LI 2 - Semestre 4 Septembre 2025

Université d'Artois
Faculté des Sciences Jean Perrin

Algorithmique et Programmation 3

TP2: Complexité

1. Écrire une fonction qui calcule la somme des entiers pairs dans une liste et évaluer sa compexité.

```
>>> somme_pairs([1,4,3,5,10,2])
16
```

2. Écrire une fonction qui calcule le nombre de couples d'éléments dans une liste dont la somme est supérieure strictement à une valeur donnée et évaluer sa complexité.

```
>>> pairs_sup_k([1,4,3,5,10,2],7)
7
```

3. Écrire la fonction `somme_factorielles` permettant de calculer la somme suivante : $somme\_factorielles(n) = \sum_{k=0}^{n} k!$ et évaluer sa complexité.

```
>>> somme_factorielles(0)
1
>>> somme_factorielles(3)
10
```

4. Écrire un programme permettant de trouver le premier élément qui se répète dans une liste d'entiers.

```
>>> l = [1,2,3,5,7,5,2,12,1]
>>> premiere_repetition(l)
5
```

5. Étant donnée une matrice bidimensionnelle, écrire un algorithme qui imprime toutes les diagonales.

   Exemple : pour la matrice

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

   Le résultat attendu est :

```
1
5 2
9 6 3
13 10 7 4
14 11 8
15 12
16
```

6. Étant donnée une liste d'entiers dont chacun apparaît deux fois sauf un seul qui apparaît une seule fois. Écrire une fonction permettant de trouver ce nombre. Donner sa complexité et proposer un algorithme en complexité linéaire.

```
>>> l = [1,2,3,5,7,5,2,1,3]
>>> intrus(l)
7
```

7. Étant donnée une liste d'entiers non triés, écrire un programme qui teste si elle contient tous les entiers d'un certain intervalle. Donner sa complexité et proposer un algorithme en complexité linéaire.

```
>>> l = [4,1,2,3,5,7,5,2,1,3,6,9]
>>> intervalle_est_dans_liste(l,1,12)
False
>>> intervalle_est_dans_liste(l,2,5)
True
```

8. Étant donnée une liste l, écrire un programme permettant de trouver une instance de i, j, k tels que

$$0 \leq i < j < k < n \quad \text{et} \quad l[i] < l[j] < l[k].$$

```
>>> l = [4,0,7,3,5,7,5,2,1,3,6,9]
>>> triplet_avec_ordre(l)
(1,3,4)

>>> l = [14,12,7,5,5,3,0]
>>> triplet_avec_ordre(l)
(-1,-1,-1)
```

   — Donner sa complexité et proposer un algorithme en complexité linéaire.

9. Dominant d'une liste :un élément est majoritaire (aussi appelé dominant) dans une liste de taille n, c'est-à-dire un élément qui apparaît strictement plus de $n/2$ fois.
   — Écrire une fonction retourne un dominant s'il existe et évalue sa complexité.
   — Implémenter l'algorithme de **Boyer–Moore Majority Vote** [1] (linéaire en temps).
   — Expliquer l'idée de l'algorithme. Vérifier la complexité.

1. https ://en.wikipedia.org/wiki/Boyer%E2%80%93Moore_majority_vote_algorithm