entier `n` est premier, on parcourt les entiers entre 2 et $\sqrt{n}$. Écrire un programme qui détermine si `n` est premier. On utilisera la variable booléenne `premier` et le la fonction `sqrt` du module `math` (`import math as m`).

Le variant de boucle

Théorème

Toute suite à valeurs entières positives, strictement décroissante, ne peut prendre qu'un nombre fini de valeurs.

Définition du variant de boucle

Un variant de boucle est une expression prenant des valeurs entières et positives et qui décroît strictement à chaque passage dans la boucle. Il montre ainsi qu'une boucle finit par s'arrêter.

Exemple 1

```
n=10
while n>=0:
    print("Demat")
    n=n-1
```

n est un variant de boucle

Exemple 2

```
n=0
while n<=10:
    print("Demat")
    n=n+1
```

10-n est un variant de boucle

Exemple 3

```
def pgcd(a,b):
    while b!=0:
        a,b=b,a%b
    return a
```

b est un variant de boucle

Correction d'une boucle : Invariant de boucle – Principe

Algo 1

```
compteur =1
produit=a
Tant que compteur <=b
    produit =produit+a
    compteur=compteur+1
Fin Tant que
Retourner produit
```

Algo 2

```
compteur =1
produit=0
Tant que compteur <b
    produit =produit+a
    compteur=compteur+1
Fin Tant que
Retourner produit
```

Algo 3

```
compteur=0
produit=0
Tant que compteur<b
    produit =produit+a
    compteur=compteur+1
Fin Tant que
Retourner produit
```

Invariant de boucle

Pour une boucle un invariant de boucle est :

- une propriété (ou un ensemble de propriétés) qui est vraie avant l'entrée dans la boucle et qui reste vraie après chaque passage dans la boucle.
- cette propriété reste inchangée à la sortie de la boucle.

Vérification des algorithmes

Montrer que la propriété $\mathrm{produit_k} = \mathrm{k} * \mathrm{a}$ est un invariant de boucle pour la fonction produit et déterminer l'algorithme correct.

Exercices d'application

1 Factorielle

```
def factorielle(n):
    i,fact=1,1
    while i<=n:
        fact=fact*i
        i=i+1
    return fact
```

Soit $\text{fact}_0 = 1$ la valeur initiale de fact et $\text{fact}_k = k!$ avec $k \in [\![1; n]\!]$. Montrer que fact_k est un invariant de boucle.

2 Fonction inconnue 1

```
def finconnue(a,b):
    q=0
    while a>=b:
        a=a-b
        q=q+1
    return(q,a)
```

1. Prouver la terminaison du programme. Pour cela on cherchera un variant de boucle.
2. Monter que l'expression : $q_k b_k + a_k = a_i$ est un invariant de boucle où $k \in \mathbb{N}$ et a_i la valeur initiale de a.

3 Fonction inconnue 2

```
def fonction(n):
    r,i=2,0
    while i<n:
        r=r*r
        i=i+1
    return r
```

1. Donner une preuve formelle de terminaison de cette fonction.
2. Soit $r_0 = 2$ et $\forall k \in [\![1; n]\!]$, $r_k = 2^{2^k}$.
   Montrer que r_k est un invariant de boucle.

Exercice de synthèse : Exponentiation rapide

On souhaite calculer la puissance entière d'un nombre réel.

1. Écrire une fonction `puiss` qui admet comme argument un nombre réel `x`, un entier strictement positif `n` et qui retourne x^n en utilisant exponentiation « naïve » c'est-à-dire en multipliant n fois par lui même x.
2. Écrire une fonction `puiss_rec` en utilisant un programme récursif.
3. Démontrer la terminaison de la fonction `puiss_rec`.
4. Démontrer la correction de la fonction `puiss_rec`.
5. Évaluer la complexité de la fonction `puiss_rec`. On pourra considérer que l'entier n est une puissance de 2.
6. On souhaite améliore la complexité de la fonction `puiss_rec` en utilisant les propriétés : $x^0 = 1$, $x^{2p} = (x^p)^2$ et $x^{2p+1} = x\,(x^p)^2$. Évaluer la complexité de la fonction ci-dessous.

```
def puiss_rapide(x,n):
    if n==0:
        return 1
    elif n%2==0:
        return puiss_rapide(x,n//2)**2
    else:
        return x*puiss_rapide(x,(n-1)//2)**2
```

7. Écrire une fonction `puiss_rapide2` permettant d'optimiser la fonction `puiss_rapide` avec une seul appel de la fonction récursive dans le corps de la fonction. Évaluer la complexité de la fonction `puiss_rapide2`