Algorithme glouton

1. Définition et idée générale

Un algorithme glouton fait, à *chaque* étape, le *meilleur choix* ***possible*** *à cette étape*, sans jamais revenir en arrière pour réévaluer les choix précédents.

> ⚠️ **Gros problème !**
>
> Le meilleur choix « local » à chaque étape ne garantit pas d'obtenir, à l'issue du processus complet, une solution globale **optimale**.

2. Illustration

Le glouton se comporte comme quelqu'un qui ferait toujours le meilleur choix *sur le moment*, sans tester les alternatives à long terme. Un comportement « court-termiste », en somme.

3. Applications

Un algorithme glouton est souvent rapide à l'exécution, et c'est là son principal intérêt. Mais, comme indiqué, il peut échouer à trouver la solution **optimale** sur certains problèmes. On l'utilise surtout pour faire du « vite et bien », plutôt que du « parfait mais lent ».

3.1. Exemple 1 : le rendu de monnaie (succès relatif)

Le rendu de monnaie (https://fr.wikipedia.org/wiki/Probl%C3%A8me_du_rendu_de_monnaie) est un problème d'algorithmique classique (implémenté dans les caisses enregistreuses, les distributeurs de billets, les « machines à monnaies »...).

Énoncé : étant donné un système de monnaie (pièces ou billets, peu importe), comment rendre une somme donnée de façon optimale, c'est-à-dire avec le nombre minimal de pièces et billets ?

Solution en Python :

```python
def rendu_glouton(montant : int, pieces : list) -> list:
    """
    Prend en paramètre un montant et une liste triée de pièces (des entiers).
    On considère qu'on dispose d'une quantité illimitée de pièces, ici.
    Renvoie la liste des pièces qu'il faut rendre pour résoudre le problème.
    """
    resultat = []
    for piece in sorted(pieces, reverse=True):  # Prendre la plus grande pièce en 1er
        while montant >= piece:                 # Tant que c'est possible...
            resultat.append(piece)              # ...ajouter cette plus grande pièce
            montant -= piece                    # Actualiser le montant restant à rendre
    return resultat
```

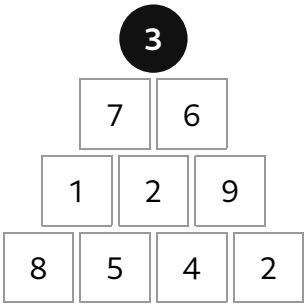
Cet algorithme prend d'abord la plus grande pièce possible, puis recommence avec la somme restante.

> ⚠️ **Subtilité**
>
> Dans ce cas précis, le glouton **peut** ou non être optimal. Par optimal, on veut dire « réussir à rendre la monnaie en trouvant **le** nombre **minimal** de pièces ». **Mais cela dépend du *système de pièces !***
>
> - Dans le système de pièces européen (en centimes : 1, 2, 5, 10, 20, 50, 100, 200), l'algorithme glouton est *toujours* optimal (on peut le *démontrer*).
>   **Exemple :** pour rendre 37, il donnera 20+10+5+2.
> - Dans le système de pièces (1, 3, 4), l'algorithme glouton **n**'est **pas** optimal.
>   **Exemple :** pour rendre 6, il donnera 4+1+1, alors que 3+3 est optimal.

3.2. Exemple 2 : pyramide de nombres (échec fréquent)

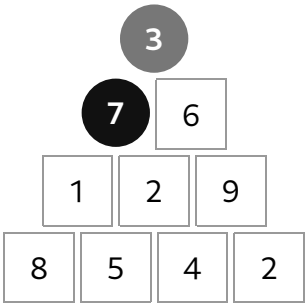
Un exemple classique de situation conduisant à un résultat sous-optimal est celui de la **pyramide de nombres**. Observez la situation ci-dessous. On part du sommet, et à chaque étape on ne peut que descendre : soit vers la case en dessous à gauche, soit vers la case en dessous à droite. L'algorithme consiste à **toujours faire le *meilleur choix possible***, c'est-à-dire celui qui **maximise la somme des nombres rencontrés.**

Étape 1



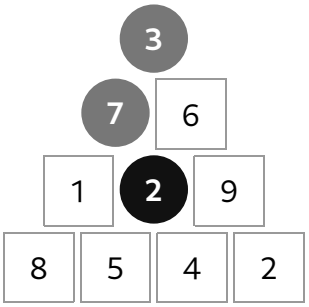
Somme *partielle* : 3

Étape 2



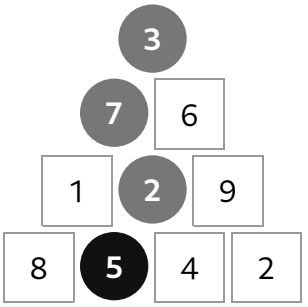
Somme *partielle* : 10
(car 7 > 6)

Étape 3



Somme *partielle* : 12
(car 2 > 1)

Étape 4



Somme **finale** : 17
(car 5 > 4)

Pourtant le meilleur chemin est 3 → 6 → 9 → 4, somme 22.

Conclusion : un choix local maximal à chaque étape peut mener à une somme totale inférieure à celle d'un autre chemin.

Remarque : *la valeur 9 est inaccessible depuis 7 (trop à droite) ; de même, 8 est inaccessible depuis 2 (trop à gauche).*

4. En bref

- Le glouton est simple et rapide.
- Il n'est correct que pour certains problèmes.
- Il faut distinguer « bon choix local » et « solution optimale globale ».

> ⚠️ **Piège sur l'optimalité !**
>
> **NE PAS** croire qu'un algorithme glouton est *toujours* optimal : on l'a vu, cela peut être faux.

5. Liens avec d'autres notions

- Le **rendu de monnaie** est un exemple classique.
- Le **sac à dos** montre souvent les limites du glouton.
- La **programmation dynamique** est l'alternative quand le glouton échoue.
- La **complexité** du glouton est souvent plus faible que celle d'une exploration exhaustive.