📚 Guide Complet : Comprendre les Algorithmes - Part 1 et Part 2

🎯 Vue d'Ensemble Simplifiée

Imaginez que vous avez une machine à café avec plusieurs boutons. Chaque bouton affecte plusieurs voyants ou compteurs. Votre objectif : trouver quels boutons presser pour atteindre un état désiré.

Part 1 : Les boutons sont des interrupteurs (ON/OFF) **Part 2** : Les boutons sont des compteurs (0, 1, 2, 3...)

🔵 PART 1 : Élimination de Gauss en GF(2)

🤔 Le Problème en Termes Simples

Exemple Concret

Vous avez 4 voyants : `[.##.]` (éteint, allumé, allumé, éteint) Et 3 boutons :

- Bouton A : inverse les voyants 0 et 2
- Bouton B : inverse les voyants 1 et 3
- Bouton C : inverse les voyants 0 et 1

Question : Quels boutons presser pour allumer exactement les voyants 1 et 2 ?

La Particularité

- Presser un bouton **deux fois** = ne rien faire du tout
- On cherche : presser chaque bouton **0 ou 1 fois**

📊 Étape 1 : Transformer en Équations

Représentation Visuelle

```
État actuel   : [0, 0, 0, 0]  (tous éteints)
État désiré   : [0, 1, 1, 0]  (voyants 1 et 2 allumés)

Boutons :
A : affecte voyants 0,2  →  A = [1, 0, 1, 0]
B : affecte voyants 1,3  →  B = [0, 1, 0, 1]
C : affecte voyants 0,1  →  C = [1, 1, 0, 0]
```

En Équations

Pour chaque voyant, écrivons une équation :

```
Voyant 0 : A + C = 0  (on veut 0, donc éteint)
Voyant 1 : B + C = 1  (on veut 1, donc allumé)
Voyant 2 : A     = 1  (on veut 1, donc allumé)
Voyant 3 : B     = 0  (on veut 0, donc éteint)
```

Important : Dans GF(2), l'addition est le XOR (ou exclusif)

- 0 + 0 = 0
- 0 + 1 = 1
- 1 + 0 = 1
- 1 + 1 = 0 ← Deux 1 s'annulent !

🔧 Étape 2 : Construire la Matrice Augmentée

Matrice : Tableau Organisé

```
      A  B  C | Cible
     ─────────────────
V0 │  1  0  1 │  0
V1 │  0  1  1 │  1
V2 │  1  0  0 │  1
V3 │  0  1  0 │  0
```

Lecture :

- Ligne 1 : "A + C doit égaler 0"
- Ligne 2 : "B + C doit égaler 1"
- etc.

🎨 Étape 3 : Élimination de Gauss (Le Cœur de l'Algo)

Objectif : Forme "Échelonnée"

On veut transformer la matrice pour avoir une forme "en escalier" :

```
1 * * | *
0 1 * | *
0 0 1 | *
0 0 0 | *
```

Animation Étape par Étape

État Initial

```
     A  B  C | Cible
    ─────────────────
V0 │  1  0  1 │  0   ← Pivot ligne 0, colonne A
V1 │  0  1  1 │  1
V2 │  1  0  0 │  1
V3 │  0  1  0 │  0
```

Étape 1 : Éliminer A dans les autres lignes

"Éliminer" = rendre à 0 en utilisant XOR avec la ligne pivot

```
V2 a un 1 dans colonne A
→ V2 = V2 XOR V0

Avant V2: [1, 0, 0, 1]
V0:       [1, 0, 1, 0]
XOR:      [0, 0, 1, 1]  ← Nouveau V2
```

Résultat après élimination de A :

```
     A  B  C | Cible
    ─────────────────
V0 │  1  0  1 │  0   ✓ Pivot trouvé
V1 │  0  1  1 │  1
V2 │  0  0  1 │  1   ← A éliminé
V3 │  0  1  0 │  0
```

Étape 2 : Passer à la colonne B

```
      A  B  C | Cible
     ─────────────────
V0 │  1  0  1 │  0
V1 │  0  1  1 │  1   ← Pivot ligne 1, colonne B
V2 │  0  0  1 │  1
V3 │  0  1  0 │  0
```

Éliminer B dans V3 :

```
V3 = V3 XOR V1
[0, 1, 0, 0] XOR [0, 1, 1, 1] = [0, 0, 1, 1]
```

Résultat :

```
      A  B  C | Cible
     ─────────────────
V0 │  1  0  1 │  0
V1 │  0  1  1 │  1
V2 │  0  0  1 │  1   ← Pivot colonne C
V3 │  0  0  1 │  1
```

Étape 3 : Finaliser avec colonne C

Éliminer C dans toutes les autres lignes :

```
V0 = V0 XOR V2  →  [1, 0, 1, 0] XOR [0, 0, 1, 1] = [1, 0, 0, 1]
V1 = V1 XOR V2  →  [0, 1, 1, 1] XOR [0, 0, 1, 1] = [0, 1, 0, 0]
V3 = V3 XOR V2  →  [0, 0, 1, 1] XOR [0, 0, 1, 1] = [0, 0, 0, 0]
```

Forme Finale (Échelonnée Réduite)

```
      A  B  C | Cible
     ─────────────────
V0 │  1  0  0 │  1   ← A = 1
V1 │  0  1  0 │  0   ← B = 0
V2 │  0  0  1 │  1   ← C = 1
V3 │  0  0  0 │  0   ← Toujours vrai
```

✅ Étape 4 : Lire la Solution

La matrice finale nous donne directement la réponse :

```
A = 1  ⟶  Presser le bouton A une fois
B = 0  ⟶  Ne PAS presser le bouton B
C = 1  ⟶  Presser le bouton C une fois
```

Vérification :

```
État initial : [0, 0, 0, 0]
Presser A    : [1, 0, 1, 0]  (inverse V0 et V2)
Presser C    : [0, 1, 1, 0]  (inverse V0 et V1)
             = [0, 1, 1, 0]  ✓ C'est bien notre cible !
```

Total : 2 pressions (A et C)

🌟 Cas Spécial : Variables Libres

Que se passe-t-il si on a moins d'équations que de boutons ?

Exemple :

```
2 voyants, 3 boutons

      A  B  C | Cible
     ─────────────────
V0 │  1  0  1 │  1
V1 │  0  1  1 │  0
```

Après élimination :

```
      A  B  C | Cible
     ─────────────────
V0 │  1  0  1 │  1   ← A + C = 1
V1 │  0  1  1 │  0   ← B + C = 0
```

Observation : C n'a pas de ligne à lui ! → C est une **variable libre** (on peut choisir 0 ou 1)

Essayer Toutes les Possibilités

Si C = 0 :

- A = 1, B = 0, C = 0 → 1 pression

Si C = 1 :

- A = 0, B = 1, C = 1 → 2 pressions

Minimum : 1 pression ✓

🔢 Pourquoi GF(2) ?

GF(2) = Corps de Galois à 2 éléments

C'est juste un nom compliqué pour dire :

- On travaille avec 0 et 1
- Addition = XOR (ou exclusif)
- Presser 2 fois un bouton = ne rien faire

Propriétés magiques :

```
1 + 1 = 0  (deux allumages = éteindre)
-1 = 1     (annuler = refaire)
```

📈 Complexité Part 1

Temps : $O(n^2 \times m + 2^f \times m)$

- n = nombre de voyants
- m = nombre de boutons
- f = nombre de variables libres (généralement 0-5)

Exemple : 10 voyants, 12 boutons, 2 variables libres → $10^2 \times 12 + 2^2 \times 12 = 1200 + 48 = 1248$ opérations

---

🟢 PART 2 : Programmation Linéaire en Entiers (ILP)

🤔 Le Problème en Termes Simples

Exemple Concret

Vous avez 3 compteurs : `{5, 8, 3}` (on veut atteindre ces valeurs) Et 3 boutons :

- Bouton A : +1 aux compteurs 0 et 2
- Bouton B : +1 au compteur 1
- Bouton C : +1 aux compteurs 0 et 1

Question : Combien de fois presser chaque bouton pour atteindre exactement {5, 8, 3} ?

La Différence avec Part 1

- On peut presser un bouton **plusieurs fois** (0, 1, 2, 3, ...)
- Les effets **s'additionnent** (pas de XOR)

📊 Étape 1 : Modéliser le Problème

Représentation Visuelle

```
Compteur 0 : Doit valoir 5
Compteur 1 : Doit valoir 8
Compteur 2 : Doit valoir 3

Boutons :
A : +1 aux compteurs 0,2
B : +1 au compteur 1
C : +1 aux compteurs 0,1
```

En Équations Mathématiques

```
Compteur 0 : A + C = 5
Compteur 1 : B + C = 8
Compteur 2 : A     = 3
```

Avec :

- A, B, C ≥ 0 (entiers positifs)

- Minimiser : A + B + C (total de pressions)

🎯 Étape 2 : Résoudre le Système

Méthode Simple : Substitution

Étape 2.1 : Compteur 2

```
Compteur 2 : A = 3
```

→ **A = 3** (évident !)

Étape 2.2 : Compteur 0

```
Compteur 0 : A + C = 5
           : 3 + C = 5
           : C = 2
```

→ **C = 2**

Étape 2.3 : Compteur 1

```
Compteur 1 : B + C = 8
           : B + 2 = 8
           : B = 6
```

→ **B = 6**

Vérification

```
Compteur 0 : A + C = 3 + 2 = 5 ✓
Compteur 1 : B + C = 6 + 2 = 8 ✓
Compteur 2 : A     = 3     = 3 ✓
```

Total : A + B + C = 3 + 6 + 2 = **11 pressions**

🔴 Problème : Et Si C'est Plus Compliqué ?

Exemple Difficile

```
Compteur 0 : A + B = 10
Compteur 1 : A + C = 8
Compteur 2 : B + C = 12
```

Il y a **plusieurs solutions** :

- A=6, B=4, C=2 → Total = 12
- A=5, B=5, C=3 → Total = 13
- A=4, B=6, C=4 → Total = 14

Question : Laquelle minimise le total ?

→ C'est là qu'on utilise la **Programmation Linéaire** !

🧮 Qu'est-ce que la Programmation Linéaire ?

Définition Simple

Un problème avec :

1. **Variables** : ce qu'on cherche (A, B, C)
2. **Contraintes** : équations à respecter
3. **Objectif** : ce qu'on veut minimiser/maximiser

Visualisation 2D

Imaginons 2 boutons seulement : A et B

```
Contraintes :
- A + B ≥ 10  (ligne 1)
- A ≥ 0       (ligne 2)
- B ≥ 0       (ligne 3)

Objectif : Minimiser A + B
```

Graphique :

```
  B
  |
15 |    Zone impossible
  |   (A + B < 10)
10 |  /────────────────
  | / Zone possible
 5 |/  (A + B ≥ 10)
  |
  └──────────────────> A
  0  5  10  15

Point optimal : A = 0, B = 10
ou A = 10, B = 0 (au choix)
```

🤖 Comment le Solveur ILP Fonctionne

Méthode : Branch and Bound (Exploration Intelligente)

C'est comme chercher le chemin le plus court dans un labyrinthe.

Étape 1 : Relaxation Continue

D'abord, on **ignore** la contrainte "entiers" :

```
Solution continue : A = 3.5, B = 6.5, C = 2.2
Total = 12.2
```

→ C'est une **borne inférieure** : on ne peut pas faire mieux que 12.2

Étape 2 : Arrondir et Tester

On essaie d'arrondir intelligemment :

```
Essai 1 : A = 3, B = 7, C = 2  →  Vérifie les contraintes ? NON
Essai 2 : A = 4, B = 6, C = 2  →  Vérifie les contraintes ? NON
Essai 3 : A = 3, B = 6, C = 3  →  Vérifie les contraintes ? OUI ✓
Total = 12
```

Étape 3 : Optimisation

Le solveur explore intelligemment l'espace :

```
Arbre de décision :

              Racine
             /     \
          A=3       A=4
         /  \      /  \
      B=6   B=7  B=5   B=6
      /\    /\   /\    /\
    C=2 C=3 ...  ...   ...

Test chaque branche, élimine celles qui sont pires
```

🎨 Visualisation Complete d'un Exemple

Problème

```
Compteurs cibles : {3, 5, 4}

Boutons :
A : affecte compteurs 0, 2
B : affecte compteurs 1, 2
C : affecte compteurs 0, 1
```

Étape par Étape

1. Modélisation

```
Compteur 0 : A + C = 3
Compteur 1 : B + C = 5
Compteur 2 : A + B = 4
```

2. Visualisation des Contraintes

```
État après A=1 : {1, 0, 1}
État après B=1 : {1, 1, 2}
État après C=1 : {2, 2, 2}
État après A=2 : {3, 2, 3}
État après B=3 : {3, 5, 6}  ← Trop pour compteur 2 !
```

3. Recherche de la Solution

Le solveur teste systématiquement :

```
Essai avec A=1, B=3, C=2 :
  Compteur 0 : 1 + 2 = 3 ✓
  Compteur 1 : 3 + 2 = 5 ✓
  Compteur 2 : 1 + 3 = 4 ✓
  Total : 1 + 3 + 2 = 6 pressions
```

Est-ce optimal ? Le solveur explore d'autres combinaisons...

```
Essai avec A=2, B=2, C=1 :
  Compteur 0 : 2 + 1 = 3 ✓
  Compteur 1 : 2 + 1 = 3 ✗  (on veut 5)

Non valide, on continue...
```

Solution finale : A=1, B=3, C=2 (6 pressions) ✓

📊 Comparaison Part 1 vs Part 2

Tableau Récapitulatif

```
┌─────────────────────┬───────────────────────┬───────────────────────┐
│ Critère     │ Part 1     │ Part 2     │
├─────────────────────┼───────────────────────┼───────────────────────┤
│ Domaine        │ {0, 1}          │ ℕ (0,1,2,3,...) │
│ Opération      │ XOR (inverse)    │ Addition        │
│ Algorithme     │ Gauss GF(2)      │ ILP (minilp)    │
│ Complexité     │ O(n²m)           │ Exponentiel*    │
│ Solution unique │ Parfois non     │ Souvent oui     │
│ Difficulté     │ Polynomial       │ NP-complet      │
└─────────────────────┴───────────────────────┴───────────────────────┘

* En pratique, très rapide grâce aux optimisations
```

🎓 Résumé pour les Nuls

Part 1 : Comme un Puzzle Sudoku

1. Chaque bouton = une contrainte

2. On transforme en tableau
3. On simplifie ligne par ligne (élimination)
4. On lit la solution directement

Astuce : Les "1 + 1 = 0" rendent tout magique !

Part 2 : Comme un Jeu d'Optimisation

1. On cherche des nombres entiers
2. On doit respecter des contraintes
3. On veut minimiser le total
4. Un algorithme intelligent explore les possibilités

Astuce : Le solveur ILP fait le travail difficile pour nous !

💡 Points Clés à Retenir

Part 1

- ✅ Simple à comprendre (XOR)
- ✅ Toujours rapide
- ⚠️ Parfois plusieurs solutions possibles

Part 2

- ✅ Plus flexible (plusieurs pressions possibles)
- ✅ Solution souvent unique
- ⚠️ Nécessite un solveur spécialisé

🧪 Exercice Pratique

Essayez de résoudre à la main :

```
Part 1 :
Cible : [1, 0, 1]
Boutons : A(0,1), B(1,2), C(0,2)

Part 2 :
Cible : {2, 3, 1}
Boutons : A(0), B(1), C(0,1), D(2)
```

Solutions :

- Part 1 : A=1, B=1, C=0 (2 pressions)
- Part 2 : A=2, B=3, C=0, D=1 (6 pressions)

Essayez de suivre les étapes décrites ci-dessus ! 🎯