Approche de Résolution Approche Force Brute :

- Génère tous les sous-ensembles possibles de demandes.
- Vérifie la compatibilité de chaque sous-ensemble.
- Sélectionne le sous-ensemble compatible de taille maximale.
- **Complexité** : $O(2^n \times n^2)$ (inefficace pour de grandes entrées).

Utilisation de Force Brute

- **Craquage de mots de passe** : En cyber sécurité, les attaques par force brute consistent à essayer systématiquement toutes les combinaisons possibles pour deviner un mot de passe. Cette méthode est simple mais peut être très longue et gourmande en ressources.

- **Cryptanalyse** : La méthode de force brute est utilisée pour casser des algorithmes cryptographiques en essayant toutes les clés possibles jusqu'à trouver la bonne. C'est pourquoi les algorithmes de chiffrement puissants utilisent des clés longues pour rendre les attaques par force brute impraticables.

- **Résolution de puzzles** : La force brute peut être appliquée pour résoudre des puzzles tels que le Rubik's Cube, les Sudoku et d'autres problèmes combinatoires en recherchant toutes les configurations possibles jusqu'à trouver la solution.

- **Problèmes de recherche** : En informatique, les algorithmes de force brute sont utilisés pour résoudre des problèmes de recherche, comme trouver un élément spécifique dans une liste ou résoudre le problème du voyageur de commerce en vérifiant toutes les routes possibles.

- **Problèmes d'optimisation** : La force brute peut être utilisée pour trouver la solution optimale à des problèmes comme le problème du sac à dos, où toutes les combinaisons possibles d'objets sont considérées pour déterminer la valeur maximale qui tient dans une capacité donnée.

- **Recherche de chaînes** : En traitement de texte, les méthodes de force brute peuvent être utilisées pour rechercher une sous-chaîne dans une chaîne en vérifiant chaque position possible jusqu'à ce qu'une correspondance soit trouvée.

- **Théorie des graphes** : Les approches par force brute peuvent être utilisées pour résoudre des problèmes en théorie des graphes, tels que trouver le chemin le plus court, le flux maximum ou les circuits hamiltoniens en explorant tous les chemins ou combinaisons possibles.

I. Problème de location d'un camion

Problème :

Le problème consiste à **maximiser le nombre de clients satisfaits** en sélectionnant un sous-ensemble de demandes de location d'un camion qui ne se chevauchent pas. Chaque demande est représentée par un intervalle de temps $[d_i, f_i]$, où d_i est la date de début et f_i est la date de fin. Deux demandes sont **compatibles** si leurs intervalles **ne se chevauchent pas**.

Objectif :

Trouver un sous-ensemble maximal de demandes compatibles deux à deux.

2. **Exemple**

Données :

- **Considérons les demandes suivantes :**

Demande	**Début (d_i)**	**Fin (f_i)**
e1	1	3
e2	2	5
e3	3	7
e4	6	9
e5	8	10

Résolution avec l'Approche Force Brute :

1. Générer tous les sous-ensembles possibles (il y a 2^5=32 sous-ensembles).
2. Vérifier la compatibilité de chaque sous-ensemble.
3. Trouver le sous-ensemble compatible de taille maximale.

Liste des 32 Combinaisons

1. **{}** (ensemble vide)
2. **{e1}**
3. **{e2}**
4. **{e3}**
5. **{e4}**
6. **{e5}**
7. **{e1,e2}**
8. **{e1,e3}**
9. **{e1,e4}**
10. **{e1,e5}**
11. **{e2,e3}**
12. **{e2,e4}**
13. **{e2,e5}**
14. **{e3,e4}**
15. **{e3,e5}**
16. **{e4,e5}**
17. **{e1,e2,e3 }**

18. **{e1,e2,e4 }**
19. **{e1,e2,e5}**
20. **{e1,e3,e4}**
21. **{e1,e3,e5}**
22. **{e1,e4,e5}**
23. **{e2,e3,e4}**
24. **{e2,e3,e5}**
25. **{e2,e4,e5}**
26. **{e3,e4,e5}**
27. **{e1,e2,e3,e4}**
28. **{e1,e2,e3,e5}**
29. **{e1,e2,e4,e5}**
30. **{e1,e3,e4,e5}**
31. **{e2,e3,e4,e5}**
32. **{e1,e2,e3,e4,e5}**

Vérification de la Compatibilité

Pour chaque combinaison, on vérifie si les demandes sont **compatibles** (c'est-à-dire qu'elles ne se chevauchent pas). Par exemple :

1. **{e1,e2}** :
   - $e1$: [1, 3]
   - $e2$: [2, 5]
   - **Incompatibles** (chevauchement entre 2 et 3).
2. **{e1,e3}** :
   - $e1$: [1, 3]
   - $e3$: [3, 7]
   - **Compatibles** (pas de chevauchement).
3. **{e1,e3,e5}** :
   - $e1$: [1, 3]
   - $e3$: [3, 7]
   - $e5$: [8, 10]
   - **Compatibles** (aucun chevauchement).

4. **{e2,e3,e4}** :
    - *e2 : [2, 5]*
    - *e3 : [3, 7]*
    - *e4 : [6, 9]*
    - **Incompatibles** (chevauchements multiples).

Résultat :

Le sous-ensemble maximal trouvé est :

- *e*1 : [1, 3]
- *e*3 : [3, 7]
- *e*5 : [8, 10]

II. Problème de rendu de monnaie

Problème

Le **problème de rendu de monnaie** consiste à déterminer le nombre minimum de pièces nécessaires pour rendre une certaine somme d'argent, en utilisant des pièces de valeurs données. Par exemple, si les pièces disponibles sont de 1, 2 et 5 unités, et que nous devons rendre 6 unités, une solution optimale serait d'utiliser une pièce de 5 et une pièce de 1 (total de 2 pièces).

Exemple

Données :

- Pièces disponibles : {1, 2, 5}
- Somme à rendre : 6

Combinaisons possibles :

1. **1, 1, 1, 1, 1, 1** (6 pièces de 1)
2. **1, 1, 1, 1, 2** (4 pièces : 4 × 1 + 1 × 2)
3. **1, 1, 2, 2** (4 pièces : 2 × 1 + 2 × 2)
4. **2, 2, 2** (3 pièces : 3 × 2)
5. **1, 5** (2 pièces : 1 × 1 + 1 × 5)

Résultat optimal :

- **1, 5** (2 pièces)