

Métropole – 2026 – sujet1 - Correction

Exercice 2 (6 points)

Partie A – Représentation binaire des configurations

1. Nombre de configurations

Chacun des 9 jetons peut être sur 2 faces (Recto ou Verso), indépendamment des autres : il y a $2 \times 2 \times \dots \times 2 = 2^9 = 512$ configurations.

2. Ligne 1, ligne 2, colonne 2 depuis la figure 1

La figure 1 correspond à la liste $[0, 0, 1, 0, 0, 1, 1, 1, 0]$. On applique les coups :

coup	configuration obtenue
départ	$[0, 0, 1, 0, 0, 1, 1, 1, 0]$
ligne 1	$[1, 1, 0, 0, 0, 1, 1, 1, 0]$
ligne 2	$[1, 1, 0, 1, 1, 0, 1, 1, 0]$
colonne 2	$[1, 0, 0, 1, 0, 0, 1, 0, 0]$

Tous les jetons ne sont pas sur Recto : cette séquence **ne mène pas à la victoire**.

👉 *Commentaire : en jouant colonne 3 à la place de colonne 2, on gagne : ligne 1, ligne 2, colonne 3 retourne tous les jetons sur Recto.*

3. Entier de la figure 1

001001110 en binaire : $0 \times 256 + 0 \times 128 + 1 \times 64 + 0 \times 32 + 0 \times 16 + 1 \times 8 + 1 \times 4 + 1 \times 2 + 0 \times 1 = 64 + 8 + 4 + 2 = 78$.

4. Fonction représentant

```
def representant(liste):  
    n = 0  
    for bit in liste:  
        n = 2 * n + bit  
    return n
```

👉 *Commentaire : à chaque bit, on décale l'écriture d'un rang vers la gauche ($\times 2$) puis on ajoute le bit (méthode de Horner). Autre version : $n = n + \text{liste}[i] * 2^{**(\text{len}(\text{liste}) - 1 - i)}$. On vérifie : `representant([0, 0, 1, 0, 0, 0, 0, 1, 1])` renvoie 67.*

5. Fonction xor_etendu

```
def xor_etendu(l_x, l_y):
    resultat = []
    for i in range(len(l_x)):
        resultat.append(xor(l_x[i], l_y[i]))
    return resultat
```

6. Fonction configurations_suivantes

```
def configurations_suivantes(config):
    config_bin = binaire(config)
    voisins = []
    for nom in coups_possibles:
        coup = coups_possibles[nom]          # ligne facultative
        suivante = xor_etendu(config_bin, coup)
        voisins.append(representant(suivante))
    return voisins
```

👉 *Commentaire : on vérifie : `configurations_suivantes(67)` renvoie [387, 123, 68, 359, 209, 10, 338, 23], comme sur la figure 3. Le return est hors de la boucle (dans l'énoncé, il est mal aligné).*

Partie B – Graphe des configurations

7. Arête $x \rightarrow y$ donc arête $y \rightarrow x$

S'il existe une arête $x \rightarrow y$, c'est qu'un coup c transforme x en y : $y = x \oplus c$. Rejouer le même coup donne $y \oplus c = (x \oplus c) \oplus c = x$, car $c \oplus c$ ne contient que des 0 et $x \oplus 0 = x$. Jouer c depuis y ramène à x : **l'arête $y \rightarrow x$ existe.**

👉 *Commentaire : autrement dit, retourner deux fois les mêmes jetons les remet dans leur état initial : le graphe est non orienté.*

8. Dictionnaire ou matrice d'adjacence

Le graphe a $2^9 = 512$ sommets.

– Une **matrice d'adjacence** contient $512 \times 512 = 2^{18}$ entiers (262 144), presque tous nuls.

– Avec le **dictionnaire**, chaque sommet a exactement 8 voisins : 512 clés et $512 \times 8 = 2^{12}$ valeurs, soit $2^9 + 2^{12} = 4\,608$ entiers.

Le dictionnaire occupe environ **57 fois moins** de mémoire : le graphe est peu dense, ce qui justifie ce choix.

9. Fonction `parcours_profondeur`

```
def parcours_profondeur(graphe, configuration, vus):  
    vus.append(configuration)  
    for s in graphe[configuration]:  
        if s not in vus:  
            parcours_profondeur(graphe, s, vus)
```

10. La configuration 4 est perdante

```
vus = []  
parcours_profondeur(graphe, 4, vus)  
print(511 not in vus)    # affiche True
```

👉 *Commentaire : la victoire (tous les jetons Recto) correspond à la liste [1, 1, 1, 1, 1, 1, 1, 1, 1], c'est-à-dire l'entier $2^9 - 1 = 511$. Le parcours depuis 4 visite tous les sommets atteignables ; 511 n'en fait pas partie, donc 4 est perdante. Le parcours ne visite que 128 sommets : seules 128 des 512 configurations sont gagnantes.*

11. Le parcours le plus adapté

Le **parcours en largeur** : il visite les sommets par distance croissante au départ (1 coup, puis 2 coups, etc.). Le premier chemin trouvé vers 511 est donc le plus court. Le parcours en profondeur trouve un chemin, mais pas forcément le plus court.

👉 *Commentaire : par exemple, depuis la configuration 67, le parcours en largeur trouve une victoire en 4 coups : ligne 1, colonne 1, colonne 3, diagonale 1.*