ALGORITHMIQUE

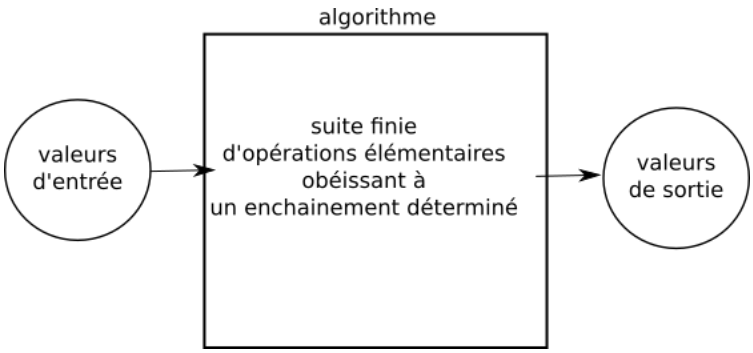
Qu'est-ce qu'un algorithme ?

Voici deux définitions trouvées dans la littérature :

- Procédure de calcul bien définie qui prend en entrée une valeur ou un ensemble de valeur, et qui donne en sortie une valeur ou un ensemble de valeur.
- Un algorithme est la spécification d'un schéma de calcul sous forme d'une suite finie d'opérations élémentaires obéissant à un enchainement déterminé.

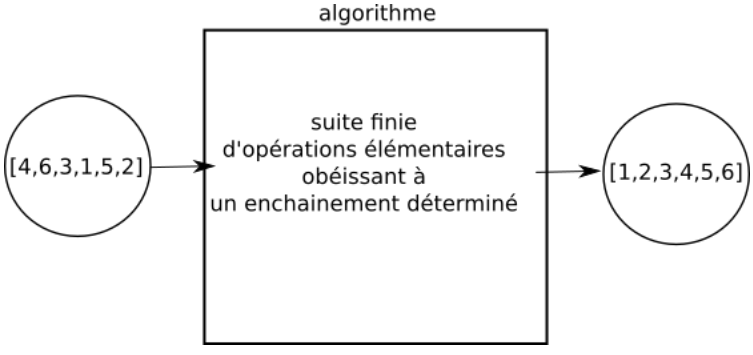
Pour concevoir un algorithme, la 2e définition nous précise qu'il est nécessaire de définir "une suite finie d'opérations élémentaires obéissant à un enchainement déterminé". L'ordre dans lequel nous allons définir nos "opérations élémentaires" va donc avoir son importance.

On pourrait résumer ce qu'est un algorithme par le schéma suivant :

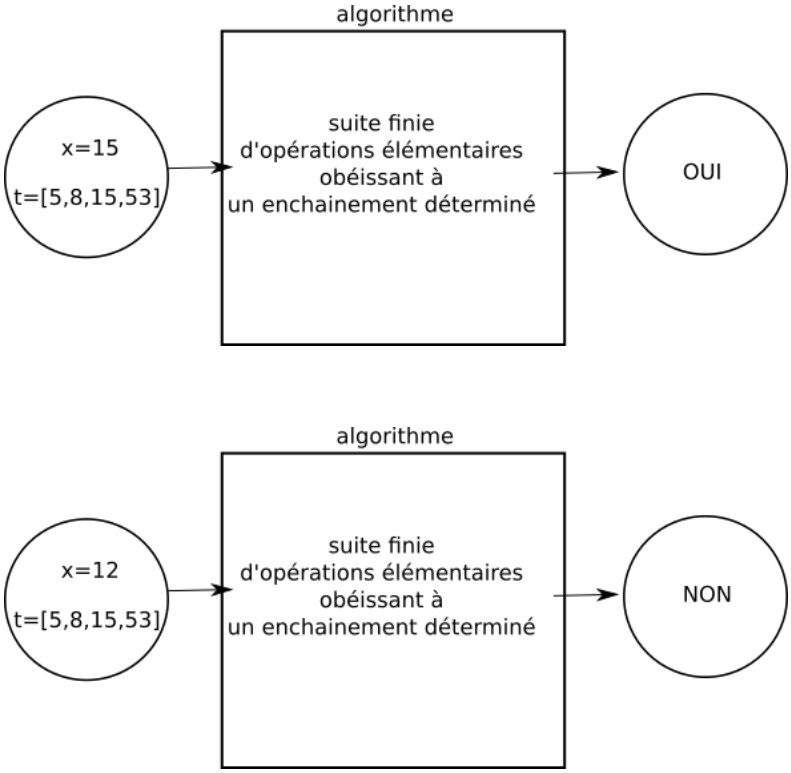


Prenons un exemple concret :

Nous allons étudier cette année, ainsi que l'année prochaine, des algorithmes de tri pour les tableaux (un tableau ressemble beaucoup à une liste en Python, même si ce n'est pas exactement la même chose). Nous avons en entrée un tableau non trié et nous obtenons en sortie un tableau trié :



Prenons l'exemple d'un algorithme qui prend en entrée un tableau t d'entiers et un entier x, et qui "répond" par "oui" ou par "non" à la question "x est-il présent dans le tableau t ?". Dans ce cas, la "valeur de sortie" sera "oui" ou "non".



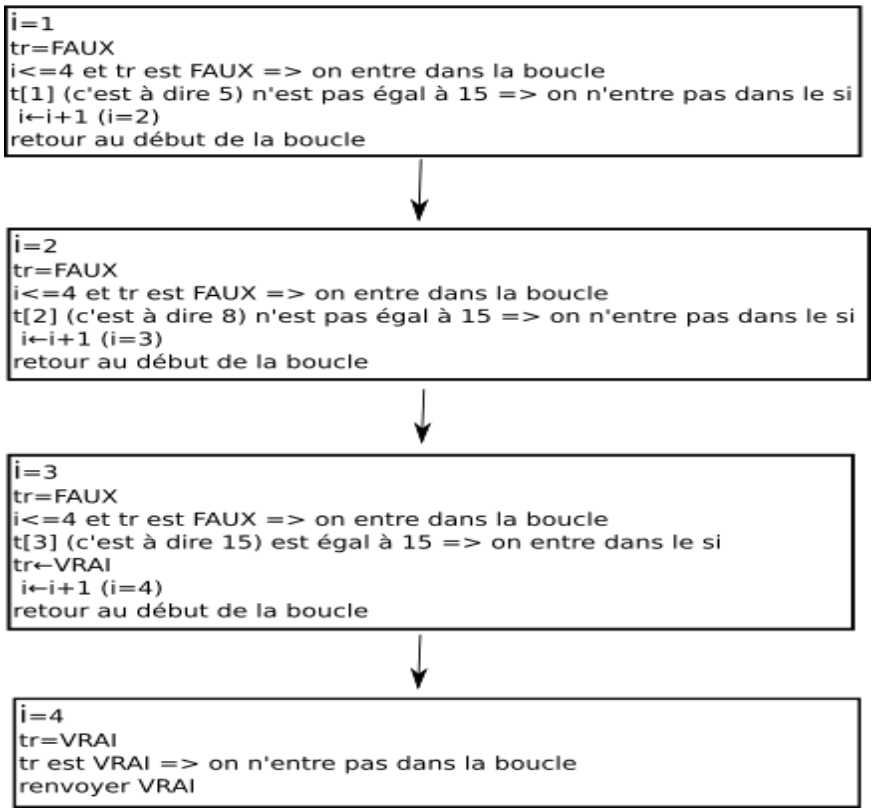
Essayons d'écrire l'algorithme décrit ci-dessus :

Nous devons trouver la suite d'opérations élémentaires qui permettra de répondre à la question : "x est-il présent dans le tableau t ?"

```
VARIABLE
t : tableau d'entiers
x : nombre entier
tr : booléen (VRAI ou FAUX)
i : nombre entier
DEBUT
tr ← FAUX
i ← 1
tant que i<=longueur(t) et que tr==FAUX:
  si t[i]==x:
    tr ← VRAI
  fin si
  i ← i+1
fin tant que
renvoyer la valeur de tr
FIN
```

La première chose à faire quand on étudie un algorithme, c'est de le "faire tourner à la main" : on "exécute" l'algorithme en utilisant uniquement une feuille et un crayon. Voilà ce que cela pourrait donner avec l'algorithme que nous venons d'écrire :

```
t = [5,8,15,23]
x= 15
```



*Créez un document word que vous nommerez « **algorithmes_NOM_Prénom** » dans lequel vous rédigerez les exercices suivants :*

Exercice 1 :

Faites "tourner à la main" l'algorithme "x est-il présent dans le tableau t ?" avec t=[5,8,15,53] et x=12

Exercice 2 :

Écrivez un algorithme permettant de trouver le plus grand entier présent dans un tableau. Vous ferez "tourner à la main" votre algorithme en utilisant le tableau t = [3,5,1,8,4,2].

Exercice 3 :

Écrivez un algorithme permettant de calculer la moyenne de tous les entiers présents dans un tableau. Vous ferez "tourner à la main" votre algorithme en utilisant le tableau t = [3,5,1,8,4,3].

Les algorithmes de tri

Les algorithmes de tri des éléments d'un tableau ont une place à part en algorithmique. En effet, ils sont souvent utilisés pour mettre en évidence certains concepts algorithmiques (concepts que l'on retrouve dans d'autres types d'algorithmes). Nous allons commencer par 2 algorithmes "classiques" : le tri par insertion et le tri par sélection.

Tri par insertion

Entrons tout de suite dans le vif du sujet, voici l'algorithme du tri par insertion :

```
VARIABLE
t : tableau d'entiers
i : nombre entier
j : nombre entier
k : nombre entier
DEBUT
j←2
tant que j<=longueur(t):   //boucle 1
  i←j-1
  k←t[j]
  tant que i>0 et que t[i]>k:   //boucle 2
    t[i+1]←t[i]
    i←i-1
  fin tant que
  t[i+1]←k
  j←j+1
fin tant que
FIN
```

Remarque : il est possible de mettre des commentaires à l'aide de "//" afin de rendre la compréhension des algorithmes plus aisée

Exercice 4 :

Poursuivez le travail commencé ci-dessous (attention de bien donner l'état du tableau à chaque étape)

t = [27, 10, 12, 8, 11]

j=2
j<longueur(t) => on "entre" dans la boucle 1
i←j-1 (i=1)
k←t[j] (k=10)
i>0 et t[1]>k => on "entre" dans la boucle 2
t[i+1]←t[i] (t[2]=27)=> t=[27, 27, 12, 8, 11]
i←i-1(i=0)
retour au début de la boucle 2

↓

j=2
i=0
k=10
i=0 => on n'entre pas dans la boucle 2
t[i+1]←k (t[1]=10) => t=[10, 27, 12, 8, 11]
j←j+1 (j=3)
retour au début de la boucle 1

↓

j=3
j<longueur(t) => on "entre" dans la boucle 1
i←j-1 (i=2)
k←t[j] (k=12)
i>0 et t[2]>k => on "entre" dans la boucle 2
t[i+1]←t[i] (t[3]=27)=> t=[10, 27, 27, 8, 11]
i←i-1 (i=1)
retour au début de la boucle 2

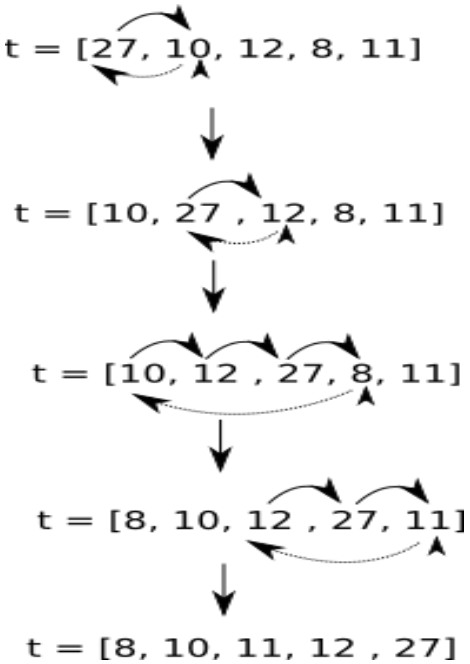
↓

j=3
i=1
k=12
t[1]<k => on n'entre pas dans la boucle 2
t[i+1]←k (t[2]=12)=> t=[10, 12, 27, 8, 11]
j←j+1 (j=4)
retour au début de la boucle 1

↓

.....

On peut résumer le principe de fonctionnement de l'algorithme de tri par insertion avec le schéma suivant :



Exercice 5 :

Essayez de produire le même type de schéma explicatif que ci-dessus avec le tableau t = [12, 8, 23, 10, 15]

Tri par sélection

```
VARIABLE
t : tableau d'entiers
i : nombre entier
min : nombre entier
j : nombre entier
DEBUT
i←1
tant que i<longueur(t):   //boucle 1
  j←i+1
  min←i
  tant que j<=longueur(t):   //boucle 2
    si t[j]<t[min]:
      min←j
    fin si
    j←j+1
  fin tant que
  si min≠i :
    échanger t[i] et t[min]
  fin si
  i←i+1
fin tant que
FIN
```

Exercice 6 :

Poursuivez le travail commencé ci-dessous (attention de bien donner l'état du tableau).

t=[12, 8, 23 ,10, 15]

i = 1
i < longueur(t) (1<5) ⟹ on entre dans la boucle 1
j = i+1 (j=2)
min = i (min = 1)
j <= longueur(t) (2<=5) ⟹ on entre dans la boucle 2
t[j] < t[min] (8<12) ⟹ on entre dans le si
min = j (min = 2)
j = j+1 (j = 3)
Retour au début de la boucle 2

t=[12, 8, 23 ,10, 15]

i = 1
j = 3
min = 2
j <= longueur(t) (3<=5) ⇒ on entre dans la boucle 2
t[j] > t[min] (23>8) ⇒ on n'entre pas dans le si
j = j+1 (j = 4)
Retour au début de la boucle 2

t=[12, 8, 23 ,10, 15]

i = 1
j = 4
min = 2
j <= longueur(t) (4<=5) ⇒ on entre dans la boucle 2
t[j] > t[min] (10>8) ⇒ on n'entre pas dans le si
j = j+1 (j = 5)
Retour au début de la boucle 2

t=[12, 8, 23 ,10, 15]

i = 1
j = 5
min = 2
j = longueur(t) (5=5) ⇒ on entre dans la boucle 2
t[j] > t[min] (15>8) ⇒ on n'entre pas dans le si
j = j+1 (j = 6)
Retour au début de la boucle 2

t=[12, 8, 23 ,10, 15]

i = 1
j = 6
min = 2
j > longueur(t) (6>5) ⇒ on n'entre pas dans la boucle 2
min ≠ i (2≠1) ⇒ on entre dans le si
échanger t[i] et t[min] (t[1] et t[2]) ⇒ **t = [8,12,23,10,15]**
i = i+1 (i=2)
Retour au début de la boucle 1

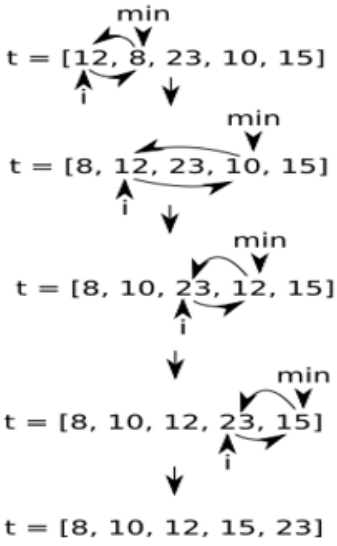
t=[8, 12, 23 ,10, 15]

i = 2
i < longueur(t) (2<5) ⟹ on entre dans la boucle 1
j = i+1 (j=3)
min = i (min =2)
j <= longueur(t) (3<5) ⟹ on entre dans la boucle 2
t[j]>t[min] (23>12) ⟹ on n'entre pas dans le si
j = j+1 (j=4)
Retour au début de la boucle 2

t=[8, 12, 23 ,10, 15]

i = 2
j = 4
min = 2
j < longueur(t) (4<5) ⟹ on entre dans la boucle 2
t[j]<t[min] (10<12) ⟹ on entre dans le si
min = j (min = 4)
j = j+1 (j=5)
Retour au début de la boucle 2

On peut résumer le principe de fonctionnement de l'algorithme de tri par sélection avec le schéma suivant :



Exercice 7 :

Essayez de produire le même type de schéma explicatif que ci-dessus avec le tableau t = [15, 16, 11, 13, 12]