# Mathématiques PT. 2025 - 2026

Mathématiques & Informatique - F. Gaunard
http://frederic.gaunard.com
Lycée Voltaire, Paris 11e.

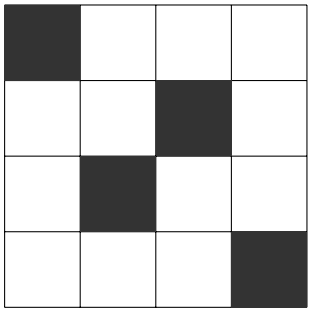
## Préparation à l'oral

*Math II - Informatique*
*Semaine du 2 Juin*

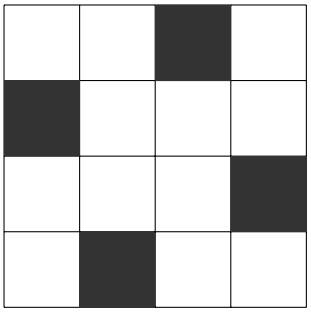
---

## Sujet OB-INFO-1

Soit $n \in \mathbb{N}^*$ un entier. À chaque quadrillage de taille $n \times n$ est associée une liste comme ci-dessous.
Chaque quadrillage ne contient, par ligne et par colonne, qu'une seule *figure*.



`L = [1, 3, 2, 4]`



`L = [2, 4, 1, 3]`

**1**. Écrire une fonction `tester(L)` qui renvoie un booléen selon que le quadrillage correspondant à la liste `L` existe ou non.

**2**. **a**. Écrire une fonction `outil(L)` qui prend en argument une liste `L` et renvoie une liste dont chaque composante est une liste à deux éléments : chacun des éléments $x$ de `L` et la liste `L'` $= \texttt{L} \setminus \{x\}$.
Par exemple `outil([1, 2, 3])` renverra $[[1,[2,3]],[3,[1,2]],[2,[1,3]]]$.
**b**. Trouver une fonction récursive qui renvoie tous les quadrillages de taille $n$ qui existent.

**3**. **a**. Écrire une fonction `contact(L, i,j)` qui vérifie si deux *figures* d'un même quadrillage `L` sont en contact par leur diagonale. *Cette fonction renverra* 1 *ou* 0.
**b**. Écrire une fonction qui permet de trouver tous les quadrillages de taille $n$ sans contact.

## Sujet OB-INFO-1 : Math II 2025, Solution

1. On doit avoir une permutation des $n$ entiers et donc aucune répétition. Une manière de faire est la suivante

```
def tester(L):
    return sort(L)==list(range(1, len(L)+1)
```

ou aussi

```
def tester(L):
    diff=[ ]
    for x in L :
        if x in diff :
            return False
        diff.append(x)
    return len(diff)==len(L)
```

2. **a**. Sans trop de difficulté non plus. On concatène.

```
def outil(L) :
    liste=[ ]
    for k in range(len(L)):
        liste.append([L[k], L[:k]+L[k+1:]])
    return liste
```

**b**. On utilise la fonction précédente !

```
def quadrillages(L):
    if len(L) == 0:
        return [[ ]] # pas de quadrillage donc liste avec la liste vide
    result = [ ]
    for x, reste in outil(L):
        for q in quadrillages(reste):
            result.append([x] + q)
    return result
```

Cette fonction renvoie le résultat voulu si l'on prend `L` = [`k for k in range(1, n+1)`] en argument.

3. 
```
def contact(L, i, j):
    if abs(i - j) == 1 and abs(L[i] - L[j]) == 1:
        return True
    return False
```

4. On commence par déterminer si une grille a un contact.

```
def test_contact(L):
    n = len(L)
    for i in range(n-1):
        if contact(L, i, i+1):
            return True

    return False

def quad_sans_contact(n):
    L=[k for k in range(1, n+1)]
    Q=quadrillages(L)
    res=[]
    for q in Q :
        if not test_contact(q) :
            res.append(q)
    return res
```