

ÉTUDE DES ALGORITHMES. VARIANT ET INVARIANT DE BOUCLE.

1. Preuve d'arrêt d'un algorithme

On considère un algorithme constitué d'une ou plusieurs boucles (boucle **for** ou boucle **while**).

Prouver que cet algorithme se termine revient à prouver que, quelles que soient les données choisies, chaque boucle de l'algorithme ne s'exécute qu'un nombre fini de fois.

- Dans le cas d'une boucle **for i in range(a,b)**, il y a $b - a$ itérations donc l'algorithme se termine nécessairement sous réserve que le corps de boucle ne contienne qu'un nombre fini d'instructions.
- Le cas d'une boucle **while** est plus problématique car le nombre d'itérations dépend des données et de la condition d'arrêt.

Exemple :

```
def somme(L):  
    """somme des termes d'une liste"""  
    compteur = 0  
    n = len(L)  
    s = 0  
    while compteur < n:  
        s = s + L[compteur]  
    return s
```

Pour une liste non vide, ce programme ne se termine pas car la valeur de **compteur** n'évolue pas entre le début et la fin de l'itération. **compteur** vaut toujours 0 et la condition **compteur < n** est toujours vérifiée.

Pour que l'algorithme soit correct, il faut rajouter **compteur = compteur + 1**.

Un critère possible pour prouver la terminaison d'une boucle **while**, est de trouver une suite à valeurs entières positives strictement décroissante car elle a alors un nombre fini de termes.

Définition : variant de boucle

Un variant de boucle est une quantité positive, à valeurs dans $\mathbb{N}$, dépendant des variables de la boucle, qui décroît strictement à chaque passage dans la boucle.

Il montre ainsi qu'une boucle finit par s'arrêter.

Exemple :

```
def somme(L):  
    """somme des termes d'une liste"""  
    compteur = 1  
    n = len(L)  
    s = L[0]  
    while compteur < n:  
        s = s + L[compteur]  
        compteur = compteur + 1  
    return s
```

On note c_0 la valeur initiale de $n - \text{compteur}$.

On note c_i la valeur de $n - \text{compteur}$ à la fin de la i ème itération.

- Avant la première exécution de la boucle, $c_0 = n - 1$.
- A la fin du premier tour, $c_1 = n - 2$.
- On suppose que $c_i = n - i - 1$.
Alors, à la fin du $(i+1)$ ème tour, $c_{i+1} = n - (i + 1) - 1$ car le compteur est augmenté de 1.

La suite $(c_i)_i$ est strictement décroissante et à valeurs entières positives donc la boucle se termine effectivement.

2. Invariant de boucle

Ce n'est pas parce qu'un algorithme se termine qu'il est correct. Il faut vérifier qu'il renvoie effectivement le résultat attendu. Pour cela, on utilise un invariant de boucle.

Définition : invariant de boucle

Un invariant de boucle est une (ou plusieurs) propriété(s) qui relie les variables du programme telle que :

- elle est vraie avant la première exécution de la boucle
- à chaque itération, si la propriété est vraie en entrée de boucle, elle est vraie en sortie de boucle.

Une telle propriété est alors vraie après l'exécution complète de la boucle (il s'agit d'un raisonnement par récurrence).

Exemple

```
def somme(L):
    """somme des termes d'une liste"""
    compteur = 1
    n = len(L)
    s = L[0]
    while compteur < n:
        s = s + L[compteur]
        compteur = compteur + 1
    return s
```

On note s_0 la valeur initiale de s .

On note s_i la valeur de s à la fin de la i ème itération.

On propose l'invariant de boucle $s_i = L[0] + \dots L[i]$:

- Avant la première exécution de la boucle, $s_0 = L[0]$.
- A la fin du premier tour, $s_1 = L[0] + L[1]$.

- On suppose que $s_i = \sum_{k=0}^i L[k]$.

Alors, à la fin du $(i+1)$ ème tour, $s_{i+1} = s_i + L[i+1]$ car, lors de l'exécution de $s=s+L[\text{compteur}]$, le compteur vaut $i+1$.

Comme la variable `compteur` prend les valeurs de 1 à $n-1$, la boucle effectue $n - 1$ tours donc $s = \sum_{k=0}^{n-1} L[k]$: elle contient bien la somme de tous les termes de la liste.

Remarques

- Il n'existe pas de méthode générale pour trouver un bon invariant : c'est une démarche intuitive qui dépend de la compréhension que l'on a de l'algorithme à prouver.
- On peut également construire un invariant en utilisant le début de la i ème itération.