

# 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

Mathis HAMMEL

[@MathisHammel](#)

CTO @ Agoratlas

Expert en programmation compétitive

Créateur de challenges

CodinGame, MDF, CTF...

Social data analyst

# 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 :  $N \Rightarrow$  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 = 🤯

# Complexité - Temps de calcul

		Ordi actuel	10x plus puissant	1000x plus puissant
Somme	$N$	X	10 X	1000 X
Doublons	$N^2$	X	3.1 X	31 X
Cadenas	$10^N$	X	X + 1	X + 3

Il n'y a qu'à attendre de meilleures machines ?

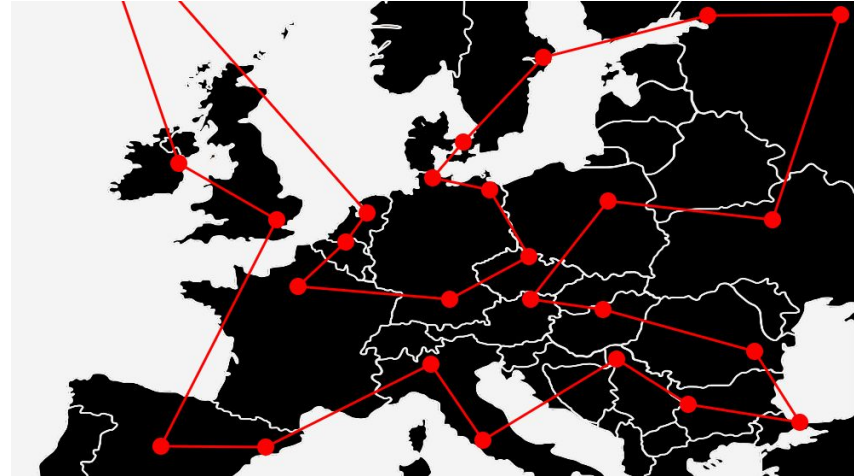
# Complexité - Temps de calcul

Complexité	1	$\log(N)$	$N^2$	$N^3$	$2^N$	$N!$
Volume max	$\infty$	" $\infty$ "	100 000	3 000	34	14

# 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)

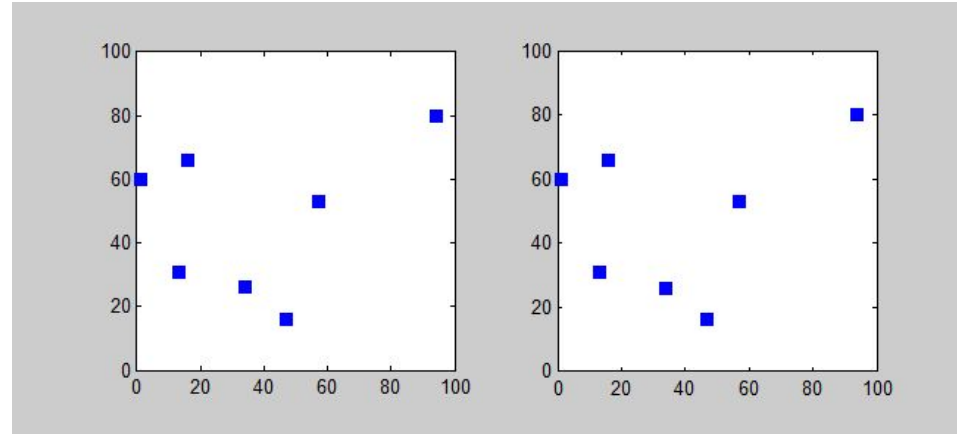
Solution exacte ?

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 ?

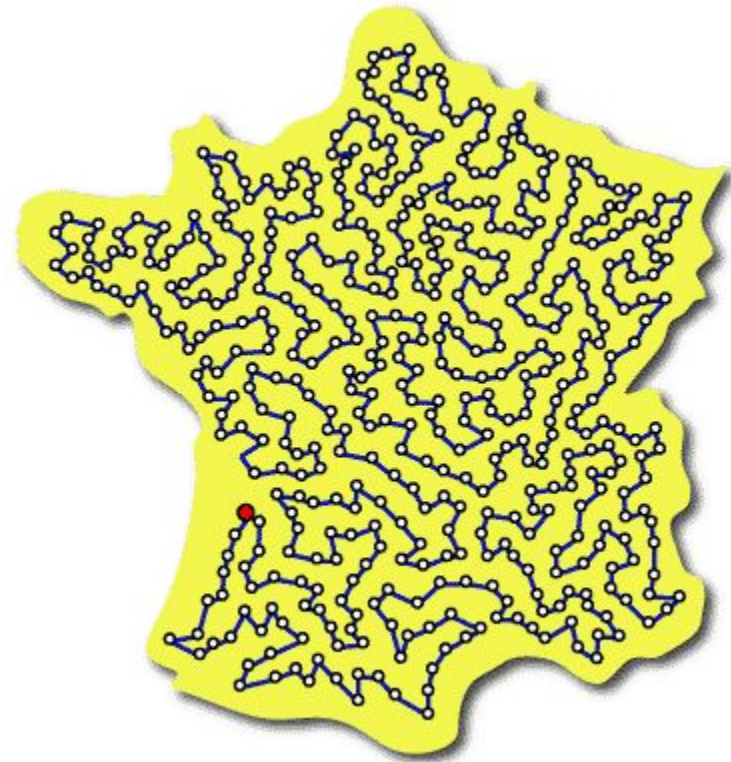
1 000 000\$ si vous trouvez ! ( $P \stackrel{?}{=} NP$ )

# Optimisation

Problème impossible ? Non !

Nouvel objectif :

Trouver une bonne solution,  
pas forcément la meilleure





# Familles de problèmes

## Problèmes de tournées

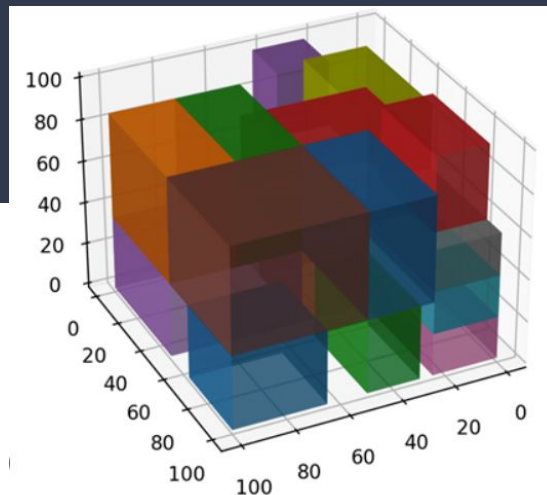
- Transports
- Supply chain
- Livraisons



# Familles de problèmes

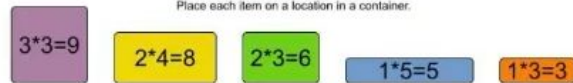
## Bin packing

- Remplissage de conteneur
- Découpe (tissu, pare brises...)

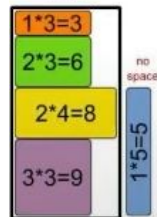


Bin packing

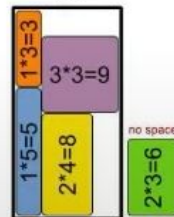
Place each item on a location in a container.



Largest size first



Largest side first



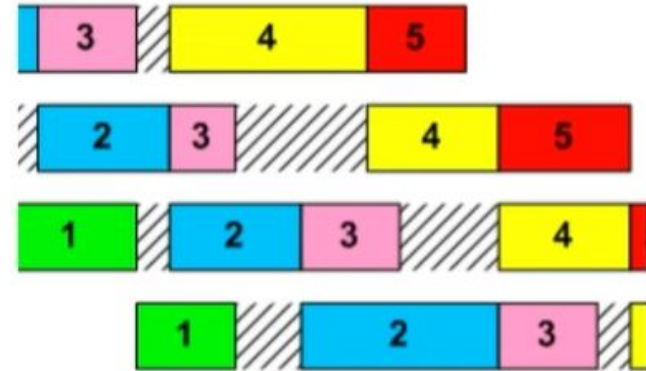
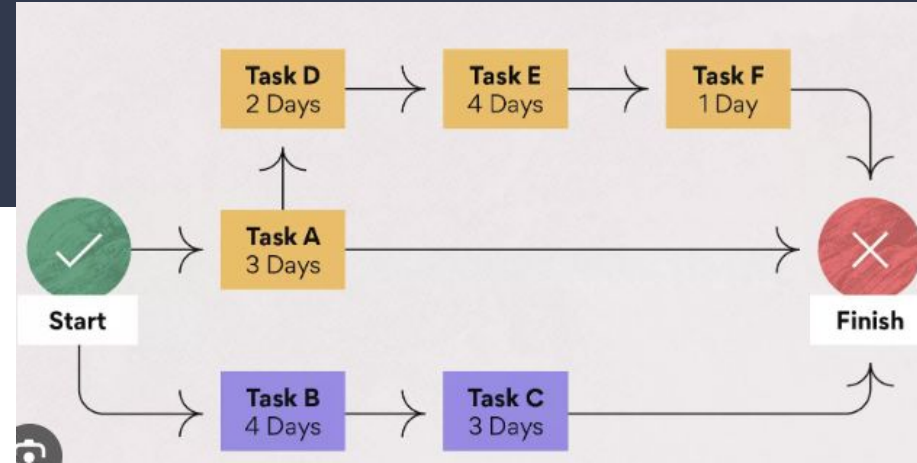
Drools Planner



# Familles de problèmes

## Ordonnancement

- Planning
- Usinage
- Noyau OS



# Familles de problèmes

Tout n'est pas un problème d'optim !

Montant d'une facture

Recherche de produit

API classique de BDD

# 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

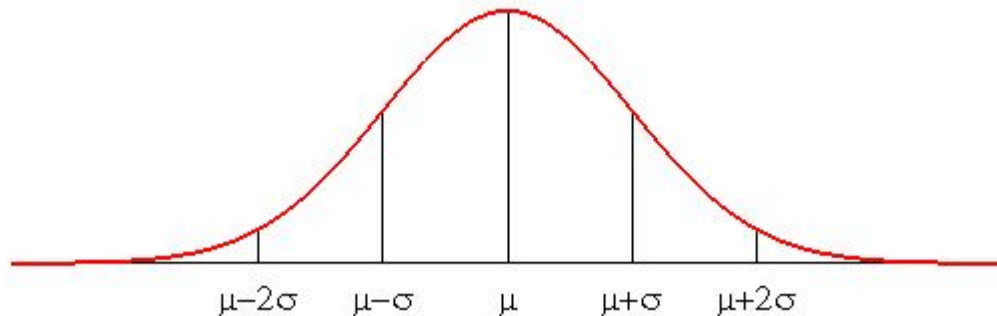
But : vérifier que le reste du programme marche, et apparaître sur le scoreboard

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

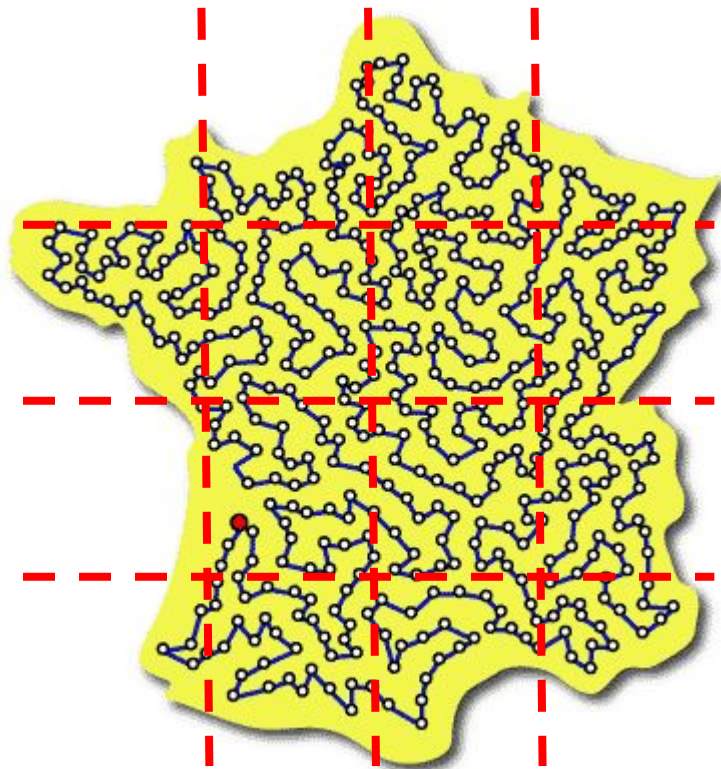
Une solution “intelligente”, tout dépend du problème posé.

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

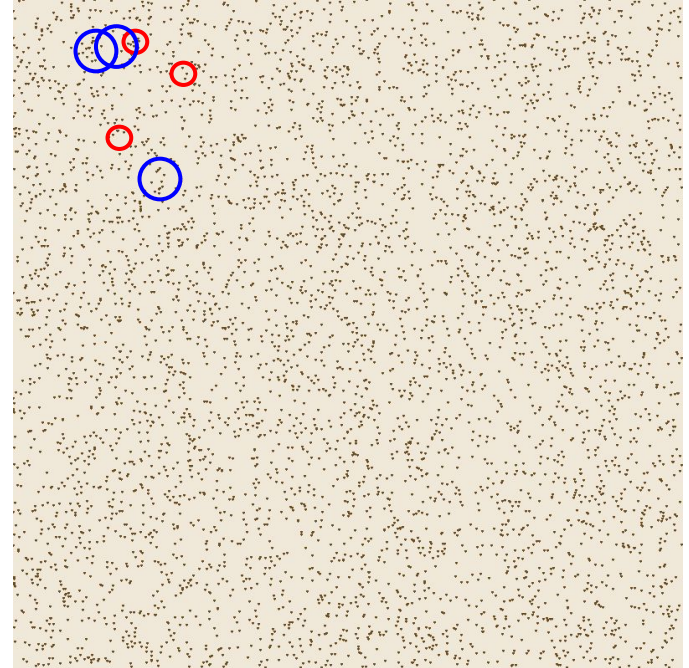
# Mise en pratique

5000 objets à surveiller

Caméras de surveillance

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

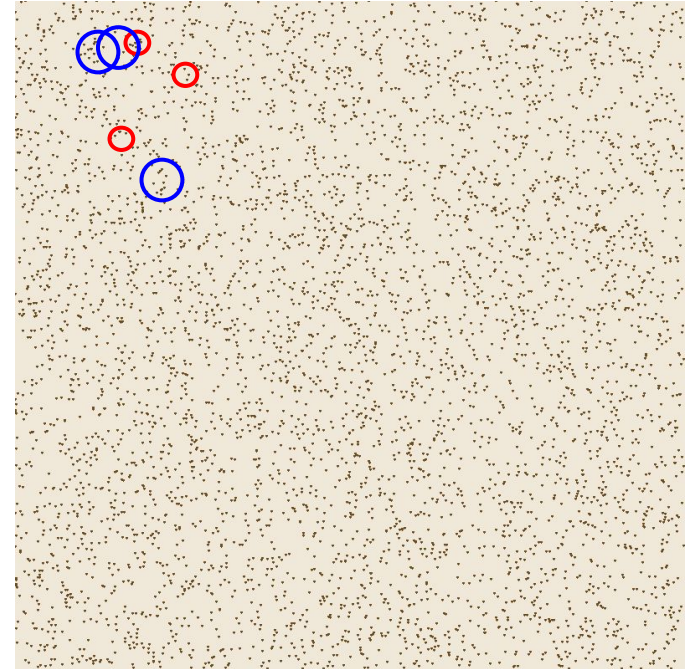
Surveiller tous les objets avec  
le coût le plus bas possible



# Mise en pratique

Trouver une stratégie

- Minimale
- Monte Carlo
- Greedy



# Le challenge !

Présentation

Premier dataset

Score

Starter kit

# Challenge - Présentation

Dessiner une fresque le plus rapidement possible



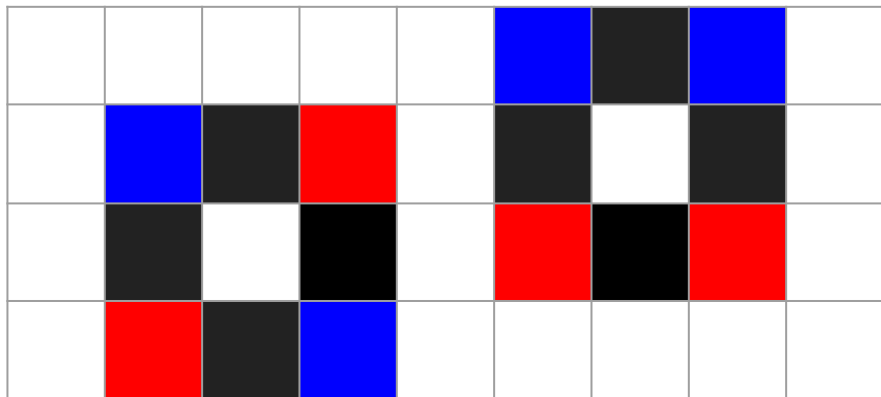
# Challenge - Présentation

Analogue à certaines problématiques réelles :

- Optimisation de processus industriels
- Rendu graphique
- Algorithmes de compression
- Machine Learning

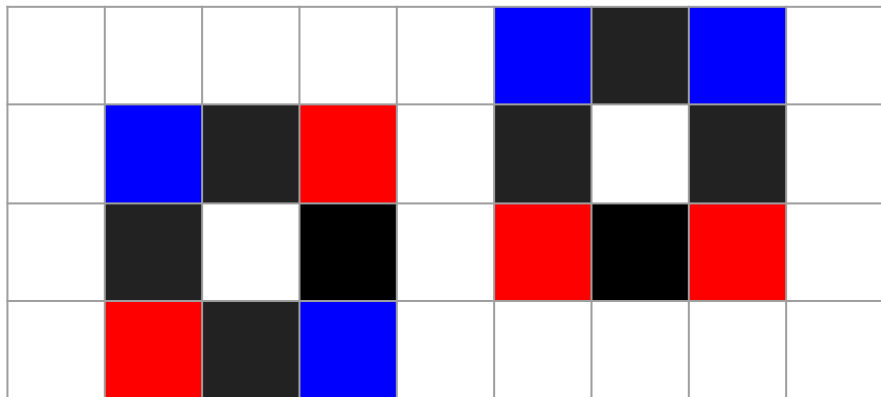
# Challenge - Dataset 1

Objectif : dessiner cette forme



# Challenge - Dataset 1

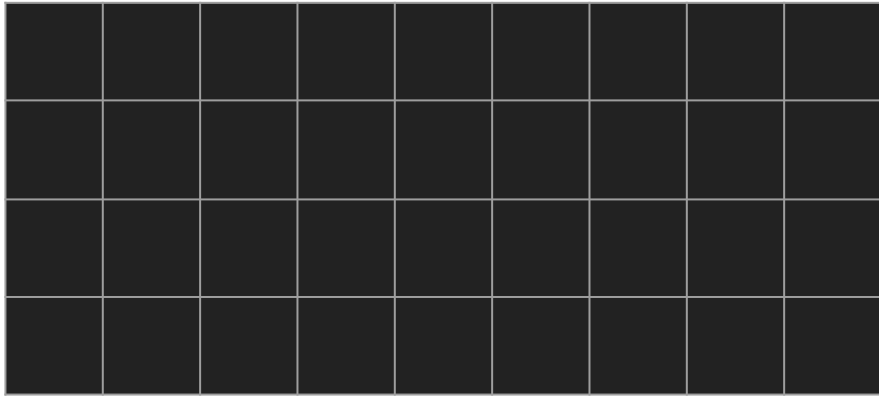
Objectif : dessiner cette forme



```
1  {
2    "comment": "Exemple de fresque",
3    "maxActions": 8,
4    "maxJokers": 2,
5    "maxJokerSize": 8,
6    "grid": [
7      [1, 1, 1, 1, 1, 2, 0, 2, 1],
8      [1, 2, 0, 3, 1, 0, 1, 0, 1],
9      [1, 0, 1, 0, 1, 3, 0, 3, 1],
10     [1, 3, 0, 2, 1, 1, 1, 1, 1]
11   ]
12 }
```

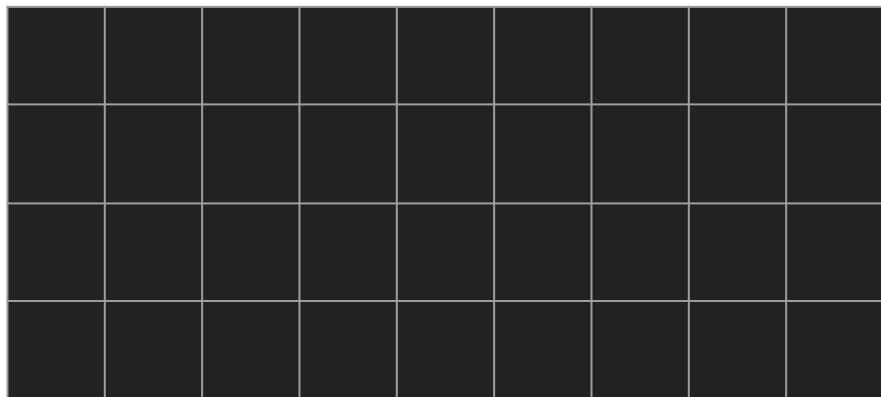
# Challenge - Dataset 1

Initialement, grille entièrement  
noire (valeur 0)



# Challenge - Dataset 1

Deux actions possibles : RECT et JOKER



RECT 4 0 4 3 1

# Challenge - Dataset 1

Deux actions possibles : RECT et JOKER

	0	1	2	3	4	5	6	7	8
0									
1									
2									
3									

RECT 4 0 4 3 1

# Challenge - Dataset 1

Deux actions possibles : RECT et JOKER

	0	1	2	3	4	5	6	7	8
0									
1									
2									
3									

RECT 4 0 4 3 1

# Challenge - Dataset 1

Deux actions possibles : RECT et JOKER

	0	1	2	3	4	5	6	7	8
0									
1									
2									
3									

RECT 4 0 4 3 1



# Challenge - Dataset 1

Deux actions possibles : RECT et JOKER

	0	1	2	3	4	5	6	7	8
0									
1									
2									
3									

RECT 4 0 4 3 1

# Challenge - Dataset 1

Deux actions possibles : RECT et JOKER

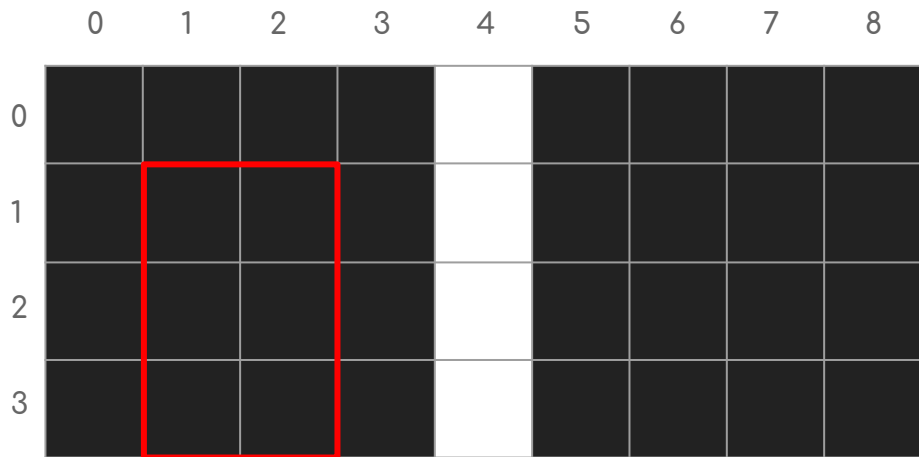
	0	1	2	3	4	5	6	7	8
0									
1									
2									
3									

RECT 4 0 4 3 1

JOKER 1 1 2 3

# Challenge - Dataset 1

Deux actions possibles : RECT et JOKER



RECT 4 0 4 3 1

JOKER 1 1 2 3

# Challenge - Dataset 1

Deux actions possibles : RECT et JOKER



RECT 4 0 4 3 1

JOKER 1 1 2 3

# Challenge - Dataset 1

Deux actions possibles : RECT et JOKER

3 limites :

- Nombre d'actions
- Nombre de JOKER
- Taille des JOKER

Limites spécifiques à chaque dataset

```
1  {
2      "comment": "Exemple de fresque",
3      "maxActions": 8,
4      "maxJokers": 2,
5      "maxJokerSize": 8,
6      "grid": [
7          [1, 1, 1, 1, 1, 2, 0, 2, 1],
8          [1, 2, 0, 3, 1, 0, 1, 0, 1],
9          [1, 0, 1, 0, 1, 3, 0, 3, 1],
10         [1, 3, 0, 2, 1, 1, 1, 1, 1]
11     ]
12 }
```

# Challenge - Score

Si l'image est parfaitement reproduite :

$$\text{score} = 1,000,000 \times \text{nbActionsMax} / \text{nbActionsUtilisées}$$

Sinon :

$$\text{score} = 1,000,000 \times \text{nbPixelsCorrects} / \text{nbPixels}$$

# Challenge - Score

## Exemple 1

La solution proposée dessine  
31 pixels corrects sur 36

Score =  $1M \times 31 / 36$

→ 861 111 points

```
1  {
2      "comment": "Exemple de fresque",
3      "maxActions": 8,
4      "maxJokers": 2,
5      "maxJokerSize": 8,
6      "grid": [
7          [1, 1, 1, 1, 1, 2, 0, 2, 1],
8          [1, 2, 0, 3, 1, 0, 1, 0, 1],
9          [1, 0, 1, 0, 1, 3, 0, 3, 1],
10         [1, 3, 0, 2, 1, 1, 1, 1, 1]
11     ]
12 }
```

# Challenge - Score

## Exemple 1

La solution proposée dessine  
36 pixels corrects sur 36  
en 7 actions

Score =  $1M \times 8 / 7$

→ 1 142 857 points

```
1  {
2      "comment": "Exemple de fresque",
3      "maxActions": 8,
4      "maxJokers": 2,
5      "maxJokerSize": 8,
6      "grid": [
7          [1, 1, 1, 1, 1, 2, 0, 2, 1],
8          [1, 2, 0, 3, 1, 0, 1, 0, 1],
9          [1, 0, 1, 0, 1, 3, 0, 3, 1],
10         [1, 3, 0, 2, 1, 1, 1, 1, 1]
11     ]
12 }
```



# Challenge - Score

6 datasets à résoudre, de taille variable

Sur chaque dataset, le score est normalisé par le score du meilleur : si vous avez 500 000 et que le meilleur a 1 200 000, vous marquez 416 666 points

N'hésitez pas à faire plusieurs soumissions !

# Challenge - Starter kit

Plusieurs fichiers vous sont fournis pour démarrer

- Script de calcul du score
- Script de validation
- Exemple d'algorithme solution

# Conseils



# Méthodologie

- Premier dataset à la main (30 min max)
- Automatiser une strat minimale (fin de journée max)
- Trouver une meilleure idée
- Implémenter la nouvelle idée



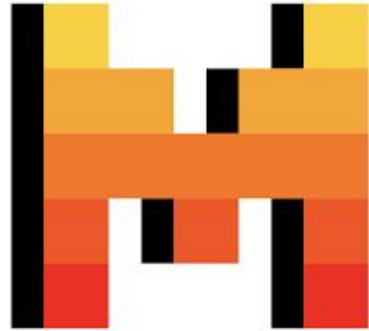
Très rapide

# Trouver une meilleure idée

- Discutez, dessinez
- Approche globale ou approche par dataset
- Pas de solutions trop complexes (trop vite) !

Votre code est amené à évoluer : une bonne conception dès le début est cruciale

# Implémenter l'idée



# Organisation

Présence : soumission chaque jour

Présentation finale :

- Top 3 + 2 équipes au hasard
- 5 minutes
- Présentation des stratégies (qui ont marché ou pas !)
- Visualisations

# Support

Teams -> Pb généraux

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

Pas d'aide sur votre code !



Merci !

Questions ?