Numérique et sciences informatiques

Guillaume Le Blanc Jean-Pierre Vallon

16 avril 2020

Table des matières

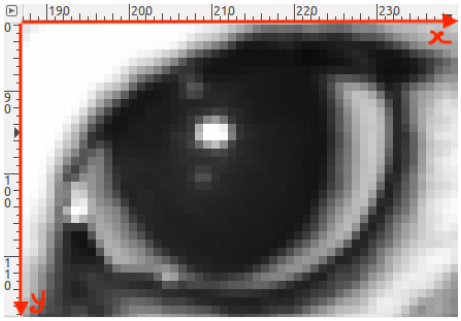
Chapitre 1

Représentation des données : types et valeurs de base

1.1 Des 0 et des 1

Les machines actuelles (ordinateurs, tablettes, smartphones..) traitent les données (images, sons, textes) sous la forme de bits 0 ou 1, parce que (pour simplifier) le 0 est associé à une tension électrique en-dessous d'un certain seuil et le 1 est associé à une tension électrique en dessus de ce seuil.

Par exemple une image en nuances de gris comme celle du chat ci-dessus est composée de pixels (picture element) visibles lorsqu'on a fait un zoom (voir ci-dessous)avec l'aide d'un éditeur d'images comme GIMP



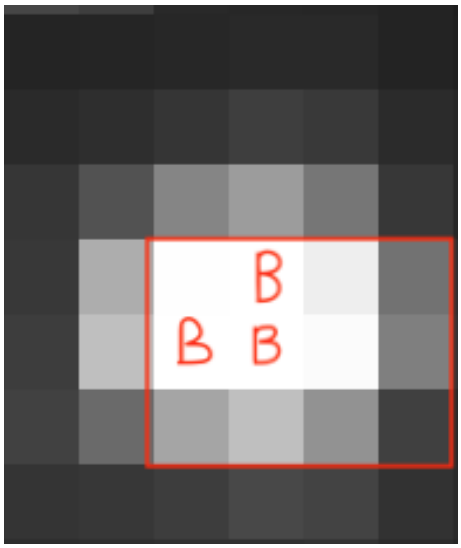
L'intensité de gris pour chaque pixel est un entier entre 0 et 255. 0 correspond au noir et 255 au blanc.

On observe la présence de pixels "blancs" à l'intérieur de l'oeil gauche du chat autour du pixel (209,95) (on repère les pixels suivant les axes du repère en rouge)

Nous pouvons observer cela en utilisant la bibliothèque PIL de Python , on fait afficher les intensités des pixels de coordonnées (209,94), (210,94), (209,95) et (210,95)

```
>>> from PIL import Image

>>> image = Image.open("/Users/vallon/Documents/chat.png")

>>> image.getpixel((209,94))
254

>>> image.getpixel((210,94))
255

>>> image.getpixel((209,95))
255

>>> image.getpixel((210,95))
255

>>> image.getpixel((207,91))
36
```

Les 3 pixels "blancs" sont visibles sur le zoom suivant :



Ensuite à l'aide d'un programme Python on fait afficher les intensités des pixels du rectangle rouge

```
from PIL import Image

#on charge l'image du chat
image1 = Image.open("/Users/vallon/Documents/chat.png")

for y in range(94,97):
    print()
    for x in range(209,213):

        print(image1.getpixel((x,y)),end=' ')
```

On obtient :

```
254 255 238 114
255 255 251 127
165 191 146 64
```

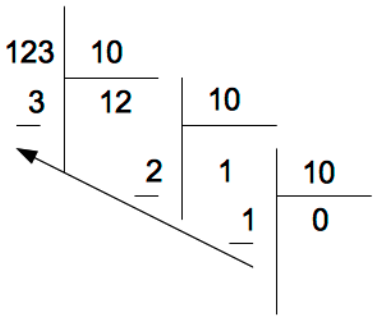
Ces nombres sont pour les humains, les machines travaillent plutôt avec des 0 et des 1

```
11111110 11111111 11101110 1110010
11111111 11111111 11111011 1111111
10100101 10111111 10010010 1000000
```

254 et 11111110 représente la même intensité 254 pour les humains et 11111110 pour les machines
Comment interpréter 11111110 ?

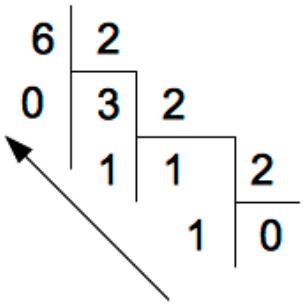
On écrit plutôt $(254)_{10} = (11111110)_2$ à la fois pour dire que c'est la même intensité mais aussi pour préciser la base avec laquelle on calcule
$(254)_{10} = 4 \times 10^0 + 5 \times 10^1 + 2 \times 10^2$
On dit que 4 est le chiffre des unités, 5 celui des dizaines 10^1 et 2 celui des centaines 10^2
Au lieu de procéder avec des puissances de 10 on va procéder avec des puissances de 2, cependant cette fois ci nous aurons comme chiffres que 0 et 1

Ce qui signifie que $(11111110)_2 = 0 \times 2^0 + 1 \times 2^1 + 1 \times 2^2 + 1 \times 2^3 + 1 \times 2^4 + 1 \times 2^5 + 1 \times 2^6 + 1 \times 2^7$
Coment obtient on les chiffres ?Par quel algorithme les obtient on ? On pose la division



euclidienne de 123 par 10 puis celle de 12 (le quotient de la division précédente) par 10 puis celle de 1 (le quotient de la division précédente) par 10 et puisque le quotient de la dernière division est nul on arrête l'algorithme
Les chiffres de la décomposition de 123 en base 10 sont les **restes**

1. Le premier reste obtenu est le chiffre des unités
2. Le dernier donne le nombre de la plus grande puissance de 10 contenue dans le nombre donné



Décomposons 6 maintenant en puissances de 2 On observe le même phénomène les chiffres sont les restes....
Voici l'algorithme en base $b \leqslant 10$

Algorithme 1 : décomposition d'un entier n en base b $\leqslant 10$

Données : Un entier naturel n, une base b $\leqslant 10$
Résultat : les chiffres de n à la volée dans l'ordre chiffre des unités vers chiffre

```
1  début
2      nombre ← n
3      base ← b
4      quotient ← nombre
5      tant que  quotient ≠ 0 faire
6          chiffre ← nombre % 10
7          afficher chiffre
8          quotient ← nombre // base
9          afficher reste
10     fin
11 fin
```

En Python

```
nombre = int(input("Entrez un entier naturel "))
base     = int(input("Entrez une base b <= 10 "))
quotient = nombre

while quotient > 0:
    chiffre = quotient \% base
    print(chiffre,end = " ")
    quotient = quotient // base
```

Il existe des outils pour "voir" les images numériques ou les programmes informatiques tel qu'ils sont dans la machine ou presque.
Presque car on préfère utiliser une expression des nombres, intermédiaire entre le binaire (machine) et le décimal(humain), appelée la notation hexadécimale (on verra pourquoi après)
Ainsi voici un morceau de l'image ci-dessus en hexadécimal :

```
00000000: 50 32 0A 23 20 43 52 45 41 54 4F 52 3A 20 47 49
00000010: 4D 50 20 50 4E 4D 20 46 69 6C 74 65 72 20 56 65
00000020: 72 73 69 6F 6E 20 31 2E 31 0A 33 32 30 20 32 34
00000030: 30 0A 32 35 35 0A 31 0A 31 0A 31 0A 31 0A 31 0A
00000040: 32 0A 32 0A 33 0A 33 0A 32 0A 32 0A 32 0A 32 0A
00000050: 31 0A 31 0A 31 0A 31 0A 32 0A 33 0A 33 0A 36 0A
00000060: 31 30 0A 31 36 0A 32 31 0A 34 30 0A 38 30 0A 31
00000070: 31 35 0A 31 32 31 0A 31 31 38 0A 31 32 30 0A 31
00000080: 32 31 0A 31 32 33 0A 31 32 34 0A 31 32 33 0A 31
00000090: 32 33 0A 31 32 39 0A 31 33 30 0A 31 32 38 0A 31
000000A0: 33 30 0A 31 33 38 0A 31 34 32 0A 31 33 36 0A 31
000000B0: 32 33 0A 31 32 35 0A 31 33 35 0A 31 33 39 0A 31
000000C0: 33 37 0A 31 34 32 0A 31 33 39 0A 31 33 36 0A 31
000000D0: 34 35 0A 31 34 32 0A 31 33 38 0A 31 34 30 0A 31
000000E0: 34 33 0A 31 34 31 0A 31 33 37 0A 31 34 30 0A 31
000000F0: 33 39 0A 31 32 37 0A 31 31 33 0A 39 39 0A 31 30

00000100: 30 0A 31 31 37 0A 31 32 35 0A 31 32 33 0A 31 32
00000110: 31 0A 31 33 33 0A 31 33 39 0A 31 33 34 0A 31 32
00000120: 30 0A 31 31 30 0A 31 31 34 0A 31 31 31 0A 31 31
00000130: 31 0A 31 30 31 0A 38 39 0A 31 31 31 0A 31 32 36
00000140: 0A 31 32 38 0A 31 33 38 0A 31 33 35 0A 31 33 31
00000150: 0A 31 32 38 0A 31 34 35 0A 31 37 30 0A 31 38 35
00000160: 0A 31 37 35 0A 31 35 37 0A 31 33 34 0A 31 31 32
00000170: 0A 39 35 0A 38 39 0A 31 31 32 0A 39 36 0A 34 35
00000180: 0A 33 33 0A 32 36 0A 32 32 0A 32 34 0A 31 37 0A
00000190: 39 0A 31 33 0A 32 39 0A 32 35 0A 31 35 0A 31 31
000001A0: 0A 36 0A 36 0A 36 0A 35 0A 36 0A 31 30 0A 39 0A
```

1.2 Représentation en base b $\leqslant$ 10

Théorème 1 *Tout entier naturel $n \geqslant 1$ peut se décomposer dans la base $2 \leqslant b \leqslant 10$ sous la forme $n = c_0 + c_1 \times b^1 + c_2 \times b^2 + ... + c_m \times b^m = (c_m...c_1c_0)_b$*

où $c_i \in \{0, 1, ..., b-1\}$ et m l'unique entier tel que $b^m \leqslant n < b^{m+1}$
L'écriture de n $(c_m...c_1c_0)_b$ comporte $m+1$ chiffres

Preuve

Soit $m \geqslant 1$ l'unique entier tel que $b^{m-1} \leqslant n < b^m$. On peut donc diviser n par b , $m-1$ fois

La division euclidienne de n par b donne $n = ba_1 + r_1$ avec $0 \leqslant r_1 < b$

On recommence et $a_1 = ba_2 + r_2$ avec $0 \leqslant r_2 < b$ et $a_2 < a_1$ et $n = b(ba_2 + r_2) + r_1 = b^2a_2 + r_2 \times b + r_1$

Soit $a_2 = 0$ et $n = r_2 \times b + r_1$ est l'écriture de n soit on recommence et $a_2 = ba_3 + r_3$ avec $0 \leqslant r_3 < b$ donc $n = b^3a_3 + r_3 \times b^2 + r_2 \times b + r_1$ et $a_3 < a_2 < a_1$

Et puisqu'on ne peut pas avoir dans $\mathbb{N}$ une suite d'entiers strictement décroissante forcément si on divise m fois on obtient $a_m = 0$

A la fin $n = r_m b^{m-1} + ... + r_3 \times b^2 + r_2 \times b + r_1$

Exemple

$13 = 2 \times 6 + 1$
$6 = 2 \times 3 + 0$
$3 = 2 \times 1 + 1$
$1 = 2 \times 0 + 1$
Donc $13 = (1101)_2$

1.2.1 Exercices

Ex 1

1. Décomposer en base 2 , 10, 15, 16, 32, 127, 255 et 2019
2. Que vaut $(1010)_2$ en base 10 ? Que vaut $(101010)_2$ puis $(10101010)_2$en base 10. Quelle est la logique ?

Ex 2

1. Calculer $(1010)_2 + (11)_2$ et vérifier la compatibilité avec la même opération en base 10
2. Calculer $(1110)_2 + (11)_2$ et vérifier la compatibilité avec la même opération en base 10

Ex 3

Modifier l'algorithme de décomposition de telle sorte qu'on fasse l'économie de la dernière division

Ex 4

Décomposer en base 8 : 16, 31,64 et 2019

Ex 5

Pourquoi les informaticiens confondent Noël et Halloween ?
Parce que Dec 25 = Oct 31

Ex 6

Classer par ordre croissant de capacité :
2000 octets, 15 Ko, 4 Go, 1,5 Mo, 1 To, 2 millions de bits, 2 000 000 bytes

Ex 7

Il n'y a pas que des ordinateurs qui "travaillent" avec des 0 et des 1 mais aussi des machines analogiques
Qu'est qu'une machine *analogique* ?

Ex 8

Il n' y a que 10 types de personnes, ceux qui comprennent le binaire et ceux qui ne le comprennent pas

1.3 Représentation en base 16

Un **octet** est composé de 8 bits

$(c_7c_6c_5c_4c_3c_2c_1c_0)_2 = c_7\times2^7+c_6\times2^6+...+c_3\times2^3+...+c_0 = 2^4(\underline{c_7 \times 2^3 + c_6 \times 2^2 + ... + c_4})+ \underline{c_3 \times 2^3 + ... + c_0}$

Or les quantités soulignées sont comprises entre 0 et 15 d'où le théorème

Théorème 2 *Tout octet est traduit avec 2 symboles en base 16 où les chiffres sont 0 jusqu'à 9 puis A,B,C,D,E et F*
Avec $A_{16} = 10_{10}$, $B_{16} = 11_{10}$, $C_{16} = 12_{10}$, $D_{16} = 13_{10}$, $E_{16} = 14_{10}$ *et* $F_{16} = 15_{10}$

Exemple

$254 = (1111\ 1110)_2 = (FE)_{16} = 0$xfe

1.3.1 Exercices

Ex 1

Convertir les entiers suivants en hexadécimal : 9,10,15,16,31 ,32, 255,256 et 2020

Ex 2

Que vaut 0xfff en base 10 ?

Ex 3

If only DEAD people understand hexadecimal, how many people understand hexadecimal ?

Ex 4

1. Voici une série d'octets séparés par / en binaire les traduire en hexadécimal : 1000 0011 / 0001 1010/1111 1111/1111 1100/0101 1010
2. Voici une série d'octets séparés par / en hexadécimal les traduire en binaire : 0xff / 0xaa / 0xbe/ 0x1a/0xa1

1.4 Représentation binaire d'un entier relatif

Comment représenter les entiers négatifs ?

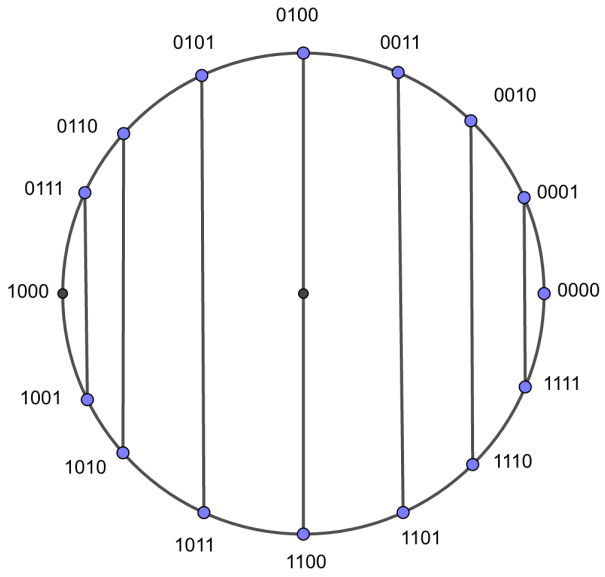
1. *Une première approche* : Par exemple sur un octet on va coder 4 par 0000 0100 et pour coder -4 on va utiliser **un bit significatif** celui qui est le "plus à gauche" dit **bit de poids fort** que l'on va mettre à 1 pour dire que ce nombre est négatif et on ne change pas les autres bits ce qui donne
   $-4 = (10000100)_2$
   Calculons $-4 + 4 = (10001000)_2$ qui vaut -8 avec notre convention au lieu de 0 ce qui est un problème !
2. *Une deuxième approche* Supposons que les entiers sont codés uniquement sur 4 bits et plaçons les mots de 4 bits dans l'ordre croissant de 0000 jusqu'à 1111 sur un cercle de la manière suivante : on apparie ensemble les mots de telle sorte que la somme des deux mots donne toujours 1 0000 que l'on va assimiler à 0
   On a donc apparié les entiers positifs et leur opposé on choisit pour positifs les mots de 4 bits **dont le bit de poids fort est 0** et les négatifs ceux dont le bit de poids fort est 1
   Ainsi 2 est codé par 0010 et -2 par 1110

Théorème 3 *(Complément à 2) Si les entiers relatifs sont codés sur n bits alors :*

1. *Il y a $2^{n-1} - 1$ mots ayant un bit de poids fort égale à 0, ils servent à coder les entiers positifs*
2. *Les mots précédents ont un opposé obtenu en remplaçant chaque chiffre par son complément et en ajoutant 1*

Preuve

Soit $m = c_n c_{n-1}....c_0$ avec $c_i \in \{0, 1\}$ et $c_n = 0$ et où $\overline{c_i}$ le complémentaire de c_i
Considérons $\overline{m} = \overline{c_n}\overline{c_{n-1}}....\overline{c_0}$ donc $m + \overline{m} = 11....11$ (que des 1)
donc $m + \overline{m} + 1 = m + (\overline{m} + 1) = 1\ 00...00$ que l'on assimile à 0, et l'opposé de m et $\overline{m} + 1$



Exemple

Sur 16 bits 2019 est codé par 0000 0111 1110 0011 donc pour trouver le code de son opposé on prend le complémentaire 1111 1000 0001 1100 puis on ajoute 1 ce qui donne pour le codage de -2019 –> 1111 1000 0001 1101

1.4.1 Exercices

Ex 1

Sur 8 bits coder -1 et -4

Ex 2

Sur 16 bits coder -15,-16,-31 et -32

Ex 3

Sur 32 bits coder -7 et -8

Ex 4

1. Les systèmes d'exploitation actuels codent les entiers sur 64 bits. Quel est le plus grand entier naturel représentable en machine ?
2. Combien peut-on coder de nombres négatifs sur 64 bits ?

1.5 Représentation approximative d'un nombre réel

Les vecteurs suivants $\overrightarrow{u} = \begin{pmatrix} 0,1 \\ 0,3 \end{pmatrix}$ et $\overrightarrow{v} = \begin{pmatrix} 1 \\ 3 \end{pmatrix}$ sont colinéaires en effet $\overrightarrow{u} = 0,1 \times \overrightarrow{v}$

La fonction suivante devrait retourner la valeur Vraie or ce n'est pas le cas !!!
`print(colineaires(0.1,0.3,1,3))` retourne False

```
def determinant(xU,yU,xV,yV):
    return xU*yV - yU*xV

def colineaires(xU,yU,xV,yV):
    return determinant(xU,yU,xV,yV) == 0
```

```
print(colineaires(0.1,0.3,1,3))
```

Que s'est il passé ?

1. Les nombres ont une représentation **finie** en machine !

Avec les puissances de 10 on a les nombres **décimaux** par exemple $1,75 = 1 + 7 \times 10^{-1} + 5 \times 10^{-2}$

Avec les puissances de 2 on a les nombres **dyadiques**par exemple $1,75 = 1 + 1 \times 2^{-1} + 1 \times 2^{-2} = 1 + \frac{1}{2} + \frac{1}{4}$

On va donc noter $1,75 = (1,11)_2$

Les " premières" puissances de $\frac{1}{2}$ sont dans le tableau suivant :

$(\frac{1}{2})^n = 5^n \times 10^{-n}$

n	1	2	3	4	5	6	7	8
$(\frac{1}{2})^n$	0,5	0,25	0,125	0,0625	0,03125	0,015625	0,0078125	0,00390625

Problème : Tout nombre réel est-il dyadique ?

Théorème 4 *Il existe des nombres réels qui ne sont pas dyadiques, par exemple 0,1*

Preuve Calculons la représentation dyadique de 0,1 :

On cherche n entier naturel tel que $(\frac{1}{2})^n \leqslant 0,1 \leqslant (\frac{1}{2})^{n-1}$

On trouve $n = 4$

Ensuite on calcule 0,1-0,0625 = 0,0375 et on recommence avec 0,0375

Maintenant 0,0375 = 0,03125 + 0,00625 donc pour l'instant $0,1 = (\frac{1}{2})^4 + (\frac{1}{2})^5 + 0,00625$

On continue avec 0,00625, or 0,00625 = 0,00390625 + 0,00234375

On continue et on obtient que 0,1 a **un développement dyadique illimité** et puisqu'une machine est finie la représentation en machine de 0,1 est **tronquée**

$0,1 = (0,0001100110011001100.....)_2$

Retenir

Les calculs sur les nombres flottants ne sont parfois pas exacts

Il ne faut JAMAIS tester une égalité entre deux nombre flottants mais utiliser une marge d'erreur relative.

On corrige donc la fonction `sontColineaires(xU,yU,xV,yV)`

1.5.1 Exercices

Ex 1

Faire un programme Python qui affiche les 20 premières puissances de $\frac{1}{2}$

Ex 2

Quel est le nombre décimal représenté par $(1010,1010)_2$

Ex 3

Donner la représentation dyadique du nombre décimal $3,1415$

Ex 4

```
>>> def determinant(xU,yU,xV,yV):
        return xU*yV - yU*xV

>>> def sontColineaires(xU,yU,xV,yV):
        return determinant(xU,yU,xV,yV) == 0

>>> print(sontColineaires(0.1,0.3,1,3))
False
>>> print(determinant(0.1,0.3,1,3))
5.551115123125783e-17
```

```
>>> def sontColineaires(xU,yU,xV,yV):
        return abs(determinant(xU,yU,xV,yV)) < 6*10**(-17)

>>> print(sontColineaires(0.1,0.3,1,3))
True
>>> print(sontColineaires(0.1,0.3,1.000000000001,3))
False
```

Existe-t-il des nombres dyadiques qui ne sont pas décimaux ?

Ex 5

Lire le document `https://docs.python.org/fr/3.7/tutorial/floatingpoint.html`

Bilan : Codage des nombres

Ex 1

1. Coder les puissances de 2 en binaire de 2^1 jusqu'à 2^{10}
2. On appelle nombre de Mersenne un nombre de la forme $M_n = 2^n - 1$
   Coder M_n en binaire

Ex 2

Convertir les expressions suivantes du binaire au décimal

1. $(10010010)_2$
2. $(11001101)_2$
3. $(11100010)_2$

Ex 3

Coder en hexadécimal les nombres suivants

1. les puissances de 2
2. les nombres de Mersenne
3. votre date de naissance

Ex 4

Coder les couleurs RVB suivantes en couleurs html

1. (127 ;127 ;127)
2. (255 ;63 ;10)
3. (255,255,255)

Ex 5

Coder les couleurs html en RVB

1. aabbcc
2. 00ff00
3. 999a9b

Ex 6

Coder en binaire sur 8 bits (complément à 2)

1. -1
2. -4
3. -15

Ex 7

Les codes suivants sont ceux de nombres négatifs codés sur un octet. Quels sont ils ?

1. $(1100\ 0011)_2$
2. $0010\ 0011)_2$
3. $(1000\ 0010)_2$

Ex 8

Que vaut $(1, 11)_2$?, $(1, 1101)_2$?

1.6 Expressions booléennes

Une expression booléenne (vient de **George Boole**, mathématicien britannique du XIX ^{e} siècle) est une expression pouvant être vraie ou fausse mais pas les deux
Par exemple l'expression $1 < 2$ est évaluée à vraie (True) dans l'interpréteur Python alors que $1 > 2$ est évaluée à fausse (False)
Dans la suite, pour des raisons de simplicité on associe la valeur 1 à la valeur Vraie

```
>>> 1 < 2
True
>>> 1 > 2
False
```

(True) et la valeur 0 à la valeur Faux (false)
Les **conditions d'arrêt** ou les test sont des expressions booléennes parfois complexes

```
>>> type(1)
<class 'int'>
>>> type(True)
<class 'bool'>
>>> 1 == True
True
>>> 0 == False
True
```

car composées d'autres expressions booléennes reliées entre elles par des **opérateurs booléens** ou **connecteurs booléens** (Voir exercice) Un **opérateur booléen** permet de connecter deux expressions booléennes pour former une nouvelle expression booléenne
Nous utiliserons principalement les opérateurs **et**, **ou** puis **non**

Voici la table de vérité de l'opérateur **et**

a	b	a et b
0	0	0
0	1	0
1	0	0
1	1	1

On retiendra que **a et b n'est vrai que lorsque a et b sont vrais** de ce fait le langage Python n'évalue pas l'expression b si a est fausse

Voici la table de vérité de l'opérateur **ou**

a	b	a ou b
0	0	0
0	1	1
1	0	1
1	1	1

On retiendra que a et b n'est faux que lorsque a et b sont faux de ce fait le langage Python n'évalue pas l'expression b si a est vraie

Voici la table de vérité de l'opérateur **non**

a	non a
0	1
1	0

On verra en exercices qu'il existe 16 connecteurs logiques pour deux expressions booléennes et que tous ces connecteurs s'expriment en fonction des connecteurs `non`, `et`, `ou`
Ces connecteurs apparaissent donc comme des connecteurs fondamentaux

1.6.1 Exercices

Ex 1

1. Combien de connecteurs logiques à 2 places ?
2. Combien de connecteurs logiques à 3 places ?
3. Combien de connecteurs logiques à n places ?

Ex 2

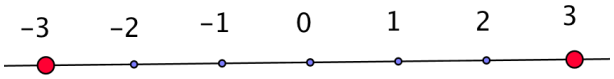
1. Montrer que `non(a et b)` a la même table de vérité que `non(a) ou non(b)`
2. En déduire que le connecteur logique `et` s'exprime en fonction des connecteurs `non` et `ou`
3. Montrer que `non(a ou b)` a la même table de vérité que `non(a) et non(b)`
4. Montrer que le connecteur logique `ou` s'exprime en fonction des connecteurs `non` et `et`

Ex 3

Un opérateur logique très utilisé est le ou exclusif ou xor (en anglais) défini comme le `ou` à la différence que `xor(1,1) = 0`
Exprimer `xor` en fonction de `et`, `ou` et `non`

Ex 4

Un jeton part de l'origine et se déplace aléatoirement ainsi : Si le lancer d'une pièce a donné **Pile** alors le jeton se déplace d'une unité vers la droite sinon il se déplace d'une unité vers la gauche et le déplacement s'arrête lorsque l'abscisse du jeton est 3 ou -3. Il s'agit de compter le nombre moyen de lancers de la pièce par déplacement
Compléter l'algorithme suivant



Algorithme 2 : Déplacement du jeton

Données : Un jeton à l'origine, un intervalle gradué [-3 ;3] et un générateur de nombres aléatoires
Résultat : Le nombre de lancers de la pièce

```
1 début
2 |  position ← 0
3 |  nbLancers ← 0
4 |  tant que ..... faire
5 |  |  piece ← entierAleatoire(0,1)
6 |  |  .....
7 |  fin
8 |  afficher(nbLancers)
9 fin
```

1.7 Représentation d'un texte en machine

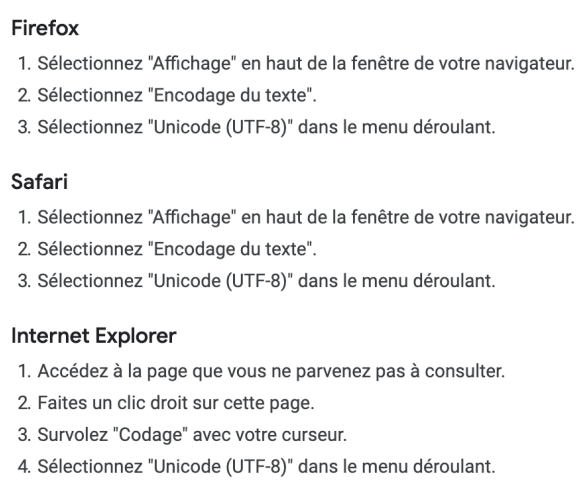
Lire le texte en anglais ici `https://docs.python.org/fr/3.6/howto/unicode.html` concernant l'histoire du codage des caractères et répondre aux questions suivantes :

1. En quelle année ASCII a été standardisé ?
2. Quels sont les points faibles de ASCII ?
3. Combien de bits un code ASCII ?
4. Quelle plage pour A ; ;B et a..b ?
5. Coder "Bonjour"
6. Coder "bonjour il fait beau"
7. Combien de bits un code Unicode ?
8. Qu'est qu'un glyphe ? Décrire celui de A
9. Que signifie cette phrase ? "Most Python code doesn't need to worry about glyphs ; figuring out the correct glyph to display is generally the job of a GUI toolkit or a terminal's font renderer."
10. Qu'est ce que Latin-1 ? et UTF-8 ?

1.7.1 Exercices

Ex 1

Exercice à faire avec les navigateurs (Firefox,Safari et Internet Explorer)



Firefox

1. Sélectionnez "Affichage" en haut de la fenêtre de votre navigateur.
2. Sélectionnez "Encodage du texte".
3. Sélectionnez "Unicode (UTF-8)" dans le menu déroulant.

Safari

1. Sélectionnez "Affichage" en haut de la fenêtre de votre navigateur.
2. Sélectionnez "Encodage du texte".
3. Sélectionnez "Unicode (UTF-8)" dans le menu déroulant.

Internet Explorer

1. Accédez à la page que vous ne parvenez pas à consulter.
2. Faites un clic droit sur cette page.
3. Survolez "Codage" avec votre curseur.
4. Sélectionnez "Unicode (UTF-8)" dans le menu déroulant.

Changer le paramétrage d'encodage du navigateur et sélectionner Iso Latin 1 comme encodage au lieu de utf-8, puis télécharger le fichier texte suivant contenant des caractères accentués puis ouvrir ce fichier avec un éditeur de texte. Qu'observez vous ?

Ex 2

Aller ici `https://www.labri.fr/perso/betrema/verlaine/automne.html`. Comment expliquez vous que les caractères accentués sont remplacés par des points d'interrogation ?

(Indication : Si vous faites afficher le code source de la page `https://www.labri.fr/`

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

`perso/betrema/verlaine.html` vous obtenez ceci :
maintenant si vous réglez votre navigateur en iso Latin 1 et rechargez la page vous verrez les caractères accentués !. Conclure)

Chapitre 2

Représentation des données : types construits

Pour une image couleur en mode "RGB" c'est à dire R pour rouge, G pour Green,(Vert) et B pour bleu, à chaque pixel est associé un tuple ou n-uplet donnant la couleur du pixel sous la forme d'un triplet de nombres entiers, les intensités de Rouge, de Vert et de Bleu Par exemple le pixel repéré par le tuple (150,50) a pour couleur (205,102,103)

```
>>> from PIL import Image

>>> image = Image.open("/Users/vallon/Documents/lena.png")

>>> image.mode
'RGB'

>>> image.getpixel((150,50))
(205, 102, 103)
```

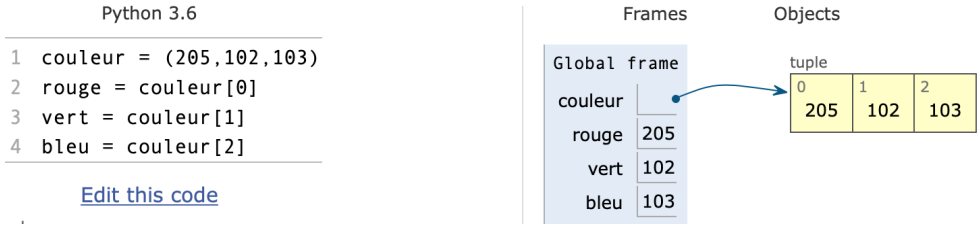
On observe donc que les informations :

1. position d'un pixel, par exemple (150,50)
2. taille de l'image ici (512,512)
3. couleur d'un pixel par exemple (205,102,103)

sont représentés en mémoire par une collection de plusieurs entiers appelée tuple

```
>>> image.size
(512, 512)

>>> taille = image.size

>>> type(taille)
<class 'tuple'>
```

2.1 Tuples



On va utiliser PythonTutor pour visualiser les propriétés des tuples.

1. Dans un premier temps (voir figure ci-dessus) on observe "une petite flèche" qui associe la variable **couleur** au triplet de nombres en mémoire (205,102,103).
   Cette "flèche" signifie *qu'à la variable couleur est associée non pas la valeur (205,102,103)* mais **une référence** c'est à dire concrètement une adresse en mémoire qui permettra de retrouver en mémoire *la valeur* du tuple (205,102,103).

   Il est important de bien comprendre cette association **variable** en tant qu'étiquette et **référence** en tant qu'adresse mémoire permettant de retrouver la **valeur** en mémoire
2. On peut avoir accès aux composantes d'un tuple, ainsi pour avoir la composante en rouge de la couleur il suffit de faire `couleur[0]` (voir ci-dessus)
3. Mais on ne peut pas changer ces valeurs on dit que le type tuple est **immuable** L'instruction `couleur[0] = 255` conduirait à une erreur
4. On veut écrire une fonction en Python qui prend en paramètres les anciennes coordonnées (x,y) d'un point et qui retourne les nouvelles coordonnées milieu de (x,y) et de (a,b) un point fixe du plan
   Une possiblité est :

```
def milieu(x,y):
    nouvellePosition = ((a + x)/2 , (b + y)/2)
    return nouvellePosition
```

   Mais en Python on peut ne pas mettre des parenthèses l'interpréteur saura interpréter un tuple

```
def milieu(x,y):
    nouvellePosition = (a + x)/2 , (b + y)/2
    return nouvellePosition
```

que l'on peut encore simplifier

```
def milieu(x,y):
    return    (a + x)/2 , (b + y)/2
```

2.2 Tableaux

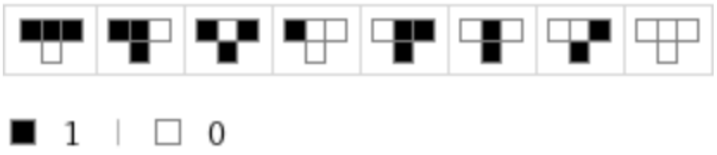
Au stade suivant **on veut à la fois avoir une collection d'entiers référencé par une seule variable mais aussi pouvoir modifier chacune des valeurs**

Ce type de structure de données est appelé généralement **tableau** en Python on appelle cette structure une **liste**

Pour bien comprendre l'intérêt des listes nous allons étudier un automate cellulaire à une dimension.

Pour commencer lire et comprendre l'article suivant `https://fr.wikipedia.org/wiki/Automate_cellulaire` (seulement la partie "les automates cellulaires les plus simples")

Nous allons nous intéresser à l'automate suivant la règle 110 schématisée ci-dessous

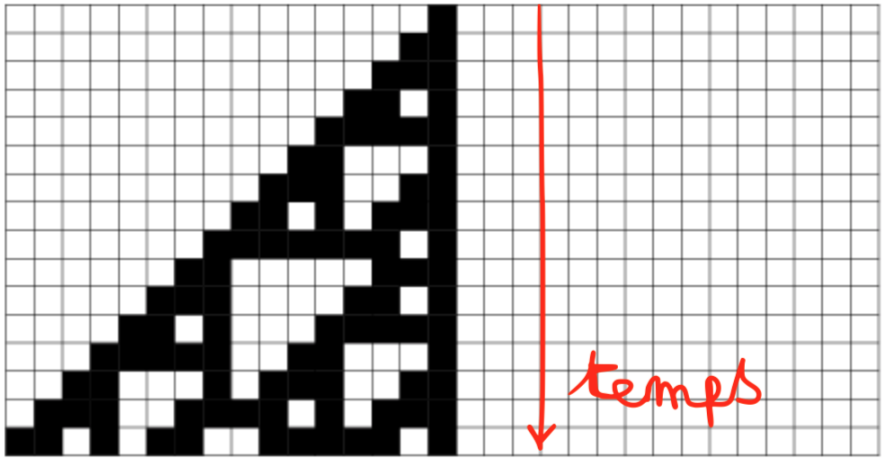


La règle s'appelle règle 110 car un état d'une cellule et de ses deux voisins correspond à un mot écrit avec trois bits donc 8 possibilités et par conséquent définir l'état futur d'une cellule correspond à énumérer toutes les possibilités sur un octet donc 256 possibilités

Voir aussi l'animation dans l'article Wikipedia suivant `https://en.wikipedia.org/wiki/Rule_110`

L'état initial de la colonie est la liste ne contenant que des 0 sauf en son "milieu"

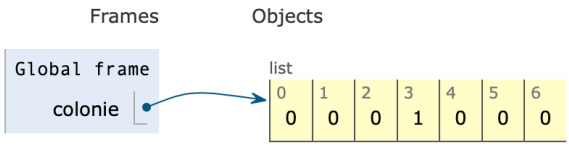
Voici l'évolution de la colonie "au cours du temps" au bout de 15 itérations



1. **Initialisation :** On crée une liste `colonie` ne contenant que trente 0 et un 1 au "milieu"
   Comment ? S'il n' y avait eu que 7 éléments on aurait pu le faire **en extension** ainsi

```
colonie = [0,0,0,1,0,0,0]
```

   et on aurait eu



2. Mais il y a 31 éléments on risque de se tromper en procédant par extension, alors il faut procéder de manière "logique"
   Soit ainsi :

```
colonie = [] //liste vide
//on ajoute à la liste trente et un 0
for i in range(31):
    colonie.append(0)
// on affecte au seizième champ noté 15, la valeur 1
colonie[15] = 1
```

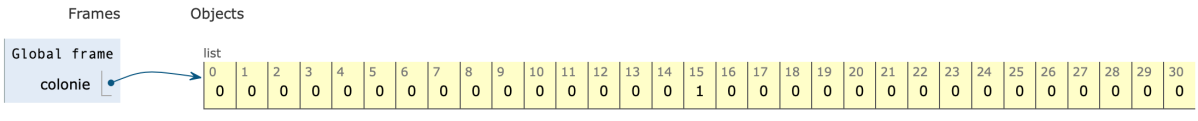
   ou encore

```
 //la liste colonie contient trente et un 0 par compréhension
colonie = [0 for i in range(31)]
// on affecte au seizième champ noté 15, la valeur 1
colonie[15] = 1
```

   ou encore

```
  //la liste colonie contient trente et un 0 par compréhension
colonie = [0] *31
// on affecte au seizième champ noté 15, la valeur 1
colonie[15] = 1
```

   on obtient alors ceci



3. **Itération**Comment créer l'état de la colonie à l'instant t + 1 à partir de l'état de la colonie à l'instant t ?
   Puisque l'état d'une cellule à l'instant t + 1 est déterminée par son état à l'instant

t ainsi que l'état de ses deux voisins à l'instant t on ne peut pas créer le nouvel état de la ccolonie uniquement avec la seule variable `colonie`
Il nous faut une deuxième variable `temp` de type liste et ayant autant de cellules que la variable `colonie` et on va procéder ainsi

Algorithme 3 : Automate cellulaire 110

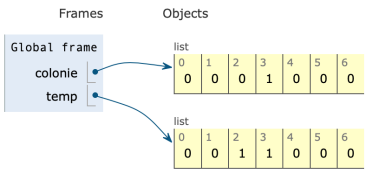
Données : La liste colonie et un nombre n
Résultat : Un entier i, la première position de la valeur dans le tableau

```
1 début
2 |  dessiner(colonie)
3 |  Répéter n fois
4 |    Pour chaque élément de colonie faire
5 |      Construire l'état suivant de élément dans temp en appliquant les règles
6 |    colonie ← temp
7 |    dessiner(colonie)
8 fin
```

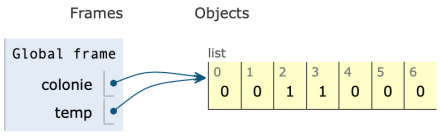
4.

5. Ce qui donne en Python

```
colonie = [0]*31
colonie[15] = 1
dessiner(colonie)
for i in range(15):
    temp = [0]*31
    for j in range(1,30):
        temp[j] = regle(colonie[j-1],colonie[j],colonie[j+1])
    colonie = temp
    dessiner(colonie)
```

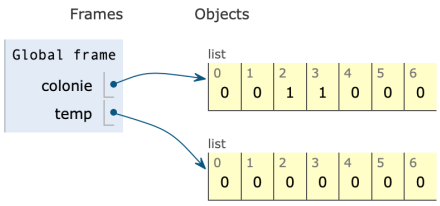


6. Voici une image de ce qui se passe à chaque tour de boucle
La variable `colonie` est associée à une référence dont la valeur contient l'état actuel de l'automate
La variable `temp` est associée à une référence dont la valeur contient l'état futur de l'automate



7. L'instruction `colonie = temp` fait que les variables `colonie` et `temp` pointent vers la même référence dont la valeur est l'état futur de la colonie

Pour faire une nouvelle génération il suffit de recréer une liste `temp` ne contenant que des 0 puis de se servir de `temp` pour construire la nouvelle génération



8. La fonction regle peut être vue comme une expression booléenne dépendant de a = colonie[j-1] , b = colonie[j] et c = colonie[j+1]
   On vérifiera en exercice que
   regle(a,b,c) est équivalente à (a et xor(b,c) ou (non a et (b ou c))

2.2.1 Exercices : Tuples - Listes

Ex 1

Ecrire une fonction `minMax(tuple)` en Python qui prend en paramètre un tuple de nombres et qui retourne un tuple composé en premier du plus petit de ces nombres et en second du plus grand de ces nombres
Le tuple contient au moins un nombre
Par exemple `minMax(8,12,30.6)` retourne le tuple (8,30.6)

Ex 2

Ecrire une fonction `moyEtendue(tuple)` en Python qui prend en paramètre un tuple de nombres et qui retourne un tuple composé de la moyenne des nombres et l'étendue des nombres
Le tuple contient au moins un nombre
Rappel : L'étendue est la différence entre le plus grand et le plus petit des nombres
Par exemple `moyEtendue(5,10,23.8,41.2)` retourne le tuple (20,36.2)

Ex 3

Définir une fonction python `negation(liste)` où liste est une liste de 0 et de 1 et negation(liste) retourne la liste où les 1 sont changés en 0 et les 0 en 1

Par exemple `negation([1,0,0,1,0,0])` retourne la liste [0,1,1,0,1,1]

Ex 4 : opérateur &

Le & (lire et) correspond à la fonction logique `et` mais appliqué aux représentations des nombres en binaire, bit par bit.
Ainsi par exemple 110 & 100 = 100
Tester cette fonction à la console puis définir une fonction `et(liste1,liste2)` où liste1 et liste2 sont deux listes de même longueur contenant que des 0 et des 1 et retournant la liste résultat de liste 1 & liste2

Ex 5

Définir une fonction Python `decToBin(n)` qui retourne une liste L de 0 et de 1 et telle L[i] est le chiffre de rang i de l'écriture de n en base 2

Ex 6

Définir une fonction Python `binToDec(l)` qui retourne un entier n en base 10 à partir d'une liste l de 0 et de 1 et telle l[i] est le chiffre de rang i de l'écriture de n en base 2 (Utiliser la méthode de Horner)
(la méthode de Horner consiste par exemple pour calculer $4x^3+3x^2+2x+1$ à disposer le calcul ainsi $((4\times x+3)\times x+2)\times x+1$
Sur cet exemple sans la méthode de Horner il y a 6 multiplications, avec 3 multiplications

Ex 7

Voici un programme en python

```
liste1 = [1,1]
liste2 = [0,0]
liste1[0] = 2
liste1 = liste2
print(liste1[0])
```

Qu'observe-t-on à l'écran ? (Justifier)
A :) 2 B) : 1 C) : 0

Ex 8

Voici un programme en python

```
liste1 = [0,0,0]
liste2 = [1,1,1]
liste3 = [liste1,liste2]
liste2 = liste1
print(liste3)
```

Qu'observe-t-on à l'écran ? (Justifier)

A :) [liste1,liste2] B) : [[0,0,0],[1,1,1]] C) : [[0,0,0],[0,0,0]]

Ex 9

Etant donné une liste de points du plan par exemple `[(0,1),(1,2),(1,1),(1.2,3)]` il s'agit de trouver le point le point le "plus haut et le plus à droite possible" dans le plan

On peut comparer les tuples avec l'ordre du dictionnaire ainsi `(0,1) < (1,2)` est vrai ainsi que `(1,1) < (1,2)`

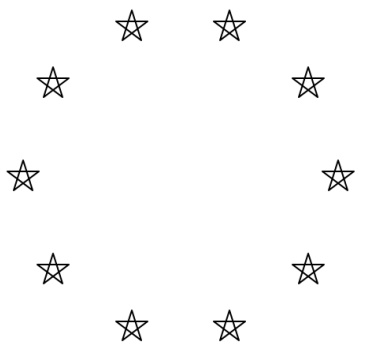
Définir une fonction `pointSupDroit(liste)` où liste est une liste de tuples correspondant à des coordonnées de points dans le plan

Ex 10

Engendrer par compréhension :

1. une liste de 100 booléens False
2. une liste de 100 caractères "#"
3. la liste des cubes des 10 premiers entiers

Ex 11

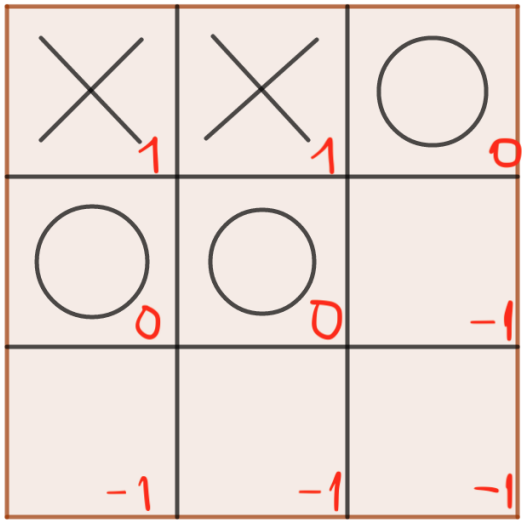


Il s'agit de faire dessiner par la Tortue dix étoiles disposées en cercle

1. Créer une liste angles en compréhension de $\frac{2k\pi}{10}$ pour k variant de 0 jusqu'à 9
2. Créer une liste de tuples en compréhension $(100\cos(a), 100\sin(a))$ pour a variant dans la liste angles
3. Finir en itérant sur la liste de tuples et dessiner une étoile

2.3 Tableaux à 2 dimensions

On veut simuler le jeu du Tic Tac Toe. Il faut mémoriser l'état du jeu par un tableau 2D. Que peut être un tableau 2D ? On associe 0 au rond O , 1 à la croix X et -1 pour

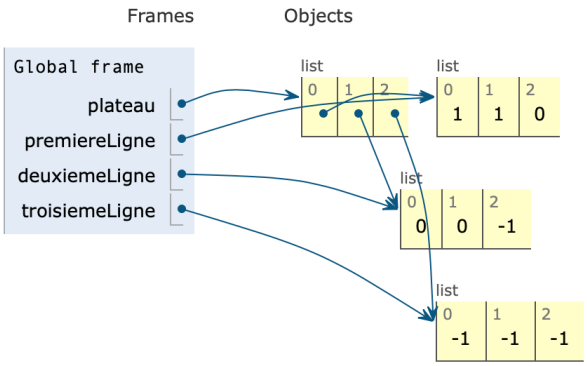


tac toe.png

une case non jouée. Une façon de faire est de "découper" le tableau ci-dessus en lignes et le tableau sera en mémoire sous la forme d'une liste de liste (en extension)

```
plateau = [[1,1,0],
           [0,0,-1],
           [-1,-1,-1]]
premiereLigne = plateau[0]
deuxiemeLigne = plateau[1]
troisiemeLigne = plateau[2]
```

Regardons avec Python tutor :



Si le joueur qui joue avec les X veut jouer sur la deuxième ligne et la troisième colonne, il va sélectionner la ligne puis la colonne, ce qui donne

```
plateau[1][2] = 1
```

2.3.1 Exercices : Tableaux 2D

Ex 1

Construire le tableau 2D 10 x 5 des entiers pairs de 2 à 100 **en compréhension**

Ex 2

1. Ecrire **en extension** un tableau 2D de 8 lignes et 4 colonnes pour la règle 110 de l'automate cellulaire
   `regle = [[False,False,False,False],`
   `[ [False,False,True,True],...]`
   Les quatrièmes éléments sont les résultats de la règle sur les trois premiers éléments
2. Définir une fonction `etatSuivant(numCellule,colonie,regle)` qui retourne l'état futur de la cellule `colonie[numCellule]` en suivant la règle dans le tableau `regle`

Ex 3

Voici un programme en Python :

```
regles = [[False]*4]*8
regles[0][2] = True
print(regles[1][2])
```

Qu'observe-t-on à l'écran ?

A) `True` B) `False`

Ex 4

Voici un programme en Python :

```
regles = [[False]*4 for i in range(8)]
regles[0][2] = True
print(regles[1][2])
```

Qu'observe-t-on à l'écran ?

A) `True` B) `False`

Ex 5

Ecrire une fonction `affichePlateau()` qui affiche le plateau du jeu tictactoe à partir d'un tableau 2D `plateau` contenant les caractères 'O','X', et '#' sous la forme d'un tableau 3 x 3 par exemple :

```
# X O
# X #
# # #
```

Ex 6

1. Ecrire une fonction `alignementLigne(numLigne)` qui retourne vrai s'il y a alignement sur la ligne de numéro `numLigne` du plateau du Tic Tac Toe
2. idem pour un alignement sur une des colonnes
3. idem pour un alignement sur la première et la deuxième des diagonales

2.4 Dictionnaires

Un dictionnaire associe à une clé **clé unique** une **valeur**
C'est une structure de données de correspondance, on parle aussi de tableau associatif
Par exemple dans le dictionnaire des variables gérées par Python , à un nom de variable est associée soit une valeur soit une référence, ainsi dans l'exemple suivant à la variable `quotient` est associée la valeur 3

```
>>> quotient = 3
>>> vars()
{'quotient': 3, '__loader__': <class '_frozen_importlib.BuiltinImporter'>, '__spec__': None, '
__package__': None, '__doc__': None, '__name__': '__main__', '__builtins__': <module 'builtins
' (built-in)>}
```

Un autre exemple :
Nous avons tous dans nos téléphones un dictionnaire où sont associés à des noms, des numéros de téléphone, par exemple :

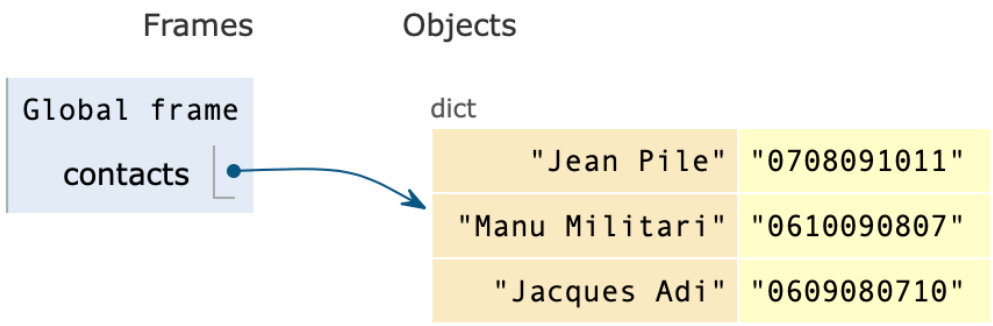
```
contacts = { "Jean Pile": "0708091011",
 "Manu Militari": "0610090807",
 "Jacques Adi ": "0609080710"}
```

Le dictionnaire `contacts` ci-dessus a été défini en extension mais on aurait pu à partir d'un dictionnaire vide ajouter des éléments au fur et à mesure, par exemple

```
contacts = { }
contacts["Jean Pile"] = "0708091011"
contacts["Manu Militari"] = "0610090807"
contacts["Jacques Adi"] = "0609080710"
```

Il n'y a pas de structure d'ordre dans un dictionnaire d'ailleurs à l'affichage les éléments n'apparaissent pas forcément dans l'ordre d'insertion dans le dictionnaire. Ainsi par exemple

```
>>> contacts = {}
>>> contacts["Jean Pile"] = "0708091011"
>>> contacts["Manu Militari"] = "0610090807"
>>> contacts["Jacques Adi"] = "0609080710"
>>> contacts
{'Jean Pile': '0708091011', 'Jacques Adi': '0609080710', 'Manu Militari': '0610090807'}
```



Quelles sont les principales opérations que l'on peut faire sur un dictionnaire ?

1. Se demander si une clé se trouve dans le dictionnaire

```
>>>"Jean Pile" in contacts
True
```

ou

```
>>>"Jean Pile" in contacts.keys()
True
```

contacts est un **objet** et `keys()` est une fonction spécifique ou méthode de l'objet `contacts` et exercer une certaine action sur un objet par l'intermédiaire d'une méthode `f(...)` s'écrit `objet.f(...)`
La méthode `keys()` appliquée à l'objet `contacts` retourne les clés du dictionnaire `contacts` sous forme de liste

2. Supprimer une clé et sa valeur

```
>>>del contacts["Jean Pile"]
```

3. Parcourir la liste des clés pour faire un traitement sur les valeurs associées, par exemple afficher les noms des numéros commençant par "06"

```
for cle in contacts.keys():
    if contacts[cle][1] == "6":
        print(cle," a pour numéro de téléphone : ",
        contacts[cle])
```

En effet `contacts[cle]` est une chaîne de caractères et `contacts[cle][1]` est le deuxième caractère de cette chaîne de caractères

4. Il existe aussi la méthode `values()` qui appliquée à l'objet `contacts` retourne la liste des valeurs. On n' aurait pu afficher que les numéros de téléphone commençant par 06 **sans leur antécédents** c'est à dire les noms

```
for val in contacts.values():
    if val[1] == "6":
        print(val)
```

5. Il existe aussi la méthode `items()` qui appliquée à l'objet `contacts` retourne la liste des tuples (clés,valeurs)

```
for couple in contacts.items():
    if couple[1][1] == "6":
        print(couple[0]," a pour numéro de téléphone : ",
        couple[1])
```

Ici couple est un tuple dont le premier élément couple[0] est une clé et le deuxième élément coulple[1] est une valeur de type chaîne de caractères.
couple[1][1] est le deuxième caractère de cette chaîne

2.4.1 Exercices : Dictionnaires

Ex 1

Le serveur DNS (Dynamic Name System) d'authentification confirme l'association d'un nom de domaine à une adresse IP (Internet Protocol)
On dispose du dictionnaire suivant :

```
dns = {"fr.wikipedia.org":"91.198.174.192",
"mathly.fr":"85.236.158.88",
"gallerianazionale.com":"94.130.217.148"}
```

1. Qu'affiche l'instruction `print(dns["mathly.fr"])` ?
2. Qu'affiche l'instruction `len(dns.keys())` ?

Ex 2

On dispose d'un dictionnaire `volcans` qui associe à des noms de volcans (de type str) des couples (type,date) où type est le type du volcan, le caractère "E" pour explosif ou "e" pour effusif et date la date de la dernière éruption connue.
Par exemple l'instruction `print(volcans["Piton de la Fournaise"])` a affiché `("e","01/11/2019")`
On dit que le Piton de la Fournaise est en cours d'éruption

1. Définir une fonction `volcansEffusifs(volcans)` qui à partir du dictionnaire `volcans` retourne la liste des volcans effusifs
2. Définir une fonction `volcansEnCoursEruption(volcans)` qui à partir du dictionnaire `volcans` retourne la liste des volcans en cours d'éruption

Ex 3

A chaque nom de polygone régulier (type str) on lui associe le couple (type tuple) constitué du nombre de sommets (type int) et du nombre de diagonales (type int)
On dispose d'un dictionnaire `nbDiagonales`
Par exemple l'instruction `print(nbDiagonales["carré"])` a affiché `(4,2)`

1. On aimerait trouver une loi qui relie le nombre de sommets (ou de côtés) du polygone et le nombre de diagonales
   Pour cela construire une liste `nbDiag` à partir du dictionnaire `nbDiagonales` en parcourant les valeurs de ce dictionnaire telle que
   `nbDiag[n]` est le nombre de diagonales d'un polygone à `n` sommets et
   `nbDiag[n]` $= 0$ si $0 \leqslant n \leqslant 3$
   Définir une fonction `creeListe(nbDiagonales)` qui crée la liste `nbDiag` à partir du dictionnaire `nbDiagonales`
2. Etant donnée une liste `liste1` on définit la liste des variations `liste2` telle que `liste2[n] = liste1[n] - liste1[n-1]` et `liste2[0] =0` Définir une fonction `creeListeVariations(liste)` qui retourne la liste des variations de `liste`
3. Les valeurs de la liste des variations de la liste `nbDiag` nous emmènent à penser que la loi qui relie le nombre de sommets (ou de côtés) du polygone et le nombre de diagonales est un trinôme $d(n) = an^2 + bn$ avec $d(3) = 0$ et $d(4) = 2$. Quelle est cette loi ?
4. Utiliser cette loi pour engendrer **par compréhension** le dictionnaire `nbDiagonales`

Problème :

Une expression littérale étant donnée par exemple sous cette forme '5 - 2' c'est à dire une chaîne de caractères représentant un nombre **puis un espace** puis un symbole parmi + - * / **puis un espace** puis à nouveau une chaîne de caractères représentant un nombre, il s'agit **d'évaluer** cette expression littérale c'est à dire de lui attribuer la valeur numérique correspondante, ici 3

On va utiliser la fonction `split()` qui appliquée à une chaîne de caractères contenant

```
>>> expression = '5 - 2'
>>> elt1, op, elt2 = expression.split(' ')
>>> type(expression.split(' '))
<class 'list'>
>>> print(elt1)
5
>>> type(elt1)
<class 'str'>
```

un séparateur ici l'espace, "coupe" cette chaîne selon le séparateur et les éléments sont des chaînes de caractères insérés dans une liste

On utilise un dictionnaire pour **associer** au symbole une **fonction** qui permet ensuite d'évaluer l'expression

```
def somme(x,y):
    return x + y

def difference(x,y):
    return x - y

def mult(x,y):
    return x*y

def div(x,y):
    if y == 0:
        return 'infini'
    else:
        return x/y

operateurs = {'+' : somme,
              '-' : difference,
              '*' : mult,
              '/' : div}

def evaluation(expression):
    elt1, op, elt2 = expression.split(' ')
    operateur = operateurs[op]
    return operateur(int(elt1),int(elt2))
```

```
#-----------------main----------------------
expression = '6 + 7'
print(evaluation(expression))
```

Le nom d'une fonction est une **variable** associée à une **référence**, on voit cette association ci-dessous dans le dictionnaire `vars()`

```
>>> def somme(x,y):
        return x + y

>>> def difference(x,y):
        return x - y

>>> def mult(x,y):
        return x*y

>>> def div(x,y):
        if y == 0:
                return 'infini'
        else:
                return x/y
```

```
>>> vars()
{'mult': <function mult at 0x1129e3510>, '__package__': None, 'div': <function div at 0x1129e3598>, '__do
__': None, 'difference': <function difference at 0x101e4cbf8>, 'somme': <function somme at 0x10ffce510>,
__name__': '__main__', '__builtins__': <module 'builtins' (built-in)>, '__loader__': <class '_frozen_impo
tlib.BuiltinImporter'>, '__spec__': None}
```

On retrouve cette association dans le dictionnaire `operateurs` que l'on peut créer dynamiquement en créant chaque association en autant d'instructions

```
>>> operateurs = {}
>>> operateurs['+'] = somme
>>> operateurs['-'] = difference
>>> print(operateurs)
{'+': <function somme at 0x10ffce510>, '-': <function difference at 0x101e4cbf8>
>>> operateurs['+'](7,5)
12
```

```
>>> operateurs['*'] = mult
>>> operateurs['/'] = div
>>> operateurs.keys()
dict_keys(['+', '/', '*', '-'])
>>> operateurs.values()
dict_values([<function somme at 0x10ffce510>, <function div at 0x1129e3598>, <function mult at 0x1129e351
>, <function difference at 0x101e4cbf8>])
>>> operateurs.items()
dict_items([('+', <function somme at 0x10ffce510>), ('/', <function div at 0x1129e3598>), ('*', <function
mult at 0x1129e3510>), ('-', <function difference at 0x101e4cbf8>)])
```

Chapitre 3

Traitement de données en tables

Les bases de données jouent un rôle important dans le stockage "intelligent" de données. A travers un exemple nous allons étudier le traitement de données en tables. Nous allons considérer une petite base de données concernant un certain nombre de films qu'un cinéphile pourrait constituer et stocker sur un serveur.

3.1 Indexation de tables

Cette base de données est constituée de quatre tables
Chaque table (ou fichier) peut être vu comme un tableau 2D constitué d'un certain nombre de champs
Un tableau 2D peut être vu comme un ensemble de lignes ou tuples ou n-uplets ou vecteurs **Une table ne doit pas contenir de doublons c'est à dire des vecteurs identiques**
Deux vecteurs sont identiques si leurs coordonnées ou champs sont égaux deux à deux
La première table est la table Film (idFilm, titre, année, genre, idRéalisateur, codePays) dont les champs sont :

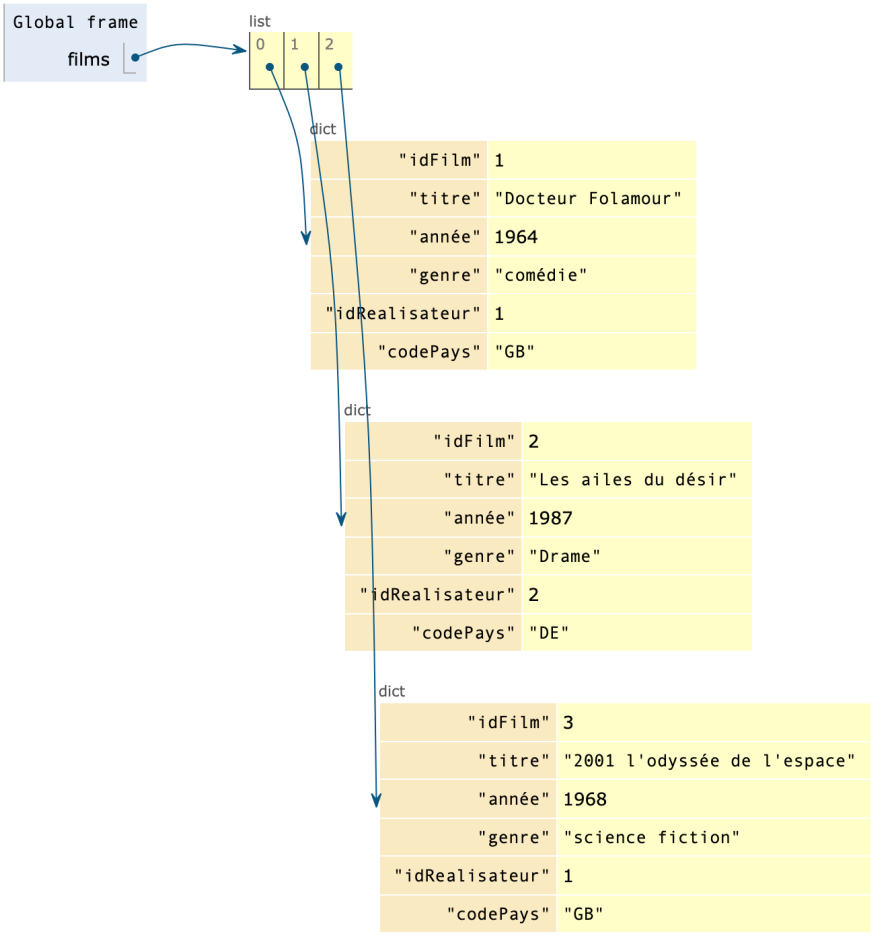
1. **idFilm** est un identificateur sous la forme d'un *entier*
2. **titre** est une *chaîne de caractères* qui est le titre du film
3. **annee** est un *entier* qui est l'année de sortie du film
4. **genre** est une *chaîne de caractères* qui est le genre du film
5. **idRealisateur** est une *entier* caractérisant le réalisateur
6. **codePays** est une *chaîne de caractères* caractérisant le pays principal où le film a été produit

idFilm	titre	année	genre	idRéalisateur	codePays
1	Docteur Folamour	1964	Comédie	1	GB
2	Les ailes du désir	1987	Drame	2	DE
3	2001 l'odyssée de l'espace	1968	Science Fiction	1	GB
4	Le dictateur	1940	Drame	8	US
5	Les tontons flingueurs	1963	Comédie	9	FR

Une façon d'implémenter cette table en Python est de faire une liste de dictionnaires

```
films = [{'idFilm':1,'titre': 'Docteur Folamour','année':1964,
'genre':'Comédie','idRealisateur':1,'codePays':'GB'},
{'idFilm':2,'titre': 'Les ailes du désir','année':1987,
'genre':'Drame','idRealisateur':2,'codePays':'DE'},
{'idFilm':3,'titre': '2001 l\'odysee de l\'espace','année':1968,
'genre':'Science-Fiction','idRealisateur':1,'codePays':'GB'}]
```

En travaux pratiques on construira la liste à partir d'un fichier .csv
Voici une visualisation avec Pythontutor de cette structure de données



La seconde table est la table Artiste (idArtiste, nom, prénom, annéeNaiss) dont les champs sont :

1. **idArtiste** un entier qui identifie l'artiste (clé primaire)
2. **nom** une chaîne de caractères , le nom de l'artiste
3. **prenom** une chaîne de caractères , le prénom de l'artiste
4. **anneeNaiss** un entier , l'année de naissance de l'artiste

idArtiste	nom	prénom	anneeNaissance
1	Kubrick	Stanley	1928
2	Wenders	Wim	1945
3	Sellers	Peter	1935
4	Falk	Peter	1927
5	Ganz	Bruno	1941
6	Dommartin	Solweig	1961
7	Dullea	Keir	1936
8	Chaplin	Charlie	1889
9	Lautner	Georges	1926
10	Ventura	Lino	1919
11	Blier	Bernard	1916
12	Brown	Peter	1935

La troisième est la table Rôle (idFilm, idActeur, nomRôle) dont les champs sont :

1. **idFilm** un entier qui identifie le film
2. **idActeur** un entier qui identifie l'acteur ayant joué dans ce film
3. **nomRole** une chaine de caractères désignant le rôle de l'acteur

idFilm	idActeur	nomRole
1	3	Group Captain Lionel Mandrake / président Merkin Muffley / Dr Strangelove
2	4	Peter Falk
2	5	Damiel
2	6	Marion
3	7	David Bowman
4	8	Adenoid Hynkel, le dictateur de Tomanie / un barbier juif
5	10	Fernand Naudin
5	11	Raoul Volfoni

La quatrième est la table Pays (code, nom, langue) dont les champs sont :

1. **code** une chaîne de caractères désignant le pays producteur du film
2. **nom** une chaîne de caractères désignant le nom du pays producteur du film
3. **langue** une chaîne de caractères désignant la langue principale du pays producteur du film

code	nom	langue
FR	France	Français
US	Etats Unis	Anglais
GB	Angleterre	Anglais
DE	Allemagne	Allemand

3.1.1 Exercices

Ex 1

Pourquoi ne pas créer directement une seule table pour la base de données de films ? (Indication : Considérer les artistes qui jouent dans un film...)

Ex 2

1. Proposer une implémentation pour la table Artiste
2. Afficher tous les prénoms des artistes (sans doublon)

3.2 Recherche dans une table

Il s'agit maintenant de rechercher des informations intéressantes dans les tables de la base de données et ces opérations sont d'autant plus intéressantes que la base de données est volumineuse

On pourrait se demander par exemple : *"Quels sont les titres de toutes les comédies ?"*
On parcourt la liste films et pour chaque élément (ou ligne) si le champ genre est "comedie" alors on fait afficher le titre du film ce qui donne en Python

```
for film in films:
    if film['genre'] == 'comedie':
        print(film['titre'])
```

On pourrait se demander par exemple : *"Quels sont les titres de tous les films qui ne sont pas des comédies ?"*
On parcourt la liste films et pour chaque élément (ou ligne) si le champ genre est différent de "comedie" alors on fait afficher le titre du film ce qui donne en Python

```
for film in films:
    if film['genre'] != 'comedie':
        print(film['titre'])
```

On pourrait se demander par exemple : *"Quels sont les comédies produites au XX ième siècle ?"*
On parcourt la liste films mais cette fois ci la condition est une expression booléenne "genre == comedie" et "annee < 2000"

```
for film in films:
    if film['genre'] == 'comedie' and film['date'] < 2000:
        print(film['titre'])
```

On pourrait se demander par exemple : *"Quels sont les comédies produites en France au XX ième siècle ?"*
On parcourt la liste films mais cette fois ci la condition est une expression booléenne "genre == comedie" et "annee < 2000" et codePays=='FR'

```
for film in films:
    if film['genre'] == 'comedie' and \
    film['date'] < 2000 and \
    film['codePays'] == 'FR':
        print(film['titre'])
```

On pourrait se demander par exemple : *"Quels sont les comédies ou les drames de la base de données ?"*

On parcourt la liste films mais cette fois ci la condition est une expression booléenne "genre == comedie" ou "genre =="drame"

```
for film in films:
    if film['genre'] == 'comedie'  or  film['genre'] == 'drame':
        print(film['titre'])
```

3.3 Tri d'une table

Toutes les opérations que l'on fait sur une table doivent toujours redonner une autre table avec la règle fondamentale qu'il n'y a pas de doublons Si on se demande par exemple :*A partir de la table Artiste d'obtenir la table PrénomsDesArtistes et date de Naissance* on va avoir un problème car deux artistes ont le même prénom, et la même date de Naissance on ne peut donc pas juste prendre la colonne des prénoms et la colonne des dates de naissance dans la table Artiste car sinon on n'aura pas une table
Comment faire ?
En triant la colonne des prénoms par ordre lexicographique les deux prénoms "Peter" vont se retrouver à côté l'un de l'autre ce qui nous permettra d'en garder un seul

3.3.1 Exercices

Ex 1

Lire le document `https://docs.python.org/fr/3/howto/sorting.html` et expérimenter (et comprendre) la méthode suivante

```
>>> films = [{'idFilm':1,'titre': 'Docteur Folamour','année':1964,
'genre':'Comédie','idRealisateur':1,'codePays':'GB'},
{'idFilm':2,'titre': 'Les ailes du désir','année':1987,
'genre':'Drame','idRealisateur':2,'codePays':'DE'},
{'idFilm':3,'titre': '2001 l\'odysee de l\'espace','année':1968,
'genre':'Science-Fiction','idRealisateur':1,'codePays':'GB'}]
```

```
>>> sorted(films, key=lambda film: film['année'])
[{'codePays': 'GB', 'année': 1964, 'titre': 'Docteur Folamour', 'idRealisateur': 1, 'idFilm': 1, 'genre': 'Comédie'},
{'codePays': 'GB', 'année': 1968, 'titre': "2001 l'odysee de l'espace", 'idRealisateur': 1, 'idFilm': 3, 'genre': 'Sci
ence-Fiction'}, {'codePays': 'DE', 'année': 1987, 'titre': 'Les ailes du désir', 'idRealisateur': 2, 'idFilm': 2, 'gen
re': 'Drame'}]
```

Ex 2

Implémenter les tables cette fois-ci sous la forme d'une liste de tuples

3.4 Fusions de tables

On aimerait maintenant "voir" un film de manière un peu plus globale en "fusionnant" les tables Films et Artistes
Comment ?

On va trier le champ "idRealisateur" dans la table Films puis le champ "idArtiste" dans la table Artiste puis fusionner

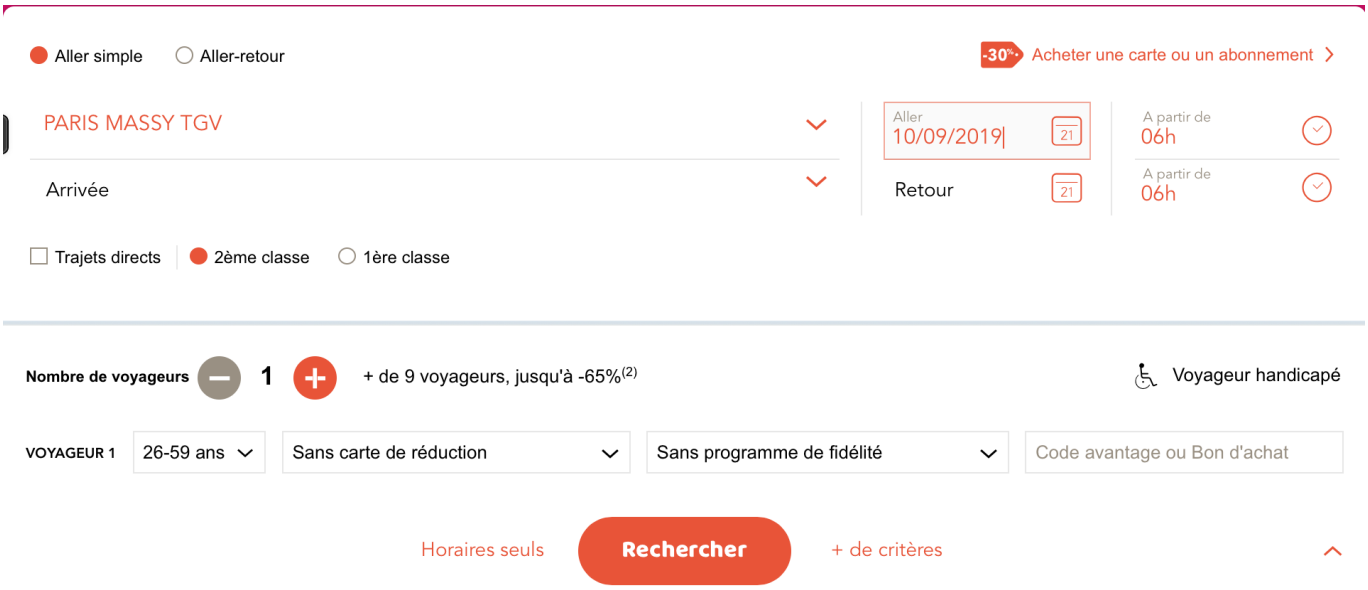
Chapitre 4

Interactions entre l'homme et la machine sur le Web

Lorsqu'on navigue sur le Web nous ne faisons pas que lire passivement du texte mais parfois agissons sur le document que nous lisons.
Regardons sur un site de réservation en ligne d'un billet de TGV

4.1 Modalités de l'interaction entre l'homme et la machine

4.1.1 Les différents composants graphiques permettant d'interagir avec une application Web



La plupart du temps nous retrouvons un peu partout les même éléments :

1. **des cases à cocher** comme celle associée à Trajets directs. En cliquant à l'intérieur de cette case on peut voir une encoche apparaître $\vee$ ce qui signifie que ce choix Trajets directs a été sélectionné

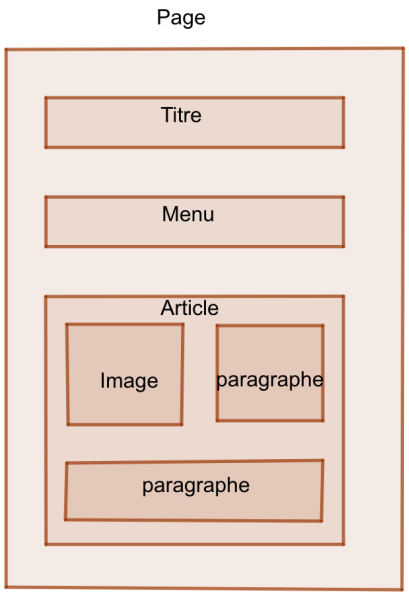
2. **des boutons radio** .
   En cliquant dans ces boutons ils deviennent colorés ce qui permet de faire un choix, par exemple sur le site de réservation on peut ainsi choisir un direct ou un aller retour
3. **des menus déroulants** qui nous propose des choix que l'on peut sélectionner ainsi la date du départ est un menu déroulant ainsi que le lieu du départ.
4. **des curseurs**. Si on clique sur le symbole $\oplus$ on peut augmenter d'une unité le nombre de voyageurs alors que si on clique sur le symbole $\ominus$ on diminue d'une unité le nombre de voyageurs
5. **des boutons** comme le bouton **Rechercher** qui lance la recherche d'un voyage avec les critères sélectionnés à condition que l'on clique dessus

4.1.2 Les bases du HTML et du CSS

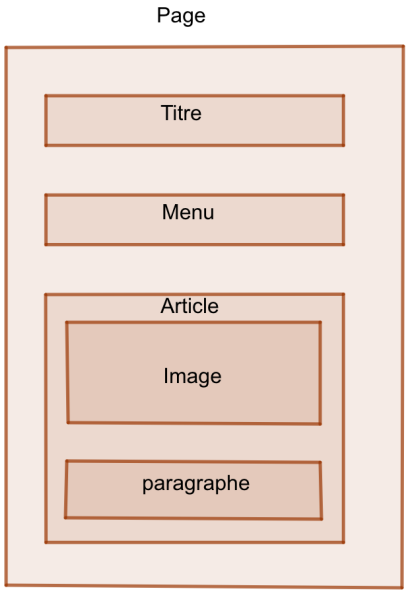
Ce qui suit est un exposé "simple" de quelques éléments des langages HTML et CSS Pour avoir des informations supplémentaires on peut consulter le site de la fondation Mozilla `https://developer.mozilla.org/fr/docs/Apprendre`
Il faut distinguer le fond ou la structure de la page de la forme (ou la mise en page) de la page
On peut regarder une page Web comme la Une d'un journal papier : certains éléments forment la structure de la page :

1. Titre ou en-tête
2. Menu
3. Article
4. Image
5. paragraphe(s)

Une mise en page possible peut ressembler à



Mais d'autres mises en page sont possibles, comme :



1. D'où un langage, le langage HTML (pour HyperText Markup Language), pour décrire la structure de la page
2. Et un deuxième langage, le langage CSS (pour Cascading Style Sheets), pour décrire la mise en page de la page

Voici la copie d'écran d'une page Web très simple

Comment écrire une page Web ?

HTML

CSS

Codage d'une page Web

```
<!DOCTYPE html >
<html lang="fr">
   <head>
       <meta charset="utf-8" />
      <!--<link  href="feuilleSty.css" rel="stylesheet"/>-->
      <title>HTML et CSS</title>
   </head>

   <body>
    <header>
         <h1>Comment écrire une page Web ?</h1>
    </header>

        <nav>

            <p><a href="html.html">HTML</a></p>
            <p><a href="css.html">CSS</a></p>

        </nav>

       <article>
           <h2>Codage d'une page Web</h2>
           <img width = "450"  align = "middle" src="images/html.png"/>
           <p>Deux langages sont nécessaires pour écrire une page Web</p>
           <ol>
               <li>Le langage HTML (pour HyperText Markup Language),
                pour décrire la structure de la page</li>
               <li>Le langage CSS (pour Cascading Style Sheet),
               pour décrire la mise en page de la page</li>
           </ol>
        </article>
  </body>
</html>
```

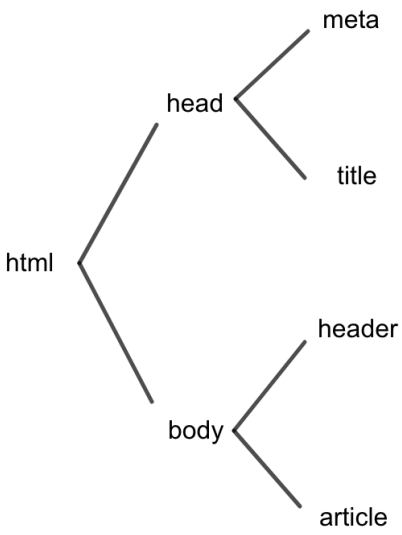
Deux langages sont nécessaires pour écrire une page Web

1. Le langage HTML (pour HyperText Markup Language), pour décrire la structure de la page
2. Le langage CSS (pour Cascading Style Sheet), pour décrire la mise en page de la page

Voici le contenu du fichier `exemple.html` qui a donné la page ci-dessus

```
1  <!DOCTYPE html >
2  <html lang="fr">
3     <head>
4         <meta charset="utf-8" />
5        <!--<link  href="feuilleSty.css" rel="stylesheet"/>-->
6        <title>HTML et CSS</title>
7     </head>
8
9     <body>
10     <header>
11          <h1>Comment écrire une page Web ?</h1>
12     </header>
13
14         <nav>
15
16              <p><a href="html.html">HTML</a></p>
17              <p><a href="css.html">CSS</a></p>
18
19         </nav>
20
21        <article>
22            <h2>Codage d'une page Web</h2>
23            <img width = "450"  align = "middle" src="images/html.png"/>
24            <p>Deux langages sont nécessaires pour écrire une page Web</p>
25            <ol>
26                <li>Le langage HTML (pour HyperText Markup Language),
27                 pour décrire la structure de la page</li>
28                <li>Le langage CSS (pour Cascading Style Sheet),
29                pour décrire la mise en page de la page</li>
30            </ol>
31        </article>
32   </body>
33 </html>
```

1. Le langage HTML est un langage à **balises**. La plupart des balises sont ouvrantes et fermantes comme des parenthèses, par exemple :
   (a) Tout le document est encadré par <html>...</html> (ligne2 et ligne 33)
   (b) Les balises <header>.....</header> (ligne 10 et 12) encadrent tout ce qui concerne l' en-tête de la page
   (c) Les balises <article>.....</articles> encadrent tout ce qui concerne le coeur de la page
2. Une balise a des paramètres ou descripteurs ainsi la balise <html> a pour paramètre lang="fr"
3. Il existe une relation de **parentalité** entre les balises. On dit par exemple que la balise <html> est le parent de la balise <body> ou encore que la balise <body> est un enfant de la balise <html>, parce que la balise <html> "encadre" la balise <body>
   ce qui permet de dessiner une sorte d'arbre généalogique ou DOM (pour Document Object Model) du document HTML, dont voici une version incomplète
4. La balise <head>...</head> donne des informations sur le document , ainsi l'encodage des caractères du document est fait en utf-8. On remarque au passage la présence d'une des rares balises n'ayant qu'un seul terme, la balise meta



La balise <title>...</title> contrairement à ce que l'on pourrait croire ne concerne pas le titre visible de la page mais le titre de l'onglet associé à la page

5. La balise <body>...</body> délimite tout ce qui est visible dans la page
6. Une balise très importante <a>.... crée des **liens** entre les pages et le paramètre href ici donne la référence **relative** (on reviendra plus tard sur cette notion) de la page.
   Lorsqu'on déplace le curseur de la souris sur une page Web, s'il y a un lien la "flèche" se transforme en une "main avec un index pointé" ce qui signifie que l'on peut cliquer à cet endroit pour visualiser une autre page Web

7. Dans la structure article on observe un titre de niveau 2 avec la balise <h2>
8. La balise <img /> est une des rares balises à être seule, elle a plusieurs paramètres comme src (pour source) pour indiquer où se trouve l'image
9. La balise <ol> (pour ordered list) permet de faire des listes numérotées. Chaque élément de la liste est encadrée par les balises <item> et </item>
10. La balise <link href="feuilleSty.css"/> permet de relier au fichier `exemple.html` un fichier CSS `feuilleSty.css` qui "mettra en forme" le fichier précédent
    Il faut "décommenter" la ligne où se trouve la balise <link/> pour la rendre active, c'est à dire enlever les caractères <!– et –>

Ce que fait le fichier `feuilleSty.css` pour l'instant c'est de mettre en évidence d'abord les trois structures descendantes de la balise body, en mettant une bordure bleue autour de l'en-tête, une verte autour du menu et une rouge autour de l'article

Comment écrire une page Web ?

HTML

CSS

Codage d'une page Web

```
<!DOCTYPE html >
<html lang="fr">
   <head>
       <meta charset="utf-8" />
      <!--<link  href="feuilleSty.css" rel="stylesheet"/>-->
      <title>HTML et CSS</title>
   </head>

   <body>
    <header>
         <h1>Comment écrire une page Web ?</h1>
    </header>

        <nav>

             <p><a href="html.html">HTML</a></p>
             <p><a href="css.html">CSS</a></p>

        </nav>

       <article>
           <h2>Codage d'une page Web</h2>
           <img width = "450"  align = "middle" src="images/html.png"/>
           <p>Deux langages sont nécessaires pour écrire une page Web</p>
           <ol>
               <li>Le langage HTML (pour HyperText Markup Language),
                pour décrire la structure de la page</li>
               <li>Le langage CSS (pour Cascading Style Sheet),
               pour décrire la mise en page de la page</li>
           </ol>
        </article>
  </body>
</html>
```

Deux langages sont nécessaires pour écrire une page Web

1. Le langage HTML (pour HyperText Markup Language), pour décrire la structure de la page
2. Le langage CSS (pour Cascading Style Sheet), pour décrire la mise en page de la page

Voici le code CSS qui a permis cela :

```css
body
{
   /*background-color:#FFE4B5;
   border:2px solid black;  */
}

header
{
   /*text-align:center;
   background-color:#E4B5FE;*/
   border:2px solid blue;
}

nav
{
   border:2px solid green;
   /*text-align:center;*/
}

article
{
   border:2px solid red;
}
```

On constate qu'un fichier CSS consiste à donner des propriétés de mise en forme aux balises par exemple
`header` {
border :2px solid blue ;
}
met une bordure bleue de 2 pixels autour de l'en-tête de la page
Les "structures" sont alignées sur le bord droit de la fenêtre on va "centrer" les structures header, nav et article en les décalant du bord
On va utiliser la propriété **padding** voir `https://developer.mozilla.org/fr/docs/Web/CSS/padding-left`
On va aussi mettre de la couleur de fond aux structures avec la propriété `background-color` pour finalement obtenir une page :
On peut avec le même fichier HTML avoir deux fichiers CSS différents et avoir deux mises en forme différentes (Voir TP)

Comment écrire une page Web ?

HTML

CSS

Codage d'une page Web

```
<!DOCTYPE html >
<html lang="fr">
   <head>
       <meta charset="utf-8" />
      <!--<link  href="feuilleSty.css" rel="stylesheet"/>-->
      <title>HTML et CSS</title>
   </head>

   <body>
    <header>
         <h1>Comment écrire une page Web ?</h1>
    </header>

        <nav>

             <p><a href="html.html">HTML</a></p>
             <p><a href="css.html">CSS</a></p>

        </nav>

       <article>
           <h2>Codage d'une page Web</h2>
           <img width = "450"  align = "middle" src="images/html.png"/>
           <p>Deux langages sont nécessaires pour écrire une page Web</p>
           <ol>
               <li>Le langage HTML (pour HyperText Markup Language),
                pour décrire la structure de la page</li>
               <li>Le langage CSS (pour Cascading Style Sheet),
               pour décrire la mise en page de la page</li>
           </ol>
       </article>
  </body>
</html>
```

Deux langages sont nécessaires pour écrire une page Web

1. Le langage HTML (pour HyperText Markup Language), pour décrire la structure de la page
2. Le langage CSS (pour Cascading Style Sheet), pour décrire la mise en page de la page

Comment écrire une page Web ?

HTML CSS

Codage d'une page Web

```
<!DOCTYPE html >
<html lang="fr">
   <head>
       <meta charset="utf-8" />
      <!--<link  href="feuilleSty.css" rel="stylesheet"/>-->
      <title>HTML et CSS</title>
   </head>

   <body>
    <header>
         <h1>Comment écrire une page Web ?</h1>
    </header>

        <nav>

             <p><a href="html.html">HTML</a></p>
             <p><a href="css.html">CSS</a></p>

        </nav>

       <article>
           <h2>Codage d'une page Web</h2>
           <img width = "450"  align = "middle" src="images/html.png"/>
           <p>Deux langages sont nécessaires pour écrire une page Web</p>
           <ol>
               <li>Le langage HTML (pour HyperText Markup Language),
                pour décrire la structure de la page</li>
               <li>Le langage CSS (pour Cascading Style Sheet),
               pour décrire la mise en page de la page</li>
           </ol>
       </article>
  </body>
</html>
```

Deux langages sont nécessaires pour écrire une page Web

1. Le langage HTML (pour HyperText Markup Language), pour décrire la structure de la page
2. Le langage CSS (pour Cascading Style Sheet), pour décrire la mise en page de la page

4.1.3 Exercices : HTML et CSS

Ex 1

Compléter le DOM du fichier `exemple.html`

Ex 2

La structure de la page `html.html` est quasiment identique à celle de `exemple.html` avec quand même quelques modifications

1. Pour pouvoir revenir vers la page `exemple.html` insérer un lien dans le menu autour du mot Accueil
2. On suppose que l'on a une icône `accueil.png` représentant une maison . Faire un lien autour de l'icône permettant de revenir vers la page `exemple.html`

Ex 3

Essayer d'écrire la page `css.html` sans regarder le cours ni la documentation

Ex 4

On veut mettre en évidence les mots "structures" et "mise en page" dans l'article.
Allez chercher dans le Web des informations sur une balise qui met en gras le texte encadré, et sur une autre balise qui souligne le texte encadré
Réécrire les paragraphes en question en utilisant ces balises

Ex 5

On considère maintenant que la mise en évidence d'une partie d'un texte n'est pas le fait du langage HTML mais du langage CSS
On va donc créer une balise ... avec lesquelles on va encadrer du texte que l'on veut mettre en valeur

1. Encadrer les mots "structures" et "mise en page" dans l'article avec cette balise
2. Dans le fichier CSS associé au fichier `exemple.html` ajouter
   .important{
   color :#FF0000 ;
   }
   Allez chercher dans le Web comment mettre en gras et/ou surligner avec CSS

Ex 6

Dans le menu les éléments sur lesquels il y a un lien sont soulignés et sont en bleu. On aimerait changer leur couleur et qu'ils ne soient pas soulignés.
Allez voir sur le Web comment faire cela

4.1.4 Evènements

Nous avons vu précédemment que les éléments graphiques, boutons, curseurs, cases à cocher ne **réagissent que si on clique avec la souris ou avec le doigt (tablette ou smartphone**.
On parle de **programmation évènementielle**. La machine est au service de l'humain et attend **la réalisation d'évènements** pour **déclencher une action** bien précise
Ceci n'est pas réservé qu'au Web on retrouvera la programmation évènementielle avec le module tkinter de Python.
. Nous nous intéresserons qu'à deux évènements

1. Survoler avec le doigt ou avec la souris
2. Cliquer avec le doigt ou avec la souris

Nous avons revu les bases du HTML et du CSS pour structurer et mettre en forme les pages Web. Uniquement avec ces deux langages il nous est impossible de faire de la programmation évènementielle. Nous allons utiliser quelques éléments d'un langage, Javascript pour pouvoir dans une situation simple faire de la programmation évènementielle
Pour commencer on va écrire une page HTML contenant un bouton.
Si on clique sur ce bouton la page change de couleur aléatoirement
Nous allons expliquer le programme ci-dessous

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Changement de la couleur d'arrière-plan</title>
    <style>
      button {
        margin: 10px;
      }
    </style>
  </head>
  <body>
    <button>Changement de couleur</button>
    <script>
      var btn = document.querySelector('button');

      function random(number) {
        return Math.floor(Math.random()*number);
      }

      function changeCouleurBg() {
        var rndCol = 'rgb(' + random(255) + ',' + random(255) + ',' + random(255) +
        document.body.style.backgroundColor = rndCol;

      }
      btn.addEventListener('click', changeCouleurBg);
    </script>
  </body>
</html>
```

1. Entre la ligne 14 et 27 entre les balises <script> et </script> est inséré le code Javascript
2. Javascript permet de travailler sur la structure arborescente de la page HTML ou **DOM**
   La racine de l'arbre est la balise <html>, les noeuds **non-terminaux** sont les balises html de la page et les noeuds **terminaux** sont les textes
3. En Javascript comme en Python tout est **objet** (il y aura un cours en Terminale sur la programmation objet)
   Nous avons vu les types de bases et les types construits. La notion de classe permet de créer d'autres types.
   Ainsi Document est une classe de Javascript et document un objet (une réalisation particulière de la classe Document).
   Enfin la méthode querySelector() appliqué à l'objet document permet de se déplacer dans le DOM et d'une certaine manière permet de "pointer" sur le noeud <button> du DOM
4. A la ligne 26, la méthode `addEventListener()` associe à un évènement "click" ayant lieu sur le bouton une fonction `changeCouleurBg`

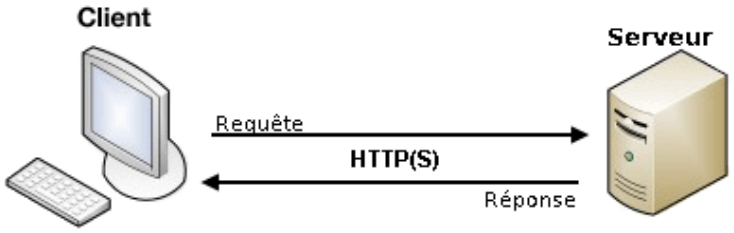
4.1.5 Exercices : HTML /CSS et Javascript

Ex 1

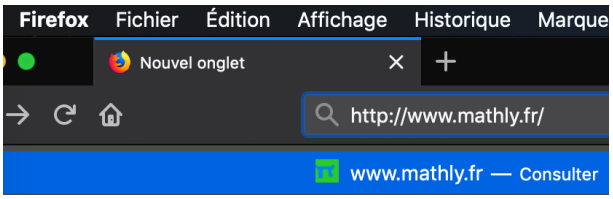
```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
 <link  href="style.css" rel="stylesheet"/>
<title>Première N.S.I</title>
 </head>

<body>
<h1>Quelques langages de programmation</h1>
<ul>
<li>Python</li>
<li>C</li>
<li>Java</li>
</ul>
<script type="text/javascript"
  src="liste.js">

</script>
<button>Ajouter des langages</button>
</body>
 </html>
```

1. Quel est le DOM de cette page html ?
2. Quels sont les noeuds et les noeuds littéraux ?
3. Faire un script Javascript qui ajoute les langages suivants OCaml ,C++ et php

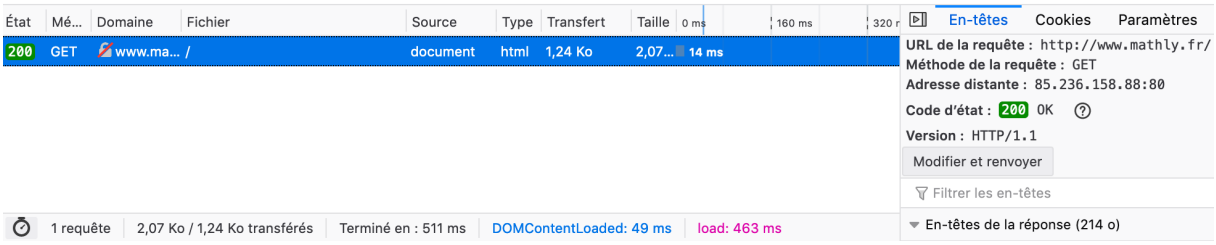
4.2 Interaction client-serveur



La plupart du temps quand on surfe sur le Web de lien en lien **en tant que client** on sollicite une information (texte, image, vidéo, son) qui se trouve sur un **serveur** distant suivant un **protocole** de communication nommé **HTTP** pour (Hyper Text Transfert Protocol) ou encore HTTPS (S pour Secure) lorsque les données sont chiffrées si on fait des achats en ligne par exemple
Si j'utilise un navigateur (Firefox par exemple) je peux entrer l'URL (pour Uniform Resource Locator) de la page d'un site que je connais sans forcément passer par un **moteur de recherche**



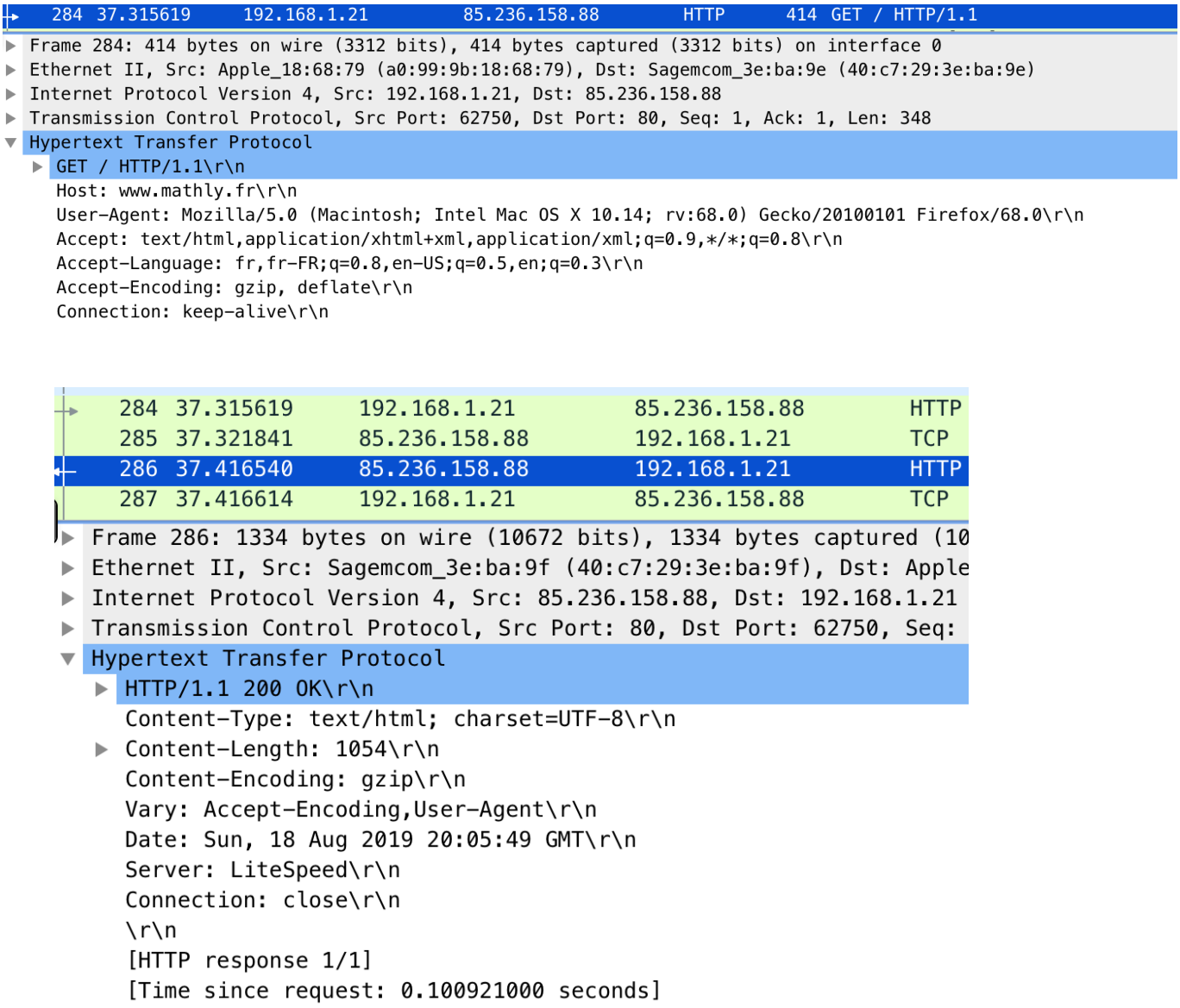
Une fois la page chargée, je peux avec l'aide de la console Web avoir des informations réseau sur le chargement de la page On voit que :



1. La méthode de la requête est GET
2. Le code est 200 (la page a bien été chargée) (un code d'erreur connu est le fameux 404)
3. On a **l'adresse IP** du serveur distant

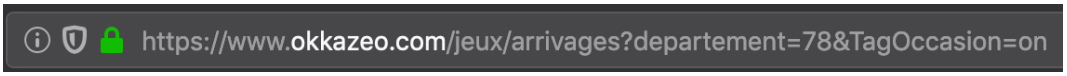
Pour être plus précis nous utilisons le logiciel Wireshark qui permet d'analyser les paquets d'information

Voici la demande (frame 284) par le client 192.168.1.21 au serveur 85.236.158.88 de la page d'accueil du site mathly.fr
Quelques millièmes de seconde après (frame 286) arrive la réponse Avec Wireshark on

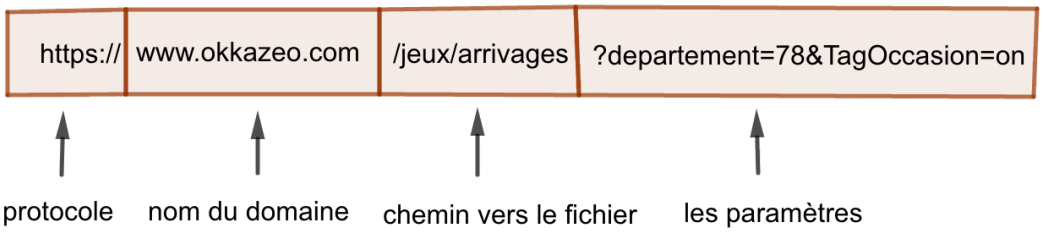


peut observer que les éléments d'une page arrivent séparément ainsi le fichier css arrive après (certes quelques centièmes de seconde), le javascript encore après
Ces éléments d'information sont ensuite "lus" par le navigateur du client et l'utilisateur peut voir la page Web demandée sur son écran
On peut affiner la requête en ajoutant des paramètres dans l'URL.
Regardons sur un exemple.
Si vous allez sur le site okkazeo.com , vous pouvez voir en fonction du département des jeux de société d'occasion
Ainsi par exemple on peut cliquer sur la carte de France et choisir le département 91 on obtient alors dans la barre d'URL Après le point d'interrogation la valeur du paramètre departement est 91, ainsi si on veut maintenant choisir le département 78, on peut soit revenir à la carte soit directement modifier dans la barre URL le departement

en remplaçant 91 par 78
Une fois que l'on a choisi le département 78 on peut faire un second choix et ne vouloir que les jeux d'occasion en cochant une case pour cela et ensuite en cliquant sur Filtrer on observe alors dans la barre d'URL Dans la barre URL on observe que la requête est

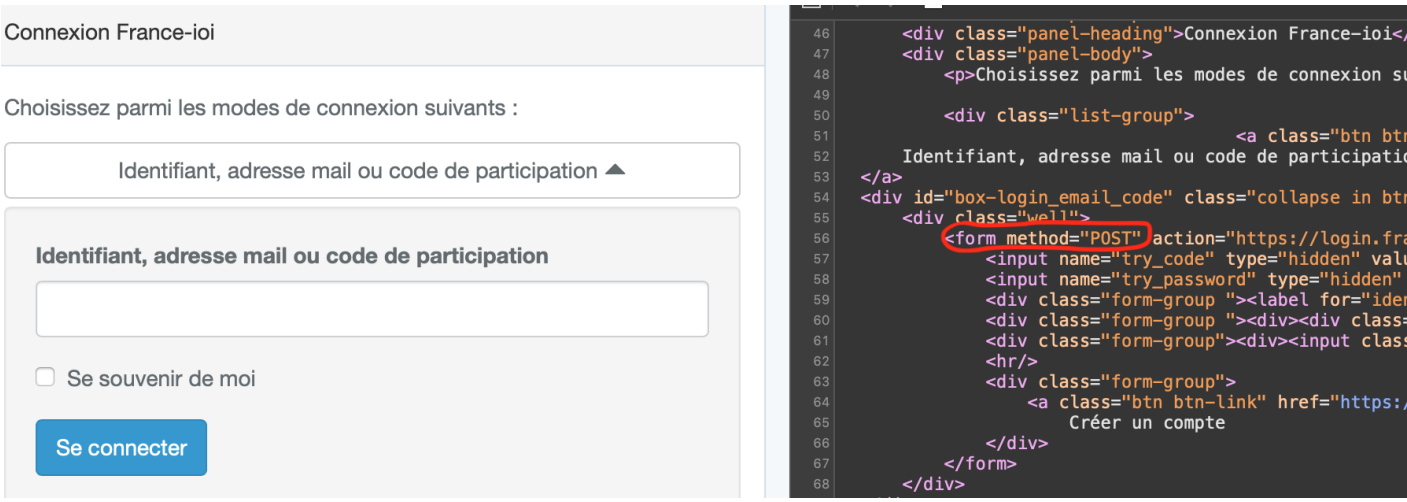


devenu departement=78&TagOccasion=on
On peut affiner la requête en ajoutant un paramètre avec le symbole & pour et
Ainsi si on voulait uniquement les jeux neufs on peut modifier l'URL ainsi departement=78&TagNeuf=on
En conclusion on retiendra la structure de l'URL d'une page



4.3 Formulaire d'une page web

Sur de nombreux sites on peut se connecter par exemple pour laisser un commentaire sur un site d'informations ou pour pouvoir y programmer comme sur le site de **France I.O.I** à l'aide **d'un formulaire**
On voit ci-dessous le formulaire à l'écran et le code de la page. Un formulaire est codé par les balises <form> et </form>.
Il existe principalement deux méthodes la méthode GET et la méthode POST (voir le site france ioi)
En travaux pratiques nous verrons les deux méthodes et leur différence

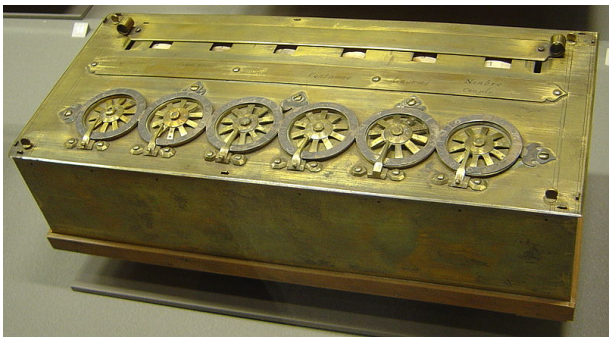


Chapitre 5

Architecture

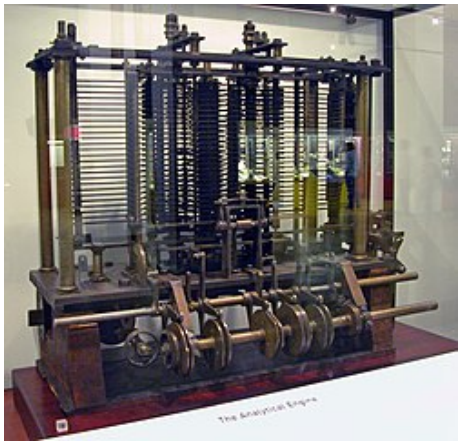
5.1 Un peu d'Histoire

La Pascaline de Pascal (1650)
Il s'agit de l'une des premières calculatrices construites avec des roues dentées. Un procédé astucieux permet de réaliser des retenues*(Visible au musée des arts et métiers à Paris)*



Métier à tisser de Jacquard (1801) Il s'agit d'un automate capable de créer un tissu à partir d'une sorte de programme fait avec des cartes perforées
(Visible au musée des arts et métiers à Paris)

La machine analytique de Charles Babbage(1830, jamais achevée) D'une certaine manière la machine analytique de Babbage est une version du métier à tisser de Jacquard destiné aux calculs.



Ada Lovelace et Babbage mettent aux points les premiers programmes écrits pour une machine *(Visible au Science Museum à Londres)*

Algèbre de Boole (XIX ème siècle)

Par analogie avec le calcul numérique, Boole introduit un calcul sur des grandeurs (les booléens) n'ayant que deux valeurs possibles (vraie ou faux) de telle sorte que le **ou** correspond à l'addition et le **et** correspond à la multiplication
Ainsi (A ou B)et$C \iff A$ et C ou A et B correspond à $(A+B)\times C = A\times C + B\times C$

Problème de décidabilité de Hilbert (1900)

Peut -on trouver un ***algorithme*** *permettant de décider, pour n'importe quelle équation diophantienne (c'est-à-dire équation polynomiale à coefficients entiers), si cette équation possède des solutions entières ?*
Par exemple $3x+2y=1$ est une équation diophantienne ayant une infinité de solutions les couples d'entiers $(-1+2k, 2-3k)|k \in \mathbb{Z}$ alors que l'équation $3x+2y^2=1$ n'a pas de solution

Machine de Turing (1935)

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO
THE ENTSCHEIDUNGSPROBLEM

By A. M. Turing.

[Received 28 May, 1936.—Read 12 November, 1936.]

Pour donner une réponse au problème de Hilbert, Turing introduit un objet théorique, appelé depuis lors *machine de Turing* qui va inspirer ensuite Von Neumann pour concevoir ce qui est considéré comme étant **le premier ordinateur** l'EDVAC
Voir ici une animation `https://interstices.info/comment-fonctionne-une-machine-de-turing/`

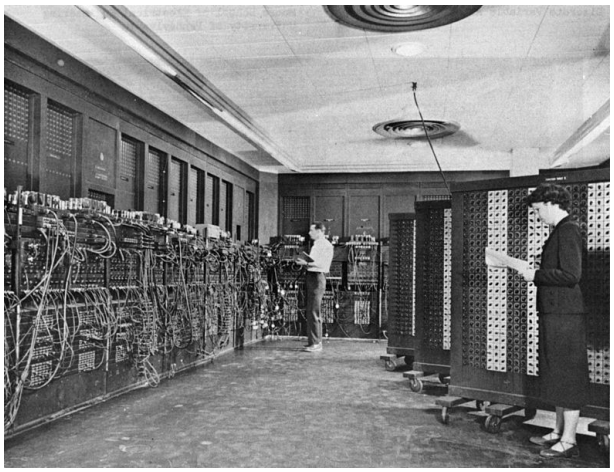
Le calculateur Harvard Mark 1 (1937)

Le calculateur mesurait environ 16 mètres de long et 2,6 mètres de hauteur et pesait 5 tonnes. Ce calculateur a été construit à partir des plans de la machine de Babbage. La technologie utilisée est celle des relais électromécaniques.

Le calculateur ENIAC(1943)

(Electronic Numerical Integrator And Computer) Ce calculateur a été construit en collaboration avec l'armée américaine et l'université de Pennsylvanie

La technologie est celle des tubes à vides electroniques (17468 tubes). La machine dans son ensemble pesait environ 30 tonnes pour une surface au sol de 160 mètres carrés



Machine de Von Neumann

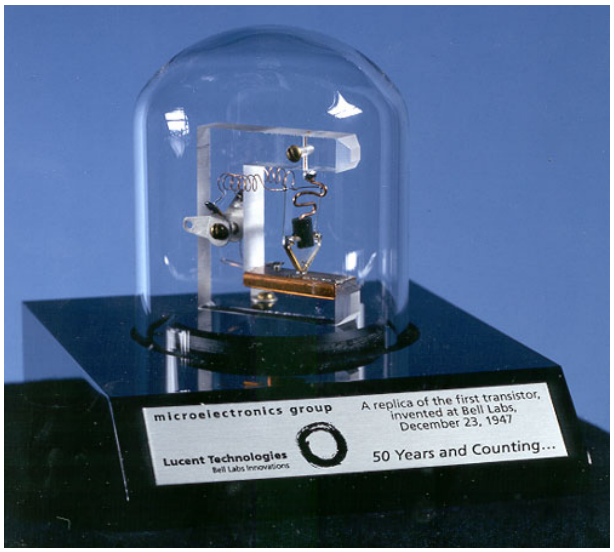
Eckert et Mauchly, les deux principaux concepteurs de l'ENIAC avec Von Neumann définissent la machine de Von Neumann **(le programme et les données se trouvent dans la même mémoire)** et mettent au point l'EDVAC (Electronic Discrete Variable Automatic Computer) considéré comme le premier ordinateur

Invention de l'assembleur (1954)

Les programmes de l'EDVAC étaient écrits en **binaire**, le langage machine. Progressivement la nécessité de créer un langage intermédiaire entre le langage des humains et celui de la machine s'est fait sentir. Le langage assembleur est apparu en 1954

Invention du transistor (1947-> 1953)

Une des révolutions du vingtième siècle est l'invention du **transistor** qui remplacera les tubes à vides dans les ordinateurs et permettra leur miniaturisation



Fortran (Formula Translator) (1957)

Dans le prolongement de l'assembleur apparaît le premier langage de programmation plus proche de l'Humain que de la machine. Destiné aux calculs scientifiques le FORTRAN sera étudié dans les universités jusqu'à la fin du vingtième siècle

Compilateur (1957)

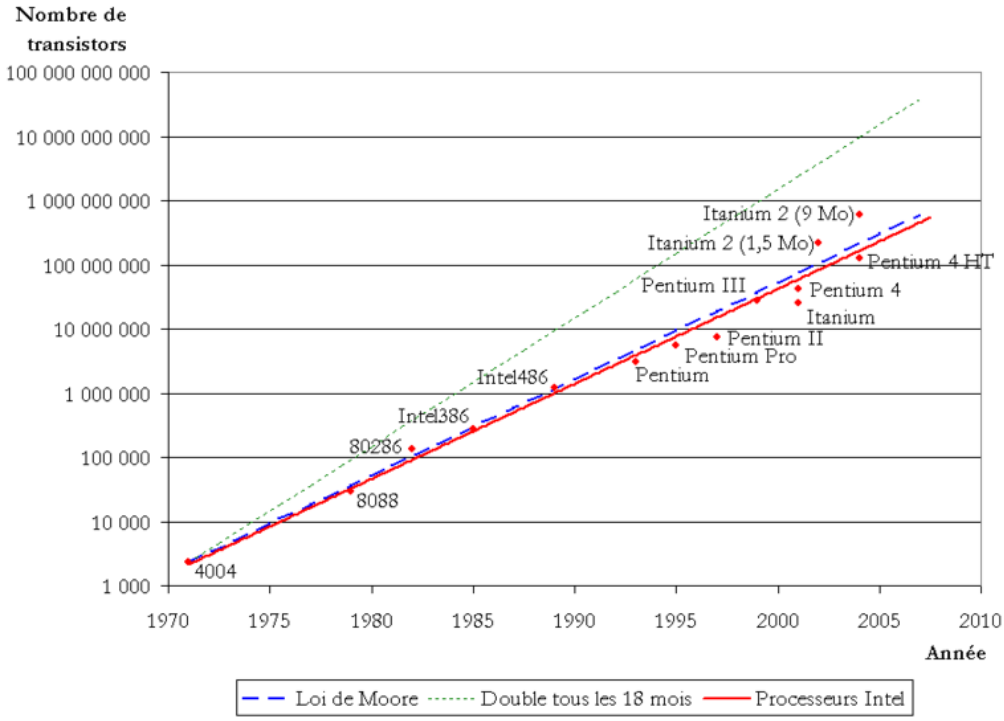
Un compilateur est un programme qui traduit un programme écrit dans un langage évolué comme Fortran , C, Java, C++ en langage machine

Loi de Moore (1965)

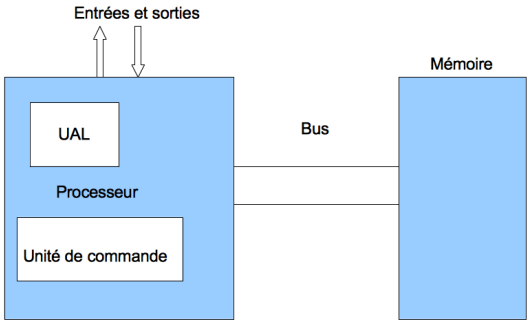
"Le nombre de transistors dans un microprocesseur double à peu près tous les deux ans"

Processeur 8080 (Intel) 1974

Microprocesseur 8 bits comportant 6000 transistors et une horloge de 2MHz
La miniaturisation des transistors conduit aux **circuits intégrés**



5.2 Modèle d'architecture séquentielle (Machine de Von Neumann)



L'originalité de la machine de Von Neumann est que **le programme et les données** sont dans la mémoire (vive) de l'ordinateur. Les éléments principaux sont :

1. un **processeur** constitué d'une :
   (a) **Unité arithmétique et logique (UAL)**
   (b) Des **registres** (des mémoires peu nombreuses et accessibles à l'unité arithmétique et logique pour y stocker des résultats de calculs)

(c) Une **unité de commande (UC)**

2. un *Bus de données* (fils permettant la circulation des données entre la mémoire vive et le processeur)
3. un *Bus d'adresses* (fils permettant la circulation des adresses mémoire entre la mémoire vive et le processeur)
4. un *Bus de commandes* (fils permettant la circulation des instructions entre la mémoire vive et le processeur)
5. **La mémoire Vive**
6. **Divers périphériques (écran, clavier,etc... :**

Voici le cycle principal (en boucle) du processeur

1. (phase lecture) Lit une instruction dans la mémoire à l'adresse n spécifiée par le registre PC (program counter) et la stocke dans un registre particulier appelé registre d'instruction
2. (phase décodage et exécution) L'instruction est découpée en bits significatifs, puis les bits significatifs sont distribués à tous les éléments du processeur (par exemple à l'UAL) pour exécuter des tâches précises (addition , opération logique etc...)

   Parmi ces bits significatifs dans certaines instructions peuvent figurer des indications de saut d'instruction ce qui signifie que la prochaine instruction n'est pas à l'adresse $n+1$ (pour faire un test si ...alors... par exemple) Le résultat de l'exécution peut intervenir aussi dans un saut (boucle)

   Autrement dit l'actuelle instruction contient des indications avant et/ou après exécution décidant l'adresse de la prochaine instruction

Nous allons maintenant entrer plus en détail à travers quelques exemples "simples" sur le langage assembleur

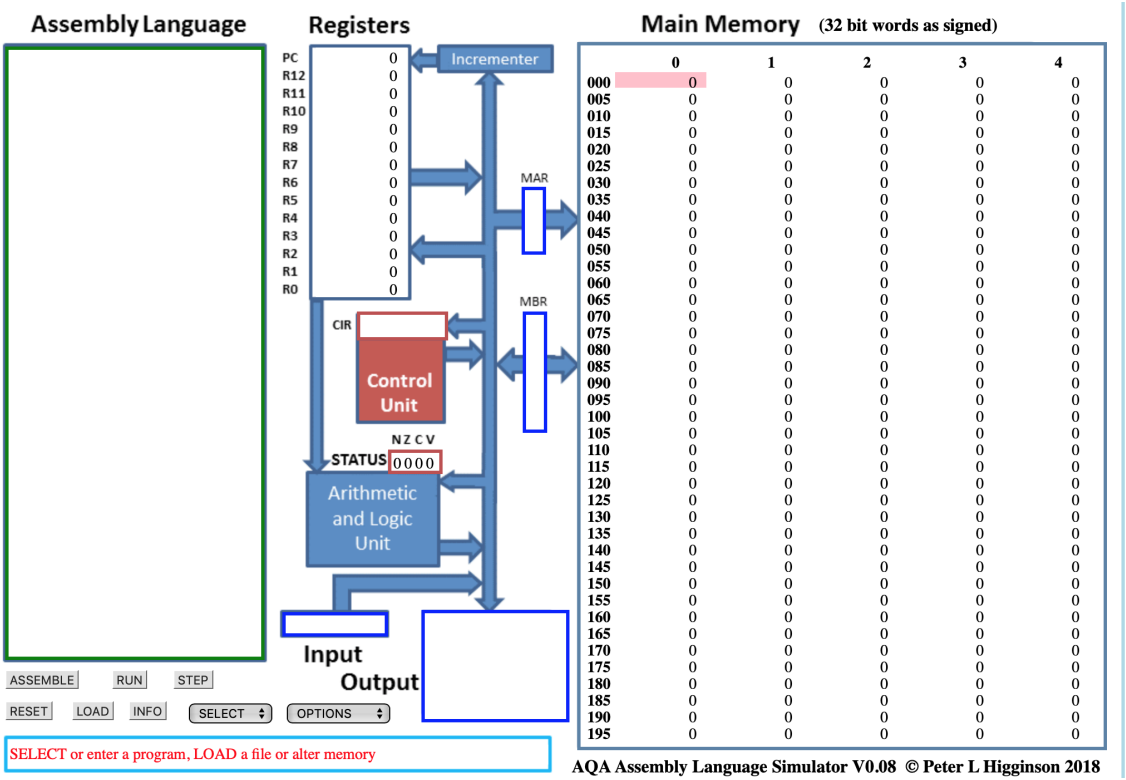
5.2.1 Langage machine vs Assembleur

Pour visualiser le fonctionnement du processeur nous allons utiliser le simulateur en ligne `http://www.peterhigginson.co.uk/AQA/` mis au point par Peter Higginson d'un processeur ARM

ARM est un fabricant de processeurs intervenant dans les téléphones portables et les tablettes. Il existe d'autres fabricants de processeurs comme Intel, AMD pour les ordinateurs personnels et les ordinateurs portables
Il existe deux grandes familles de processeurs :

1. Les processeurs RISC (Reduced Integration Set Computing) : ARM, ...
   Le processeur n'a qu'un nombre limité d'instructions de base

2. Les processeurs CISC (Complex Integration Set Computing) : Intel, AMD
   Le processeur a un grand nombre d'instructions



Le langage machine ARM est constitué de mots de 32 bits (4 octets) dont le sens est donné ci-dessous par une copie d'écran d'un manuel d'un processeur ARM
Le langage machine des processeurs x86 INTEL est différent mais la philosophie générale est dans l'ensemble la même

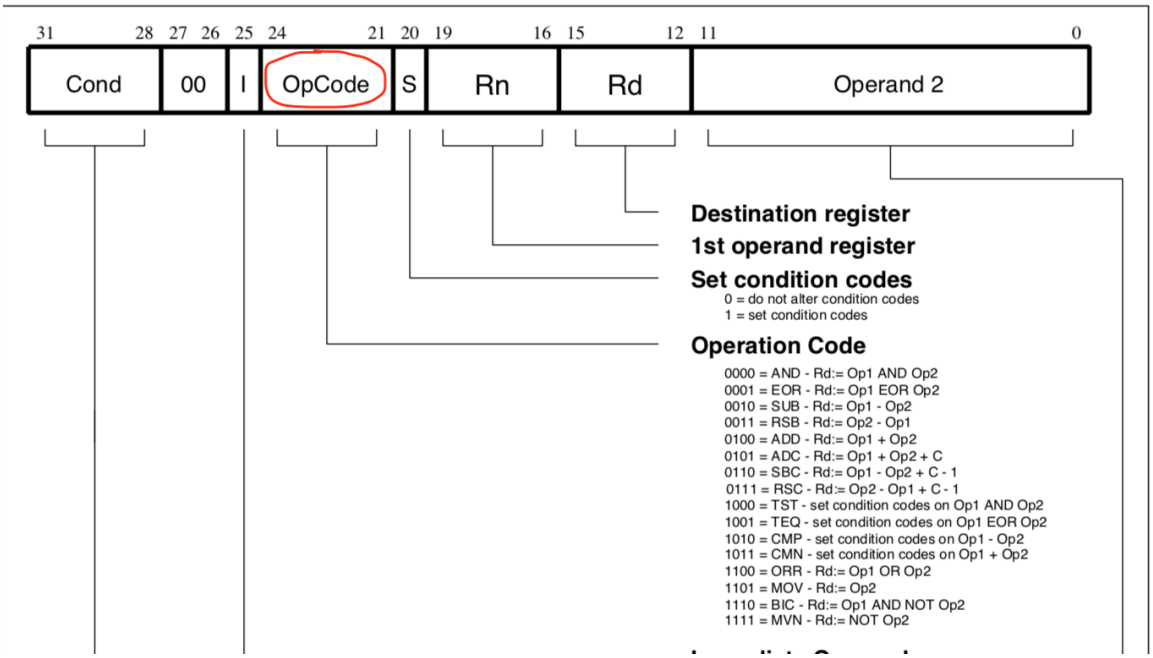
l' opCode se trouve sur les 4 bits de 21 à 24 ainsi par exemple **1101** signifie mettre une valeur numérique dans un registre alors que **0100** signifie ajouter une valeur ou le contenu d'un registre au contenu d'un autre registre et mettre le résultat dans un registre de destination

Ainsi 11100011 10100000 00000000 00000100 signifie en langage machine "Mettre la valeur numérique 4 dans le registre R0" ce qui est codée dans le langage **assembleur** MOV R0, #4
Voici "quelques phrases" du langage assembleur :
Les calculs ou les tests logiques se font avec l'UAL et les registres

1. **MOV R1,#4** signifie "Mettre la valeur décimale 4 dans le registre R1"
2. **MOV R1,R0** signifie "Mettre la valeur contenue dans le registre R0 dans le registre R1"
3. **ADD R1,R0,#4** signifie "Ajouter 4 à la valeur contenue dans R0 et mettre le résultat dans R1"
4. **ADD R2,R1,R0** signifie "Ajouter le contenu de R0 à celui de R1 et mettre le résultat dans R2"
5. **SUB R1,R0,#2** signifie "Soustraire 2 à la valeur contenue dans le registre R0 et mettre le résultat dans le registre R1"

The data processing instruction is only executed if the condition is true. The conditions are defined in ***Table 4-2: Condition code summary*** on page 4-5.

The instruction encoding is shown in ***Figure 4-4: Data processing instructions*** below.



Le lien entre les registres et la mémoire. On ne peut pas utiliser des valeurs numériques avec LDR et STR

1. **STR R0,1** signifie "Enregistrer (Store Register) la valeur contenue dans le registre R0 à la place en mémoire repérée par l'adresse 1"
2. **LDR R1,3** signifie "Mettre la valeur enregistrée dans la mémoire à l'adresse 3 dans le registre R1"

Le rôle de la commande HALT est d'arrêter l'exécution du programme et de séparer la zone du programme en mémoire de la zone des données.
Par exemple on peut utiliser des étiquettes (labels) comme des noms de variables pour stocker des valeurs numériques en mémoire, ainsi dans le programme suivant `nombre1` et `nombre2` sont des noms de variables apparaissant après HALT
LDR R0,nombre1
LDR R1,nombre2
ADD R1,R0,R1
HALT
nombre1 : 3
nombre2 : 4

Branchements ou sauts
Pour faire des tests et des boucles on peut faire des branchements après des comparaisons
B 4 est une instruction qui "fait sauter" le programme à l'instruction se trouvant à l'adresse mémoire 4
B `findesi` est une instruction qui "fait sauter" le programme à l'instruction se trou-

vant à l'adresse étiquetée par le mot findesi
On peut "conditionner " un saut à une comparaison dans ce cas on fait succéder dans l'ordre d'abord une comparaison avec le contenu d'un registre, puis un saut conditionné par le résultat de la comparaison précédente
CMP R0,#val permet de comparer la valeur val au contenu du registre
En fonction du résultat de la comparaison on peut faire différents sauts :
BEQ `si` si c'est vrai (1) le programme saute à l'étiquette si
BNE `sinon` si c'est faux (0) le programme saute à l'étiquette sinon
BLT `pluspetit` si le contenu de R0 est strictement plus petit que la valeur, le programme saute à l'étiquette pluspetit
BGT `pluspetit` si le contenu de R0 est strictement plus grand que la valeur, le programme saute à l'étiquette plusgrand
Par exemple on entre un nombre entier si ce nombre est égal à 4 on lui ajoute 1 sinon on lui retranche 1

```
INP R0,2
CMP R0,#4
BNE sinon
ADD R0,R0,#1
B findesi
sinon :
SUB R0,R0,#1
findesi :
OUT R0,4
HALT
```

On aurait pu aussi écrire (en nous obligeant à gérer nous même la mémoire)

```
INP R0,2 (adrr 0)
CMP R0,#4 (adrr 1)
BNE 5 (adrr 2)
ADD R0,R0,#1 (adrr 3)
B 6 (adrr 4)
SUB R0,R0,#1 (adrr 5)
OUT R0,4 (adrr 6)
HALT (adrr 7)
```

Boucles
On entre un nombre au clavier et on répète 4 fois on ajoute 5 à ce nombre :

```
INP R0,2
MOV R1,#0
boucle :
CMP R1,#3
BGT finDeBoucle
ADD R0,R0,#5
ADD R1,R1,#1
B boucle
finDeBoucle :
OUT R0,4
HALT
```

5.2.2 Exercices : Assembleur

Ex 1

Au musée des Arts et Métiers à Paris sont exposés des ordinateurs anciens comme l'ordinateur IBM 7030 et le CRAY 2 . En allant sur le Web rechercher les informations suivantes

1. Taille de l'ordinateur
2. Prix à l'époque de l'ordinateur
3. Nombre de transistors
4. Fréquence de l'horloge
5. Comparer avec les caractéristiques de votre smartphone et de votre ordinateur portable (processeur,fréquence d'horloge)

Ex 2

Aller sur le site d'Intel et relever les caractéristiques des processeurs INTEL de 2020 . Que signifie le terme lithographie ?

Ex 3

A l'aide du simulateur vérifier les opCodes de ADD et SUB

Ex 4

1. Que signifie l'instruction **ADD** R0,R0, #10 ?
2. L'instruction suivante est elle valide **ADD** R0,R0, R0 ?
3. Ecrire en assembleur successivement "La prochaine instruction à exécuter se situe en mémoire vive à l'adresse 100. Si la valeur contenue dans le registre R0 est égale à 1 alors la prochaine instruction à exécuter se situe à l'adresse 90"

Ex 5

Faire un programme en Assembleur pour obtenir le nombre 84

1. à partir de 50, 25, 8, 7, 3 et 1
2. à partir de 100 , 25, 8, 7, 3 et 1

Ex 6

1. Comprendre en Python l'instruction 5 » 2 et 5 « 2 (à la console)
2. Lire dans la documentation le sens des mnémoniques LSR et LSL

Ex 7

Ecrire en assembleur un programme qui affiche le plus grand de deux entiers entrés au clavier

Ex 8

Ecrire en assembleur : "Si un nombre entré est divisible par 2 afficher la moitié de ce nombre sinon afficher le triple de ce nombre plus 1"

Ex 9

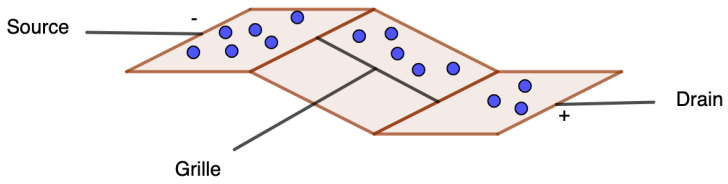
Ecrire en assembleur un programme qui calcule la somme $1 + 2 + 3 + ... + n$ où n est un entier entré au clavier

Ex 10

Aller voir sur le Web l'algorithme de la multiplication égyptienne. Coder cet algorithme en Python, en Javascript et en assembleur

5.3 Le processeur en tant qu'assemblage de transistors

Le transistor (1947)



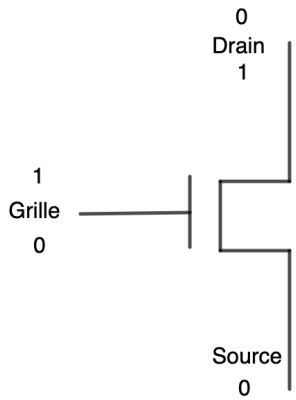
De manière imagée un transistor est constituée de trois éléments :

1. Une **source** d'où vont "partir" des électrons
2. Un **drain** (analogie avec l'écoulement de l'eau) qui va "collecter" ces électrons
3. Une **grille** qui va "freiner" l'écoulement des électrons de la source vers le drain

Voici de manière simplifiée le fonctionnement du transistor

1. Lorsque la tension de la grille est égale à une valeur seuil ($\simeq 2,9\ V$) il n'y a plus de déplacement d'électrons **il n'y a pas de tension ou de différence de potentiel entre la source et le drain**
   Symboliquement on dit que la grille est à 1 et le drain et la source sont à 0
2. Lorsque la tension de la grille est nulle il y a déplacement d'électrons entre la source à 0 et le drain à 1
   Symboliquement on dit que la grille est à 0 et le drain est à 1 et la source est à 0

Voici la représentation symbolique d'un **transistor NMOS** (Metal Oxyde Semiconductor de type N) Sans entrer dans les détails on réalise encore la fonction logique

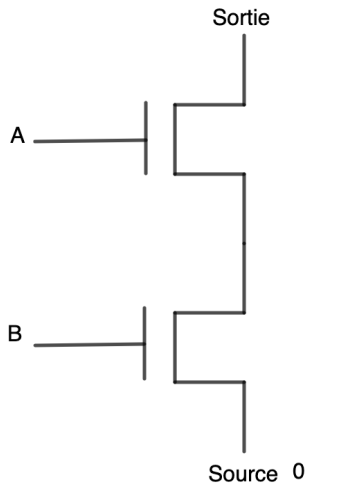


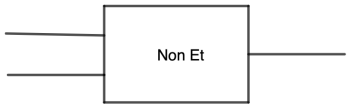
`non` mais dans un volume "très petit" et de manière plus fiable et moins énergivore
Pour éviter des "courants de fuite" et de la dissipation d'énergie sous forme de chaleur on associe au transistor, non pas une résistance ce qui serait un frein à la miniaturisation mais un transistor complémentaire de type P jouant le rôle de résistance
Pour montrer le coté concret de la fonction logique `non` on parle de **porte logique** `non` que l'on représentera ainsi

Comment ensuite construire les autres portes ?

la porte Non Et (Nand)

En mettant deux transistors NMOS en série on réalise la porte logique Non Et (Nand en Anglais) représentée à partir de maintenant sous la forme symbolique Ensuite à partir





de la porte Non Et et la porte Non on réalise les autres portes par assemblage successifs

5.3.1 Exercices : Portes Logiques

Ex 1

1. Vérifier qu'en mettant deux transistors NMOS en parallèle on réalise la porte logique **Non Ou**
2. En déduire la porte **Ou** avec 3 transistors
3. Vérifier que x ou $y \iff$ non(non(x) et non(y)) en déduire une réalisation de la porte logique **Ou** à partir des portes **Non et** et **Non**. Combien de transistors ?

Ex 2

Construire un circuit réalisant le **Ou exclusif**

Ex 3

On définie la fonction **multiplexeur** par mux(a,b,c) = (non(a) et b) ou (a et c)
On dit que a est le **sélecteur** qui permet d'aiguiller et de faire passer sur le même fil deux bits.
Si le sélecteur est à 1 c'est le bit c qui passe sinon c'est le bit b
Construire un circuit réalisant cette fonction
On suppose que l'on a désormais une porte logique **mux**

Ex 5

On veut réaliser un circuit **additionneur 1 bit** avec retenue d'entrée cin et de sortie cout.
Ce circuit a 3 entrées a,b et cin et deux sorties s (la somme) et cout la retenue de sortie

1. Construire la table de vérité de la fonction cout(a,b,cin)
2. En considérant cin comme le **sélecteur** de la fonction multiplexage, vérifier que cout(a,b,cin) = mux(cin,a et b,a ou b)
3. Construire la table de vérité de la fonction s(a,b,cin)
4. En considérant cin comme le **sélecteur** de la fonction multiplexage, exprimer s(a,b,cin) en fonction de mux
5. En déduire la réalisation d'un circuit additionneur 1 bit

Ex 6

1. Construire un circuit à quatre entrées et trois sorties qui ajoute deux nombres sur deux bits
2. Construire un circuit à seize entrées et neuf sorties qui ajoute deux octets

Ex 7

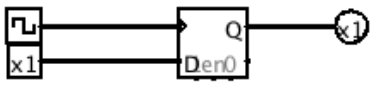
Construire un circuit à 9 entrées $a_0, ..., a_8$ et d et 8 sorties $b_0, ..., b_8$ de telle sorte que :

1. Si $d = 1$ alors $b_i = a_i$
2. Si $d = 0$ alors $b_i = a_{i-1}$ pour i de 1 à 7 et $b_0 = 0$

Ce circuit réalise un décalage d'un bit sur la gauche ce qui revient à multiplier par 2

5.4 Le temps et la mémoire

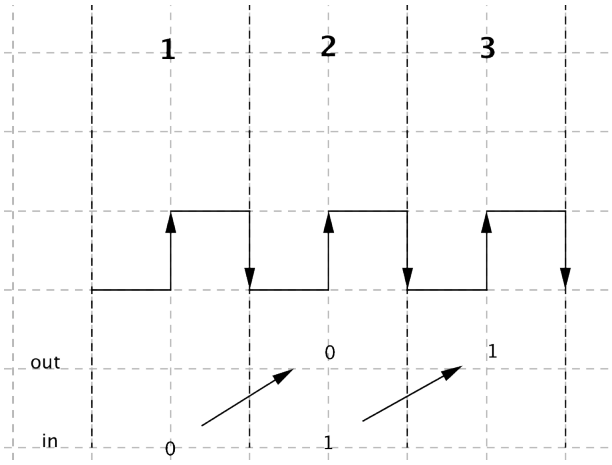
La mémoire est liée au temps. Sans temps pas de mémoire.
L'élément de base pour les circuits **séquentiels** est le data-flip-flop L'entrée et la sortie



du data flip flop sont des fonctions du temps `entrée(t)` et `sortie(t)` et le data flip flop est caractérisé par :

`sortie(t)` = `entrée(t - 1)`

On voit sur le schéma suivant que la valeur de la sortie est celle de l'entrée au cycle précédent

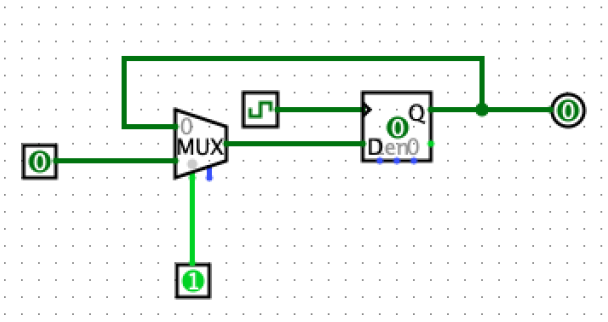


A partir du Data Flip Flop on construit le **registre à 1 Bit** premier élément **capable de mémoriser un bit** dont le contrat est le suivant :

1. Il a 3 entrées : une pour l'horloge, une pour la donnée (1 Bit) et enfin une pour un sélecteur (avec un multiplexeur)
2. une sortie

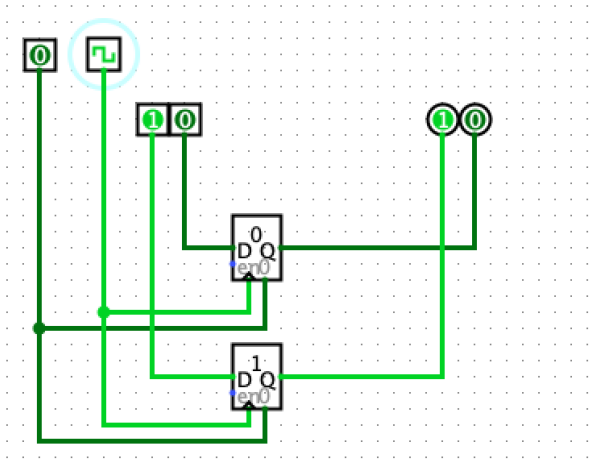
si `sel = 1` alors `sortie(t)` = `entrée(t - 1)`
si `sel = 0` alors `sortie(t)` = `sortie(t - 1)` (**mémorisation**)

voici le circuit



Register.png

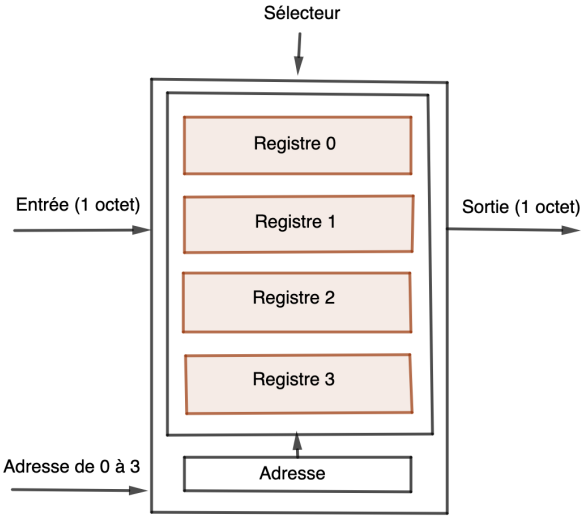
A partir du registre 1 bit on construit un registre à 2 bits et ainsi de site



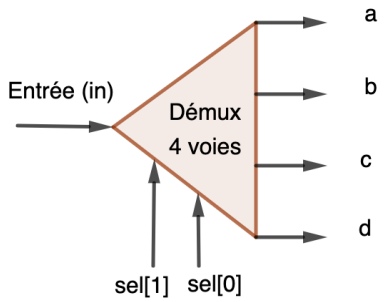
5.4.1 Exercices : Temps et mémoire

Ex 1

La mémoire RAM (Random Acces Memory). Ce terme vient du fait que l'accès à cette mémoire en lecture et écriture pour chaque élément de la mémoire se fait avec la même vitesse.
On veut construire un circuit RAM avec 4 registres de 8 bits chacun, en entrée



Pour sélectionner le registre que l'on veut modifier en écriture en fonction de son adresse on utilise un Démultiplexeur 4 voies



Voici les caractéristiques du démultiplexeur 4 voies :
Par exemple si on veut modifier le contenu du registre 1 il faut que `in` soit égal à 1 et `sel[0]` soit égal à 0 et `sel[0]` à 1, ainsi le bit à 1 "ressort" par `b` qui est relié au sélecteur du registre 1 qui va donc être modifié en écriture au prochain cycle d'horloge

sel[1]	sel[0]	a	b	c	d
0	0	in	0	0	0
0	1	0	in	0	0
1	0	0	0	in	0
1	1	0	0	0	in

Ex 2

On a vu dans la partie assembleur le rôle important joué par le registre PC qui contient l'adresse mémoire de la **prochaine** instruction exécutée par le processeur. La plupart du temps l'adresse mémoire est incrémentée de 1 à chaque cycle d'horloge sauf lorsqu'il y a un saut dans le programme.
Le registre PC est semblable à un registre "normal" avec en plus du bit `sel` servant à le modifier en écriture, il y a deux bits de contrôles supplémentaires, `inc` pour incrémentre de 1 et `reset` pour remettre à 0
Voici leur **intérêt** :

```
si reset(t-1) = 1 alors out(t) = 0
sinon si load(t-1) = 1 alors out(t) = int(t-1)
sinon si inc( t-1) = 1 alors out(t) = out(t-1) + 1
sinon out(t) = out(t-1)
```

5.5 Réseaux

5.5.1 Réseau local associé à une box

Les fournisseurs d'accès à internet louent des box, ce qui permet aux particuliers d'être connectés à Internet. Il est possible aussi d'accéder à **l'interface** de la box et d'avoir ainsi des informations sur la box et le réseau local associé.
Regardons un exemple de réseau local associé à une box :

Ex 1

Demandez à vos parents de pouvoir avoir accès à l'interface de votre box et de pouvoir voir les informations du réseau local.(Pour Orange dans la barre URL de votre navigateur entrer 192.168.1.1 (on verra plus loin ce que c'est))

La plupart du temps nous retrouvons les même éléments :

1. un **routeur -commutateur** appelée communément **box** louée à un fournisseur d'accès à internet (F.A.I) :
   (a) le **routeur** sert à router les informations entre les ordinateurs du réseau local vers les ordinateurs extérieurs au réseau local
   (b) le **commutateur** (switch) sert à aiguiller les informations entre les éléments du réseau local. Par exemple il peut y avoir une panne sur internet mais le réseau local continue de fonctionner
2. des ordinateurs connectés à ce routeur (smartphones, tablettes, ordinateurs,tv,imprimante,...) pour pouvoir aller sur internet (le réseau mondial)
3. On constate qu'il y a un réseau **filaire** et un réseau **sans fil** (wi-fi)

Comment les différents ordinateurs sont différenciés et reconnus par la box ?
La box est aussi un **serveur DHCP** (Dynamic Host Configuration Protocol) c'est à

Baux DHCP valides			
nom	adresse IP		adresse MAC
[redacted]	IPv4 :	192.168.1.11	38:94:96:8E:3A:05
[redacted]	IPv4 :	192.168.1.21	A0:99:9B:18:68:79
vhjp	IPv4 :	192.168.1.22	94:DB:C9:0D:F0:E4
[redacted]	IPv4 :	192.168.1.16	1C:91:48:79:75:A5
[redacted]	IPv4 :	192.168.1.10	7C:03:D8:37:E4:09
[redacted]	IPv4 :	192.168.1.14	80:4E:70:03:0D:70
[redacted]	IPv4 :	192.168.1.12	10:F0:05:89:F0:AD
vhjp-1	IPv4 :	192.168.1.22	10:BF:48:03:B7:65

dire qu'à chaque ordinateur qui se connecte au réseau local , la box attribue un identifiant unique sur le moment **l'adresse IP** pour **Internet Protocol** une série de 4 nombres entre 0 et 255 (1 octet)
Dans notre exemple l'ordinateur appelé vhjp a pour adresse IP 192.168.1.22 (voir tableau).
Si un ordinateur se déconnecte puis se reconnecte un peu plus tard il peut très bien avoir une autre adresse IP, voilà pourquoi on parle **d'adressage dynamique**
Parmi les élèves de 1NSI du LVC certains ont peut-être déjà paramétré leur box pour avoir une adresse IP **statique** (fréquent chez les joueurs et les utilisateurs de peer to peer)

L'ordinateur nommé vhjp-1 est en fait le même que vhjp, la box différencie le mode de connexion l'un est filaire (par câble **ETHERNET**) avec une adresse **MAC** (pour Media Access Control) 10 :BF :48 :03 :B7 :65 et l'autre est sans fil (wi-fi) pour (wireless fidelity) avec l'adresse MAC 94 :DB :C9 :0D :F0 :E4
Un ordinateur actuellement est équipé de plusieurs cartes réseaux et chaque carte réseau a une adresse MAC unique (6 octets écrits en hexadécimal)

Ex 2

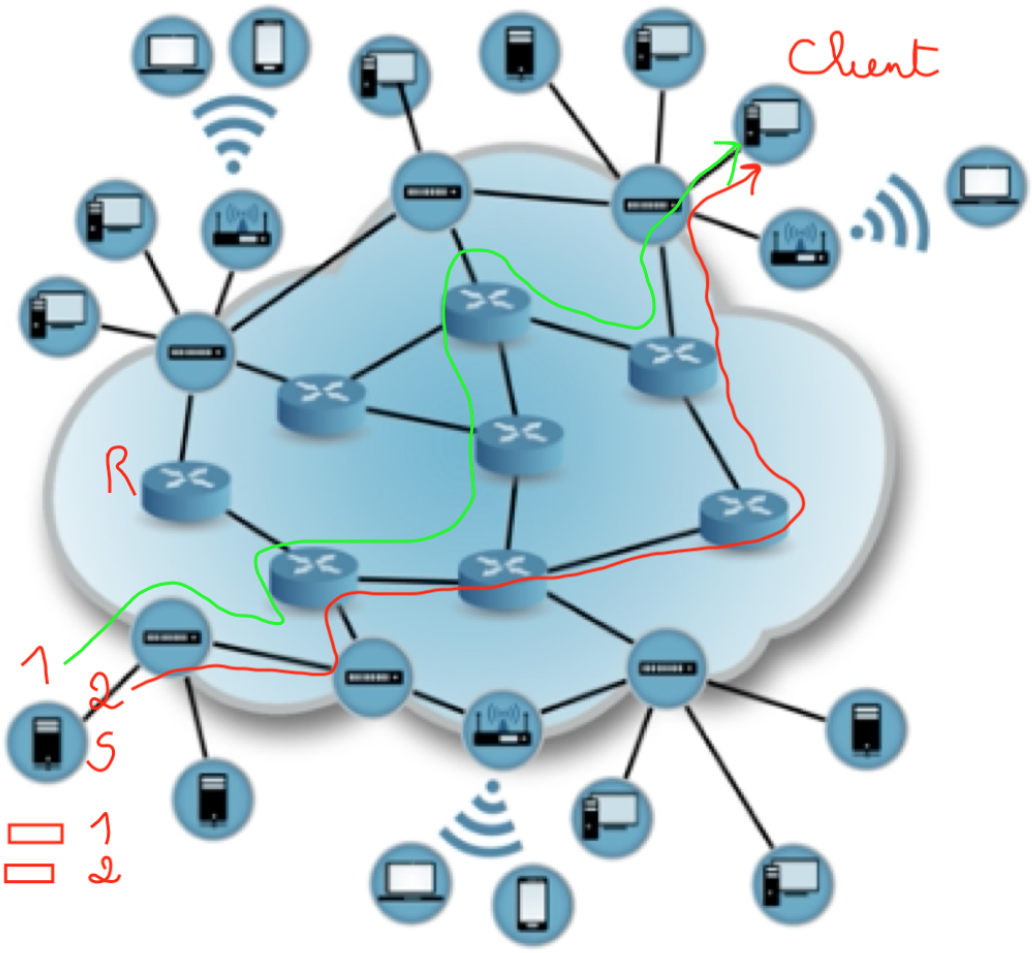
1. Connecter votre ordinateur portable par wi-fi à votre réseau local et aussi par câble RJ45 et regarder dans l'interface de la box l'adresse IP attribuée par votre box à votre ordinateur et les deux adresse MAC
2. Retrouver ces informations en allant dans la console Windows de votre ordinateur portable
   (a) Ouvrir un terminal (pour les machines Windows faire : Windows + R puis dans la barre entrer cmd puis exécuter
   (b) Dans le terminal entrer la commande ifconfig pour les machines UNIX ou ipconfig pour les machines Windows
   (c) Relever les paramètres réseau de votre machine : Le type de connexion (filaire ou non filaire), L'adresse IPv4 de votre machine, L'adresse IPv4 de la passerelle (routeur d'entrée du réseau local = votre box), En passant par le Web et le site monip.com relever l'adresse IP publique de votre machine
   (d) Repérer dans l'interface de votre box un autre élément du réseau local, un ordinateur, une imprimante etc...et relever son adresse IP (locale) disons par exemple 192.168.1.15

Dans le terminal de votre ordinateur entrer ping 192.168.1.15 et exécuter Arrêter le ping avec CTRL -C puis noter le temps moyen aller-retour(round-trip avg) de connexion entre votre ordinateur et celui de votre voisin Noter vos observations (Vous venez de tester la connexion entre votre ordinateur et l'autre élément du réseau)

(e) Recommencer avec un ordinateur d'adresse IP 202.158.196.134 (situé en Australie)

5.5.2 Interconnexion des réseaux

Tous les réseaux locaux sont interconnectés. Voici une image globale et symbolique (simplifiée) de cette interconnexion



Imaginons qu'un client (en haut de l'image) veuille télécharger un document relativement volumineux (un pdf de plusieurs pages) chez un serveur S (en bas à gauche)

Le document est **découpé** en paquets étiquetés et numérotés puis ces paquets vont suivre **des trajectoires indépendantes** le long du trajet du serveur au client.
Les routeurs R (symboles avec 4 flèches comme des carrefours) font la circulation des paquets, de manière autonome à l'aide d'algorithmes
On a dessiné pour l'exemple deux trajectoires différentes pour les paquets 1 et 2

Ensuite chez le client le document est reconstitué en tenant compte de la numérotation des paquets.

l'idée de la commutation des paquets signifie donc que :

1. Le document au delà d'une certaine taille (1500 octets) est découpée en paquets
2. Ces paquets sont commutés (aiguillés) indépendamment les uns des autres en fonction de la densité de circulation sur le réseau

5.5.3 Adressage IPv4 et IPv6

Les adresses IP ci-dessus sont **privées** et dans tous les réseaux domestiques il y a les mêmes adresse privées de 192.168.1.11 à 192.168.1.140

Pourquoi n'a-t-on pas donné une adresse IPv4 unique à chaque ordinateur dans le monde ?

Il n'y en a plus assez surtout qu'à l'heure actuelle (2019-2020) on parle de **l'Internet des objets (IOT)**, c'est à dire la connexion aussi de robots divers etc.... En prévision de l'augmentation des objets connectés on a prévu l'adressage IPv6 (Internet protocol version 6) : une adresse IPv6 est constitué de 8 groupes de 2 octets or chaque octet est constitué de 8 bits donc une adresse est constituée de $8 \times 2 \times 8 = 128$ bits ce qui fait 2^{128} adresses IPv6 possibles

Ex 3

1. En utilisant l'approximation $2^{10} \simeq 10^3$ convertir 2^{128} en puissance de 10
2. Convertir la superficie de la Terre qui est de 510 millions de km^2 en cm^2
3. Combien peut-on mettre d'adresses IPv6 par cm^2 de la surface de la Terre ?

Comment alors mon ordinateur personnel peut-il être reconnu dans internet ?

Si on prend deux ordinateurs différents du réseau local précédent on observe qu'ils ont la même adresse publique 90.92.205.162 qui est en fait l'adresse IP de la box dans internet

(Faire l'expérience chez vous de trouver votre adresse IP publique en passant par un site Web comme `monip.com`)

Les deux ordinateurs ont la même adresse IP **publique** mais pas la même adresse privée. La box sert de traducteur entre les deux adresses, on dit que la box est aussi un serveur **NAT** (pour Network Adress Translation)

5.5.4 DNS (facultatif)

Puisque chaque ordinateur est repéré par une adresse IP comment l'information peut circuler de mon ordinateur à un ordinateur distant dont je ne connais pas l'adresse mais uniquement le nom de domaine par exemple youtube.com ou fr.wikipedia.org

De la même manière il nous est plus facile de retenir les noms de nos proches que leur numéro de téléphone et nous avons besoin d'un annuaire qui fait le lien entre nom et numéros de téléphone.

Au début d'Internet ces annuaires étaient des fichiers texte peu volumineux disponibles

Le serveur DNS sert à convertir les noms de domaine en adresses IP et inversement.

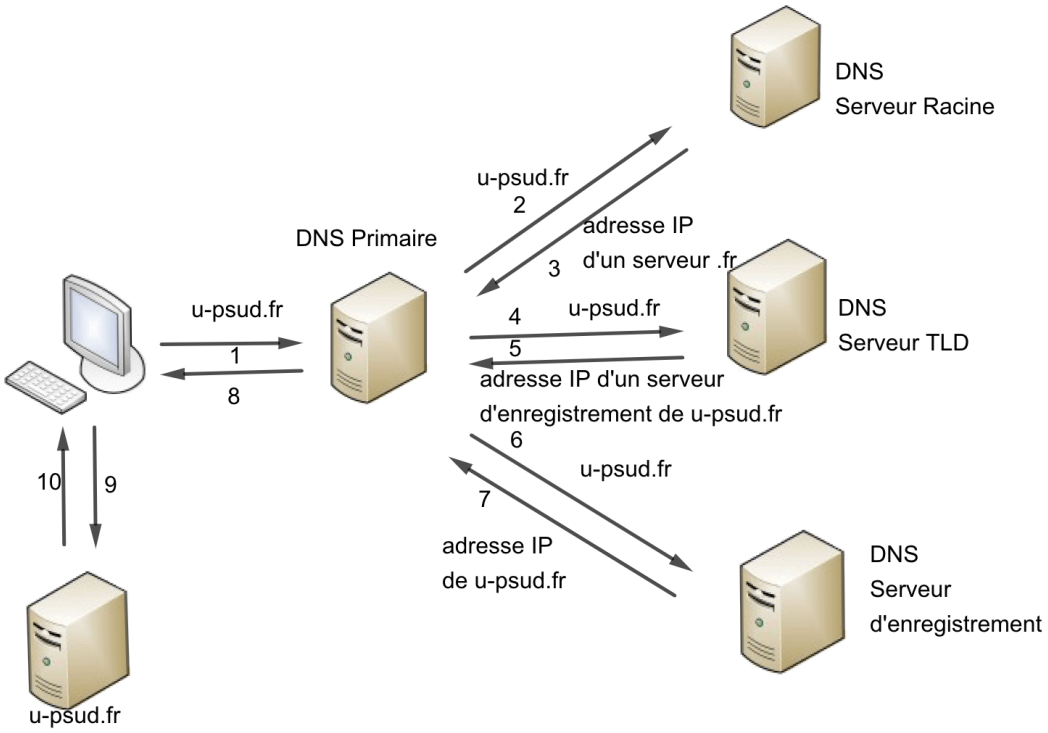
Configuration DNS

DNS primaire IPv4	80.10.246.1
DNS secondaire IPv4	81.253.149.9

sur chaque ordinateur individuel mais avec la croissance d'Internet il a fallu **distribuer** ces connaissances sur des serveurs.

Des serveurs traduisent les **noms de domaines** en adresses IP on les appelle serveurs DNS pour Domain Name System
Par exemple la box du réseau local étudié est en relation avec deux serveurs DNS

DNS.png

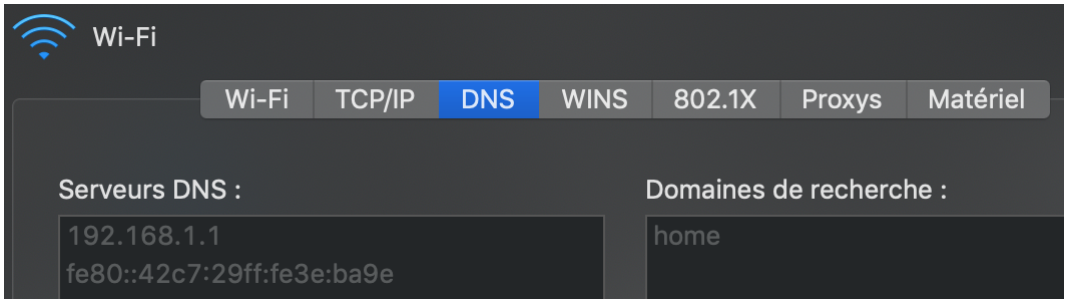


En passant par le site de RIPEstat `https:stat.ripe.net` on observe que l'adresse IP 80.10.246.1 est un serveur DNS d'Orange
Imaginons que nous voulons entrer en relation avec le serveur de l'université d'Orsay dont le nom de domaine est `u-psud.fr`

1. Dans un premier temps le client 192.168.1.21 va par l'intermédiaire de la box envoyer la requête `u-psud.fr` au serveur DNS primaire 80.10.246.1

2. Ce dernier va transmettre la requête à un serveur DNS racine qui va retourner au serveur DNS primaire l'adresse IP d'un serveur DNS .fr
3. Le serveur DNS primaire va transmettre la requête `u-psud.fr` à ce serveur DNS .fr qui va lui retourner l'adresse IP d'un serveur "sûr" qui authentifie l'adresse IP de `u-psud.fr`. (Ce dernier serveur joue un rôle analogue au services de l'Etat qui authentifient les actes de propriété)
4. Finalement le serveur DNS primaire envoie `u-psud.fr` à ce serveur d'authentification qui va lui retourner l'adresse IP de `u-psud.fr`. Tous ces échanges n'ont duré en tout qu'une dizaine de millisecondes
5. Maintenant le client 192.168.1.21 peut entrer en relation avec le serveur de `u-psud.fr`

Sur chaque ordinateur personnel on peut modifier le DNS dans les préférences système de l'ordinateur On voit ici que le serveur DNS est celui lié à 192.168.1.1 c'est à



dire la box, cependant on peut si on le veut ajouter un autre D.N.S comme le 1.1.1.1

5.5.5 Systèmes autonomes

```
[macbook-pro-de-vallon:~ vallon$ traceroute example.com
traceroute to example.com (93.184.216.34), 64 hops max, 52 byte packets
 1  livebox (192.168.1.1)  2.157 ms  1.293 ms  1.197 ms
 2  80.10.235.169 (80.10.235.169)  2.943 ms  4.355 ms  2.879 ms
 3  ae107-0.ncidf203.aubervilliers.francetelecom.net (193.249.213.242)  3.512 ms
  4.225 ms  3.420 ms
 4  ae41-0.niidf201.aubervilliers.francetelecom.net (193.252.98.161)  3.752 ms
3.648 ms  3.675 ms
 5  81.253.184.182 (81.253.184.182)  3.814 ms  3.854 ms  4.202 ms
 6  hundredgige0-2-0-3.partr1.paris.opentransit.net (193.251.133.77)  4.563 ms
    hundredgige0-2-0-2.partr1.paris.opentransit.net (193.251.133.125)  4.629 ms
*
 7  prs-b5-link.telia.net (62.115.171.226)  5.226 ms  3.988 ms  3.939 ms
 8  prs-bb3-link.telia.net (213.155.130.20)  91.108 ms
    prs-bb4-link.telia.net (213.155.130.22)  92.986 ms  87.100 ms
 9  ash-bb4-link.telia.net (62.115.112.242)  89.679 ms
    ash-bb3-link.telia.net (62.115.122.159)  87.647 ms
    ash-bb4-link.telia.net (62.115.112.242)  88.171 ms
10  ash-b1-link.telia.net (62.115.143.79)  81.171 ms  81.409 ms
    ash-b1-link.telia.net (213.155.136.39)  87.082 ms
11  verizon-ic-315152-ash-b1.c.telia.net (213.248.83.119)  91.335 ms  87.447 ms
 88.381 ms
12  152.195.64.129 (152.195.64.129)  88.158 ms  82.113 ms  88.100 ms
13  93.184.216.34 (93.184.216.34)  85.790 ms  86.898 ms  81.234 ms
```

La connexion entre un client et un serveur n'est pas directe, il existe de nombreux relais entre le client et le serveur on appelle ces relais des **routeurs** et leur rôle est fondamental dans l'acheminement des données entre le client et le serveur grâce à l'adressage IP
Dans un terminal si on entre la commande `traceroute example.com` ou `tracert example.com`, on observe la suite des routeurs reliant la box 192.168.1.1 au serveur 93.184.216.34 hébergeant le domaine example.com
A l'aide du site RIPEstat on peut ranger les routeurs en trois groupes ou **systèmes autonomes** le premier celui de ORANGE, le second celui de TELIA et enfin le dernier celui de EDGE-CAST
Il existe plus de 50 000 systèmes autonomes, on peut voir ici les principaux systèmes autonomes `http://as-rank.caida.org`.
Au lieu de regarder Internet comme un ensemble d'ordinateurs interconnectés dans le monde on peut regarder Iaussi nternet comme un ensemble de réseaux (les systèmes autonomes) interconnectés
Un organisme (Center for Applied Internet Data Analysis) a proposé une visualisation d'Internet à partir des systèmes autonomes (Voir `http://caida.org`)

Ex 3

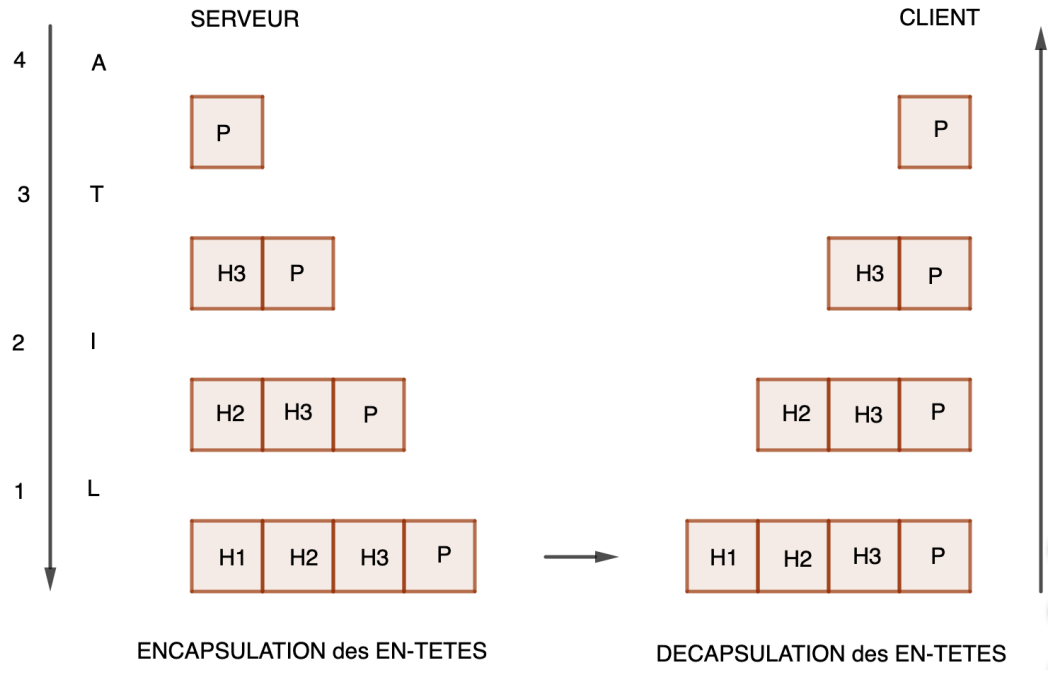
1. Dans le terminal entrer la commande tracert education.gouv.fr
   Vous observez une adresse IP entre parenthèses, notez la, c'est l'adresse IP du serveur hébergeant le site du ministère de l'éducation nationale
   Notez la suite des adresses IP (celle des routeurs)
   Puis allez sur le site RIPEstat pour avoir des informations sur ces adresses IP.
   A quels systèmes autonomes appartiennent ces adresses IP ?
2. Refaire la même manipulation avec les sites des musées d'art moderne de Rome et de New-York
3. Enfin faire tracert 202.158.196.134

5.5.6 Modèle TCP/IP

La circulation des informations dans Internet suit le protocole ou modèle TCP/IP (pour Transport Control Protocol/ Internet Protocol) crée en 1973

1. **Une application (A)** est un programme nous permettant par exemple de visionner une vidéo ou lire une page Web. Lorsqu'une source nous envoie une page Web, une application va transmettre le contenu de cette page à la **couche Transport (T)** de la même manière que l'on peut s'adresser à la Poste pour transporter un colis

   Plusieurs applications peuvent d'ailleurs utiliser en même temps les services de la couche transport par l'intermédiaire des **ports** caractérisés par un nombre entre 0 et 65535
   Par exemple le protocole http a comme numéro de port 80, le protocole ftp (File Transfert Protocol) le numéro 21
2. TCP qui ici est chargé de la **couche Transport** va découper le contenu probablement trop volumineux en **paquets** puis étiqueter chaque paquet P d'un



certain nombre d'informations, une en-tête (le destinataire , le numéro du paquet, mais aussi des informations permettant de **sécuriser l'envoi** et aussi de gérer d'éventuels "bouchons" dans Internet.
Ensuite TCP transmet les paquets étiquetés (H3-P) à la couche Internet

3. IP est chargé de la **couche Internet** . IP se charge de l'acheminement des paquets de bout en bout par l'intermédiaire des **routeurs**.
   Chaque routeur a une **table de routage**, mis à jour régulièrement, permettant de transmettre un paquet à un routeur voisin en fonction de la destination de ce paquet
   Les paquets d'un même document ne prendront pas forcément le même chemin dans Internet pour arriver à destination. IP ajoute une en-tête H2 sur chaque paquet et transmet les paquets à la couche **Liaison**
4. La couche **Liaison** se charge d'envoyer les paquets dans Internet par différents type de liaison (filaire avec le protocole ETHERNET et non filaire avec le protocole wi-fi). En fonction du protocole on a une en-tête H1 différente ajouté au paquet H2-H3-P

De la couche Application à la couche Lien c'est la **phase d'encapsulation**

Maintenant le paquet H1-H2-H3-P va être routé dans le réseau jusqu'à destination surtout grâce à H2 qui est changé à chaque routeur traversé en fonction de l'état du réseau (pannes, "bouchons")

Arrivés à destination les en-têtes sont enlevés des paquets de la couche liaison vers la couche application et TCP se charge de remettre les paquets dans l'ordre et de fournir l'information demandée à l'application

Ex 4

Dans l'interface de votre box vous verrez que l'on parle d'un autre protocole de la couche Transport, le protocole UDP (moins sécurisé que TCP)

Allez chercher sur le Web à quoi est utilisé le protocole UDP

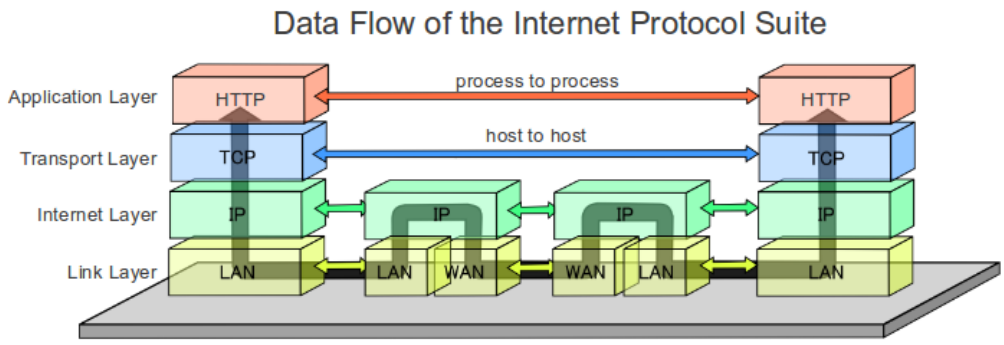


FIGURE 5.1 – Source :Wikipedia

5.5.7 Histoire

1. **1969** : Création du réseau ARPANET reliant quelques universités américaines
2. **1972** : Premier essai de ARPANET
3. **1973** : Création du protocole TCP/IP par Kahn et Cerf (prix Turing 2004)
4. **1983** : Le protocole TCP/IP devient le seul protocole sur ARPANET qui devient Internet
5. **1983** : Apparition du DNS
6. **1990** : Création du World Wide Web par Tim Berners-Lee (Prix Turing 2016) au CERN de Genève

5.5.8 QCM

Question 25 Dans un réseau informatique que peut on dire de la transmission des données par paquets ?

☐ Cela garantit que toutes les données empruntent le même chemin
☐ Cela nécessite la réservation d'un chemin entre l'émetteur et le récepteur
☐ Cela empêche l'interception des données transmises
☐ Cela assure une utilisation efficace des liens de connexion

Question 26 Quelle est l'utilité de la commande `ping` dans un réseau informatique ?

☐ tester si la connexion peut être établie avec une machine distante
☐ obtenir la route suivie par un paquet dans le réseau
☐ établir un réseau privé virtuel
☐ mesurer les performances d'une machine distante

Question 27 Citer les couches du modèle TCP/IP, en partant de la couche Application

☐ Application - Transport - Internet - Liaison
☐ Application - Internet - Transport - Lien
☐ Application - Transport - Internet - Réseau
☐ Application - Internet - Transport - Réseau

Question 28 Une des adresses suivantes n'est pas une adresse IP ?

☐ 192.168.1.21
☐ 1.1.1.1
☐ 172.20.6.100
☐ 172.26.6.256

Question 29 Une seule des phrases suivantes est fausse

☐ ftp est un protocole de la couche Application
☐ udp est un protocole de la couche Transport
☐ ip est un protocole de la couche Internet
☐ ethernet est un protocole de la couche internet

Question 30 Les ports se situent entre :

☐ La couche Application et la couche Transport
☐ La couche Transport et la couche Internet
☐ La couche Internet et la couche Liaison
☐ La couche Liaison et la couche Réseau

Question 31 Quelle est l'utilité de la commande `tracert` dans un réseau informatique ?

☐ tester si la connexion peut être établie avec une machine distante
☐ obtenir la route suivie par un paquet dans le réseau
☐ établir un réseau privé virtuel
☐ mesurer les performances d'une machine distante

Question 32 Combien y -a -t-il d'adresses IP v4 possibles ?

☐2^{16}
☐2^{24}
☐2^{32}
☐2^{64}

5.6 Systèmes d'exploitation

5.6.1 Histoire

Pour comprendre l'intérêt des systèmes d'exploitation il faut faire un peu d'Histoire :

1. **1945-1957 : Première Informatique : l'ère des ordinateurs**

   (a) Technologie principale : Les lampes à vide.

   (b) Les ordinateurs sont peu nombreux coûtent très cher et consomment beaucoup d'énergie

   (c) Ils sont principalement utilisés par les universités et l'Armée pour des calculs scientifiques

   (d) Les ordinateurs ont à leur service de nombreux chercheurs, ingénieurs et techniciens pour leur fonctionnement

   (e) Progressivement apparaissent des modifications pour améliorer leur **ergonomie** (adaptation d'un environnement de travail (outils, matériel, organisation...) aux besoins de l'utilisateur), ainsi en 1953 avec l'IBM 701 l'entrée des données se fait avec des bandes magnétiques pour gagner du temps

   (f) Avec l'IBM 704 en 1955 (image ci-dessus) **pour optimiser le temps de calcul** d'une machine qui a coûté plusieurs millions de dollars, deux des principaux clients ayant acheté cet ordinateur, General Motors et North American Aviation ont créé **le premier système d'exploitation**, nommé GM-NAA I/O pour General Motors and North American Aviation Input Output System.

   Pourquoi et comment ?

   A l'époque le temps de calcul d'une machine comme l'IBM 704 était organisé suivant un **traitement par lots** :

Un **lot** est constitué d'une entrée des données et du programme (par bandes magnétiques) puis de l'exécution du programme et enfin l'impression des résultats du programme sur un télétype (une sorte d'imprimante)
Les lots devaient ensuite se succéder en cadence pour profiter au mieux de l'ordinateur

Mais la phase d'impression, purement mécanique, ralentissait la cadence aussi les ingénieurs ont eu l'idée de détourner la phase d'impression vers une phase de mémorisation sur une mémoire auxiliaire électromagnétique et **surtout que la gestion de toutes ces phases pour tous les lots allait se faire par un super-programme**, une sorte d'arbitre des programmes, placé en premier dans la mémoire de l'ordinateur
A l'époque on a appelé ce programme, un moniteur

Pour optimiser encore plus le temps, le moniteur pouvait faire chevaucher un temps de calcul pour un programme P_1 et un temps d'entrées-sorties pour un autre programme P_2 et ainsi de suite.
A partir de maintenant les programmes clients du processeur sont appelés des processus gérés par le système d'exploitation

(g) **En gestation :**
Le transistor apparaît en 1947 il remplacera progressivement les lampes à vide

Ex 1

Lire la vidéo suivante sur Youtube (en anglais) `https://www.youtube.com/watch?v=DKaVvv15Heo` concernant la machine IBM 704 et répondre aux questions suivantes :

(a) Quels sont les langages de programmation utilisés avec l 'IBM 704 ?

(b) Chercher sur wikipedia le volume occupé par l'IBM 704

2. 1957-1970 : Deuxième Informatique : l'ère des mini-ordinateurs

Les transistors ont remplacé les tubes à vide, la taille des ordinateurs a diminué on parle maintenant de mini-ordinateurs (la taille d'un réfrigérateur pour le PDP-8 de Digital Equipment (image ci-dessus))

Cependant un transistor occupe encore une certaine place, l'époque des circuits intégrés n'est pas encore arrivée

De plus la mémoire des ordinateurs est constituée de tores de ferrites ce qui occupe aussi de la place

Pour optimiser le rendement du processeur les systèmes d'exploitation évoluent aussi :

1961 : CTSS (Compatible Time-Sharing System), est le **premier système d'exploitation de temps partagé** développé au MIT

L'échelle de temps du processeur (la micro seconde à l'époque, la nano seconde actuellement en 2020) n'est pas celle de l'Humain (la seconde) Bien que **le concept de machine de Von Neumann implique que le processeur exécute chaque instruction l'une après l'autre** on peut encore gagner du temps en entremêlant les instructions des processus, le système d'exploitation étant une sorte d'agent de la circulation faisant circuler tel processus P_1 puis tel processus P_2 etc...

L'Humain a ainsi l'illusion que plusieurs processus s'exécutent en même temps ce qui est un progrès par rapport au traitement par lots

Avec le système CTSS apparaît un langage interprété **le shell** qui permet le lancement des programmes et le "dialogue" entre l'utilisateur et le système (on verra quelques exemples plus loin)

1965 : Multics (Multiplexed Information and Computing Service), successeur de CTSS, améliore la gestion et la sécurité de la mémoire :

(a) Mémoire segmentée

(b) Mémoire virtuelle paginée

(c) Système de gestion de fichiers (que l'on regardera un peu en détail plus loin)

1969 : Unix, successeur de Multics et ancêtre des systèmes d'exploitation,Linux, Mac Os des ordinateurs Mac de Apple, iOs et Android

Ex 2

Lire la vidéo suivante sur Youtube (en anglais) `https://www.youtube.com/watch?v=XvDZLjaCJuw` à propos de Unix

On peut y voir Ken Thompson qui avait travaillé sur le système Multics au sein des laboratoires Bell, et Dennis Ritchie qui a inventé le langage de programmation C

Ils ont écrit Unix en C et ont privilégié **la portabilité de leur système** (c'est à dire le fait que ce système d'exploitation puisse fonctionner sur des machines de type différents en ne modifiant que le compilateur écrit aussi en C

(a) Lire de 2 :37 jusqu'à 5 :15 et répondre aux questions

   i. Quels sont les deux types de programme cités ?

   ii. Quelles sont les trois parties d' Unix ?

(b) Lire de 11 :12 jusqu'à 12 :57 une description du système de gestion des fichiers UNIX que l'on va voir juste après

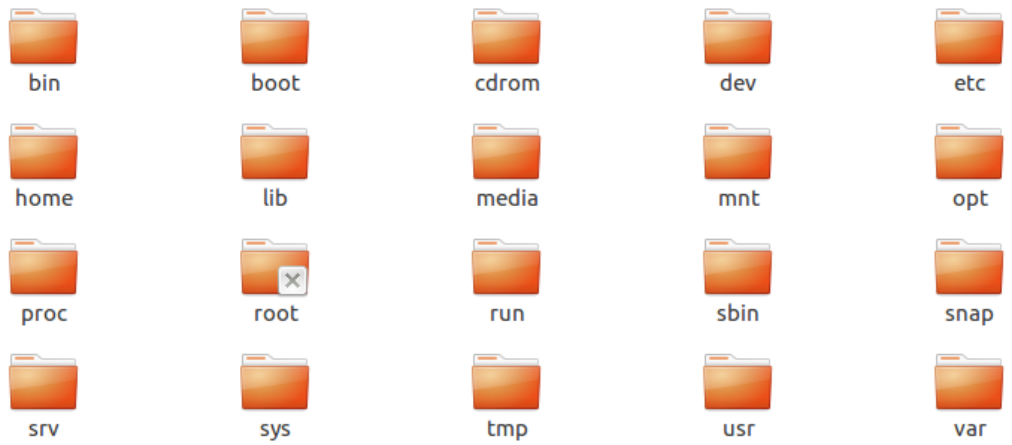
5.6.2 Découverte du système de gestion de fichiers d'Unix et du shell d'Unix

Pour pratiquer plusieurs solutions possibles :

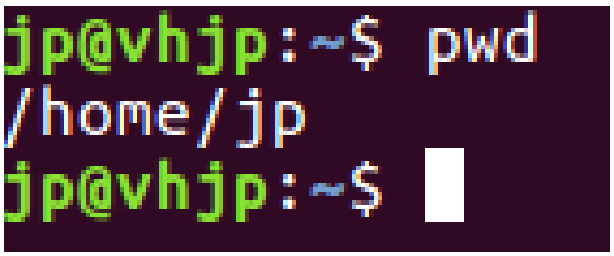
1. Soit vous avez sous la main un ordinateur avec un système LINUX (je vous conseille de le tester en live usb non persistant une version d'Ubuntu comme la 18.04 LTS
2. Soit vous avez sous la main un ordinateur Apple
3. Sinon il faut télécharger du Windows Store et installer sur votre ordinateur portable de la région , le logiciel DarkMoonX

Le but de cet exercice pratique est de copier un fichier Python nommé reduction.py (ou texte dans le répertoire Documents d'un des utilisateurs de la machine vers une clé USB , **sans utiliser le glisser-déposer de l'interface graphique**

Le système de gestion de fichiers d'Unix, structure la mémoire du disque dur (attention ! ce n'est pas la mémoire vive) sous la forme d'un arbre dont les noeuds sont soit des répertoires soit des fichiers
Un répertoire peut contenir d'autres répertoires ou des fichiers.
Voici la disposition des répertoires pour le système Ubuntu 16.04 LTS



1. Quelque soit le système UNIX que vous avez sous la main ouvrir un **terminal**
2. Le terminal nous permettra d'entrer des commandes du langage interprété **shell** et entrez la commande `pwd` pour `print working directory` qui affiche le **chemin absolu du répertoire actuel** dans lequel se trouve l'utilisateur qui a ouvert une session avec son mot de passe
   Vous devez observer ceci



Il nous faut préciser ici la notion de chemin absolu et de chemin relatif, que nous avons déjà vu lors du projet Eisbar
Un fichier ou un répertoire est **repéré absolument par rapport à la racine** /, ainsi le répertoire jp dans l'exemple ci-dessus a pour chemin absolu

/home/jp

le fichier reduction.py a pour chemin absolu :

/home/jp/Documents/reduction.py

Puisqu'il se trouve dans le répertoire Documents qui est inclus dans le répertoire jp qui est contenu dans le répertoire home qui est un "enfant" de la racine

L'inconvénient du repérage absolu est sa longueur, on lui préfère en général le repérage relatif

Qui dit relatif dit relatif par rapport à un autre fichier ou à un autre répertoire, regardons cela sur l'exemple suivant

3. Pour se déplacer dans l'arbre on peut soit "remonter" par la commande `cd ..` (pour change directory) les deux points pour signifier "vers le haut"
   Soit "descendre" vers un répertoire précis par exemple en entrant la commande `cd Documents` et

   le répertoire Documents a pour chemin relatif **par rapport à jp**

   Documents
4. Documents et Téléchargements sont deux répertoires "frères" contenus dans jp, pour passer de Documents à Téléchargements je peux entrer la commande avec une adresse relative :

   cd ../Téléchargements

   ou la commande suivante avec une adresse absolue

   cd /home/jp/Téléchargements
5. Vous pouvez faire `ls` pour visualiser le contenu du répertoire Documents et on peut observer alors On observe dans le répertoire Documents un autre réper-

   toire 2019-2020 (les répertoires n'ont pas d'extension) et d'autres fichiers comme pstree.png qui est une image et reduction.py qui est un fichier Python
6. la commande `mkdir` pour make directory permet de créer un répertoire. Ainsi l'année prochaine je vais créer un répertoire 2020-2021 dans le répertoire Documents pour y ranger mes fichiers de l'année scolaire 2020-2021
   Je peux le faire avec le langage shell en entrant la commande `mkdir 2020-2021` en m'assurant au préalable d'être placé dans le répertoire Documents

   Créer un répertoire 2020-2021 dans Documents
7. Insérer la clé USB
   Chercher son chemin absolu avec l'interface graphique

8. Pour copier le fichier reduction.py de Documents vers la clé USB il y a deux possibilités
   (a) "couper" c'est à dire déplacer le fichier avec la commande `mv <source> <cible>`, mv pour move
   En étant déjà dans Documents on va donc entrer `mv reduction.py /media/jp/cle-JP/reduc`
   (b) "copier-coller" avec la commande
   `cp <source> <cible>`, cp pour copy
   En étant déjà dans Documents on va donc entrer
   `cp reduction.py /media/jp/cle-JP/reduction.py`
   (c) Pour copier **tous** les fichiers ayant la même extension .py du répertoire Documents vers la clé USB on va faire
   `cp *.py /media/jp/cle-JP`
9. Pour supprimer un fichier on a la commande `rm` pour remove ainsi si on a fait un copier-coller du fichier reduction.py de Documents vers la clé USB et qu'après coup on veut aussi le supprimer de Documents alors on peut entrer la commande après s'être assuré d'être déjà dans Documents
   `rm reduction.py`
10. Les fichiers ont des **droits** ce qui confère une grande sécurité au Système Unix. Il y a trois types de droits :
    (a) de lecture
    (b) d'écriture
    (c) d'exécution

    Avec le langage shell on peut modifier sous certaines conditions les droits d'un fichier
    On peut visualiser les droits des fichiers et répertoire avec la commande `ls -l`
    On dit que l (pour long) est une option
11. Une dernière chose sur le shell lorsqu'on a oublié les spécificités d'une commande on utilise le manuel avec la commande `man`
    Ainsi en tapant `man ls` on aura toutes les options de la commande `ls`

5.7 Robotique

5.7.1 Rôle des capteurs et actionneurs

1. Dans la plupart des voitures il existe un radar de recul qui aide le conducteur à ne pas percuter un obstacle lors d'une marche arrière. Nous allons simuler un radar de recul à l'aide d'un microcontrôleur Arduino, un émetteur et capteur d'ultrasons et d'une diode rouge
2. Robot legoMinstorms

Chapitre 6

Algorithmique

6.1 Parcours séquentiel d'un tableau :

Problème : Je recherche un fichier nommé "algo.pdf" que je pense avoir rangé dans un dossier nommé INFO.

Pour simplifier nous supposons que ce dossier est un tableau T de chaînes de caractères ; pour chercher ce fichier on va utiliser l'algorithme suivant :

Algorithme 4 : Recherche séquentielle

Données : Un tableau T ayant n éléments de même type et comparables et une valeur v
Résultat : Un entier i, la première position de la valeur dans le tableau

```
1 début
2 |  i ← 0
3 |  tant que T[i] <> v faire
4 |  |  i ← i + 1
5 |  fin
6 fin
```

1. Les **spécifications** d'un algorithme est l'ensemble des **Données** et du **Résultat** d'un algorithme. En quelque sorte c'est une sorte de **contrat** associé à l'algorithme .
   Attention rien n'assure que ce contrat sera respecté lors de l'exécution de l'algorithme.
   Est ce que les spécifications sont suffisamment précises ?
2. Appliquer l'algorithme pour T = ["olga.txt", "algo.py","goal.csv",algo.pdf", "galo.py"] et pour la valeur v = "algo.pdf"
   Quel est la valeur de i à la fin de l'algorithme ?
3. Que se passe-t-il cette fois-ci si T = ["olga.txt", "algo.py","goal.csv", "galo.py"] ? Corriger l'algorithme
4. Implémenter cet algorithme en Python sous la forme d'une fonction qui renvoie l'entier i
5. Faire une deuxième implémentation en Python avec une boucle for et un retour dans la boucle.
6. Que se passe-t-il si le tableau est trié en ordre croissant (ordre lexicographique) ?

6.1.1 Notion de complexité :

Algorithme 5 : Recherche séquentielle

Données : Un tableau T ayant n éléments et une valeur v
Résultat : Un entier i, la première position de la valeur dans le tableau

```
1  début
2  |  i ← 0
3  |  tant que (i < n) et (T[i] <> v) faire
4  |  |  i ← i + 1
5  |  fin
6  |  si i = n alors
7  |  |  afficher("v n'est pas dans le tableau T")
8  |  sinon
9  |  |  afficher("v est dans le tableau en position",i)
10 |  fin
11 fin
```

Lorsqu'on exécute ce programme pour une liste de 10 éléments on a l'impression qu'il s'exécute presqu'instantanément.

Que se passe-t-il si la liste contient 100 000 ou 1 million d'éléments ?

On aimerait pouvoir répondre à cette question en ayant un ordre de grandeur du lien entre le temps d'exécution de l'algorithme T et la taille n du tableau

Dans la boucle `Tant que` deux opérations sont répétées : une *affectation* et deux *comparaisons*.

On peut supposer que le temps d'exécution d'une affectation peut varier d'un tour de boucle à l'autre mais ne dépassera pas une constante c_1, de même pour la comparaison le temps d'exécution d'une comparaison est plus petite qu'une constante c_2 à chaque tour de boucle. Combien de fois sera exécuté la boucle ?

On envisage deux cas extrêmes :

- **Au pire des cas** l'élément cherché est à la fin du tableau ou ne se trouve pas dans le tableau et dans ce cas le temps d'exécution $T(n) \leqslant nc_1 + 2nc_2 = (c_1 + 2c_2)n$. On dit que la complexité est linéaire dans le pire des cas, on dit aussi que la complexité est en O(n) (lire en grand O de n)
- **Au meilleur des cas** l'élément cherché est au début du tableau dans ce cas le temps ne dépend pas de n on dit que la complexité est en O(1)

6.1.2 Exercices - Parcours séquentiel d'un tableau - Complexité

Ex 1

Si sur une machine particulière il a fallu 1 s pour trouver un élément d'un tableau de taille 100 000 en dernière position, combien de temps à peu près mettra -t-on pour trouver un élément en dernière position dans un tableau de taille 1 000 000 sur cette même machine ?

Ex 2

Modifier l'algorithme de recherche séquentielle afin d'avoir comme information tous les indices i tel que $T[i] = v$. Quelle est la complexité de cet algorithme dans tous les cas ?

Ex 3

Compléter la fonction `somme (liste)` qui prend en argument une liste non vide d'entiers et renvoie la somme de ses éléments.

Compléter aussi le programme principal

Algorithme 6 : Somme des entiers contenus dans une liste

```
    début
        /* Définition de la fonction somme(liste)                    */
1       somme (T)
2       début
            Données : Un tableau T non vide ayant n entiers
            Résultat : la somme des n entiers
3           somme ← 0
4           pour  i ← 0 jusqu'à n − 1 faire
5               .....
6               i ← i + 1
7           fin
8           retourner .....
9       fin
        /* ----Programme Principal ------------                      */
10      liste ← [1,6,6,4]
11      .....
    fin
```

1. Quel est le nombre d'additions ? En déduire la complexité
2. Ecrire une fonction `moyenne (liste)` qui prend en argument une liste non vide d'entiers et renvoie la moyenne de ses éléments. On pourra utiliser la fonction `somme(liste)` ci-dessus.

Ex 4

1. On définit en mathématiques la moyenne harmonique de n nombres x_1, x_2,...,x_n comme étant l'inverse de la moyenne des inverses de ces n nombres. Ecrire une fonction `moyenneHarmonique(liste)` qui prend en argument une liste non vide d'entiers et renvoie la moyenne harmoniquede ses éléments. On pourra utiliser la fonction `moyenne(liste)` ci-dessus.
2. On définit en mathématiques la moyenne pondérée de n nombres x_1, x_2,...,x_n par des coefficients c_1, c_2, ..., c_n par :

$$\frac{c_1 \times x_1 + c_2 \times x_2 + ... + c_n \times x_n}{c_1 + c_2 + ... + c_n}$$

(a) Ecrire une fonction `moyennePonderee(valeurs,coefficients)` qui prend en argument, une liste non vide de nombres `valeurs` et une liste non vide d'entiers `coefficients` et qui renvoie la moyenne pondérée .

(b) Implémenter cette fonction en Python

(c) Calculer la complexité en temps de cette fonction en prenant uniquement en compte les multiplications

Exercices - Recherche d'un Maximum

Ex 5

On a une liste L de nombres décimaux, les températures maximales relevées à la station météo de Vélizy Villacoublay pour chaque jour du mois de Juin 2019

1. Ecrire une fonction en pseudo-code retournant le jour de la température maximale sur le mois de Juin 2019. Bien préciser les spécifications de la fonction
2. Ecrire une deuxième fonction retournant à la fois le jour et la valeur de la température maximale sur le mois de Juin 2019

Ex 6

Cette fois-ci chaque élément de la liste est une liste à deux éléments contenant la température minimale de la journée en premier, puis la température maximale de la journée ". Modifier la fonction de telle sorte que l'on ait le jour du mois de Juin où la température minimale a été la plus basse ainsi que la valeur de ce minimum, et de même pour la température maximale

Ex 7

Quelle est la complexité dans le pire des cas de la recherche d'un maximum dans une liste ?

Ex 8

Si on sait qu'une liste est **déjà** triée dans l'ordre **croissant** que suffit-il de faire pour avoir le maximum de la liste ? Quelle est alors la complexité ?

Ex 9

Une liste contient des entiers non distincts mais triés dans l'ordre croissant par exemple [1,1,2,2,2,3]
De cette liste on veut créer une nouvelle liste ne contenant qu'un exemplaire de la précédente liste mais avec un seul exemplaire des entiers ici [1,2,3]
Donner un algorithme et évaluer sa complexité

6.2 Tri par insertion et sélection

6.2.1 Tri par sélection

Dans le prolongement de la recherche d'un maximum (ou d'un minimum dans une liste on va étudier un premier algorithme de tri, le **tri par sélection**

Regardons sur un exemple, la liste T = [L,I,G,N,E] comment fonctionne le tri par sélection :

i	min	0	1	2	3	4
0	4	L	I	G	N	E
1	2	E	I	G	N	L
2	2	E	G	I	N	L
3	4	E	G	I	N	L
4	4	E	G	I	L	N

1. D'abord on parcourt toute la liste pour **sélectionner la première position du plus petit élément de la liste**, dans notre exemple la position 4 pour la lettre E (en rouge)
2. Ensuite on échange de place cet élément avec le premier élément de la liste.
3. A la deuxième itération numérotée 1, la lettre E (en vert) est bien placée et on recommence ce que l'on a fait précedemment sur le reste du tableau de la position 1 à 4

D'où l'algorithme

Algorithme 7 : Tri par sélection

Données : Un tableau T ayant $n \geqslant 2$ éléments comparables
Résultat : T est trié dans l'ordre croissant

```
1  début
2      pour debut ∈ ⟦0; n − 1⟦ faire
3          //Sélection de la position du plus petit élément dans l'intervalle
             ⟦debut; n⟦
4          min ← debut
5          pour i ∈ ⟦debut + 1; n⟦ faire
6              si T[i] <T[min] alors
7                  min ← i
8              fin
9          fin
10         //On échange les valeurs en position min et deb
11         temp ← T[debut]
12         T[debut] ← T[min]
13         T[min] ← temp
14     fin
15 fin
```

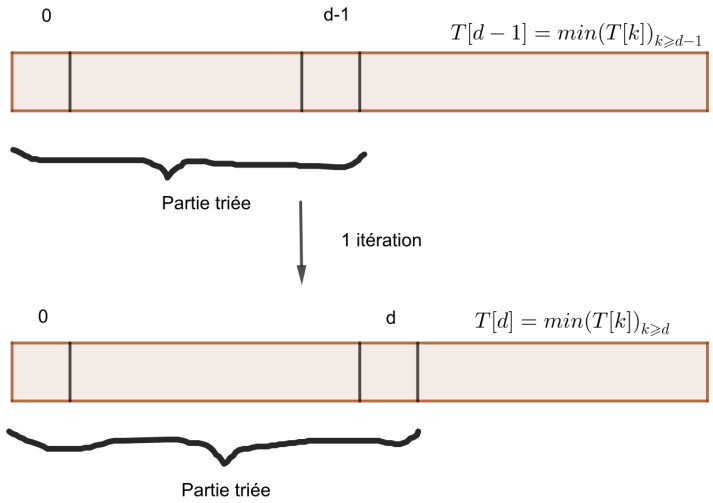
Voir ici une animation `http://www.sorting-algorithms.com/selection-sort`
Comment être sûr que ce programme est "juste" ?

6.2.2 Complexité du tri par sélection

Pour simplifier l'évaluation de la complexité on va regrouper toutes les affectations dans l'échange (lignes 11 à 13) pour un coût de c_1. Ensuite on majore les comparaisons (ligne 6) par un coût c_2

Combien d'échanges ?
Dans la suite on note d la variable *debut*. La variable d parcourt l'intervalle de 0 à $n-2$ donc il y a $n-1$ échanges
Combien de comparaisons ? Lorsque $d = 0$, i varie de 1 à $n-1$ donc $n-1$ comparaisons, lorsque $d = 1$, i varie de 2 à $n-1$ donc $n-2$ comparaisons puis finalement $d = n-2$ et i varie de $n-1$ à $n-1$ donc une seule comparaison
En tout il y a $(n-1)+(n-2)+...+2+1 = \dfrac{(n-1)n}{2} = \dfrac{n^2}{2} - \dfrac{n}{2}$ comparaisons
Donc $T(n) \leqslant (\dfrac{n^2}{2} - \dfrac{n}{2})c_2 + (n-2)c_1 = \dfrac{n^2}{2}c_2 + n(c_1 - \dfrac{c_2}{2}) - 2c_1$
Donc quand n devient "très grand" le terme en n^2 l'emporte sur celui en n qui l'emporte sur le terme constant, on va donc dire que la complexité du tri par sélection est **dans tous les cas** en $O(n^2)$ ou **quadratique**

6.2.3 Preuve du programme



Dans un premier temps on peut faire des tests (Voir TP) mais nous allons voir une **preuve** de programme qui consiste à rechercher une propriété qui reste invariante à chaque tour de boucle et qui nous aiderait pour la résolution de notre problème c'est à dire trier le tableau dans l'ordre croissant
Une propriété invariante à chaque tour de boucle s'appelle un invariant de boucle
Soit $P(d)$ "les éléments du tableau de 0 à $d-1$ sont triés dans l'ordre croissant et $T[d-1]$ est plus petit que tous les éléments après lui dans le tableau"
Montrons que $P(d)$ est un invariant de boucle pour $d \geqslant 1$:
Supposons que $P(d)$ est vraie pour $d \geqslant 1$ autrement dit tous les éléments $T[j]$ sont dans l'ordre croissant pour $0 \leqslant j \leqslant d-1$ avec $d \geqslant 1$, et $T[d-1] \leqslant T[k]$ pour tout $k \geqslant d$ au tour suivant de boucle $d \leftarrow d+1$ et alors soit $d < n-1$ soit $d = n-1$
Si $d = n-1$ alors le tableau est trié dans l'ordre croissant sinon si $d < n-1$ après avoir séléctionné dans le reste du tableau le plus petit élément et après échange $T[d]$ sera le plus petit élément de la partie du tableau entre d et $n-1$ d'où l'invariance

Enfin le programme s'arrête au bout d'un moment car l'itération se fait avec des boucles pour

6.2.4 Tri par insertion

L'idée et le nom du tri par insertion vient des jeux de cartes. Oublions les couleurs et considérons que les hauteurs sont classées ainsi dans l'ordre croissant $[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, V, D, R]$
Imaginons que l'on vous distribue 5 cartes et que la première est V donc votre main est pour l'instant [V] puis la deuxième est 4 alors puisque 4 < V on insère 4 à sa place et votre main devient [4,V], puis la troisième carte distribuée est 7 donc insère 7 à sa place donc votre main devient [4,7,V] etc...
D'où l'algorithme

Algorithme 8 : Tri par insertion

Données : Un tableau T ayant $n \geqslant 2$ éléments comparables
Résultat : T est trié dans l'ordre croissant

```
1  début
2      pour i ∈ ⟦0; n⟦ faire
3          pour j ∈⟧0; i⟧ faire
4              si T[j] <T[j-1] alors
5                  //On échange les valeurs en position j et j-1
6                  temp ← T[j]
7                  T[j] ← T[j − 1]
8                  T[j − 1] ← temp
9              sinon
10                 sortir de la boucle
11             fin
12         fin
13     fin
14 fin
```

Voici un exemple d'exécution sur le tableau T = [L,I,G,N,E]

i	j	0	1	2	3	4
		L	I	G	N	E
1	0	I	L	G	N	E
2	0	G	I	L	N	E
3	3	G	I	L	N	E
4	0	E	G	I	L	N

6.2.5 Exercices -Tris par sélection et insertion

Ex 1

Soit $S(n) = 1 + 2 + 3 + ... + n - 1 + n = n + n - 1 + ... + 3 + 2 + 1$

1. Montrer que $2S(n) = n(n + 1)$
2. Que vaut $S(n)$?
3. Que vaut $1 + 2 + 3 + 4 + ... + 1000$?

Ex 2

Faire tourner l'algorithme du tri par sélection sur le tableau T = [3,1,4,1,5] avec un tableau comme dans le cours

Ex 3

Ecrire en Python et en Javascript l'algorithme du tri par sélection

Ex 4

Faire tourner l'algorithme du tri par insertion sur le tableau T = [3,1,4,1,5] avec un tableau comme dans le cours

Ex 5

Quel algorithme (tri par insertion ou sélection) sera le plus rapide sur un tableau tel que :

1. Toutes les valeurs sont identiques
2. Le tableau est déjà trié
3. Le tableau est trié dans l'ordre descendant

Ex 6

Faire une preuve du programme du tri par insertion

Ex 7

Montrer que la complexité du tri par insertion est quadratique dans le pire des cas (tableau trié dans l'ordre descendant) et linéaire dans le meilleur des cas (tableau déjà trié)

Ex 8

Faire une preuve de l'exercice 9 (Parcours séquentiel d'un tableau)

Ex 9

Ecrire en Python et en Javascript l'algorithme du tri par insertion

6.3 Recherche dichotomique dans un tableau trié

Comment chercher un mot dans un dictionnaire ?
Parcourt-on le dictionnaire du début à la fin ?
Surtout pas car le dictionnaire est **ordonné** et en utilisant le fait qu'il soit ordonné on peut procéder de manière plus efficace
On procède plutôt **par dichotomie**, (couper en deux) c'est à dire on ouvre le dictionnaire au milieu et on compare le mot du milieu au mot recherché
Si le mot recherché "est plus petit" que le mot du milieu on réitère la recherche dans la partie du dictionnaire qui précède le mot du milieu, sinon on réitère la recherche dans la partie qui suit le mot recherché

On traduit l'idée ci-dessus sous une forme mathématique plus précise
si $x \in [a, b[$ alors **ou bien** $x \in [a, \frac{a+b}{2}[$ **ou bien** $x \in [\frac{a+b}{2}, b[$

Bien observer comment on garde **invariant** l'encadrement de v dans un intervalle du type [..,...[

L'idée ci-dessus nous amène à poser comme **invariant de boucle** :

$I(a, b) = \{0 \leqslant a < b \leqslant n-1 \;\; T[a] \leqslant x < T[b]\}$

1. L'assertion I(a,b) doit être vraie au début de la boucle (ligne 10). Compléter les lignes 4 et 7 de l'algorithme ci-dessous pour que cela soit ainsi
2. Il nous reste à trouver la condition d'arrêt de la boucle et compléter la ligne 11 de l'algorithme

   Pour comprendre comment arrêter la boucle traiter les cas particuliers

   (a) $T = \{E, G, I, K, M, M, M\}$ et $v = H$

   (b) $T = \{E, G, I, K, M, M, M\}$ et $v = K$
3. Une fois que vous avez compris comment s'arrête la boucle il faut conclure. Compléter les lignes 20 et 21

Algorithme 9 : Recherche dichotomique

`estDansTableau` (T,v)
Données : Un tableau T ayant $n \geqslant 2$ éléments comparables **trié dans l'ordre croissant**,une valeur v
Résultat : un indice de v si v est dans T, -1 sinon

```
1  début
2      a ← 0
3      b ← n − 1
4      si ..... alors
5          retourner -1
6      fin
7      si ..... alors
8          retourner n-1
9      fin
10     {I(a,b)}
11     tant que ..... faire
12         m ← E((a+b)/2)
13         si v < T[m] alors
14             b ← m
15         sinon
16             a ← m
17         fin
18         {I(a,b)}
19     fin
20     si ..... alors
21         retourner .....
22     sinon
23         retourner -1
24     fin
25 fin
```

6.3.1 Preuve

Algorithme 10 : Recherche dichotomique

`estDansTableau` (T,v)
Données : Un tableau T ayant $n \geqslant 2$ éléments comparables **trié dans l'ordre croissant**,une valeur v **Précondition**
Résultat : un indice de v si v est dans T, -1 sinon **Postcondition**

```
1  début
2      a ← 0
3      b ← n − 1
4      si v < T[0] ou v > T[n-1] alors
5          retourner -1
6      fin
7      si v = T[n-1] alors
8          retourner n-1
9      fin
10     tant que  a+1 < b faire
11         m ← E((a + b)/2)
12         si v < T[m] alors
13             b ← m
14         sinon
15             a ← m
16         fin
17     fin
18     si T[a] = v alors
19         retourner a
20     sinon
21         retourner -1
22     fin
23 fin
```

1. **Terminaison** : la grandeur $b - a \geqslant 1$, **l'amplitude de l'encadrement de v** , décroit à chaque tour de boucle car est à peu près divisée par 2 à chaque tour de boucle jusqu'à ce que $b - a = 1$
   Or la condition d'arrêt de la boucle est justement que $a + 1 \geqslant b$ (négation de $a + 1 < b$), c'est à dire $b - a \geqslant 1$ donc la boucle se termine au bout d'un nombre fini de tours
2. **Invariant de boucle** :
   A l'entrée de la boucle I(a,b) est vraie pour a = 0 et b = n-1 car à cause du test précédant la boucle on sait que $T[a] \leqslant v < T[b]$
   Montrons que I(a,b) reste vraie après une itération :
   si I(a,b) est vraie, c'est à dire $\{0 \leqslant a < b \leqslant n - 1 \;\; T[a] \leqslant v < T[b]\}$ alors les nouvelles valeurs après une itération de a et b notées a' et b' sont :
   soit $a' = E(\frac{a+b}{2})$ et $b' = b$ si $v \geqslant T[E(\frac{a+b}{2})]$ ou $a' = a$ et $b' = E(\frac{a+b}{2})$ si $v < T[E(\frac{a+b}{2})]$

Donc dans le premier cas et le deuxième cas $T[a] \leqslant v < T[b]$ avec $0 \leqslant a < b \leqslant n-1$

3. **Sortie de boucle** la boucle s'arrête lorsque b = a+1 pour $0 \leqslant a \leqslant n-2$ donc I(a,a+1) est vraie autrement dit $T[a] \leqslant v < T[a+1]$ donc :
   Soit $v = T[a]$ et on retourne a ou $T[a] < v < T[a+1]$ et on retourne -1

6.3.2 Complexité

Pour évaluer la complexité de la recherche dichotomique dans le pire des cas on va supposer que la longueur du tableau T est une puissance de 2
Regardons sur un cas particulier :
T = {E,G,I,K,M,O,Q,S} avec v = J
A chaque tour de boucle il y a 2 affectations et 3 comparaisons, combien de tours de boucles ?
On se rend compte qu'il y a 3 tours de boucle pour 2^3 valeurs
Plus généralement on a n tours de boucles pour 2^n valeurs
Il existe une fonction mathématique, appelée le logarithme en base 2
la fonction $n \to \ln_2(n)$ telle que $\ln_2(2^n) = n$
Par conséquent si la taille d'un tableau est 2^n alors la complexité de la recherche dichotomique est en $O(\ln_2(n))$
Plus généralement pour toute taille m, il existe n tel que
$2^n \leqslant m < 2^{n+1}$ avec $n = \ln_2(m)$
Par conséquent la complexité de la recherche dichotomique est en $O(\ln_2(n))$
Quelques valeurs expérimentales : On a vu en TP que dans le pire des cas pour une liste de un million d'éléments la recherche séquentielle prenait à peu près 4 centièmes de secondes alors que la recherche dichotomique sur ce même tableau trié prend ...40 micro - secondes , 1000 fois moins de temps !
Encore plus fort on verra en exercice que si la taille du tableau est multiplié par un facteur 10^6 le temps d'exécution ne fera que **doubler** pour atteindre 80 micro secondes !

6.3.3 Exercices - Recherche dichotomique

Ex 1

Exécuter l'algorithme de la recherche dichotomique avec $T = [2, 4, 6, 8, 10, 12, 14, 16]$ pour les valeurs suivantes

1. v = 1
2. v = 3
3. v = 14
4. v = 16

Ex 2

Si une valeur est dans le tableau, l'algorithme retourne-t-il forcément le plus petit indice de cette valeur ?

Ex 3

Ecrire en Python la fonction `estDansTableauDicho(liste,valeur)`

Ex 4

1. En TP on a chronométré que le tri par sélection prenait à peu près 4 centièmes de secondes pour un tableau de 1000 éléments. Justifier le temps de 1000 jours pour un tableau de 100 millions d'éléments
2. En TP on a chronométré que dans le pire des cas la recherche séquentielle prenait à peu près 4 centièmes de secondes pour un tableau d'un million d'éléments. Combien de temps prendra la recherche séquentielle dans le pire des cas pour un tableau de mille milliards d'éléments ?
3. En TP on a chronométré que dans le pire des cas la recherche dichotomiques prenait à peu près 40 micro secondes pour un tableau d'un million d'éléments. Combien de temps prendra la recherche dichotomique pour un tableau de mille milliards d'éléments ?(Indication : on admettra que $T(n) = c\ln_2(n)$ et on admettra la relation $\ln_2(n^2) = 2\ln_2(n)$ pour $n > 0$

Ex 5

Peut-on écrire a $<$ b à la place de a $+ 1 <$ b dans le programme vu en cours ?

Ex 6

Il existe d'autres façons d'écrire la recherche dichotomique, voici un autre algorithme :

1. Prouver que la boucle s'arrête
2. Prouver que $\{0 \leqslant a < b \leqslant n-1 \mid$ v n'est pas dans le tableau pour les indices dans [0,a[et]b,n-1]$\}$ est un invariant de boucle
3. Prouver qu'à la sortie de boucle l'algorithme résoud le problème de la recherche dichotomique

Algorithme 11 : Recherche dichotomique

`estDansTableau` (T,v)
Données : Un tableau T ayant $n \geqslant 2$ éléments comparables **trié dans l'ordre croissant**,une valeur v
Résultat : un indice de v si v est dans T, -1 sinon

```
1  début
2      a ← 0
3      b ← n − 1
4      {I(a,b)}
5      tant que a < b faire
6          m ← E((a + b)/2)
7          si v < T[m] alors
8              b ← m − 1
9          fin
10         sinon si v > T[m] alors
11             a ← m + 1
12         fin
13         sinon
14             retourner m
15         fin
16     fin
17     {I(a,b)}
18     si v = T[a] alors
19         retourner a
20     sinon
21         retourner -1
22     fin
23 fin
```

Ex 7

1. Montrer que $\{\ 0 \leqslant i < n\ f = i!\ \}$ est l'invariant de boucle
2. Justifier la terminaison de la boucle
3. Montrer que le programme calcule bien n !
4. Que se passe-t-il si on permute les deux instructions à l'intérieur de la boucle ?

Algorithme 12 : Calcul de n !

```
début
    i ← 0
    f ← 1
    tant que i < n faire
        i ← i + 1
        f ← i ∗ f
    fin
fin
```

6.4 Algorithmes gloutons

6.4.1 Problème de rendu de monnaie (1)

Imaginons que nous devons rendre 43 euros et que nous avons à notre disposition des pièces de 1,2,5,10 euros **avec la contrainte supplémentaire de rendre le moins de pièces possible (problème d'optimisation)** (pour simplifier il n'y a pas de billets mais que des pièces)

Une façon courante de procéder est :

1. Combien de fois **la plus grande valeur à notre disposition**, c'est à dire 10 y-a-t-il dans 43 ?

   $43 = 4 \times 10 + 3$

   Je vais donc rendre au moins 4 pièces de 10 euros.

   Après avoir divisé 43 par 10,j'exclus définitivement la valeur 10 des valeurs disponibles et je recommence le travail de division du reste c'est à dire 3 avec la plus grande des valeurs restantes c'est à dire 5
2. Combien de fois **la plus grande valeur à notre disposition**, c'est à dire 5 y-a-t-il dans 3 ?

   $3 = 0 \times 5 + 3$

   On n'utilisera donc pas de pièces de 5 euros et on exclut la valeur 5 des diviseurs
3. Combien de fois **la plus grande valeur à notre disposition**, c'est à dire 2 y-a-t-il dans 3 ?

   $3 = 1 \times 2 + 1$

   Donc pour l'instant on utilisera 4 pièces de 10 et une pièce de 2 on exclut la pièce de 2 et on continue
4. Combien de fois **la plus grande valeur à notre disposition**, c'est à dire 1 y-a-t-il dans 1 ?

   $1 = 1 \times 1$

   On rend donc 4 pièces de 10 et une pièce de 2 et une pièce de 1 euros

Problème de rendu de monnaie (2) : Imaginons que nous devons toujours rendre 43 euros et que cette fois ci nous avons à notre disposition des pièces de 1,3,7,21,30 euros avec la contrainte supplémentaire de rendre le moins de pièces possible

Procédons comme précédemment :

1. Combien de fois **la plus grande valeur à notre disposition**, c'est à dire 30 y-a-t-il dans 43 ?

   $43 = 1 \times 30 + 13$

   Après avoir divisé 43 par 30,j'exclus définitivement la valeur 30 des valeurs disponibles et je recommence le travail de division du reste c'est à dire 13 avec la plus grande des valeurs restantes c'est à dire 21
2. Combien de fois **la plus grande valeur à notre disposition**, c'est à dire 21 y-a-t-il dans 13 ?

   $13 = 0 \times 21 + 13$

   On n'utilisera donc pas de pièces de 21 euros et on exclut la valeur 21 des diviseurs

3. Combien de fois **la plus grande valeur à notre disposition**, c'est à dire 7 y-a-t-il dans 13 ?

   $13 = 1 \times 7 + 6$

   Donc pour l'instant on utilisera 1 pièce de 30 et une pièce de 13 on exclut la pièce de 7 et on continue

4. Combien de fois **la plus grande valeur à notre disposition**, c'est à dire 3 y-a-t-il dans 6 ?

   $6 = 2 \times 3$

   On rend donc 1 pièce de 30 et une pièce de 7 et deux pièces de 3 euros.

   On a donc rendu 4 pièces **mais on peut faire mieux** avec 3 pièces car $43 = 2 \times 21 + 1$

Dans un algorithme lorsqu'on choisit à chaque itération le plus grand ou le plus petit d'un ensemble de valeurs et que l'on ne revient plus jamais sur ce choix on dit que c'est un algorithme **glouton** ou que l'on a adopté une stratégie gloutonne pour résoudre notre problème

La stratégie gloutonne fait partie d'un ensemble de stratégies que vous apprendrez au Lycée. Cette année vous voyez la stratégie gloutonne et la stratégie "diviser pour régner" (vue dans la recherche dichotomique), l'année prochaine en Algorithmique on approfondira la stratégie "diviser pour régner" en étudiant le tri par fusion et on découvrira une autre stratégie "la programmation dynamique"

A ce stade une question légitime nous vient à l'esprit : Comment être sûr que la stratégie gloutonne résout bien le problème ?

On a montré que cette stratégie ne fonctionne pas dans le deuxième cas en trouvant un **contre-exemple**, par contre il faut **prouver** le programme pour être sûr qu'il fonctionne dans le premier cas (On admettra que cela fonctionne)
Voici un algorithme pour le premier cas :

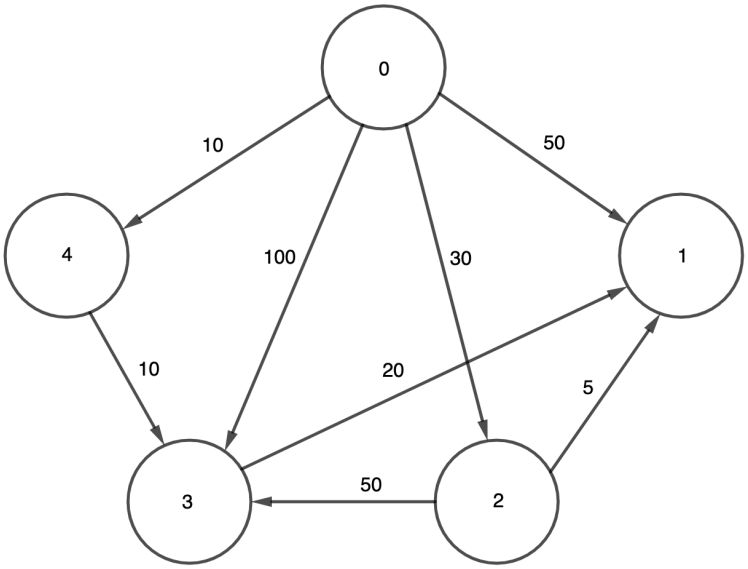
On dispose d'une fonction `div(a,b)` qui retourne le tuple (q,r) tel que $a = bq + r$ est le résultat de la division euclidienne de a par b

Algorithme 13 : Rendu de monnaie (algorithme glouton)

`renduMonnaie` (P,v)
Données : Un tableau de pièces P = [1, 2, 5, 10] et une valeur v
Résultat : Un tableau d'entiers T tel que
$v = T[0] + 2 \times T[1] + 5 \times T[2] + 10 \times T[3]$

```
1  début
2      s ← v
3      T ← [0, 0, 0, 0]
4      i ← 3
5      tant que s > 0 faire
6          T[i], s ← div(s, max(P))
7          i ← i − 1
8      fin
9      retourner T
10 fin
```

6.4.2 Plus court chemin dans un graphe (Algorithme de Dijkstra)



On appelle graphe, un ensemble de sommets, ici les éléments 0, 1, 2, 3 et 4 et d'arêtes (les flèches) reliant certains éléments entre eux
Ces arêtes sont orientées ainsi il y a une arête qui part de 0 vers 1, et valuées avec des nombres positifs par exemple, l'arête qui part de 0 vers 1 a pour valeur 50
On peut imaginer qu'un tel graphe peut représenter les temps de transport entre les différentes stations de métro d'une ville

L	0	1	2	3	4
0	∞	50	30	100	10
1	∞	∞	∞	∞	∞
2	∞	5	∞	50	∞
3	∞	20	∞	∞	∞
4	∞	∞	∞	10	∞

Un des sommets est l'origine ou **la source** (on choisit ici le sommet 0)
Le but du problème est de trouver les valeurs des plus courts chemins entre la source et les autres sommets
Pour cela on dispose du tableau D = [50, 30, 100, 10] donnant dans l'ordre les distances de 0 à 1, puis de 0 à 2, puis de 0 à 3 et enfin de 0 à 4
Ce tableau va évoluer tout au long de l'exécution de l'algorithme car on va trouver une distance plus courte pour aller de 0 à 4
L'équivalent de l'ensemble des pièces est ici l'ensemble des sommets différents de la source 0, c'est à dire C = [1,2,3,4]

1. On cherche le sommet le plus proche de la source pour les éléments de C :
   C'est le sommet 4, on l'enlève de l'ensemble C, donc C devient [1,2,3] et on regarde maintenant si on peut faire mieux dans le tableau D, en passant par le sommet 4 pour aller vers 1, 2 ou 3
   Oui car le chemin 0-4-3 prend moins de temps que 0-3 donc on remplace 100 par

10 + 10 = 20 et D = [50,30,20,10]

2. On cherche encore le sommet le plus proche de la source pour les éléments de C :
   C'est le sommet 3, on l'enlève de l'ensemble C, donc C devient [1,2] et on regarde maintenant si on peut faire mieux dans le tableau D pour aller de la source vers 1, 2 via le sommet 3
   Oui car le chemin 0-4-3-1 prend moins de temps que 0-1 donc on remplace 50 par 20 + 20 = 40 et D = [40,30,20,10]
3. On cherche encore le sommet le plus proche de la source pour les éléments de C :
   C'est le sommet 2, on l'enlève de l'ensemble C, donc C devient [1] et on regarde maintenant si on peut faire mieux dans le tableau D pour aller de la source vers 1 via le sommet 2
   Oui car le chemin 0-3-1 prend moins de temps que 0-1 donc on remplace 40 par 30 + 5 = 35 et D = [35,30,20,10]

D'où l'algorithme de Dijkstra

Algorithme 14 : Algorithme de Dijkstra (algorithme glouton)

`dijkstra` (L,s)
Données : Un tableau 2D de longueurs L de dimension $n \times n$, la source s
Résultat : Un tableau D de distances les plus courtes de la source s aux autres sommets

```
1  début
2      C ← [1, 2, ..., n − 1]
3      D ← L[0]
4      i ← 3
5      pour i ∈ ⟦0; n − 2⟦ faire
6          v ← l'élément de C qui minimise L[0]
7          C ← C − {v}
8          pour chaque élément w de C faire
9              D[w] ← min(D[w], D[v] + L[v][w])
10         fin
11     fin
12     retourner D
13 fin
```

6.4.3 Exercices - Algorithmes gloutons

Ex 1

On dispose du système de pièces suivant S = $\{1,\ 2,\ 2^2,\ ...,\ 2^7\}$
Rendre les sommes suivantes avec le système S

1. 127
2. 15
3. 255
4. A quoi cela correspond il d'un point de vue numérique ?

Ex 2

Exécuter l'algorithme de Dijkstra sur le graphe du cours

1. en prenant comme source le sommet 2
2. en rendant le graphe non orienté avec les mêmes valeurs et pour source 0
3. en rendant le graphe non orienté avec les mêmes valeurs et pour source 2

6.5 Algorithme des k plus proches voisins