DSA · FICHE PRATIQUE

Cheatsheet : Glouton, programmation dynamique et retour sur trace

Justifier un choix local, réutiliser des sous-problèmes ou explorer des choix.

Par Ali ZAINOUL · Cours gratuit

Définitions, contrats et coûts

Justifier un choix local, réutiliser des sous-problèmes ou explorer des choix.

Glouton, programmation dynamique et retour sur trace

Notion	À retenir
Glouton	Choix local ; preuve nécessaire que ce choix reste compatible avec une solution optimale.
Programmation dynamique	Sous-problèmes définis, état suffisant, cas de base, récurrence et ordre de calcul.
Retour sur trace	Choisir, explorer, puis restaurer ; coût potentiellement exponentiel.

Vérifier sur un exemple

Des pièces qui piègent le glouton

Calculez une solution optimale pour un petit montant avant de faire confiance à une stratégie locale.

Python

```python
# minimum_coins.py
coins = [1, 3, 4]
amount = 6
best = [0] + [amount + 1] * amount
for total in range(1, amount + 1):
    best[total] = min(best[total - coin] + 1 for coin in coins if coin <=
total)
print(best[amount])
# Expected output:
# 2
```

> **Résultat expliqué.** La solution est 3 + 3. Un glouton qui choisit toujours la plus grande pièce prend 4 + 1 + 1. Ici, Θ(Ak) étapes et Θ(A) mémoire pour montant A et k pièces.

> **Erreur à éviter.** Cette borne dépend de la valeur du montant, pas seulement de la longueur de son écriture : elle est pseudo-polynomiale.

▼ **Pourquoi restaurer un choix après un appel récursif de retour sur trace ?**

Pour que la branche suivante parte du même état initial, sans hériter des mutations de la branche précédente.

Télécharger cette fiche en PDF · Toutes les fiches DSA

Version web : https://www.clearcode.fr/courses/programming/dsa/cheatsheet/strategies