README

KOALA Valentin
COULIBALY Cheick Ahmed
BAZIE Dureel Donaldson
BASSOLE Yipéné Harold Ezekiel
OUATTARA Arnauld

4 mai 2025

1 Description du problème

Le problème **BADXOR** sur SPOJ demande de compter le nombre de sous-ensembles d'un tableau A dont le **XOR bitwise** des éléments n'appartient pas à un tableau B.

2 Approche de la solution

L'approche utilise une **programmation dynamique** pour calculer tous les XOR possibles des sous-ensembles de A, puis exclut ceux qui sont présents dans B.

Points clés

- Utilisation d'un tableau de taille fixe $dp[1024]$ pour stocker le nombre de sous-ensembles ayant un XOR donné.
- Optimisation des entrées/sorties avec `ios_base::sync_with_stdio(false)`.
- Gestion du modulo $10^8 + 7$ pour éviter les débordements.

3 Code source

```
// Probl me BADXOR.cpp de SPOJ r solu avec C++14

#include <iostream> // Biblioth que standard C++, Pour les
    entr es/sorties
#include <vector> // Pour d finir et manipuler les vecteurs
#include <unordered_set> // Pour les ensembles non ordonn s
//#include <bitset> //biblioth que standard <bitset>, qui permet
    de manipuler des ensembles de bits, utilis e pour manipuler
    les bits et r duire la complexit  et le temps d'ex cution
```

```cpp
using namespace std;

const int MODULO = 100000007;// Modulo pour les r sultats

// Fonction principale
int main() {
  ios_base::sync_with_stdio(false);//d sactive la synchronisation entre les flux C++ (cin, cout) et les flux C (scanf, printf). Gain de performance notable sur des grandes entr es / sorties.
    cin.tie(nullptr); //d tache cin de cout car cin est li cout par d faut
  // Lecture du nombre de cas de test
    int test_cases;
    cin >> test_cases;
  // V rification du nombre de cas de test
    if (test_cases < 1 || test_cases > 20) {
        cerr << "Le nombre de cas de test doit  tre  compris entre 1 et 20.\n";
        return 1;
    }

    for (int case_num = 1; case_num <= test_cases; ++case_num) {
        int size_A, size_B;
        cin >> size_A >> size_B;
    // V rification des tailles des tableaux
        if (size_A < 0 || size_B < 0) {
            cerr << "Invalid array sizes.\n";
            return 1;
        }
        // D finition et Lecture des   lments   de A sous forme de vecteur
        vector<int> A(size_A);
        for (int i = 0; i < size_A; ++i) {
            cin >> A[i];
            if (A[i] < 0 || A[i] > 1000) {
                cerr << "Les   lments   de A doivent  tre  compris entre 0 et 1000.\n";
                return 1;
            }
        }
    // D finition et Lecture des   lments   de B sous forme d' ensemble non ordonn
        unordered_set<int> B_set;
        for (int i = 0; i < size_B; ++i) {
            int b;
            cin >> b;
            if (b < 0 || b > 1000) {
                cerr << "Les   lments   de B doivent  tre  compris entre 0 et 1000.\n";
                return 1;
            }
      B_set.insert(b);
        }

    vector<int> dp(1024, 0); /* Cr ation et initialisation d'un vecteur de longueur 1024   lments  , chacun ayant la valeur initiale 0.
```

```
                                Taille fixe pour repr senter tous
    les XOR possibles, diff rent avec notre programme python  o
    on
                                avait utilis  un dictionnaire
    esp rant  r duire la complexit  et le temps d'ex cution*/
        dp[0] = 1; // Car le sous-ensemble vide a un XOR = 0

        for (int a : A) {
      vector<int> new_dp = dp; // Copie pour mise    jour
    simultan e sans interf rence avec anciennes valeurs
            for (int x = 0; x < 1024; ++x) {
        int new_xor = x ^ a;// Calcul du nouveau XOR
        new_dp[new_xor] += dp[x];// Ajout de la valeur actuelle au
    nouveau XOR
                if (new_dp[new_xor] >= MODULO) {
                    new_dp[new_xor] -= MODULO; // R duction modulo
     optimis e
                }
            }
      dp = move(new_dp);// Mise    jour de dp avec new_dp
        }

        int total = 0;
        for (int x = 0; x < 1024; ++x) {
            if (!B_set.count(x)) { // Exclusion des mauvais XOR
                total = (total + dp[x]) % MODULO;
            }
        }

        cout << "Case " << case_num << ": " << total << '\n';
    }

    return 0;
}
```

4 Explications techniques

- **Tableau dynamique** : Le vecteur `dp` de taille 1024 stocke le nombre de sous-ensembles pour chaque valeur XOR possible.
- **Optimisation du modulo** : La réduction modulo est faite de manière incrémentale pour éviter les débordements.
- **Complexité** : $O(N \times 1024)$ par cas de test, où N est la taille de A.

5 EXEMPLE D'ENTREE/SORTIE

Exemple d'entrée/sortie

Entrée

```
2
2 3
```

```
1 2
0 1 2
1 3
1
0 1 2
```

Sortie

```
Case 1: 1
Case 2: 0
```

Explication

- **Cas 1** :
  - Tableau A = [1, 2]
  - Tableau B = [0, 1, 2]
  - Sous-ensembles valides : seul {1, 2} a un XOR (3) qui n'est pas dans B
  - Résultat : 1
- **Cas 2** :
  - Tableau A = [1]
  - Tableau B = [0, 1, 2]
  - Le seul sous-ensemble non-vide {1} a un XOR (1) présent dans B
  - Résultat : 0