Traitement détaillé et structuré des problèmes

# Problème 1 : Somme des éléments distincts

## Description claire :

Étant donné deux tableaux (ensembles) d'éléments, calculer la somme des éléments qui ne figurent pas simultanément dans les deux ensembles.

## Exemple détaillé :

Ensemble 1 : [3, 1, 7, 9]  
Ensemble 2 : [2, 4, 1, 9, 3]  
  
Éléments distincts uniquement dans l’ensemble 1 : 7  
Éléments distincts uniquement dans l’ensemble 2 : 2, 4  
  
Calcul de la somme : 7 + 2 + 4 = 13

## Algorithme structuré :

- Étape 1 : Initialiser la somme à zéro.  
- Étape 2 : Pour chaque élément du premier ensemble, vérifier s'il existe dans le deuxième ensemble.  
 Si l'élément n'est pas trouvé, l’ajouter à la somme.  
- Étape 3 : Répéter le processus pour chaque élément du deuxième ensemble.

# Problème 2 : Produit scalaire et orthogonalité

## Description précise :

Le produit scalaire de deux vecteurs est la somme des produits de leurs composantes correspondantes. Deux vecteurs sont orthogonaux si leur produit scalaire est nul.

## Exemple clair du produit scalaire :

V1 = [1, 2, 3]  
V2 = [4, 5, 6]  
  
Produit scalaire = (1×4) + (2×5) + (3×6) = 4 + 10 + 18 = 32

## Exemple d’orthogonalité :

V1 = [2, -2, 1], V2 = [1, 1, 2]  
Produit scalaire = 2×1 + (-2)×1 + 1×2 = 2 - 2 + 2 = 2 ≠ 0 (non orthogonaux)  
  
V1 = [1, 0, -1], V2 = [1, 5, 1]  
Produit scalaire = 1×1 + 0×5 + (-1)×1 = 1 - 1 = 0 (orthogonaux)

## Algorithme précis et structuré :

- Procédure (dot\_product) :  
 Initialiser ps = 0  
 Pour chaque élément, ps = ps + (v1[i] × v2[i])  
  
- Fonction orthogonalité :  
 Calculer produit scalaire  
 Si produit scalaire = 0, vecteurs orthogonaux.

## Pseudo-code explicatif :

fonction dot\_product(v1, v2):  
 ps = 0  
 pour i de 1 à taille(v1):  
 ps = ps + v1[i] × v2[i]  
 retourner ps  
  
fonction estOrthogonaux(v1, v2):  
 retourner dot\_product(v1, v2) == 0