AGR. Introduction

I/ Origine de l'algorithmique

Al Khawarizmi (IXème siècle) était un savant de Bagdad, originaire d'Asie centrale (l'Ouzbékistan actuel) et est considéré comme le « père de l'algèbre ». Ses écrits en arabe ont permis la diffusion jusqu'en Europe des chiffres indiens dont le zéro -très rapidement adoptés par les pays musulmans- et de l'algèbre grâce notamment à Fibonacci. Il a classifié les algorithmes existant à son époque et son nom est à l'origine du mot « algorithme ».

L'algorithme d'Euclide (trouver le plus grand diviseur commun entre deux nombres -PGCD-) est peut-être le premier algorithme non trivial, mais on trouve des algorithmes dans le calcul à Babylone datant d'environ 2300 ans.

L'algorithme d'Euclide en Python

```
def PGCD(M,m) : # M > m, algorithme d'Euclide
    r = M % m     # Reste de la division euclidienne de M par m
    if r == 0 :
        return m
    else :
        return PGCD(m,r)

print(PGCD(18,12))
print(PGCD(19,7))
```

```
6
1
```

Les PGCD écrits à la main

PGCD(18,12) = 6
18 = 12 x 1 + **6**
12 = **6** x 2 + 0

PGCD(19,7)
19 = 7 x 2 + **5**
7 = **5** x 1 + **2**
5 = **2** x 2 + **1**
2 = **1** x 2 + 0

A savoir : Un algorithme est une suite ordonnée et finie d'instructions conduisant à un résultat.

Donald Knuth a énoncé quelques règles dans un ouvrage , *The Art of Computer Programming* en 1962. Voici les cinq caractéristiques qu'il présente :

- Un algorithme doit toujours se terminer après un nombre fini d'étapes.
- Chaque étape d'un algorithme doit être définie précisément, les actions à mener doivent être spécifiées rigoureusement et sans ambiguïté pour chaque cas.
- Un algorithme a des entrées, zéro ou plus, quantités qui lui sont données avant et/ou pendant son exécution.
- Un algorithme a une ou plusieurs sorties, quantités qui ont une relation spécifiées avec les valeurs d'entrées.
- Les instructions doivent être suffisamment basiques pour pouvoir être exécutées de manière exacte à la main.

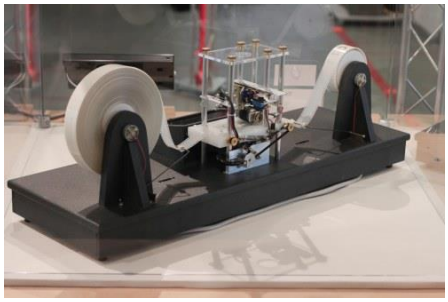
En savoir plus sur Donald Knuth ici : https://www.babelio.com/auteur/Donald-Knuth/208907

II/ Machine de Turing

1/ Principe de fonctionnement

En 1936, Alan Turing présente sa « machine » **théorique**.

Le premier élément abstrait est un **ruban** ayant un commencement mais une longueur infinie et qui représente la mémoire de l'ordinateur (qui elle est finie).
Le second élément abstrait est une **tête de lecture** qui lit, écrit et se déplace sur ce ruban. On peut modifier à l'infini une écriture sur le ruban.
Il faut définir des caractères mais aussi des « cases blanches » permettant de délimiter la partie du ruban à lire, cela forme un alphabet.
Une table de transition comprend l'intégralité des actions à mener en fonction de l'état de la tête de lecture et du caractère lu.

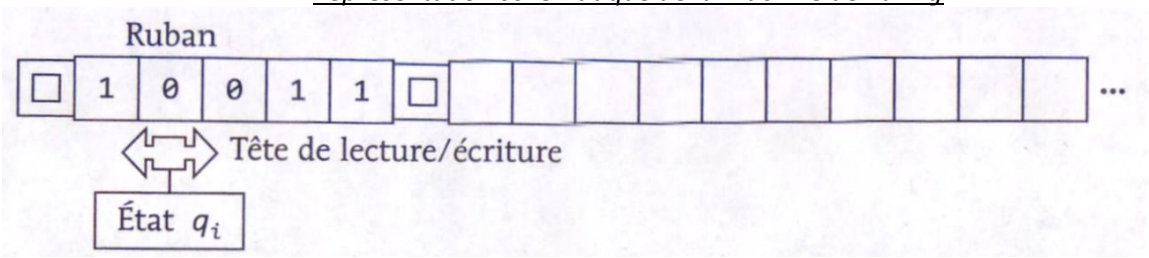


Une machine de Turing
Source : https://nucambiguous.wordpress.com/

Il y a trois actions possibles qui dépendent de l'état du pointeur et du caractère pointé :

- **Changer l'état du pointeur** (mode lecture, mode écriture ou mode déplacement).
- **Changer** le **caractère pointé** sur le ruban (alphabet).
- **Déplacer** le pointeur sur le ruban.

Représentation schématique de la machine de Turing



Dans cet exemple, le pointeur peut se déplacer entre les cases blanches et lire et/ou écrire sur 5 cases en fonction de la table de transition. L'alphabet est composé des symboles [0 ; 1 ; □].

A voir en vidéo : fonctionnement de la machine de Turing : https://www.youtube.com/watch?v=X610pII4_J8
Auteur : Maths Adultes, Gilles Bailly Durée : 49 min 47 *(à regarder de 9 min à 16 min)*.

2/ Complexité d'un algorithme

La machine de Turing permet de mesurer la **complexité** d'un algorithme, c'est-à-dire de déterminer le nombre d'instructions élémentaires nécessaires avant d'obtenir le résultat souhaité.
Modéliser un algorithme par la machine de Turing permet de répondre à trois questions fondamentales :

- L'algorithme va-t-il se terminer ?
- Quelle est l'efficacité en temps de l'algorithme (complexité temporelle) ?
- Quelle place en mémoire va-t-il utiliser (complexité spatiale) ?

En pratique, on se contente de décomposer l'algorithme en « opérations élémentaires » de complexité constante comme une opération mathématique, une affectation, une comparaison etc.

On parle de **complexité linéaire** si le nombre d'opérations est proportionnel à celui des données et de **complexité quadratique** s'il est proportionnel au carré de celui des données.