Rapide introduction en algorithme

1) Qu'est ce qu'un algorithme ?

C'est une méthode qui permet de procéder pour faire quelque chose comme trier des objets, situés des villes sur une carte, faire apparaître des pubs selon nos envies… .

Il est composé d'instructions et d'opérations réalisées, dans un ordre précis sur des données afin de produire un résultat, et souvent résoudre un problème plus ou moins complexe.

2) Les fondations d'un algorithme :

Un programme se construit comme un bâtiment: brique après brique, pas à pas. Un ordinateur, si on lui demande pas, il ne fera rien, logique ?

Donc il faut couper un algorithme en petit problème et lui expliquer tout en détail.
Pour communiquer avec un ordinateur, on utilise un langage de programmation comme le c, java, python….

3) Les variables

Une variable est une manière de faire référence à un objet. C'est comme un boite on l'on stock une valeur comme un string, int, bool, float…..
Exemple en js :

```
var myVar = 12;
```

4) Les fonctions

C'est un bloc de code qui regroupe plusieurs actions. Ce bloc est bien sûr réutilisable autant de fois que l'on veut.

En javascript une fonction s'écrit comme cela :

```
function addition(a,b){
return a + b)}
console.log(addition(1+2))
```

5) Les boucles

C'est une structure qui répète la même action plusieurs fois de suite. Il se compose de trois choses :

- initialisation (1)
- condition (2)
- incrémentation ou décrémentation) (3)

La boucle for se répéter jusqu'à atteindre un chiffre en particulier

```
for (var i = 0; i < 10; i++)
```

La boucle while se répète jusqu'à ce que la condition soit atteinte

```
while (i<5){
console.log(i++)
```

6) Les conditions

Appelé if/else permet de de vérifier une condition. Si la condition est validée alors elle renvoie true sinon false.

en javascript :
if(a < b){
return true;
} else{
return false
}.

Si les conditions sont trop nombreuses, il est préférable d'utiliser un switch. Il vérifie

en javascript:
switch (variable){
case 1: "grand":
return true;
break;
case 2: "petit:
return false;
break
default:
break;

7) Les types de données

- les nombres (int, float, double…)
- string (les chaînes de caractères)
- booléens (true or false)

8) Agencer les informations entre elles

Les différents types ou structures de données permettent de stocker plusieurs données de même type ou de types différents dans un seul conteneur.

- Opération :
  Le CRUD (créer, lire, modifier, supprimer)

- Tableaux :
  C'est une structure qui permet de mémoriser plusieurs données de type semblable ou différent.. en JS = var array = [12, "lol"].

Dans plusieurs langages, la taille d'un tableau est fixe. On ne peut modifier ou supprimer les données d'un tableau. Cependant il est possible de le faire en changeant grâce à l'index.

- Liste:

Moins flexible que le tableau, il rend l'ajout et la suppression de valeur très facile.

- Tables de hachage:

Il permet d'associer une clé à une valeur et non plus à un indice. Il ne comporte pas d'ordre comparé à une liste ou tableau. Il est identifiable uniquement via sa clé. Exemple en python :

```
>>> lolita = {"name": "Dolorès", "address": "New Hampshire"}
>>> lolita["address"]
"New Hampshire"
```

9) Les piles et files.

Les premières informations arrivées dans le système seront les premières traitées.

- Les piles

Il s'agit d'une structure de données qui donne accès en priorité aux dernières données ajoutées. Ainsi, la dernière information ajoutée sera la première à en sortir.

Ils sont les données LIFO (LAST IN FIRST OUT) qui signifie, dernier ajout, premier parti. Ils sont très pratiques lorsque nous aurons à utiliser en premier les dernières données ajoutées.

Par exemple, elles sont utilisées dans les plateformes de streaming musical. Lorsque on demande à la plateforme de lire une chanson à la suite, elle va ajouter cette dernière tout en haut de la liste "en attente de lecture".

- Les files

C'est une structure de données dans laquelle on accède aux éléments suivant la règle du premier arrivé premier sorti, ou encore FIFO (First In First Out).

Dans une logique de la vie réelle, on peut la comparer à une file d'attente chez McDonald. Le premier arrivé, le premier servi.

Pile = Initialise, EstVide, EstPleine, AccederSommet, Empiler, Depiler, Vider, Détruire

- En résumé

Les LIFO (PILE) s'utilisent pour traiter en priorité les dernières données arrivées. En exemple la file d'actualité de facebook.

Les FIFO (FILE) s'utilisent pour traiter un flux de données par ordre d'arrivée. C'est le cas d'un todo list (votre première action est celle que vous devez réaliser en premier.)

Files = Initialiser, EstVide, EstPleine, AccederTete, Enfiler, Defiler, Vider,Detruire

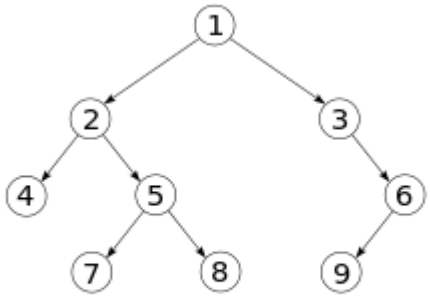
Exemple d'une pile de données en javascript qui est demandé d'ajouter un dernier élément d'un tableau et supprimer le premier arrivé.

```
function nextInLine(arr, item){
var removed=""
arr.push(item);
var removed= arr.shift()
return removed;
}

var testArr = [1,2,3,4,5]
//Output [2,3,4,5,6] A VOIR MA DOCUMENTATION SUR LA  FILE D'ATTENTE EN JS
```

10)Les arbres

- Arbres binaires



Exemple d'arbre binaire

Comme dans un arbre généalogique, toutes les cellules sont des cellules filles, sauf une qui est la cellule mère. Si l'arbre n'est pas vide, une seule cellule n'a pas de cellule mère et celle-ci est appelée racine de l'arbre.

Pourquoi l'appelle t'on arbre binaire ? Car chaque cellule mère a 0,1 ou 2 cellules enfants qui elles-mêmes sont liées à d'autres cellules.

Pour terminer, un arbre binaire est une structure analogue à une liste chainée, sauf que chaque cellule possède jusqu'à deux suivantes. Par convention on convient d'appeler fils gauche et fils droit les deux fils d'un nœud. Le fils gauche et le fils droit d'un nœud peuvent ne pas exister. L'accès à l'arbre est donné par un pointeur qui contient l'adresse de la racine.

- Parcourir un arbre binaires

Pour trouver une information dans un arbre binaire, il faut parcourir tous ses nœuds grâce à une fonction récursive.

En conclusion, les arbres sont utiles pour trier les informations.

- Un graphe

Il peut représenter un réseau routier ou aérien. Ils sont utilisés par des algorithmes dont le but est de trouver le plus court chemin entre un point A et B comme google maps facebook qui utilise les graphes pour trouver les amis les plus proches.

Sur facebook chaque réseau est représenté par un cercle (appelé sommet). Les amis sont reliés par des lignes(appelées arcs).

11) Trier les informations

Trier les chiffres du plus petit au plus grand est facile quand il y en a 10 mais difficile quand il y en a 1000000.

- Le tri à bulles

Cet algorithme est avancé dans une liste d'éléments. Il compare les données deux à deux et les échanges si la première valeur est plus élevée que la seconde.

```
def inter(a_list):
    for i in range(0, len(a_list) - 1):
        for j in range(0, len(a_list) - 1):
            if a_list[j + 1] < a_list[j]:
                a_list[j+1], a_list[j] = a_list[j], a_list[j+1]
    return a_list
```

Il existe trop d'algorithmes de tri pour en parler ici. Ainsi il serait plus sage de chercher sur google les différents types de tri.

12) La récursivité

En programmation nous pouvons utiliser des boucles pour répéter une opération; Nous appelons cela des boucles *itératives.*

- En Programmation :

C'est une fonction qui fait référence à elle-même. Deux fonctions peuvent s'appeler l'une l'autre, on parle alors de récursivité croisée.

En mathématique, les factorielles : une factorielle d'un entier naturel n est le produit des nombres entiers strictement positifs inférieurs ou égaux à n : n! = n-1*n
Exemple factoriel de 10! : 1*2*3*4*5*6*7*8*9*10 = 3 628 800.

En python la récursivité :

```
def factorielle_recursive(n):
        return n*factorielle_recursive(n-1)
```

Cet algorithme est infini car il est appelé n=3,n=2,n=1,n=0,n=-1….. . Et la solutions sera d'y ajouter un condition d'arrêt:

```
def factorielle_recursive(n):
        if n<=1:
        return 1:
        else:
        return n*factorielle_recursive(n-1)
```

Un appel récursif doit obligatoirement être dans une instruction conditionnelle

- Suite de fibonacci

C' est une liste de nombres entiers. Elle commence par les nombres 0 et 1. On appelle un nombre de cette liste un terme. Chaque terme est la somme des deux termes qui le précèdent.

Par exemple, si la suite de Fibonacci débute ainsi : 0,1,1,2,3,5,8,13,21… vous voyez que 0 +1 donne 1, que 1 +2 = 3….

Il existe plusieurs manières de résoudre les problèmes liés à la suite de fibonacci.

- Algorithme naïf

Algorithme mal pensé et pas adapté

- Les piles d’appels

Les piles d’appels permettent à l'ordinateur de retenir le résultat de tous les calculs récursifs avant de finir sa boucle. Il est généré automatiquement par le système.

Dans le cas d’un appel récursif, c’est exactement ce qui se passe ! Lorsque nous calculons une factorielle, nous devons exécuter n(-1) puis n(-2)... jusqu'à arriver au nombre que nous souhaitons calculer.

valeur de n	résultat dans la pile
n(-9)	1
n(-8)	2
n(-7)	3
n(-6)	4
n(-5)	5
n(-4)	6
n(-3)	7
n(-2)	8
n(-1)	9
n	10

Lorsque la dernière fonction récursive est appelée, l’ordinateur “dépile”. Autrement dit, il va chercher dans la pile le dernier élément enregistré et ainsi de suite jusqu’à arriver en bas de la pile.

- En résumé: la récursivité

Il permet de décomposer un problème en un ou plusieurs sous-problèmes du même type.

Les sous problèmes doivent être de taille plus petite que la problème initial.

La décomposition doit en fin de compte conduire à un élémentaire qui, lui,n’est pas décomposé en sous-problèmes.

- Structure d’une fonction récursive

```
mafonction(param1, param2):
    début
        si condition faire
            retourner calcul
        sinon faire
            mafonction(param1, param2)
            retourner quelque-chose
        fin condition faire
    fin
fin ma fonction
```

13) Aide à la programmation

- Se remettre en math avec les cours de terminal s et première s
- Algorithme sur coding game et france ioi