TD sur le chapitre 1 : algorithmes, preuves et complexité

Partie 1 : Écriture d'algorithmes simples

Exercice 1.1 : Écrivez l'algorithme permettant de calculer les fonctions suivantes :
- La valeur absolue d'un réel *x*, sans utiliser l'opérateur mathématique « |…| » (qui s'écrit « abs( ) » en Python).
- Le maximum de trois réels *a*, *b* et *c*.
- Le nombre de racines réelles d'un polynôme du second degré.

Exercice 1.2 : Écrivez l'algorithme récursif permettant de calculer les fonctions suivantes :
- La fonction factorielle.
- La fonction Fibonacci, définie par : $Fibo(0) = 0$, $Fibo(1)=1$, et $Fibo(n+2)=Fibo(n+1)+Fibo(n)$.
- La fonction de Syracuse, définie par : $S(1)=1$, $S(n)=S(n/2)$ si *n* est pair, et $S(n)=S(3n+1)$ sinon.
- Le nombre de chiffres d'un entier, sans utiliser l'opérateur mathématique *log*.

Exercice 1.3 : Écrivez un algorithme récursif permettant de calculer les fonctions suivantes :
- La fonction « multiplication(*a*,*b*) », avec *b* un entier positif, sans utiliser l'opérateur mathématique « * » mais en réalisant plusieurs additions successives.
- La fonction « puissance(*a*,*b*) », avec *b* un entier positif, sans utiliser l'opérateur de puissance (qui s'écrit « ** » ou « pow( ) » en Python), mais en réalisant plusieurs multiplications successives.

Partie 2 : Preuves d'algorithmes

Exercice 2.1 : Prouvez l'algorithme qui calcule le maximum de trois réels.

Exercice 2.2 : Écrivez l'algorithme « puissance(a,b) » de la partie 1 de manière itérative (c'est-à-dire, sans appel récursif), et prouvez le.

Remarque : D'autres preuves seront faites dans le chapitre 2, une fois que les types de données abstraits auront été étudiés.

Partie 3 : Complexité d'algorithmes

Exercice 3.1 : Est-ce qu'un algorithme en $O(n^2)$ est aussi en $O(n)$? Est-ce qu'un algorithme en $O(n^2)$ est aussi en $O(n^3)$?

Exercice 3.2 : Simplifiez les complexités suivantes :
- $O(2n^3+4n^2-8)$
- $O(n^3-50n)+17n.O(n)+3.O(1)$
- $O(2^n+n^2)$

Exercice 3.3 : Calculez la complexité de l'algorithme PGCD-v1 du cours.

Exercice 3.4 : Calculez la complexité de l'algorithme d'Euclide (algorithme PGCD-v2 du cours).

Exercice 3.5 (TP) : Dans cet exercice, nous allons étudier empiriquement le temps de calcul de deux algorithmes.

- Implémentez les deux algorithmes de calcul de PGCD (PGCD-v1 et l'algorithme d'Euclide) en Python.
- À l'aide de la commande « time », mesurez le temps nécessaire pour exécuter 10^6 fois chacun des algorithmes, sur les mêmes données d'entrée (choisies relativement grandes). Est-ce que vous observez une différence de temps de calcul ?
- Représentez graphiquement le temps de calcul de chaque algorithme, en fonction de la taille de l'entrée. Vous pourrez utiliser la librairie « matplotlib » de Python.
- Ajoutez au graphique précédent la fonction *2.08.log(n)*, où *n* est l'entrée du PGCD (par exemple, la somme de *a* et *b*).

Remarque : D'autres calculs de complexité seront faits dans le chapitre 2, une fois que les types de données abstraits auront été étudiés.