ALGORITHMIQUE - Introduction

1 - Définition

Un algorithme décrit un processus permettant de résoudre un problème.

Donald Knuth en 1964 donne les caractéristiques importante d'un algorithme :

- Un algorithme doit toujours se terminer en un **nombre fini d'étapes**
- Chaque étape d'un algorithme doit être **décrit précisémment**, sans ambiguité
- Un algorithme a des **entrées** (0 ou plus)
- Un algorithme a une ou plusieurs **sorties** qui ont une relation spécifiée avec les entrées
- Les instructions doivent être **suffisamment basiques** pour pouvoir être exécutées de manière exacte, en un **temps fini**, par une personne utilisant un papier et un crayon

On utilisera le langage naturel pour la description d'un algorithme et leur traduction dans un langage de programmation permettra de vérifier certaines notions : nombres d'opérations-étapes effectuées (**coût**), **correction**, **terminaison** (validité).

2 - Un exemple

Prenons un algorithme pas très récent (on remonte à l'Antiquité vers 300 avant JC) : l'algorithme d'Euclide.

Son objectif est de déterminer le PGCD de deux entiers naturels. On verra que les algorithmes ne s'appliquent pas qu'à des situations mathématiques (heureusement ?).

En langage naturel :

- ```
    Soit m et n deux entiers naturels, m >= n
    Etape 1 : on divise m par n (division entière) en notant r le reste
  (0<=r<n)
    Etape 2 : si r=0, terminé, le pgcd est n
    Etape 3 : sinon on remplace m par n et n par r et on recommence
  l'étape 1
  ```

1. Teste cet algorithme à la main pour m = 686 et n= 56.
  ```

En Python :

- ```
  def pgcd(m,n):
      r=m%n
      while r>0:
          m=n
          n=r
          r=m%n
      return n
  ```

2. Teste ce programme et vérifie les valeurs précédentes.

En javascript :

- ```
  function pgcd(var m:int,var n:int){
      var r=m%n
      while r>0 {
          m=n
          n=r
          r=m%n
      }
      return n
  }
  ```

3 - Coût

Connaître un algorithme permet de déterminer le nombre d'opérations, de comparaisons, d'affectations ... donc le nombre d'instructions réalisées.

Bien souvent, on déterminera un encadrement de ce coût, le coût exact dépendant des valeurs en entrées.

Avec l'algorithme d'Euclide, une mesure du coût est par exemple le nombre de divisions effectuées :

1. pour 686 et 56, détermine le nombre de divisions utilisées
2. pour 685 et 2, détermine le nombre de divisions utilisées
3. de manière générale, trouve un majorant du nombre de divisions utilisées avec l'algorithme d'Euclide

4 - Compteur

Une fois un algorithme programmé dans un langage, on peut ajouter un ou des compteurs au sein des boucles pour récupérer le nombre d'opérations, comparaisons ... effectuées .

1. Teste le programme suivant :

```
def pgcd2(m,n):
    r=m%n
    compteur=1
    while r>0:
        m=n
        n=r
        r=m%n
        compteur=compteur+1
    return n,compteur
```

2. Voici un autre algorithme :

```
Soit a et b deux entiers naturels
Etape 1 : si a = b, on affiche a
Etape 2 : sinon si a>b alors on remplace a par a-b et si a<b alors on
remplace b par b-a et on recommence l'étape 1
```

a. Traduis cet algorithme en Python

b. Teste ton programme avec a = 686 et b = 56. Explique le résultat.

c. Ajoute un compteur afin d'évaluer le nombre de soustractions.

5 - Terminaison

Un algorithme ne doit contenir qu'un nombre fini d'étapes : il doit se terminer quelques soient les valeurs en entrées.

On doit donc vérifier la terminaison des algorithmes utilisés.

Dans le cas de l'algorithme d'Euclide, il s'agit de trouver le dernier reste non nul.

Avec les notations utilisées, on a 0 <= r < n <= m : les restes calculés sont une successsion d'entiers strictement décroissante. Celle-ci se terminera forcément par zéro (on rentre encore dans des considérations mathématiques) : l'algorithme d'Euclide se termine.
