TD 1 - Master 1 Bio-informatique

Octobre 2014

Exercice 1

Soit $f(n) = \frac{1}{3}n(n+\frac{1}{2})(n-1)$. Donnez un $\mathcal{O}$, un Ω et un Θ de $f(n)$. Justifiez votre réponse.
Est-ce que les relations suivantes sont vraies, justifiez votre réponses.

i) $f(n) = \mathcal{O}(n^{10})$
ii) $f(n) = \frac{1}{3}n^3 + \mathcal{O}(n^2)$
iii) $f(n) = n^3 + \mathcal{O}(n^2)$
iv) $\mathcal{O}(n^2) = \mathcal{O}(n^3)$
v) $\mathcal{O}(n^3) = \mathcal{O}(n^2)$
vi) $\Omega(n^2) = \Omega(n^3)$
vii) $\Omega(n^3) = \Omega(n^2)$
viii) $\Theta(n^2) = \Theta(n^3)$
ix) $\Theta(n^3) = \Theta(n^2)$

Montrez que $\sqrt{n} = \mathcal{O}(n)$.
Montrez que $\ln(n) = \mathcal{O}(n)$.

Pour les exercices suivants, vous écrirez les programmes sur papier, analyserez la complexité dans le pire cas et dans le meilleur cas, puis vous dessinerez ces complexités à l'aide d'un graphique. Ensuite, vous testerez votre programme à l'aide de l'ordinateur et vous vérifierez l'exactitude de vos calculs en mesurant le temps d'exécution de votre programme et en affichant la courbe de complexité.

Pour mesurer le temps d'exécution de votre programme, vous pouvez utiliser le module *timeit* qui est dédié à la mesure du temps d'exécution des programmes. Pour afficher la courbe, vous pouvez utilisez la fonction *pyplot* du module *matplotlib*. Pour générer des tableaux aléatoires d'entier ou de réel, vous pourrez utiliser le module *random*.

Exercice 2

Que fait le programme suivant :

```
1 def prog(n):
2     t = []
3     if n%2 == 0 :
4         t = range(n)
5     print( t )
```

Exercice 3

Écrire un programme qui prend en paramètre un entier n et qui calcule $\sum_{i=0}^{j} i$.

Exercice 4

Écrire un programme qui prend en paramètre un entier n et qui calcule $\sum_{j=0}^{n} \sum_{i=0}^{j} i$.

Exercice 5

Soit T une liste d'entier, proposez un programme qui recherche le plus grand élément dans T.

Exercice 6

Soit T une liste d'entier, proposez un algorithme qui calcule la somme des éléments de T.

Exercice 7

Soit T une liste d'entier, proposez un algorithme qui cherche l'entier le plus fréquent dans T.

Exercice 8

Soit $(U_n)_{n\in\mathbb{N}}$ la suite définie récursivement par : $U_n = U_{n-1} + U_{n-2}$, $U_0 = 1$ et $U_1 = 1$. Proposer un programme qui prend en paramètre un entier n et qui calcule U_n.

Exercice 9

Écrire un algorithme qui détermine si n est un nombre premier.

Exercice 10

Écrire un algorithme qui prend en paramètre un entier positif n et un réel a et qui calcule a^n sans utiliser la fonction *pow* de python.

Exercice 11

On suppose que vous disposez d'une fonction `proposer_solution(n)` qui prends en paramètre un entier n et qui renvoie vrais si n est égal à un entier secret m qui est compris entre 0 et 1000. Proposez un algorithme qui trouve l'entier m.

Pour tester votre algorithme, vous utiliserez l'implémentation suivante de `proposer_solution(n)` :

```
1 import random
2 class Jeu :
3     def __init__(self):
4         self.m = random.randint(0,1000)
5     def proposer_solution( self, n );
6         return self.m == n
```

Ce code s'utilise de la façon suivante :

```
1 j = Jeu()
2 j.proposer_solution( 4 )
3 j.proposer_solution( 9 )
4 j.proposer_solution( 999 )
```

Exercice 12

Écrire un programme qui prend en paramètre un liste d'entier T et un entier n et qui renvoie la position dans la liste T du premier entier égal à n si n est présent dans la liste, sinon il renvoie -1. Proposez un algorithme plus rapide si la liste est déjà triée.