Devoir sur table n°2

INFORMATIQUE

Jeudi 15 Décembre

Rappel des consignes

Lorsqu'on écrit un code Python, :

- faire attention à ce que les indentations soient visibles sur la copie ;
- commenter le code de façon à expliquer les grandes étapes de l'algorithme en ajoutant un commentaire en fin de ligne de code après le symbole ♯.

Exercice 1

Ecrire une fonction `renverser` qui à une liste renvoie la liste renversée.
Remarque : On n'utilisera **pas** la méthode `reverse` déjà implémentée dans Python.

```
>>>L=[2,8,-1,7]
>>>renverser(L)
[7,-1,8,2]
```

Exercice 2

1) Ecrire une fonction `maximum(liste)` qui renvoie le maximum d'une liste de nombres non triée.
Remarque : On n'utilisera **pas** la fonction `max` déjà implémentée dans Python.

```
>>>L=[2,8,-7,3]
>>>maximum(L)
8
```

2) Si on note n la longueur de la liste, montrer que la complexité de l'algorithme obtenu est $O(n)$ [1].

Exercice 3

L'objectif de cet exercice est de faire une liste des triangles qui vérifient les trois conditions suivantes :

- les côtés des triangles sont de mesure entière ;
- les triangles sont rectangles ;
- les triangles sont de périmètre p (la valeur de p étant fixée).

1. C'est-à-dire que le nombre d'opérations s'écrit : $an + b$

1) Une fonction rectangle :

   (a) Ecrire une fonction `rectangle(a,b,c)` qui prend comme entrée trois entiers positifs et renvoie `True` si le triangle dont les côtés de mesures a, b et c est un triangle rectangle et `False` sinon. Exemple :

```
rectangle(4,3,5)
>>> True
rectangle(2,7,1)
>>> False
```

   (b) On note N_{rect} le nombre d'opérations de cet algorithme. Calculer N_{rect}.

2) Une première fonction :

   (a) On considère la fonction `triangle` suivante. Que renvoie-t-elle ?

```
def triangle(p):
    Liste=[]
    for a in range(1,p+1):
        for b in range(1,p+1):
            for c in range(1,p+1):
                if a+b+c==p:
                    Liste.append((a,b,c))
    return(Liste)
```

   (b) Modifier la fonction `triangle` en une fonction `triangle2` pour qu'elle renvoie la liste des triangles qui vérifient les trois conditions énoncées au début de l'exercice.

   (c) Déterminer le nombre d'opérations effectuées dans la fonction `triangle2`.

3) Une deuxième fonction :

   (a) On introduit la fonction suivante :

```
def triangle3(p):
    Liste=[]
    for a in range(1,p//3+1):
        for b in range(a,(p-a)//2+1):
                if rectangle(a,b,p-a-b):
                    Liste.append((a,b,p-a-b))
    return(Liste)
```

   Expliquer pourquoi cette fonction renvoie aussi la liste des triangles cherchés.

   (b) Montrer que le nombre d'opérations N_{op} de la fonction `triangle3` vérifie :

$$N_{op} \leqslant 2 + 17\left(\frac{p^2}{12} + \frac{p}{12}\right)$$

   (c) Comparer la complexité des deux algorithmes. Lequel est le plus rapide pour des grandes valeurs de p ?