Informatique en CPGE (2014-2015)
Exercices 8

1 Exercice 1

Comment le temps d'exécution des algorithmes exprimés par les programmes suivants varie-t-il en fonction de n ?

1.

```
n = int(input('Entrer un entier strictement positif'))
for i in range(1,11):
    print(i * n)
```

2.

```
n = int(input('Entrer un entier strictement positif'))
for i in range(1,n+1):
    print(i * i)
```

3.

```
n = int(input('Entrer un entier strictement positif'))
for i in range(1,n+1):
    for j in range(1,n+1):
        print(i * j)
```

4.

```
n = int(input('Entrer un entier strictement positif'))
for i in range(1,n+1):
    p=i*2
    for j in range(1,p+1):
        print(i*j)
```

2 Exercice 2 : tableaux

Déterminer la complexité dans le pire cas (précisez quel est ce pire cas) des opérations suivantes sur un tableau $t = [t_1, \ldots, t_n]$ à n éléments non triés.

1. Affichage du k-ème élément de t.
2. Affichage des éléments de t.
3. Calcul de la somme cumulée des éléments de t.
4. Recherche séquentielle d'un élément.
5. Vérification de l'appartenance de chaque élément de t à un autre tableau T non trié.

3 Exercice 3 : validité

Ecrire un algorithme pour déterminer le plus grand élément d'un tableau et prouver sa validité.

4 Exercice 4

1. Donner l'ordre de grandeur des expressions suivantes :
   n^2+n, $\dfrac{n^2+3n+1}{n+1}$, $\dfrac{n\ln n+n^2+(\ln n)^2}{n+1}$.
2. Si $f(n)$ est une fonction en $\mathcal{O}(n)$, peut-on affirmer que $(f(n))^2=\mathcal{O}(n^2)$?
3. Si $f(n)$ est une fonction en $\mathcal{O}(n)$, peut-on affirmer que $2^{f(n)}=\mathcal{O}(2^n)$?

5 Exercice 5

Si une instruction est traitée en 10^{-7} seconde, quelle est la taille maximale que peut traiter un algorithme en $\ln(n)$, n, $n\ln n$, n^2, 2^n si on dispose de une seconde, ou d'une heure ? Et si l'instruction est traitée en 10^{-20} seconde ?

6 Exercice 6

Il existe des critères (niveau collège) qui permettent de décider si un nombre entier naturel n est divisible par 3, 9 ou 11, **en n'utilisant que l'addition d'entiers et la comparaison de deux entiers**. Ecrire des algorithmes utilisant ces critères et donner leur niveau de complexité. On raisonnera sur le nombre de chiffres dans l'écriture décimale de n, les opérations élémentaires prises en compte sont l'addition et la comparaison de deux nombres.

Rappel :

• un nombre est divisible par 3 lorsque la somme de ses chiffres est un nombre multiple de 3 ;

• un nombre est divisible par 9 lorsque la somme de ses chiffres est un nombre multiple de 9 ;

• un nombre est divisible par 11 lorsque la différence entre la somme des chiffres de rang pair et la somme des chiffres de rang impair est un multiple de 11.

Ecrire les programmes correspondant en ajoutant des compteurs permettant de dénombrer et d'afficher le nombre d'opérations élémentaires. Le nombre n sera entré au clavier par l'utilisateur ; c'est donc un objet de type **str**. On pourra utiliser des conversions entre **str** et **int** suivant les besoins, le type **str** étant pratique pour obtenir les chiffres de n, le type **int** étant nécessaire pour les additions.

7 Exercice 7

1. Ecrire un algorithme traduisant simplement avec deux boucles imbriquées le calcul de la somme $S=\sum_{i,j=0}^{n-1}\left(T[i]-T[j]\right)^2$ pour un tableau de nombres T de taille n. Donner le nombre exact d'opérations à effectuer (additions, soustractions et multiplications). Quelle est la complexité de cet algorithme ?
   Ecrire le programme et le tester avec T=[3, 7, 4, 9, 12, 2].

2. On transforme la somme S afin d'écrire un algorithme de complexité $O(n)$. On peut obtenir l'égalité :
   $S=2\left((n-1)S_n^2-2n\sum_{j=1}^{n-1}T[j]S_j\right)$, où $S_k=\sum_{p=0}^{k-1}T[p]$.
   (On démontrera cette égalité éventuellement en cours de mathématiques.)
   Ecrire l'algorithme et donner le nombre exact d'opérations à effectuer.
   Tester le programme avec le tableau de la question précédente.