NSI 1ère - Algorithmique - Tris 3

QK

troisième partie : preuves de la correction

Prouver un algorithme

Comment s'assurer que l'algorithme fait ce qu'il annonce ?

- Est-on certain d'obtenir une tableau triée à la fin ?
- Comment le prouver ?

Invariant de boucle

Un invariant de boucle est une propriété qui est vraie AVANT et APRÈS l'exécution d'un tour de la boucle.

Tri par sélection : invariant et terminaison

Tri par sélection : rappel

```
tri_selection(tableau t, entier n)
    pour i de 1 à n - 1
        min = i
        pour j de i + 1 à n
            si t[j] < t[min], alors min = j
        fin pour
        si min = i, alors échanger t[i] et t[min]
    fin pour
```

Tri par sélection : invariant de boucle

Les `i` premiers éléments sont triés.

1. C'est vrai **dès le départ** car on commence à `i = 1`
2. Cela **reste vrai** car on ajoute à droite un élément plus grand que les autres.

Terminaison du tri par sélection

Est-on certain que notre algorithme va se termine bien ?

1. À chaque tour de la boucle extérieure, la liste restante (les non triés) diminue.
2. À chaque tour de la boucle intérieure, `j` augmente. Elle s'arrête bien.

Quatrième partie : programmer les tris en Python

Convertir les algorithmes du pseudo code vers Python

Programme donné

```
def selection(a):
    n = len(a)
    for i in range(n):
        m = i  # on cherche l'indice du min
        for j in range(i+1, n):
            if a[m] > a[j]:
        m = j
    a[m],a[i] = a[i],a[m] # on echange
```

Tri par insertion : invariant

Invariant

Les `i` premiers éléments sont triés.

1. C'est vrai dès le départ car on commence à `i = 1`
2. Cela reste vrai car on ajoute le nouvel élément à sa place.

L'algorithme est correct : le résultat est une liste triée

Terminaison

1. À chaque tour de la boucle extérieure, la liste restante diminue.
2. À chaque tour de la boucle intérieure, `j` diminue. Elle s'arrête bien.

L'algorithme se termine bien.