ALGORITHMES: PREMIERS PAS - EXERCICES

1. MESURE D´UNE MOYENNE:

Créez la fonction `moyenne(liste)` qui retourne la valeur moyenne de toutes les valeurs contenues dans une liste. Donnez l´expression du coût `C(n)` de cet algorithme en fonction de n. En déduire que la complexité de cet algorithme est linéaire.

Testez votre fonction avec la liste: `notes=[10,13,8,14,11,16]`

2. EXPERIMENTONS UN TEMPS D´EXECUTION:

```
1 def RechercheMin (liste):
2     n= len(liste)
3     min=liste[0]
4     for i in range (1,n):
5         if min>liste[i]:
6             min=liste[i]
7     return min
```

Reprenons le code dont nous avons analysé la complexité en cours. Le but de l´exercice est de trouver une méthode fiable pour vérifier expérimentalement qu´elle est linéaire. Commencez par écrire cette fonction dans un nouveau programme dans THONNY.

a) Créez ensuite la fonction **`rdmTab(n)`** qui crée, affiche, puis retourne une liste de nombres aléatoires compris entre 0 et 1000. Elle prend en paramètre n, la taille de la liste créée. Testez dans la console la recherche du minimum pour différentes listes aléatoires.

Exemple: `RechercheMin(rdmTab(10))` pour chercher la valeur minimum contenu dans un tableau de 10 valeurs aléatoires.

b) Pour mesurer le temps d´exécution de cette recherche, créons la fonction `afficheTemps(n)` qui retourne le temps d´exécution de la recherche du minimum dans un tableau aléatoire de taille n.

c) Testez dans la console le temps d´exécution de la recherche du minimum pour différentes listes aléatoires de même taille (1000 par exemple). Pourquoi le temps mesuré n´est-il pas le même à chaque fois?

d) Proposez une méthode qui permet de mesurer le ***temps moyen*** d´exécution de ce temps.

Indications:
→ on modifie la fonction `afficheTemps(n)` de manière à retourner une liste contenant le temps d´exécution de la recherche du minimum de 100 listes aléatoires de taille n.
→ On utilise la fonction `moyenne(liste)` de l´exercice précédent pour calculer la valeur moyenne des temps mesurés...

3. CALCUL D´UNE FACTORIELLE

```
1 def factorielle(n):
2     p=1
3     i=n
4     while i>1:
5         p=p*i
6         i=i-1
7     return p
```

La factorielle d´un nombre entier n (nous l'avions déjà rencontrée dans le chapitre sur les fonction) est le produit de tous les entiers de 1 à n soit:

$$n! = 1 \times 2 \times 3 \times \ldots \times (n-1) \times n$$

On peut la calculer en exécutant l´algorithme implémenté en langage PYTHON présenté ci-contre.

a) Exprimez n! en fonction de (n-1)!

b) Déterminez le coût de cet algorithme en fonction de n. En déduire que la complexité de cet algorithme est linéaire.

c) On propose pour cet algorithme la propriété invariante suivante: " **A la fin de l´exécution de la k^e boucle, nous avons la relation:** $p_k \times i_k! = n!$ "

A l´aide cet invariant, établir la correction partielle de cet algorithme.

4. LES HARICOTS DE GRIES

Dans les années 1970, Dijkstra (encore lui) 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 ← .....
Nombre haricots blancs	.....	B ←	B ←	B ← .....
Nombre total dans la boite	N+B	.....	.....	.....

b) Montrez que le jeu se termine toujours.

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

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