Algorithme de Glouton (Greedy)

Définition

Un algorithme glouton (ou "greedy algorithm") est une méthode algorithmique qui construit une solution optimale en prenant, étape par étape, la décision qui semble la meilleure sur le moment, sans considérer les conséquences à long terme. Autrement dit, à chaque étape, il sélectionne l'option localement optimale en espérant qu'elle mènera à une solution globalement optimale.

Propriétés principales :

1. **Choix local optimal :** À chaque étape, on prend la meilleure décision possible à ce moment précis, sans se préoccuper des conséquences sur les étapes suivantes.
2. **Solution globale espérée :** L'objectif est que la combinaison des choix locaux optimaux aboutisse à une solution optimale globale.

Exemple 1 : Le problème de la monnaie.

Problème : Étant donné un montant à rendre et une liste des valeurs des pièces disponibles (par exemple 1, 5, 10, 20, etc.), déterminer le nombre minimal de pièces nécessaires pour rendre ce montant.

Algorithme glouton :

1. Trier les pièces disponibles par ordre décroissant.
2. Initialiser le montant restant à rendre.
3. Pour chaque pièce (de la plus grande à la plus petite) :
    - Utiliser autant de pièces de cette valeur que possible sans dépasser le montant restant.
    - Réduire le montant restant en soustrayant la valeur totale des pièces utilisées.
4. Répéter jusqu'à ce que le montant restant soit égal à 0.

Exemple pratique :

1. Montant à rendre : **37**
2. Pièces disponibles : **[20, 10, 5, 1]**

Étapes :

- On commence avec la pièce de 20 → 1 pièce utilisée (reste = 37 - 20 = 17).
- Ensuite, la pièce de 10 → 1 pièce utilisée (reste = 17 - 10 = 7).
- Puis, la pièce de 5 → 1 pièce utilisée (reste = 7 - 5 = 2).
- Enfin, la pièce de 1 → 2 pièces utilisées (reste = 2 - 2 = 0).

Résultat : 1 pièce de 20, 1 pièce de 10, 1 pièce de 5, et 2 pièces de 1.

Deuxième exemple détaillé

Cas 1 : Algorithme glouton fonctionne

- Entrée : coins = [1, 2, 5, 10, 20, 50, 100, 200], amount = 93
- Exécution :
    - Choisir 50 : amount = 93 - 50 = 43, count = 1
    - Choisir 20 : amount = 43 - 20 = 23, count = 2
    - Choisir 20 : amount = 23 - 20 = 3, count = 3
    - Choisir 2 : amount = 3 - 2 = 1, count = 4
    - Choisir 1 : amount = 1 - 1 = 0, count = 5
- Résultat : 5 (solution optimale : 50 + 20 + 20 + 2 + 1)

Questions :

1. Résoudre le problème avec la méthode Force Brute,
2. Résoudre le problème avec la méthode de Glouton,
3. Comparer l'efficacité de deux algorithmes,
4. Tester l'algorithme de glouton avec des pièces disponibles [1, 3, 4], on veut rendre 6.
5. Conclure.

Exemple 2 : Le problème location de camion.

Le problème de satisfaire le maximum de demandes avec des dates de début et de fin est souvent appelé le **problème des intervalles compatibles** ou **Interval Scheduling Maximization Problem**. Il s'agit de sélectionner le plus grand nombre de demandes compatibles entre elles (c'est-à-dire des intervalles qui ne se chevauchent pas).

Approche gloutonne :

Pour résoudre ce problème, une stratégie gloutonne est idéale. Voici les étapes :

1. Trier toutes les demandes par leur **date de fin croissante**.
2. Sélectionner les demandes dans cet ordre :
    - Ajouter une demande si sa date de début est **postérieure ou égale** à la date de fin de la dernière demande sélectionnée.
3. Répéter jusqu'à ce que toutes les demandes soient examinées.

Exemple pratique :

Données des demandes :

Nous avons les demandes suivantes, avec chaque intervalle représentant une période de réservation :

- Demande 1 : (1, 3)
- Demande 2 : (2, 5)
- Demande 3 : (4, 6)
- Demande 4 : (6, 8)
- Demande 5 : (5, 7)
- Demande 6 : (8, 9)

Étapes de résolution :

1. **Triage des demandes par date de fin croissante :** On trie les demandes en fonction de leur date de fin :
    - Demande 1 : (1, 3)
    - Demande 2 : (2, 5)
    - Demande 3 : (4, 6)
    - Demande 5 : (5, 7)
    - Demande 4 : (6, 8)
    - Demande 6 : (8, 9)
2. **Sélection des demandes compatibles :**
    - On commence avec la première demande (1, 3), elle est sélectionnée. Dernière date de fin acceptée = **3**.
    - Ensuite, on examine (2, 5) : elle **n'est pas compatible** car elle commence avant la date de fin de la demande sélectionnée (3).
    - On regarde (4, 6) : elle est **compatible** car elle commence après 3. Dernière date de fin acceptée = **6**.
    - Ensuite, (5, 7) : elle **n'est pas compatible** car elle chevauche la demande (4, 6).
    - On passe à (6, 8) : elle est **compatible** car elle commence après 6. Dernière date de fin acceptée = **8**.
    - Enfin, (8, 9) : elle est **compatible** car elle commence après 8. Dernière date de fin acceptée = **9**.

Résultat :

Demandes sélectionnées :

- (1, 3)
- (4, 6)
- (6, 8)
- (8, 9)

Nombre total de demandes satisfaites : 4.

Questions :

1. Résoudre le problème avec la méthode Force Brute,
2. Résoudre le problème avec la méthode de Glouton,
3. Comparer l’efficacité de deux algorithmes,