ALGORITHMES DE TRI : EXERCICES CORRIGÉS

"Testing shows the presence, not the absence of bugs"

E. W. Dijkstra

1. SÉLECTION : LA PREUVE PAR L´INVARIANT

```
1  def triSelection(liste):
2      n=len(liste)
3      for i in range(n):
4          indice_min=i
5          for j in range(i+1,n):
6              if liste[j]<liste[indice_min]:
7                  indice_min=j
8          liste[i],liste[indice_min]=liste[indice_min],liste[i]
9      return liste
```

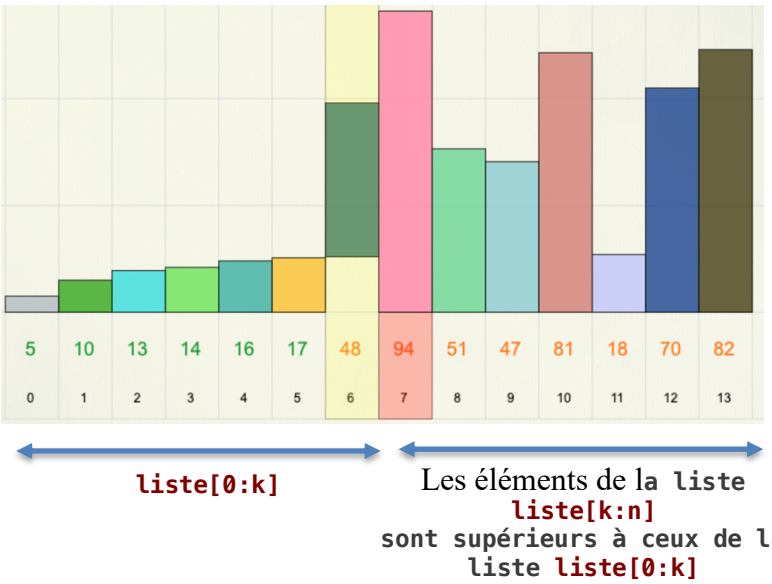
Reprenons le code du tri par sélection. Le but de l´exercice est de montrer que cet algorithme fonctionne par analyse de son invariant de boucle qui s´exprime ainsi : **`La liste liste[0:i+1] est triée en ordre croissant.`**

a) **Initialisation :** Montrez que l'invariant est vrai avant l'entrée dans la boucle et donc avant d'exécuter la première itération.

La propriété est-elle vraie avant d´entrer dans la boucle (donc i=0)? A cet instant i=0, et on peut affirmer que **`la liste liste[0:1]`** est bien triée puisqu´elle ne contient **qu´un seul élément**. **La propriété est donc bien vérifiée pour i=0.**

b) **Conservation** : Prouvez que l'invariant est conservé par une itération de boucle. Pour cela : on suppose que l'invariant est vrai **avant l'itération i** de boucle puis on montre que l'invariant est toujours vrai **après l'itération i** (et donc vrai avant l'itération i+1).

Partons du principe que la propriété est vraie au tour de boucle k : **`la liste liste[0:k]est triée.`** On peut voir sur le schéma que tous les éléments de la **`liste liste[k:n] sont supérieurs à ceux de la liste liste[0:k].`**



On peut donc affirmer l´élément inséré au rang *k+1* sera supérieur à tous ceux de la liste `liste[0:k]`. `La liste liste[0:k+1]` est donc bien triée. **La propriété est bien conservée à chaque tour de de la boucle for.**

c) **Terminaison** : On utilise le fait que l'invariant soit vrai en sortie de la boucle principale pour montrer la *correction partielle* de l'algorithme :

À la fin du dernier tour de boucle, i=n, on peut affirmer que `la liste liste[0:n]` est triée. **La liste est donc bien triée en ordre croissant à la fin de l´exécution de l´algorithme.**

2. LES HARICOTS DE GRIES

Dans les années 1970, Dijkstra propose ce petit jeu : Une boite de conserve contient une certaine quantité `B` de haricots blancs et une certaine quantité `N` de haricots noirs. On réalise les opérations suivantes :

```
TANT QUE il reste au moins deux haricots dans la boite
    On tire au hasard deux haricots
       SI ils sont de la même couleur
           on les jette
           on met un haricot noir dans la boite*
       SINON
           on jette le haricot noir
           on replace le haricot blanc dans la boite
```

** on suppose qu´on dispose d´une réserve suffisante de haricots noirs*

On cherche à savoir quelle sera **la couleur du dernier haricot qui reste dans la boite**.

a) Pour vous aider à répondre aux questions suivantes, complétez ce tableau :

	Au début	Si on tire deux blancs	Si on tire deux noirs	Si on tire un blanc et un noir
Nombre haricots noirs	**N**	N ← **N+1**	N ← **N-1**	N ← **N-1**
Nombre haricots blancs	B	B ← **B-2**	B ← **B**	B ← **B**
Nombre total dans la boite	N+B	**N+B-1**	**N+B-1**	**N+B-1**

b) Montrez que le jeu se termine toujours.

On observant la dernière ligne du tableau, on remarque quel que soit le tirage il y a **décroissance stricte de un haricot** à chaque tour de boucle. Il y aura forcément un moment où il n´en restera plus qu´un. Cette décroissance est appelée *variant de boucle.*

c) Cherchez la *propriété qui reste invariante* tout au long du jeu.

L´étude de la deuxième ligne du tableau nous permet de proposer cette propriété : **la parité du nombre de haricot** est une propriété invariante tout au long du jeu.

→ Si on a au départ un nombre pair de haricots blancs, il y aura toujours un nombre pair dans la boite.
→ Si on a au départ un nombre impair de haricots blancs, il y aura toujours un nombre impair dans la boite.

d) A l'aide de cette propriété, pouvez-vous dire la *couleur du dernier haricot* dans la boite ?

La propriété nous permet de conclure immédiatement que si le nombre de haricots blancs est impair au départ, **le dernier sera blanc** et le nombre de haricots blancs est pair au départ, **le dernier sera noir**.

Les deux situations sont équiprobables.

Pour information, voici une simulation possible du jeu en Python :

```
def recherche(liste,couleur):
    for i in range(len(liste)):
        if liste[i]==couleur:
            return i
    return None

def haricots():
    n= []
    for i in range(randint(5,10)):
        n.append('n')
    b= []
    for i in range(randint(5,10)):
        n.append('b')
    boite=n+b
    shuffle(boite)
    print(boite)
    while len(boite)>1:
        tirage=sample(boite, 2)
        if tirage[0]==tirage[1]:
            if tirage[0]=='b':
                boite.append('n')
                del boite[recherche(boite,'b')]
                del boite[recherche(boite,'b')]
            elif tirage[0]=='n':
                del boite[recherche(boite,'n')]
        else: del boite[recherche(boite,'n')]
    return boite
```