

October 8, 2025

1 Notation de Landau et complexité

Soient $U = (u_n)_{n \in \mathbb{N}}$ et $V = (v_n)_{n \in \mathbb{N}}$ deux suites à valeurs réels ou complexes.

On dit que V domine U et on note $u_n = O(v_n)$ si il existe un rang N et un nombre $\alpha > 0$ tels que pour tout indice $n > N$ on a

$$|u_n| \leq \alpha |v_n|$$

Dans le cadre d'une analyse de complexité on dit qu'une opération est en $O(1)$ si elle s'exécute en temps constant, c.a.d indépendamment des paramètres et du moment où on l'exécute.

```
[ ]: m=6#affectation O(1)
      3+5#opérations arithmétiques O(1)
      m+=1 # incrémentation O(1)
      l = [] #création d eliste vide O(1)
      l.append(56)# append est en O(1)
      l[0] = 18# écriture en O(1)
```

2 Manipulations de matrices

```
[23]: m = [[10,20,30],[40,50,60]]
      m
```

```
[23]: [[10, 20, 30], [40, 50, 60]]
```

```
[24]: len(m), len(m[0])# nb lignes, nb col
```

```
[24]: (2, 3)
```

Dans ce TP on suppose que toutes les lignes ont le même nombre de colonnes et on ne cherchera pas à vérifier ce fait (sauf indication du contraire).

```
[17]: m[0][1]
```

```
[17]: 20
```

```
[18]: m[1][2] = 100
      print(m)
```

```
[[10, 20, 30], [40, 50, 100]]
```

3 Corrigé des exercices

```
[20]: def create(n,m):  
    mat = [] # O(1)  
    for i in range(n):#n passages i+=1 en O(1)  
        ligne = []# O(1)  
        for j in range(m):#m passages j+=1 en O(1)  
            ligne.append(0) #O(1)  
        mat.append(ligne)#O(1)  
    return mat# O(1)
```

La complexité de l'appel `create(n)` est proportionnelle à la somme suivante

$$2 + \sum_{i=0}^{n-1} (1 + 2 + \sum_{j=0}^{m-1} (1 + 1)) = 2 + 3n + 2 \times nm = O(nm)$$

On dit que la complexité est *quadratique* car nm est le polynôme bivarié $n^1 m^1$ et $1 + 1 = 2$

```
[9]: create(2,3)
```

```
[9]: [[0, 0, 0], [0, 0, 0]]
```

```
[25]: def search(mat,x):  
    #hyp : toutes les lignes ont le mm nb de col  
    r = [] #O(1)  
    for i in range(len(mat)):#n passages  
        for j in range(len(mat[0])):#m passages  
            if mat[i][j]==x:#O(1)  
                r.append((i,j))#O(1)  
    return r#O(1)
```

La complexité de l'appel `search(mat,x)` est proportionnelle à la somme suivante (en notant n, m le nb de lignes et de col.):

$$2 + \sum_{i=0}^{n-1} (1 + \sum_{j=0}^{m-1} (1 + 2)) = 2 + n + 3 \times nm = O(nm)$$

les 1 : pour compter l'incrémentation du compteur.

La complexité est dite *quadratique*

```
[26]: m = [[2,3,8,2],[4,2,1,0],[6,2,0,0]]  
search(m,2)
```

```
[26]: [(0, 0), (0, 3), (1, 1), (2, 1)]
```

```
[28]: def add(m1,m2):
        #hyp : m1 et m2 ont mmes dimensions
        res = create(len(m1),len(m1[0]))
        for i in range(len(m1)):
            for j in range(len(m1[0])):
                res[i][j] = m1[i][j]+m2[i][j]
        return res
```

```
[29]: m1 = [[1,2,3],[4,5,6]]
        add(m1,m1)
```

```
[29]: [[2, 4, 6], [8, 10, 12]]
```

```
[ ]: def add(m1,m2):
        #hyp : m1 et m2 ont mmes dimensions
        res = []
        for i in range(len(m1)):
            ligne = []
            for j in range(len(m1[0])):
                ligne.append( m1[i][j]+m2[i][j])
            res.append(ligne)
        return res
```

```
[ ]: def check(m):
        """
        prend une liste de listes
        et vérifie que toutes les "lignes"
        ont même longueur
        """
        for i in rang(1,len(m)):#nb lignes passages
            if len(m[i])!=len(m[0]):#O(1)
                return False
        return True#cpx finale O(nb lignes)
```