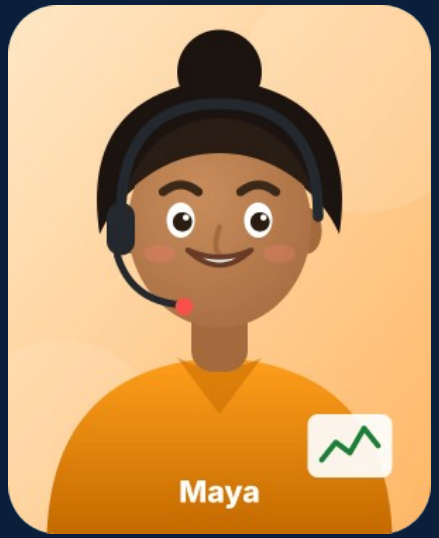
HEF-010 · MISSION 10 / 10

Face à un problème inédit

Le vrai test : un problème que tu n'as jamais vu, et la méthode pour s'en sortir.





Badge : Algorithmicien·ne

80 min · débutant

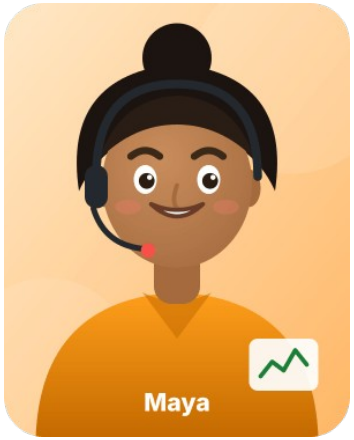
L'histoire

Dernier jour du module. Maya donne à Kodi un problème tout nouveau :

> *« Pour chaque membre, trouve la plus longue série de jours consécutifs où il a terminé au moins une mission. »*

Il n'y a pas de fonction toute faite. Kodi respire, regarde Sam, et décide d'utiliser sa méthode.

Cette mission te donne une méthode pour attaquer n'importe quel problème algorithmique.



Ta mission

À la fin, je serai capable de…

1. appliquer une méthode en 6 étapes à un problème inédit
2. choisir les bonnes briques (boucle, tri, ensemble, récursion…)
3. justifier mon choix par un compromis
4. tester ma solution sur des cas limites
5. présenter ma solution clairement

Défi d'échauffement

Défi d'échauffement

Voici un problème inconnu : *« Dans une liste de nombres, trouve deux nombres dont la somme vaut 10. »*

Sans coder, écris **trois idées** de solution, même naïves. Laquelle essaierais-tu en premier ? Pourquoi ?

Le concept

La méthode en 6 étapes



Description du schéma : six étapes de gauche à droite : comprendre le problème, trouver des exemples, trouver une idée simple, l'améliorer, coder et tester, puis comparer et expliquer.

1. **Comprendre** : entrée, sortie, règles, cas limites (mission 01).
2. **Exemples** : un petit cas, un cas moyen, un cas limite. Calcule la réponse à la main.
3. **Idée simple** : la solution « brute », même lente. Elle sert de référence.
4. **Améliorer** : cherche ce qui est répété ou inutile. Une structure aide-t-elle (tri, ensemble) ? Peut-on diviser ?
5. **Coder et tester** : contrat, code, tests, comparaison à la solution brute.

Le concept

La méthode en 6 étapes (suite)

6. **Comparer et expliquer** : temps, mémoire, clarté ; écris pourquoi tu as choisi.

Reconnaître le genre de problème

Indice dans l'énoncé	Idée à essayer
« le plus grand / petit »	parcours avec une variable
« existe-t-il », « déjà vu »	ensemble
« dans l'ordre », « le k-ième »	tri
structure imbriquée	récursion
« le plus possible » avec un critère simple	glouton (à justifier)
« partie de la liste », « moitié »	diviser pour régner

Le concept

Reconnaître le genre de problème (suite)

🧠 ***Astuce*** *: la solution brute d'abord. Elle donne la bonne réponse, sert de test, et révèle ce qui est lent.*

Exemple pas à pas

Problème de l'échauffement : trouver deux nombres dont la somme vaut 10.

1. Comprendre : entrée = liste d'entiers, sortie = un couple ou `None`. Cas limites : liste vide, un élément, deux fois le même nombre (5 + 5 avec un seul 5 ?). Décision : deux **positions différentes**.
2. Exemples : `[3, 7, 1]` → (3, 7) ; `[5]` → None ; `[5, 5]` → (5, 5).

3. Idée simple : toutes les paires.

```
def deux_somme_brute(liste, cible):
    for i in range(len(liste)):
        for j in range(i + 1, len(liste)):
            if liste[i] + liste[j] == cible:
                return (liste[i], liste[j])
    return None
```

Exemple pas à pas

Exemple pas à pas (suite)

4. Améliorer : pour chaque x, on cherche `cible - x`. Un ensemble répond vite.

```
def deux_somme_brute(liste, cible):
    for i in range(len(liste)):
        for j in range(i + 1, len(liste)):
            if liste[i] + liste[j] == cible:
                return (liste[i], liste[j])
    return None

def deux_somme(liste, cible):
    vus = set()
    for x in liste:
        if cible - x in vus:
            return (cible - x, x)
        vus.add(x)
    return None

for l in ([3, 7, 1], [5], [5, 5], []):
```

Exemple pas à pas (suite)

```
    a, b = deux_somme(l, 10), deux_somme_brute(l, 10)
    assert (a is None) == (b is None)
    print(l, a)
```

6. Comparer : brute ≈ n × n, ensemble ≈ n, au prix d'un ensemble en mémoire.

À toi, avec aide

Problème : *« Une liste de jours (nombres entiers) où un membre a terminé une mission, triée, sans doublon. Trouve la plus longue série de jours consécutifs. »* Exemple : `[1, 2, 3, 7, 8, 10]` → 3 (jours 1, 2, 3).
Suis la méthode : exemples, idée, code, tests.

À toi, avec aide · réponse

Indices progressifs

Indice 1

Garde la longueur de la série en cours et la meilleure longueur.

Indice 2

Si `jours[i] == jours[i-1] + 1`, la série continue ; sinon, elle redémarre à 1.

Indice 3

Gère la liste vide : réponse 0.

Réponse

```
def plus_longue_serie(jours):
    if not jours:
        return 0
    meilleure = courante = 1
    for i in range(1, len(jours)):
        if jours[i] == jours[i - 1] + 1:
            courante += 1
        else:
            courante = 1
        meilleure = max(meilleure, courante)
    return meilleure

assert plus_longue_serie([]) == 0
assert plus_longue_serie([5]) == 1
assert plus_longue_serie([1, 2, 3, 7, 8, 10]) == 3
assert plus_longue_serie([1, 3, 5]) == 1
print("ok")
```

À toi, en solo

Même problème, mais **sans** hypothèse : la liste n'est pas triée et peut contenir des doublons. `[10, 4, 3, 5, 3, 11, 12, 13]` → 4 (3, 4, 5 → 3 ; 10, 11, 12, 13 → 4).

Applique les 6 étapes : solution brute, solution améliorée, tests aléatoires comparant les deux, et un commentaire sur le coût.

Pistes de correction

```
import random

def serie_brute(l):
    s = set(l)
    meilleur = 0
    for x in s:
        n = 1
        while x + n in s:
            n += 1
        meilleur = max(meilleur, n)
    return meilleur

def serie(l):
    s = set(l)
    meilleur = 0
    for x in s:
        if x - 1 not in s:          # début d'une série
            n = 1
            while x + n in s:
```

Pistes de correction (suite)

```
                n += 1
            meilleur = max(meilleur, n)
    return meilleur

assert serie([]) == 0
assert serie([10, 4, 3, 5, 3, 11, 12, 13]) == 4
for _ in range(500):
    l = [random.randint(0, 15) for _ in range(random.randint(0, 12))]
    assert serie(l) == serie_brute(l)
print("ok")
```

La version améliorée ne démarre une série qu'à son premier élément : chaque nombre n'est parcouru qu'une fois, soit de l'ordre de n opérations.

Pièges classiques

Piège	Pourquoi c'est un problème	Comment l'éviter
Coder dès la lecture	Mauvais problème résolu	Commencer par les exemples
Chercher l'optimal tout de suite	On se bloque	Écrire la solution brute d'abord
Pas de référence pour tester	Aucune confiance	Comparer à la brute sur des entrées aléatoires
Ne pas expliquer son choix	Une décision sans justification est fragile	Écrire critères et compromis

Mini-boss

Mini-boss

Mini-projet du module. Choisis **un** problème ci-dessous et applique les 6 étapes. Rends un dossier d'une page + le code + les tests :

A. Trouver les deux membres les plus proches en nombre de missions terminées.

B. Regrouper des mots qui sont des anagrammes les uns des autres.

C. Vérifier si des parenthèses `()[]{}` sont bien équilibrées dans un texte.

Le dossier contient : énoncé précisé, cas limites, solution brute, amélioration, tests, mesures, conclusion.

Mini-boss · réponse

Pistes de correction

Pistes : (A) trier puis comparer les voisins ; (B) clé = lettres triées du mot, dictionnaire de listes ; (C) une pile (liste avec append/pop) : on empile les ouvrantes, on dépile en vérifiant la correspondance, la pile doit être vide à la fin. Un bon dossier montre une solution brute, une améliorée, la preuve qu'elles donnent les mêmes résultats, et un verdict argumenté.

Quiz éclair

Quiz éclair

1. La première étape de la méthode est :

A. comprendre le problème
B. coder
C. optimiser

2. La solution brute sert à :

A. être livrée telle quelle
B. servir de référence pour tester
C. remplacer les tests

3. « existe-t-il déjà », « déjà vu » suggère souvent :

A. une récursion

Quiz éclair

Quiz éclair (suite)

B. un tri
C. un ensemble

4. 🔁 *Révision mission 09* : Un compromis, c'est :

A. gagner en temps en payant en mémoire, par exemple
B. un bug
C. une erreur de syntaxe

5. 🔁 *Révision mission 08* : Un test montre :

A. qu'il n'y a plus de bug
B. qu'il y a un bug s'il échoue
C. la vitesse

Quiz éclair · réponse

Réponses

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

Je réfléchis

1. Quelle étape de la méthode as-tu le plus tendance à sauter ?
2. Quel problème de ta vie pourrais-tu aborder avec ces 6 étapes ?

À retenir

À retenir

1. Comprendre, exemples, idée simple, améliorer, coder + tester, comparer.
2. La solution brute est une référence précieuse.
3. Reconnaître le genre de problème oriente le choix de la brique.
4. Les compromis se justifient avec des mesures.
5. Un problème inédit n'est plus effrayant avec une méthode.

Félicitations

Félicitations

Tu as terminé **HEF-010 — Algorithmes**. Tu sais décrire un problème précisément, écrire et vérifier une solution, chercher, trier, récurser, comparer des approches, et attaquer un problème inédit avec une méthode.
→ Prochain module : **HEF-020 — Structures de données**

MISSION ACCOMPLIE

+120 XP

Badge débloqué : Algorithmicien·ne

Fin du module HEF-010. Bravo !

