

# Intro aux compétitions d'optimisation

Approximer pour gagner

A dark blue diagonal gradient bar that starts from the bottom left and extends towards the top right, covering the lower half of the slide.

# About me

Clément HAMMEL

 [clemhammel.bsky.social](https://clemhammel.bsky.social)

Challenge expert @ Isograd

Cofondateur @ h25

## Compétitions techniques

- CodinGame
- BattleDev
- Meilleur Dev de France

# Menu du jour

Optimisation ?

Stratégies

Le challenge !

Conseils

# Optimisation ?

- Complexité
- Temps de calcul
- TSP
- Optimisation

# Complexité - Intro

Relation

Taille de l'entrée  $\leftrightarrow$  Temps d'exécution

$N$  éléments  $\leftrightarrow f(N)$  opérations

# Complexité - Exemples



```
ma_liste = [1, 4, 2, 8, 1, 7, ...]  
somme = 0  
  
for i in ma_liste:  
    somme += i  
  
print(somme)
```

Taille de liste : N

Nb opérations : N => Complexité N

# Complexité - Exemples

```
liste = [2, 4, 1, 1, 3, ...]
doublons = []

for i in range(len(liste)):
    for j in range(i + 1, len(liste)):
        if liste[i] == liste[j]:
            doublons += 1
```

Opérations :  $N + (N-1) + (N-2) + \dots = (N^2 + N) / 2 \sim N^2$

Complexité  $N^2$

# Complexité - Exemples




Nb de chiffres :  $N$

Nb de combinaisons :  $10^N$

Complexité bruteforce :  $10^N$



# Complexité - Temps de calcul

		N = 2	N = 20	N = 10 000	N = 10 milliards
Somme	N	2	20	10 000	$10^{10}$
Doublons	$N^2$	4	400	$10^{10}$	$10^{20}$
Cadenas	$10^N$	100	10000000000 0000000000	$10^{10000}$	

# Complexité - Temps de calcul

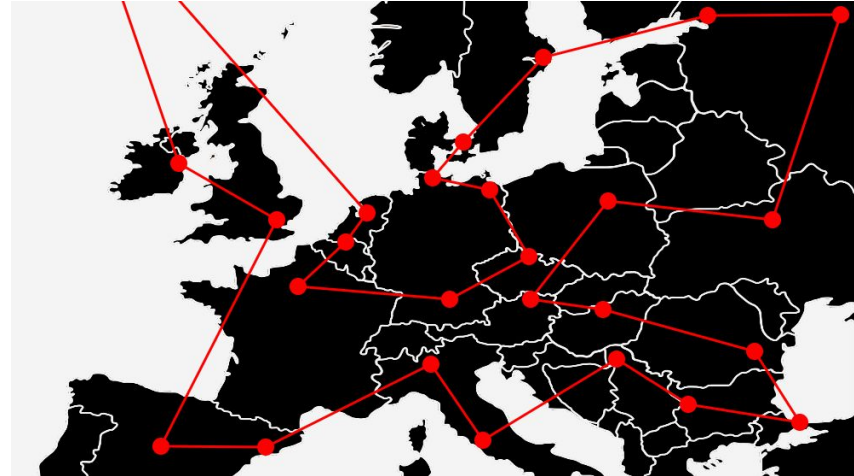
		N = 2	N = 20	N = 10 000	N = 10 milliards
Somme	N	2	20	10 000	$10^{10}$
Doublons	$N^2$	4	400	$10^{10}$	$10^{20}$
Cadenas	$10^N$	100	10000000000 0000000000	$10^{10000}$	🤯

Plus de  $10^{10}$  opérations = 🤯

# Voyageur de commerce (TSP)

Travelling salesman problem (TSP)

Objectif : trouver le trajet le plus court passant par un ensemble de  $N$  villes



# Voyageur de commerce (TSP)

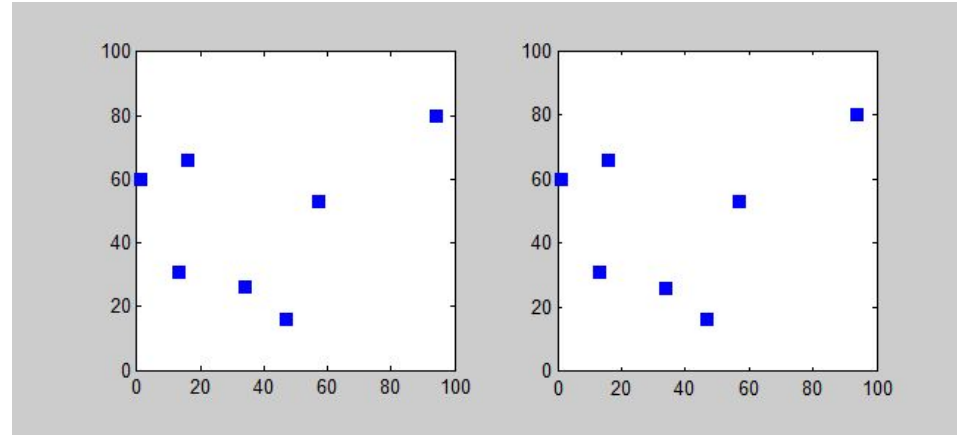
Solution exacte ?

Tester tous les chemins

- Ville 1 : N choix
- Ville 2 : (N-1) choix...

$$N \times (N-1) \times (N-2) \times \dots \times 1 = N!$$

$$14! \approx 9 \times 10^{10}$$



# Voyageur de commerce (TSP)

Meilleure solution exacte :  $2^N \cdot N^2$

Réalisable si  $N < 50$

Une meilleure solution existe ?

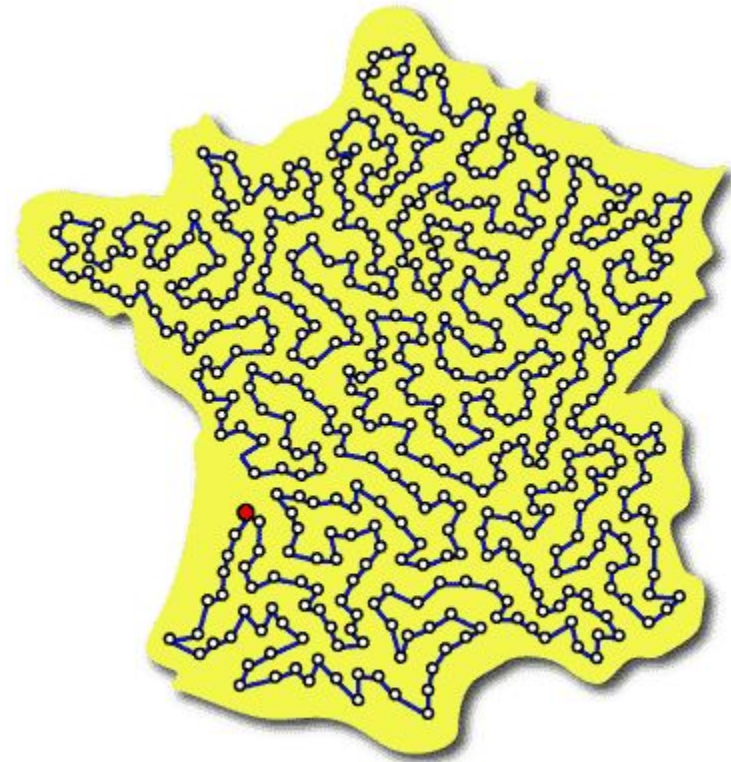
1 000 000\$ si vous trouvez ! ( $P \stackrel{?}{=} NP$ )

# Optimisation

Problème impossible ? Non !

Nouvel objectif :

Trouver une bonne solution,  
pas forcément la meilleure



# Optimisation

## “Facile”

Max d'une liste

Trier un tableau

Chemin le plus court

Pattern matching

## “Difficile”

Voyageur de commerce

Tetris

Sudoku NxN

Set cover

# Optimisation

## “Facile”

Montant d'une facture

Recherche de produit

Fusion de 2 listes

API classique de BDD

## “Difficile”

Tournée des facteurs

Placement éoliennes

Plannings

Découpe de matériaux



# Stratégies

6 niveaux de solution :

1. Minimale
2. Monte Carlo
3. Greedy
4. Greedy aléatoire
5. Customisée
6. Artillerie lourde

Puis mise en pratique !

# 1. Minimale

Stratégie la plus simple, mais la moins efficace

On cherche n'importe quelle solution valide

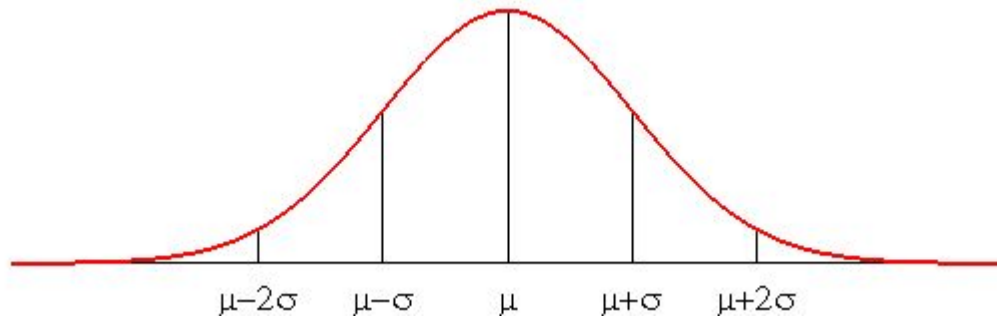
But : vérifier que le reste du programme marche, et apparaître sur le scoreboard

Exemple TSP : On parcourt les  $N$  points dans l'ordre donné. Valide mais trajet très long !

## 2. Monte Carlo

Dépend beaucoup de la puissance de calcul

On génère plein de solutions valides au hasard, et on garde la meilleure



### 3. Greedy

On construit une solution progressivement en choisissant à chaque fois la meilleure option  
heuristique !

Exemple TSP : On se dirige vers la ville non visitée la plus proche

Produit de très bonnes solutions dans certains cas

## 4. Greedy aléatoire

Combinaison greedy + Monte Carlo, on ajoute une part de random pour construire la solution

Exemples TSP :

- Choix de la ville de départ
- Aller vers une des 3 villes les plus proches

## 5. Customisée

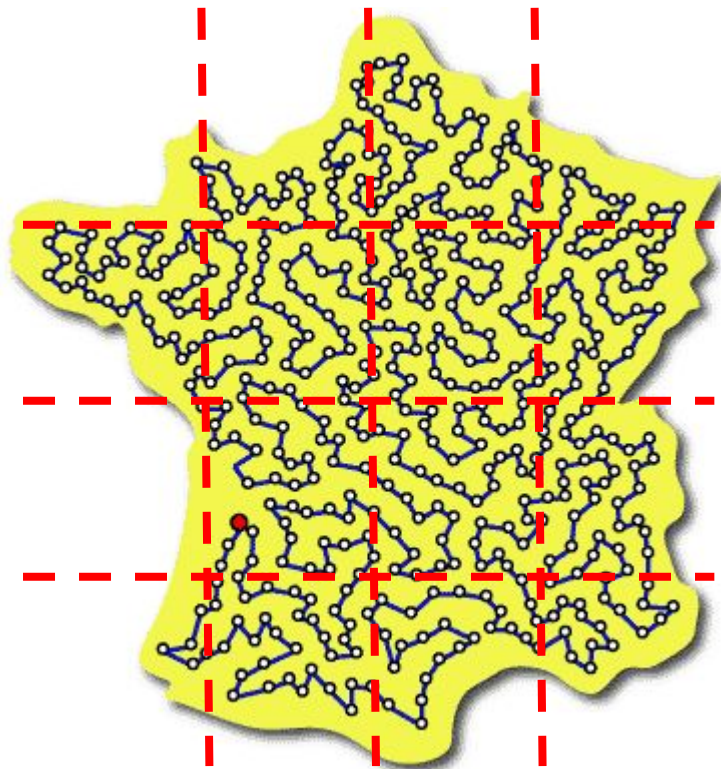
Une solution “intelligente”, tout dépend du problème posé.

Peut dépendre du dataset !

## 5. Customisée

Exemple TSP :

- Diviser les villes en petits groupes locaux
- TSP sur les groupes
- Recoller les morceaux



## 6. Artillerie lourde

Des solveurs existent pour les problèmes classiques (TSP, set cover...)

Utile dans la vraie vie, mais très dur à mettre en place

Déconseillé pour le challenge



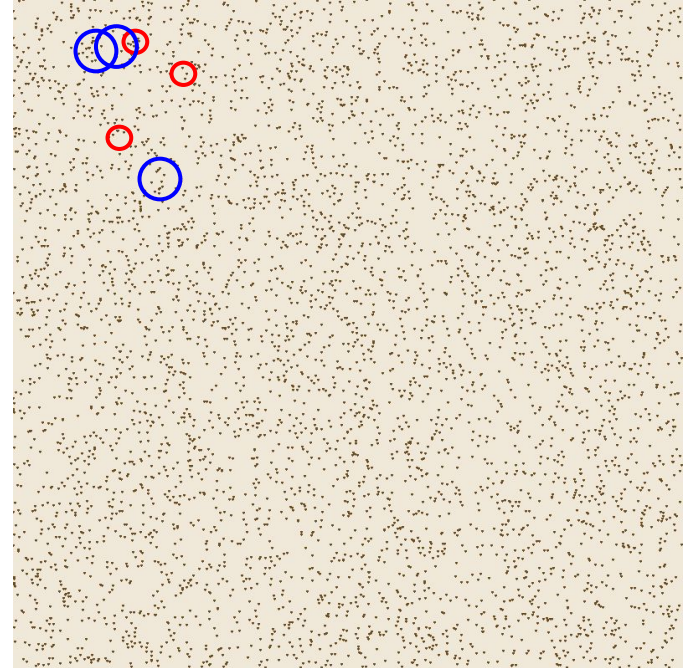
# Mise en pratique

5000 objets à surveiller

Caméras de surveillance

- Rayon 4, prix 1
- Rayon 8, prix 2

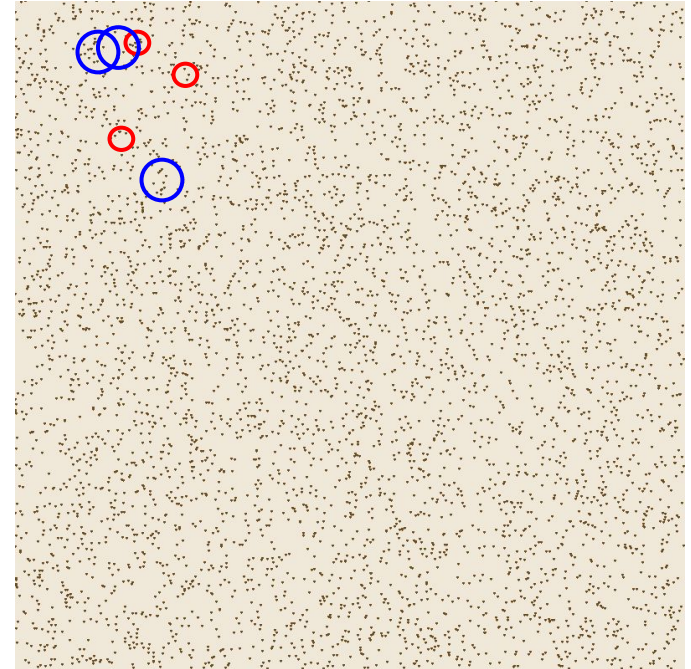
Surveiller tous les objets avec le coût le plus bas possible



# Mise en pratique

Trouver une stratégie

- Minimale
- Monte Carlo
- Greedy



# Le challenge !

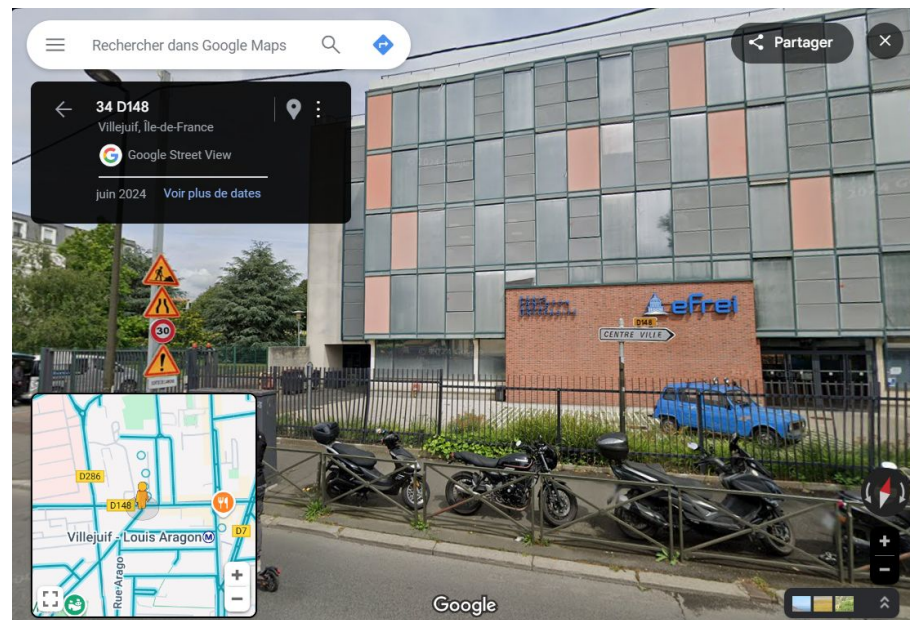
Présentation

Premier dataset

Score

Starter kit

# Challenge - Présentation



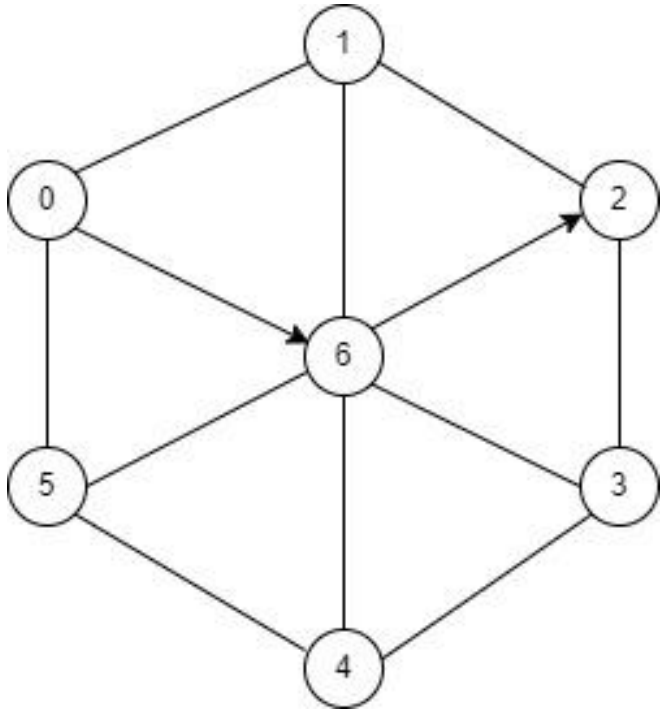
# Challenge - Présentation

Comment passer dans toutes les rues d'une ville ?

Contraintes :

- Batterie limitée, la voiture doit repasser régulièrement se charger
- Sens uniques, mais pas de culs-de-sac

# Challenge - Dataset 1



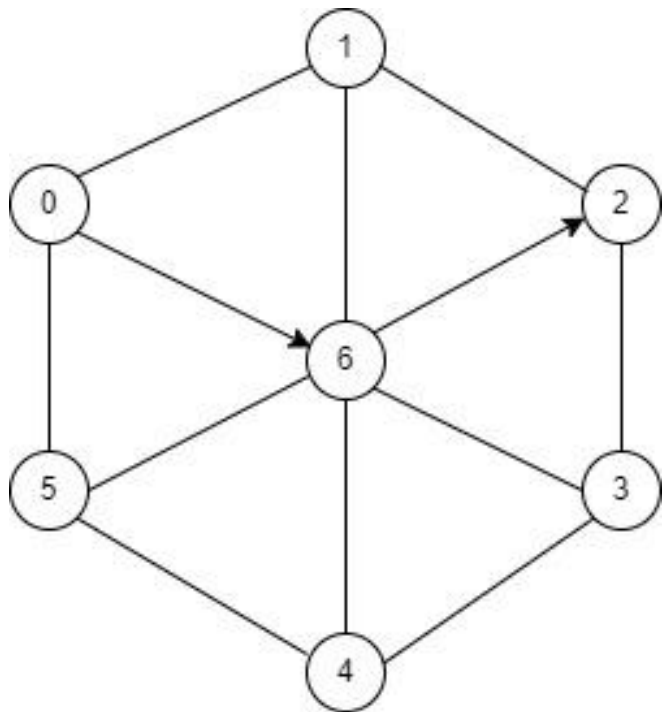
Distance entre chaque ville : 10

Batterie max : 70

2 jours max

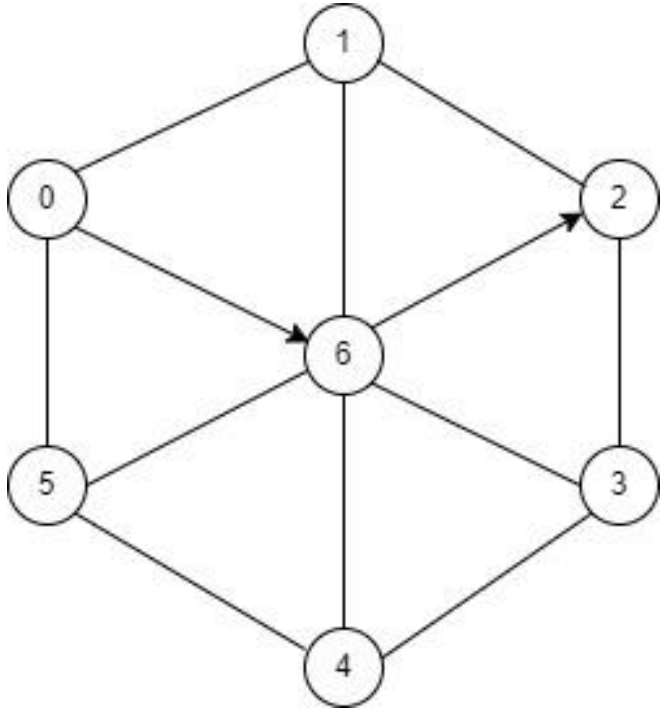
2 rues à sens unique

# Challenge - Dataset 1



```
1 {
2   "comment": "Exemple de réseau routier",
3   "batteryCapacity": 70,
4   "numDays": 2,
5   "intersections": [
6     {
7       "id": 0,
8       "lat": 5.0,
9       "lng": -8.66
10    },
11    {
12      "id": 1,
13      "lat": 10.0,
14      "lng": 0.0
15    },
16    ...
17  ],
18  "roads": [
19    {
20      "intersectionId1": 0,
21      "intersectionId2": 1,
22      "isOneWay": false,
23      "length": 10
24    },
25    {
26      "intersectionId1": 0,
27      "intersectionId2": 6,
28      "isOneWay": true,
29      "length": 10
30    },
31    ...
32  ]
33 }
```

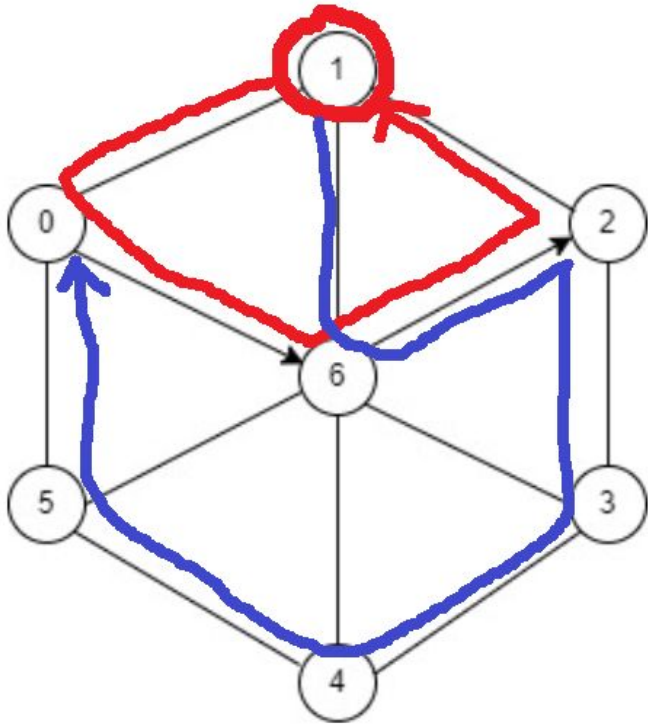
# Challenge - Dataset 1



```
1 {  
2   "chargeStationId": 1,  
3   "itinerary": [1, 0, 6, 2, 1, 6, 2, 3, 4, 5, 0]  
4 }
```



# Challenge - Dataset 1



```
1 {  
2   "chargeStationId": 1,  
3   "itinerary": [1, 0, 6, 2, 1, 6, 2, 3, 4, 5, 0]  
4 }
```

# Challenge - Score

Score : Distance parcourue + bonus si toute la ville est explorée

Classement : 1 000 000 pour la meilleure équipe, et score proportionnel pour les autres équipes

Chaque dataset ("ville") peut rapporter 1M, travaillez les tous !

# Challenge - Starter kit

7 datasets, de difficulté croissante

L'énoncé complet

La fonction de score -> à intégrer à votre code

Un exemple de soumission

Starter kit en Python

# Conseils



# Méthodologie

- Premier dataset à la main
- Automatiser une stratégie minimale
- Trouver une meilleure idée
- Implémenter la nouvelle idée



Très rapide

# Trouver une meilleure idée

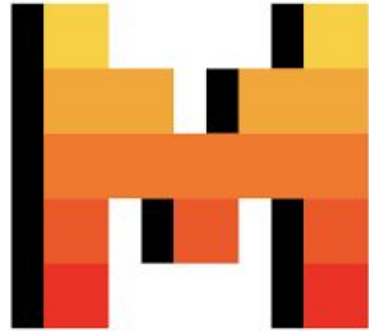
Visualisations ! 1 personne à plein temps au début

Discutez, dessinez

Approche globale ou approche par dataset

Pas de solutions trop complexes (trop vite) !

# Implémenter l'idée



# Organisation

Module n°11 - Concours d'optimisation					
	Lundi 16 décembre	Mardi 17 décembre	Mercredi 18 décembre	Jeudi 19 décembre	Vendredi 20 décembre
<b>Matin</b> 9h30-13h00	Présentation du challenge + création des équipes + coaching <b>Amphi A001</b>	Travail sur plate-forme	Travail sur plate-forme	Travail sur plate-forme	Création des slides à présenter
<b>Après-midi</b> 14h00-17h30	Travail sur plate-forme <b>Amphi A001 réservé si besoin</b>	Travail sur plate-forme	Travail sur plate-forme	Travail sur plate-forme	Présentation + Annonce du gagnant + remise de prix

Temps en présentiel	Temps en visio
---------------------	----------------



# Dernières choses

Teams

[support@isograd.com](mailto:support@isograd.com) -> Si pb de plateforme

Pas d'aide sur votre code !

Présentation à la fin

Merci !

# Questions ?

Mise en pratique :

Sujet d'entraînement Hash Code 2022