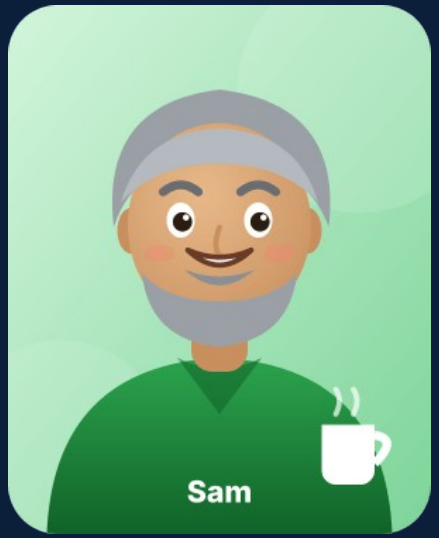
HEF-010 · MISSION 07 / 10

Glouton et diviser pour régner

Deux grandes stratégies : prendre le meilleur choix tout de suite, ou couper le problème en deux.





Badge : Stratège

75 min · débutant

L'histoire

Maya organise un atelier. Elle a **une seule salle** et une liste de présentations avec leurs horaires de début et de fin. Elle veut en caser **le plus possible**.

Kodi essaie toutes les combinaisons. Après 20 présentations, son ordinateur chauffe.



> *« Il y a une astuce, dit Sam. Une règle simple, qu'on applique à chaque étape, et qui marche ici. »*

Dans cette mission : deux stratégies pour attaquer les problèmes.

Ta mission

À la fin, je serai capable de…

1. expliquer la stratégie gloutonne avec un exemple
2. savoir que le glouton ne marche pas toujours
3. appliquer « diviser pour régner » à un problème
4. coder un tri fusion simple
5. choisir entre ces stratégies selon le problème

Défi d'échauffement

Défi d'échauffement

Tu dois rendre **63 centimes** avec le moins de pièces possible. Pièces disponibles : 50, 20, 10, 5, 2, 1. Écris ta méthode. Maintenant, imagine un pays où les pièces sont 1, 3 et 4 : rends **6 centimes**. La même méthode donne-t-elle le meilleur résultat ?

Le concept

Stratégie gloutonne

À chaque étape, on prend le **meilleur choix immédiat**, sans revenir en arrière.

Rendre la monnaie avec 50, 20, 10, 5, 2, 1 : toujours la plus grosse pièce possible. 63 = 50 + 10 + 2 + 1. C'est optimal.

Mais avec les pièces 1, 3, 4 et le montant 6 : le glouton prend 4 + 1 + 1 (3 pièces), alors que 3 + 3 (2 pièces) est meilleur. **Le glouton n'est pas toujours optimal.**

> ⚠ *Une stratégie gloutonne doit être **justifiée** (preuve ou contre-exemple). Elle est tentante parce qu'elle est simple.*

Le concept

Le cas qui marche : choisir des activités

Pour caser le plus d'activités dans une salle : **trier par heure de fin** et prendre l'activité qui finit le plus tôt, puis la suivante compatible. C'est un résultat classique (prouvé).

Diviser pour régner

1. **Diviser** le problème en sous-problèmes plus petits.
2. **Régner** : résoudre chaque sous-problème (souvent par récursion).
3. **Combiner** les résultats.

La recherche binaire est un cas simple (on garde une moitié). Le **tri fusion** en est un cas célèbre :

Le concept

Diviser pour régner (suite)

```
def tri_fusion(liste):
    if len(liste) <= 1:
        return list(liste)
    milieu = len(liste) // 2
    gauche = tri_fusion(liste[:milieu])
    droite = tri_fusion(liste[milieu:])
    return fusionner(gauche, droite)

def fusionner(a, b):
    resultat = []
    i = j = 0
    while i < len(a) and j < len(b):
        if a[i] <= b[j]:
            resultat.append(a[i]); i += 1
        else:
            resultat.append(b[j]); j += 1
    return resultat + a[i:] + b[j:]
```

Le concept

Diviser pour régner (suite)

Le tri fusion est bien plus rapide que les tris simples sur de grandes listes : de l'ordre de n × log n opérations.

Exemple pas à pas

Exemple pas à pas

Choix d'activités : `[(1, 4), (3, 5), (0, 6), (5, 7), (3, 9), (5, 9), (6, 10), (8, 11)]` (début, fin).
1. Trier par heure de fin : (1,4), (3,5), (0,6), (5,7), (3,9), (5,9), (6,10), (8,11).

2. Prendre la première, puis chaque activité qui commence après la fin de la dernière choisie :

Activité	Début ≥ dernière fin ?	Choisie ?
(1,4)	oui (rien avant)	oui, fin = 4
(3,5)	3 < 4	non
(0,6)	0 < 4	non
(5,7)	5 ≥ 4	oui, fin = 7
(3,9)	non	non
(5,9)	5 < 7	non

Exemple pas à pas

Exemple pas à pas (suite)

Activité	Début ≥ dernière fin ?	Choisie ?
(6,10)	6 < 7	non
(8,11)	8 ≥ 7	oui, fin = 11

```
def choisir_activites(activites):
    choisies = []
    derniere_fin = None
    for debut, fin in sorted(activites, key=lambda a: a[1]):
        if derniere_fin is None or debut >= derniere_fin:
            choisies.append((debut, fin))
            derniere_fin = fin
    return choisies

print(choisir_activites([(1,4),(3,5),(0,6),(5,7),(3,9),(5,9),(6,10),(8,11)]))
# [(1, 4), (5, 7), (8, 11)]
```

À toi, avec aide

Reprends la monnaie. Écris `rendre_monnaie_glouton(montant, pieces)` qui prend toujours la plus grande pièce possible. Teste avec les pièces `[50, 20, 10, 5, 2, 1]` pour 63 (résultat : `[50, 10, 2, 1]`), puis avec `[4, 3, 1]` pour 6.
Que constates-tu avec `[4, 3, 1]` ?

À toi, avec aide · réponse

Indices progressifs

Indice 1

Trie les pièces de la plus grande à la plus petite.

Indice 2

Tant que le montant restant est >= la pièce, prends-la.

Indice 3

Avec `[4, 3, 1]`, compare ton résultat à 3 + 3.

À toi, avec aide · réponse

Réponse

```
def rendre_monnaie_glouton(montant, pieces):
    rendu = []
    for p in sorted(pieces, reverse=True):
        while montant >= p:
            rendu.append(p)
            montant -= p
    return rendu

print(rendre_monnaie_glouton(63, [50, 20, 10, 5, 2, 1]))  # [50, 10, 2, 1]
print(rendre_monnaie_glouton(6, [4, 3, 1]))               # [4, 1, 1]
```

Avec `[4, 3, 1]`, le glouton donne 3 pièces alors que `[3, 3]` en utilise 2 : **contre-exemple**, le glouton n'est pas optimal ici.

À toi, en solo

Écris `maximum_diviser(liste)` qui trouve le plus grand élément par **diviser pour régner** : on coupe en deux, on trouve le maximum de chaque moitié, et on garde le plus grand. Liste non vide.
Teste avec `[3, 9, 2, 7, 5]` → 9. Dessine l'arbre des appels pour `[3, 9, 2, 7]`.

À toi, en solo · réponse

Pistes de correction

```
def maximum_diviser(liste):
    if len(liste) == 1:
        return liste[0]
    milieu = len(liste) // 2
    a = maximum_diviser(liste[:milieu])
    b = maximum_diviser(liste[milieu:])
    return a if a >= b else b

print(maximum_diviser([3, 9, 2, 7, 5]))  # 9
```

Arbre pour `[3, 9, 2, 7]` : on coupe en `[3, 9]` et `[2, 7]`. `[3, 9]` donne 9, `[2, 7]` donne 7, et le résultat final est 9.

Pièges classiques

Piège	Pourquoi c'est un problème	Comment l'éviter
Croire que le glouton est toujours optimal	Solution correcte-looking mais pas la meilleure	Chercher un contre-exemple
Mauvais critère de tri	Trier par début au lieu de fin ne marche pas	Justifier le critère
Pas de cas de base en diviser pour régner	Récursion infinie	Cas de base : taille 0 ou 1
Combinaison oubliée ou fausse	Le résultat de chaque moitié est correct, pas l'ensemble	Tester la fusion séparément

Mini-boss

Mini-boss

Un atelier a 7 présentations `[(9,10), (9,12), (10,11), (11,13), (12,14), (13,15), (14,16)]`.

1. Utilise `choisir_activites` pour trouver le maximum.
2. Essaie un autre critère glouton (trier par durée la plus courte) et trouve un cas où il donne un moins bon résultat que le tri par fin.
3. Écris `tri_fusion` et vérifie-le sur 5 listes (vide, un élément, trié, inversé, avec doublons).

Mini-boss · réponse

Pistes de correction

Résultat du critère « fin la plus tôt » : `[(9,10), (10,11), (11,13), (13,15)]` (4 présentations).

Contre-exemple au critère « plus courte durée » : `[(1,5), (4,6), (5,10)]`. La plus courte, (4,6), chevauche les deux autres, alors que (1,5) + (5,10) en caserait 2. Pour tester `tri_fusion` : compare le résultat à `sorted(...)` sur chacune des listes.

Quiz éclair

Quiz éclair

1. Une stratégie gloutonne :

A. prend le meilleur choix immédiat
B. essaie toutes les combinaisons
C. revient toujours en arrière

2. Le glouton :

A. est toujours optimal
B. n'est pas toujours optimal
C. est toujours lent

3. Diviser pour régner suit l'ordre :

A. combiner, diviser, résoudre

Quiz éclair

Quiz éclair (suite)

B. résoudre, oublier, recommencer
C. diviser, résoudre, combiner

4. 🔁 *Révision mission 06* : Un tri stable :

A. garde l'ordre des éléments égaux
B. est toujours rapide
C. modifie la liste

5. 🔁 *Révision mission 05* : La recherche binaire exige :

A. une liste courte
B. une liste triée
C. des textes

Quiz éclair · réponse

Réponses

1. **A**.
2. **B**.
3. **C**.
4. **A**.
5. **B**.

Je réfléchis

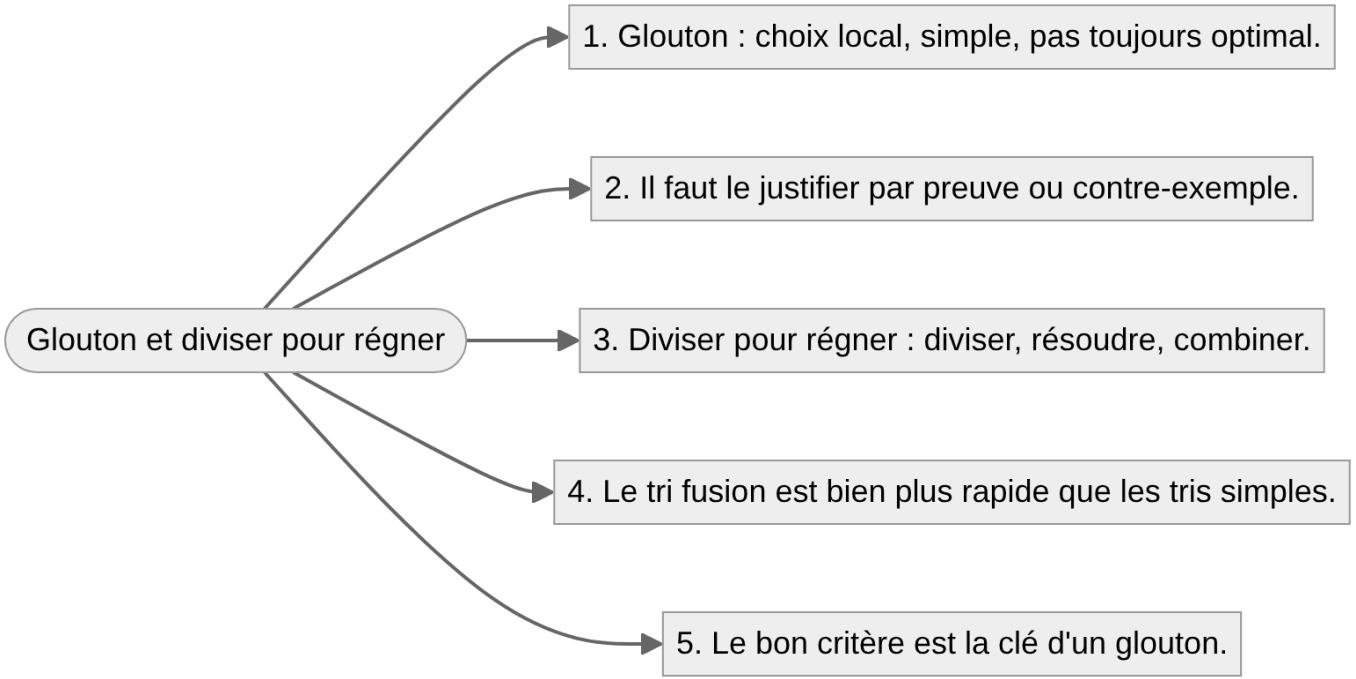
1. Dans quelle situation de ta vie as-tu pris « le meilleur choix immédiat » et l'as-tu regretté ensuite ?
2. Quel gros projet pourrais-tu couper en deux moitiés indépendantes ?

À retenir

1. Glouton : choix local, simple, pas toujours optimal.
2. Il faut le justifier par preuve ou contre-exemple.
3. Diviser pour régner : diviser, résoudre, combiner.
4. Le tri fusion est bien plus rapide que les tris simples.
5. Le bon critère est la clé d'un glouton.

À retenir

Le schéma de la mission



MISSION ACCOMPLIE

+100 XP

Badge débloqué : Stratège

Prochaine mission : Correction, tests et cas limites

