

# BATAILLE NAVALE

## Projet de validation ISN 2016

Frédéric Muller

Lycée Arbez Carme

# Plan

- 1 Les structures de données
  - Constantes de direction
  - Structure de la grille
  - Structure des joueurs
- 2 Algorithme de résolution
  - Principe général
  - Tirs en aveugle
  - Tirs ciblés
  - Algorithme complet de résolution
  - Étude statistique
- 3 Le reste dont je n'aurai pas le temps de parler
- 4 Conclusion

# Plan

- 1 Les structures de données
  - Constantes de direction
  - Structure de la grille
  - Structure des joueurs
- 2 Algorithme de résolution
  - Principe général
  - Tirs en aveugle
  - Tirs ciblés
  - Algorithme complet de résolution
  - Étude statistique
- 3 Le reste dont je n'aurai pas le temps de parler
- 4 Conclusion

# Constantes de direction

- DROITE = (1, 0)
- GAUCHE = (-1, 0)
- BAS = (0, 1)
- HAUT = (0, -1)
- TOUTES\_DIR = (1, 1)

# Plan

- 1 Les structures de données
  - Constantes de direction
  - **Structure de la grille**
  - Structure des joueurs
- 2 Algorithme de résolution
  - Principe général
  - Tirs en aveugle
  - Tirs ciblés
  - Algorithme complet de résolution
  - Étude statistique
- 3 Le reste dont je n'aurai pas le temps de parler
- 4 Conclusion

# Initialisation

Paramètres de la grille (modifiables) :

- `xmax, ymax` : dimensions de la grille
- `taille_bateaux` : liste contenant les bateaux

# Initialisation

Paramètres de la grille (modifiables) :

- `xmax, ymax` : dimensions de la grille
- `taille_bateaux` : liste contenant les bateaux

Initialisations :

- `Grille.somme_taille` : nombre total de cases à toucher (pour déterminer la fin de la partie).

# État de la grille

Dictionnaire `Grille.etat` indexé par des tuple  $(i, j)$  de coordonnées de cases.



# État de la grille

Dictionnaire `Grille.etat` indexé par des tuple  $(i, j)$  de coordonnées de cases.

`Grille.etat[(i,j)] =`

- 0 : case vide
- 1 : case touchée (ou contenant un bateau)
- -1 : case manquée ou impossible

# État de la grille

Dictionnaire `Grille.etat` indexé par des tuple  $(i, j)$  de coordonnées de cases.

`Grille.etat[(i,j)] =`

- 0 : case vide
- 1 : case touchée (ou contenant un bateau)
- -1 : case manquée ou impossible

`Grille.vides` : liste des cases vides

## Méthodes de contrôle :

- `Grille.test_case(self, case)` : True si la case est vide et dans la grille
- `Grille.is_touche(self, case)` : True si la case contient un bateau

# Espaces vides

`Grille.get_max_space(self, case, direction, face=True)` : renvoie l'espace vide maximal dans une direction.  
Si `face==True`, la détermination se fait dans les deux sens (espace libre total horizontal ou vertical).

# Espaces vides

`Grille.get_max_space(self, case, direction, face=True)` : renvoie l'espace vide maximal dans une direction. Si `face==True`, la détermination se fait dans les deux sens (espace libre total horizontal ou vertical).

`Grille.elimine_cases(self)` : élimine les cases vides dans lesquelles le plus petit bateau ne peut pas rentrer.

```
Grille.get_max_space(self, case, direction,  
face=True) :
```

Si direction == TOUTES\_DIRECTIONS :

On renvoie le maximum de get\_max\_space(case, HORIZONTAL)  
et get\_max\_space(case, VERTICAL)

direction[0] → dh

direction[1] → dv

1 → m

$\text{case}[0] \rightarrow x$

$\text{case}[1] \rightarrow y$

Tant que la case  $(x+dh, y+dv)$  est vide et est dans la grille :

$m+1 \rightarrow m$

$x+dh \rightarrow x$

$y+dv \rightarrow y$

Si face (on regarde dans l'autre sens) :

$\text{case}[0] \rightarrow x$

$\text{case}[1] \rightarrow y$

Tant que la case  $(x\text{-dh}, y\text{-dv})$  est vide et est dans la grille :

$m+1 \rightarrow m$

$x\text{-dh} \rightarrow x$

$y\text{-dv} \rightarrow y$

Retourner  $m$



# Bateaux possibles

`Grille.get_possibles(self)` renvoie :

- La liste des bateaux possibles démarrant sur chaque case (ainsi que leurs directions) dans le dictionnaire `possibles_case`  
Par exemple :  $\{(0,0):[(5,(1,0)), (5,(0,1)),\dots], (0,1):\dots\}$

Par exemple sur la case (0,0), avec la case (3,0) déjà manquée :

	0	1	2	3	4	5	6	7	8	9
0				O						
1										
2										
3										
4										
5										
6										
7										
8										
9										

[ (5,(0,1)) ]

Par exemple sur la case (0,0), avec la case (3,0) déjà manquée :

	0	1	2	3	4	5	6	7	8	9
0				O						
1										
2										
3										
4										
5										
6										
7										
8										
9										

[ (5,(0,1)) , (4,(0,1)) ]

Par exemple sur la case (0,0), avec la case (3,0) déjà manquée :

	0	1	2	3	4	5	6	7	8	9
0				O						
1										
2										
3										
4										
5										
6										
7										
8										
9										

[ (5,(0,1)) , (4,(0,1)) , (3,(0,1)) ]

Par exemple sur la case (0,0), avec la case (3,0) déjà manquée :

	0	1	2	3	4	5	6	7	8	9
0	→			O						
1										
2										
3										
4										
5										
6										
7										
8										
9										

[ (5,(0,1)) , (4,(0,1)) , (3,(0,1)) , (3,(1,0)) ]

Par exemple sur la case (0,0), avec la case (3,0) déjà manquée :

	0	1	2	3	4	5	6	7	8	9
0	↓			O						
1										
2										
3										
4										
5										
6										
7										
8										
9										

[ (5,(0,1)) , (4,(0,1)) , (3,(0,1)) , (3,(1,0)) , (2,(0,1)) ]

Par exemple sur la case (0,0), avec la case (3,0) déjà manquée :

	0	1	2	3	4	5	6	7	8	9
0	→			O						
1										
2										
3										
4										
5										
6										
7										
8										
9										

[ (5,(0,1)) , (4,(0,1)) , (3,(0,1)) , (3,(1,0)) , (2,(0,1)) , (2,(1,0)) ]

# Bateaux possibles

`Grille.get_possibles(self)` renvoie également :

- La liste des positions (et directions) de départ possibles pour chaque bateau dans le dictionnaire `possibles`  
`{5:[((0,0), (1,0)), ((0,0), (0,1)), ((1,0), (1,0)),...], 4:...}`



Par exemple, pour le bateau de taille 5 :

	0	1	2	3	4	5	6	7	8	9
0				O						
1										
2										
3										
4										
5										
6										
7										
8										
9										

[ ((0,0), (0,1)) ]

Par exemple, pour le bateau de taille 5 :

	0	1	2	3	4	5	6	7	8	9
0				O						
1										
2										
3										
4										
5										
6										
7										
8										
9										

[ ((0,0), (0,1)) , ((1,0), (0,1)) ]

Par exemple, pour le bateau de taille 5 :

	0	1	2	3	4	5	6	7	8	9
0				O						
1										
2										
3										
4										
5										
6										
7										
8										
9										

[ ((0,0), (0,1)) , ((1,0), (0,1)) , ((2,0), (0,1)) ]

Par exemple, pour le bateau de taille 5 :

	0	1	2	3	4	5	6	7	8	9
0				O						
1										
2										
3										
4										
5										
6										
7										
8										
9										

[ ((0,0), (0,1)) , ((1,0), (0,1)) , ((2,0), (0,1)) , ((4,0), (0,1)) ]

Par exemple, pour le bateau de taille 5 :

	0	1	2	3	4	5	6	7	8	9
0				O	→	→	→	→		
1										
2										
3										
4										
5										
6										
7										
8										
9										

[ ((0,0), (0,1)) , ((1,0), (0,1)) , ((2,0), (0,1)) , ((4,0), (0,1)) , ((4,0), (1,0))... ]

`Grille.get_possibles(self) :`

On met à jour la liste des cases vides

# Récupération des bateaux possibles sur chaque case

`possibles_case` est un dictionnaire indexé par les cases

Les éléments de `possibles_case` sont des listes vides

On récupère la liste unique des bateaux restants dans

`tmp_taille_bateaux`

Pour chaque case vide :

Pour chaque direction dans [DROITE, BAS] :

`tmax` est l'espace maximum dans direction

On ajoute à `possible_case[case]` les tuples (taille, direction)

pour les tailles dans `tmp_taille_bateaux`, si `taille <= tmax`

# Récupération des position possibles pour chaque bateau

possibles est un dictionnaire indexé par les tailles de bateaux

Les éléments de possibles sont des listes vides

Pour chaque case dans possible\_case :

    Pour chaque couple (taille, direction) dans possible\_case[case] :

        On ajoute (case, direction) à possibles[taille]

# Gestion de la flotte

Gestion de la liste `Grille.taille_bateaux` :

- `Grille.get_taille_max(self)` et `Grille.get_taille_min(self)` : mise à jour respectivement de la taille maximum et la taille minimum des bateaux restant à trouver
- `Grille.rem_bateau(self, taille)` : supprime un bateau de la liste `Grille.taille_bateaux`



Ajout d'un bateau :

- `Grille.test_bateau(self, bateau)` : teste si le bateau est valide
- `Grille.add_bateau(self, bateau)` : ajout d'un bateau

## Flotte aléatoire :

- `Grille.add_bateau_alea(self, taille)` : ajout d'un bateau aléatoire de taille donnée sur la grille
- `Grille.init_bateaux_alea(self)` : génération d'une flotte aléatoire

# Plan

- 1 Les structures de données
  - Constantes de direction
  - Structure de la grille
  - Structure des joueurs
- 2 Algorithme de résolution
  - Principe général
  - Tirs en aveugle
  - Tirs ciblés
  - Algorithme complet de résolution
  - Étude statistique
- 3 Le reste dont je n'aurai pas le temps de parler
- 4 Conclusion

# Grilles du joueur

- `Joueur.grille_joueur` : la grille sur laquelle le joueur place ses bateaux
- `Joueur.grille_adverse` : la grille de l'adversaire
- `Joueur.grille_suivi` : la grille sur laquelle le joueur joue ses coups

# Gestion des messages

- `Joueur.messages` : liste des messages à afficher
- `Joueur.add_message(self, message)` : ajoute le message à la liste précédente
- `Joueur.affiche_messages(self)` : vide la liste des messages en les affichant (méthode surchargée suivant l'interface)

## Vérification des bateaux coulés

`Joueur.check_coules` : vérifie les bateaux coulés sur la grille de suivi

`Joueur.check_coules(self) :`

`xmax` et `ymax` sont les dimensions de la grille et `dimensions=(xmax, ymax)`.

`coules` est une variable globale contenant la liste des cases  
marquées comme coulées

`checked` est une liste vide

Pour chaque case dans les cases jouées triées en ordre  
lexicographique :

liste\_touchees est une liste vide

Si case a été marquée comme touchée :

Si case est dans `checked` ou dans `coules` :

On continue la boucle (on ignore cette case)

Vrai  $\rightarrow$  case\_isolee

Pour d dans [DROITE, BAS] :

Si la case (case[0]+d[0],case[1]+d[1]) est hors de la grille :

On continue la boucle

Si la case (case[0]+d[0],case[1]+d[1]) est marquée touchée :

d  $\rightarrow$  direction

Faux  $\rightarrow$  case\_isolee

On casse la boucle

Si case\_isolee est Vraie :

On continue la boucle (on ignore cette case)



$0 \rightarrow k$

Tant que  $\text{case}[0] + k * \text{direction}[0] < \text{xmax}$

et  $\text{case}[1] + k * \text{direction}[1] < \text{ymax}$

et  $(\text{case}[0] + k * \text{direction}[0], \text{case}[1] + k * \text{direction}[1])$

est marquée touchée :

On ajoute la case

$(\text{case}[0] + k * \text{direction}[0], \text{case}[1] + k * \text{direction}[1])$

aux listes `checked` et `liste_touchees`

$k += 1$

Si (  
    (case[direction[1]]== 0  
        ou (case[0]-direction[0],case[1]-direction[1]) est manquée)  
et (case[direction[1]]+k== dimensions[direction[1]]  
    ou (case[0]+k\*direction[0],case[1]+k\*direction[1]) est manquée)  
) ou len(liste\_touchees)==taille\_max :  
    Enlever le bateau de taille len(liste\_touchees)  
    des bateaux restants  
    Éliminer les cases adjacentes à celles de liste\_touchees  
    Ajouter les cases de liste\_touchees à la liste coules

	0	1	2	3	4	5	6	7	8	9
0										
1	X	O	X	X	X	X	O			
2		O								
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (0,1)

	0	1	2	3	4	5	6	7	8	9
0										
1	X	O	X	X	X	X	O			
2		O								
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (0,1)

	0	1	2	3	4	5	6	7	8	9
0										
1	X	O	X	X	X	X	O			
2		O								
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (0,1)

Case isolée

	0	1	2	3	4	5	6	7	8	9
0										
1	X	O	X	X	X	X	O			
2		O								
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (0,4)

	0	1	2	3	4	5	6	7	8	9
0										
1	X	O	X	X	X	X	O			
2		O								
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (0,7)

	0	1	2	3	4	5	6	7	8	9
0										
1	X	O	X	X	X	X	O			
2		O								
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (1,1)



	0	1	2	3	4	5	6	7	8	9
0										
1	X	O	X	X	X	X	O			
2		O								
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (1,2)

	0	1	2	3	4	5	6	7	8	9
0										
1	X	O	X	X	X	X	O			
2		O								
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (1,3)

	0	1	2	3	4	5	6	7	8	9
0										
1	X	O	X	X	X	X	O			
2		O								
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (1,3)

Direction : DROITE

	0	1	2	3	4	5	6	7	8	9
0										
1	X	O	X	X	X	X	O			
2		O								
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (1,3)

k=0

liste\_touches :

[(1,3)]

	0	1	2	3	4	5	6	7	8	9
0										
1	X	O	X	X	X	X	O			
2		O								
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (1,3)

k=1

liste\_touche :

[(1,3), (2,3)]

	0	1	2	3	4	5	6	7	8	9
0										
1	X	O	X	X	X	X	O			
2		O								
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (1,3)

k=2

liste\_touche :

[(1,3), (2,3)]

	0	1	2	3	4	5	6	7	8	9
0										
1	X	O	X	X	X	X	O			
2		O								
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (1,3)

k=2

liste\_touche :

[(1,3), (2,3)]

Bateau non coulé

	0	1	2	3	4	5	6	7	8	9
0										
1	X	O	X	X	X	X	O			
2		O								
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (1,5)



	0	1	2	3	4	5	6	7	8	9
0										
1	X	O	X	X	X	X	O			
2		O								
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (1,5)

	0	1	2	3	4	5	6	7	8	9
0										
1	X	O	X	X	X	X	O			
2		O								
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (1,5)

Direction : BAS

	0	1	2	3	4	5	6	7	8	9
0										
1	X	O	X	X	X	X	O			
2		O								
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (1,5)

k=0

liste\_touche :

[(1,5)]

	0	1	2	3	4	5	6	7	8	9
0										
1	X	O	X	X	X	X	O			
2		O								
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (1,5)

k=1

liste\_touche :

[(1,5), (1,6)]

	0	1	2	3	4	5	6	7	8	9
0										
1	X	O	X	X	X	X	O			
2		O								
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (1,5)

k=2

liste\_touche :

[(1,5), (1,6), (1,7)]

	0	1	2	3	4	5	6	7	8	9
0										
1	X	O	X	X	X	X	O			
2		O								
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (1,5)

k=3

liste\_touche :

[(1,5), (1,6), (1,7)]

	0	1	2	3	4	5	6	7	8	9
0										
1	X	O	X	X	X	X	O			
2		O								
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (1,5)

Bateau non coulé

	0	1	2	3	4	5	6	7	8	9
0										
1	X	O	X	X	X	X	O			
2		O								
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (2,1)



	0	1	2	3	4	5	6	7	8	9
0										
1	X	O	X	X	X	X	O			
2		O								
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (2,1)

Direction : DROITE

	0	1	2	3	4	5	6	7	8	9
0										
1	X	O	X	X	X	X	O			
2		O								
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (2,1)

k=0

liste\_touche :

[(2,1)]

	0	1	2	3	4	5	6	7	8	9
0										
1	X	O	X	X	X	X	O			
2		O								
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (2,1)

k=1

liste\_touche :

[(2,1), (3,1)]

	0	1	2	3	4	5	6	7	8	9
0										
1	X	O	X	X	X	X	O			
2		O								
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (2,1)

k=2

liste\_touche :

[(2,1), (3,1), (4,1)]

	0	1	2	3	4	5	6	7	8	9
0										
1	X	O	X	X	X	X	O			
2		O								
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (2,1)

k=3

liste\_touche :

[(2,1), (3,1), (4,1), (5,1)]

	0	1	2	3	4	5	6	7	8	9
0										
1	X	O	X	X	X	X	O			
2		O								
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (2,1)

k=4

liste\_touche :

[(2,1), (3,1), (4,1), (5,1)]

	0	1	2	3	4	5	6	7	8	9
0										
1	X	O	X	X	X	X	O			
2		O								
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (2,1)

	0	1	2	3	4	5	6	7	8	9
0										
1	X	O	X	X	X	X	O			
2		O								
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (2,1)

Bateau coulé !



	0	1	2	3	4	5	6	7	8	9
0			O	O	O	O				
1	X	O	X	X	X	X	O			
2		O	O	O	O	O				
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (2,1)

Bateau coulé !

Élimination des cases  
adjacentes et mémorisation  
de ses cases dans la liste  
Joueur.coules

	0	1	2	3	4	5	6	7	8	9
0			O	O	O	O				
1	X	O	X	X	X	X	O			
2		O	O	O	O	O				
3		X	X							
4	O									
5		X								
6		X								
7	O	X								
8										
9										

Case : (6,1)

# Plan

- 1 Les structures de données
  - Constantes de direction
  - Structure de la grille
  - Structure des joueurs
- 2 Algorithme de résolution
  - Principe général
  - Tirs en aveugle
  - Tirs ciblés
  - Algorithme complet de résolution
  - Étude statistique
- 3 Le reste dont je n'aurai pas le temps de parler
- 4 Conclusion

# Plan

- 1 Les structures de données
  - Constantes de direction
  - Structure de la grille
  - Structure des joueurs
- 2 Algorithme de résolution
  - Principe général
  - Tirs en aveugle
  - Tirs ciblés
  - Algorithme complet de résolution
  - Étude statistique
- 3 Le reste dont je n'aurai pas le temps de parler
- 4 Conclusion

Implémenté dans la classe Ordi (Joueur)

Implémenté dans la classe `Ordi (Joueur)`

`Ordi.coup_suivant(self)` : fait jouer le coup suivant

Implémenté dans la classe `Ordi (Joueur)`

`Ordi.coup_suivant(self)` : fait jouer le coup suivant

`Ordi.case_courante` : case qui est entrain d'être jouée

Deux phases :



Deux phases :

- Phase de tirs en aveugle

Deux phases :

- Phase de tirs en aveugle
- Phase de tirs ciblés

Différents niveaux de jeu :

- `Ordi.niveau=1` : Uniquement des tirs aléatoires en aveugle et pas de tirs ciblés.

## Différents niveaux de jeu :

- `Ordi.niveau=2` : Tirs aléatoires et phase de tirs ciblés.

## Différents niveaux de jeu :

- `Ordi.niveau=3` : Tirs aléatoires sur les cases noires et phase de tirs ciblés.

## Différents niveaux de jeu :

- `Ordi.niveau=4` : Détermination de la case optimale par des échantillons et phase de tirs ciblés.

## Différents niveaux de jeu :

- `Ordi.niveau=5` : Détermination de la case optimale par le nombre de bateaux possibles sur chaque case et phase de tirs ciblés.

## Différents niveaux de jeu :

- `Ordi.niveau=6` : Idem niveau 5, mais dès que le nombre de cases vides passe en-dessous de `Ordi.seuil`, l'algorithme énumère toutes les répartitions de bateaux possibles.



# Plan

- 1 Les structures de données
  - Constantes de direction
  - Structure de la grille
  - Structure des joueurs
- 2 Algorithme de résolution
  - Principe général
  - **Tirs en aveugle**
  - Tirs ciblés
  - Algorithme complet de résolution
  - Étude statistique
- 3 Le reste dont je n'aurai pas le temps de parler
- 4 Conclusion

# Tirs aléatoires

Tirs aléatoires uniformément sur les cases vides

# Tirs aléatoires

Tirs aléatoires uniformément sur les cases vides

Peu performant

# Tirs aléatoires

Tirs aléatoires uniformément sur les cases vides

Peu performant

Facile à implémenter

# Tirs aléatoires

Tirs aléatoires uniformément sur les cases vides

Peu performant

Facile à implémenter

Seule méthode qui permet de finir la grille uniquement avec des tirs en aveugle

# Tirs sur les cases noires

Tir sur une case sur deux aléatoirement

	0	1	2	3	4	5	6	7	8	9
0										
1										
2										
3										
4										
5										
6										
7										
8										
9										

# Tirs sur les cases noires

Meilleurs performances que le précédent

# Tirs sur les cases noires

Meilleures performances que le précédent

Facile à implémenter



# Tirs sur les cases noires

Meilleures performances que le précédent

Facile à implémenter

Nécessite que le plus petit bateau à trouver soit au moins de taille 2

# Méthode par échantillonnage

Création de `nb_echantillons` de répartitions aléatoires des bateaux restants et comptage du nombre de cases occupées

# Méthode par échantillonnage

Création de `nb_echantillons` de répartitions aléatoires des bateaux restants et comptage du nombre de cases occupées

Optimisation globale

# Méthode par échantillonnage

Création de `nb_echantillons` de répartitions aléatoires des bateaux restants et comptage du nombre de cases occupées

Optimisation globale

Temps de résolution linéaire en `nb_echantillons`

```
Grille.case_max_echantillons(self, nb_echantillons):
```

probas est un dictionnaire indexé sur les cases

Pour chaque case,  $0 \rightarrow \text{probas}[\text{case}]$

On répète nb\_echantillons fois :

    grille\_tmp reçoit une copie temporaire de la grille du suivi

    On crée une flotte aléatoire sur grille\_tmp

    Pour chaque case vide dans la grille de suivi originale :

        Si la case contient un bateau dans grille\_tmp :

$\text{probas}[\text{case}] + 1 \rightarrow \text{probas}[\text{case}]$

Pour chaque case,  $\text{probas}[\text{case}] / \text{nb\_echantillons} \rightarrow \text{probas}[\text{case}]$

case\_max est la case qui a la plus grande probabilité pmax

On retourne (case\_max, pmax)

# Méthode par comptage

Comptage du nombre de bateaux possibles sur chaque case

# Méthode par comptage

Comptage du nombre de bateaux possibles sur chaque case

Optimisation locale

# Méthode par comptage

Comptage du nombre de bateaux possibles sur chaque case

Optimisation locale

Très efficace



Par exemple sur une grille vierge, il y a 7 bateaux de taille 5 sur la case (3,2) :

	0	1	2	3	4	5	6	7	8	9
0										
1										
2				1						
3										
4										
5										
6										
7										
8										
9										

Par exemple sur une grille vierge, il y a 7 bateaux de taille 5 sur la case (3,2) :

	0	1	2	3	4	5	6	7	8	9
0										
1										
2				2						
3										
4										
5										
6										
7										
8										
9										

Par exemple sur une grille vierge, il y a 7 bateaux de taille 5 sur la case (3,2) :

	0	1	2	3	4	5	6	7	8	9
0										
1										
2				3						
3										
4										
5										
6										
7										
8										
9										

Par exemple sur une grille vierge, il y a 7 bateaux de taille 5 sur la case (3,2) :

	0	1	2	3	4	5	6	7	8	9
0										
1										
2				4						
3										
4										
5										
6										
7										
8										
9										

Par exemple sur une grille vierge, il y a 7 bateaux de taille 5 sur la case (3,2) :

	0	1	2	3	4	5	6	7	8	9
0										
1										
2				5						
3										
4										
5										
6										
7										
8										
9										

Par exemple sur une grille vierge, il y a 7 bateaux de taille 5 sur la case (3,2) :

	0	1	2	3	4	5	6	7	8	9
0										
1										
2				6						
3										
4										
5										
6										
7										
8										
9										

Par exemple sur une grille vierge, il y a 7 bateaux de taille 5 sur la case (3,2) :

	0	1	2	3	4	5	6	7	8	9
0										
1										
2				7						
3										
4										
5										
6										
7										
8										
9										

`Grille.case_max(self)`

probas est un dictionnaire indexé sur les cases

Pour chaque case,  $0 \rightarrow \text{probas}[\text{case}]$

On récupère la liste placements possibles pour chaque bateau  
(dans le dictionnaire possibles)

Pour chaque taille de bateau restant :

    Pour chaque (case, direction) dans possibles[taille] :

        #La probabilité de chaque case occupée par le bateau est  
augmentée de 1

    Pour k allant de 0 à taille :

        probas[case+k\*direction] est augmentée de 1

case\_max est la case qui a le plus de possibilités pmax

On retourne (case\_max, pmax)



Bateaux de taille 5 :

	0	1	2	3	4	5	6	7	8	9
0	1									
1	1									
2	1									
3	1									
4	1									
5										
6										
7										
8										
9										

Bateaux de taille 5 :

	0	1	2	3	4	5	6	7	8	9
0	2	1	1	1	1					
1	1									
2	1									
3	1									
4	1									
5										
6										
7										
8										
9										

Bateaux de taille 5 :

	0	1	2	3	4	5	6	7	8	9
0	2	2	1	1	1					
1	1	1								
2	1	1								
3	1	1								
4	1	1								
5										
6										
7										
8										
9										

Bateaux de taille 5 :

	0	1	2	3	4	5	6	7	8	9
0	2	3	2	2	2	1				
1	1	1								
2	1	1								
3	1	1								
4	1	1								
5										
6										
7										
8										
9										

## Bateaux de taille 5 :

	0	1	2	3	4	5	6	7	8	9
0	2	3	4	5	6	6	5	4	3	2
1	3	4	5	6	7	7	6	5	4	3
2	4	5	6	7	8	8	7	6	5	4
3	5	6	7	8	9	9	8	7	6	5
4	6	7	8	9	10	10	9	8	7	6
5	6	7	8	9	10	10	9	8	7	6
6	5	6	7	8	9	9	8	7	6	5
7	4	5	6	7	8	8	7	6	5	4
8	3	4	5	6	7	7	6	5	4	3
9	2	3	4	5	6	6	5	4	3	2

## Bateaux de taille 4 :

	0	1	2	3	4	5	6	7	8	9
0	4	6	8	10	11	11	10	8	6	4
1	6	8	10	12	13	13	12	10	8	6
2	8	10	12	14	15	15	14	12	10	8
3	10	12	14	16	17	17	16	14	12	10
4	11	13	15	17	18	18	17	15	13	11
5	11	13	15	17	18	18	17	15	13	11
6	10	12	14	16	17	17	16	14	12	10
7	8	10	12	14	15	15	14	12	10	8
8	6	8	10	12	13	13	12	10	8	6
9	4	6	8	10	11	11	10	8	6	4

Tous les bateaux :

	0	1	2	3	4	5	6	7	8	9
0	10	15	19	21	22	22	21	19	15	10
1	15	20	24	26	27	27	26	24	20	15
2	19	24	28	30	31	31	30	28	24	19
3	21	26	30	32	33	33	32	30	26	21
4	22	27	31	33	34	34	33	31	27	22
5	22	27	31	33	34	34	33	31	27	22
6	21	26	30	32	33	33	32	30	26	21
7	19	24	28	30	31	31	30	28	24	19
8	15	20	24	26	27	27	26	24	20	15
9	10	15	19	21	22	22	21	19	15	10

# Méthode exhaustive

Détermination récursive de tous les arrangements de bateaux possibles  
et comptage du nombre de cases occupées



# Méthode exhaustive

Détermination récursive de tous les arrangements de bateaux possibles  
et comptage du nombre de cases occupées

Algorithme exponentiel donc inutilisable tel quel

`Grille.case_max_all(self) :`

gtmp est une copie temporaire de la grille

probas\_all est un dictionnaire indexé par les cases

Pour chaque case,  $0 \rightarrow \text{probas\_all}[\text{case}]$

Créer toutes les répartitions de bateaux sur gtmp

Récupérer la case qui en contient le plus

```
Grille.make_all(self, gtmp) :
```

# En entrée : une grille gtmp

S'il n'y a plus de bateaux dans la liste de gtmp :

    Mettre à jour les probabilités avec les cases occupées sur gtmp

    Sortir de la fonction

Récupérer les positions de bateaux possibles sur gtmp

taille est la taille du premier bateau restant

Pour (case, direction) dans possibles[taille] :

    gtmp2 est une copie temporaire de gtmp

    Ajouter le bateau (taille, case, direction) à gtmp2

    Enlever ce bateau de la liste de ceux de gtmp2

    Créer toutes les répartitions de bateaux sur gtmp2

Exemple sur une grille de taille 4×3 avec deux bateaux de tailles 3 et 2

	0	1	2	3
0	X	O		
1	X	O		
2	X	O		

	0	1	2	3
0	X	O	X	O
1	X	O	X	O
2	X	O	O	

	0	1	2	3
0	X	O	X	X
1	X	O	O	O
2	X	O		

	0	1	2	3
0	X	O	O	
1	X	O	X	O
2	X	O	X	O



	0	1	2	3
0	X	O	O	O
1	X	O	X	X
2	X	O	O	O

	0	1	2	3
0	X	O		
1	X	O	O	O
2	X	O	X	X

	0	1	2	3
0	X	O	O	X
1	X	O	O	X
2	X	O		O

	0	1	2	3
0	X	O		O
1	X	O	O	X
2	X	O	O	X

	0	1	2	3
0	X	X	X	O
1	O	O	O	
2				

		0	1	2	3
0	X	X	X	X	O
1	O	O	O		
2	X	X	O		

	0	1	2	3
0	X	X	X	O
1	O	O	O	
2	O	X	X	O

	0	1	2	3
0	X	X	X	O
1	O	O	O	O
2		O	X	X



	0	1	2	3
0	X	X	X	O
1	O	O	O	X
2			O	X

	0	1	2	3
0	O	O	O	
1	X	X	X	O
2	O	O	O	

	0	1	2	3
0				
1	O	O	O	
2	X	X	X	O

	0	1	2	3
0	X	X	O	
1	O	O	O	
2	X	X	X	O

		0	1	2	3
0	O	X	X	O	
1	O	O	O		
2	X	X	X	O	

	0	1	2	3
0		O	X	X
1	O	O	O	O
2	X	X	X	O

	0	1	2	3
0			O	X
1	O	O	O	X
2	X	X	X	O

	0	1	2	3
0	O	X	O	
1	O	X	O	
2	O	X	O	



	0	1	2	3
0	O	X	O	X
1	O	X	O	X
2	O	X	O	O

	0	1	2	3
0	O	X	O	O
1	O	X	O	X
2	O	X	O	X

	0	1	2	3
0	O	X	X	X
1		O	O	O
2				

	0	1	2	3
0	O	X	X	X
1	X	O	O	O
2	X	O		

	0	1	2	3
0	O	X	X	X
1	O	O	O	O
2	X	X	O	

	0	1	2	3
0	O	X	X	X
1		O	O	O
2	O	X	X	O

	0	1	2	3
0	O	X	X	X
1		O	O	O
2		O	X	X

	0	1	2	3
0		O	O	O
1	O	X	X	X
2		O	O	O



	0	1	2	3
0				
1		O	O	O
2	O	X	X	X

	0	1	2	3
0	X	O		
1	X	O	O	O
2	O	X	X	X

	0	1	2	3
0	X	X	O	
1	O	O	O	O
2	O	X	X	X

	0	1	2	3
0	O	X	X	O
1		O	O	O
2	O	X	X	X

	0	1	2	3
0		O	X	X
1		O	O	O
2	O	X	X	X

	0	1	2	3
0		O	X	O
1		O	X	O
2		O	X	O

	0	1	2	3
0	X	O	X	O
1	X	O	X	O
2	O	O	X	O

	0	1	2	3
0	O	O	X	O
1	X	O	X	O
2	X	O	X	O



	0	1	2	3
0			O	X
1			O	X
2			O	X

	0	1	2	3
0	X	O	O	X
1	X	O	O	X
2	O		O	X

	0	1	2	3
0	X	X	O	X
1	O	O	O	X
2			O	X

		0	1	2	3
0	O			O	X
1	X		O	O	X
2	X		O	O	X

	0	1	2	3
0	O	O	O	X
1	X	X	O	X
2	O	O	O	X

	0	1	2	3
0			O	X
1	O	O	O	X
2	X	X	O	X

	0	1	2	3
0	O	X	O	X
1	O	X	O	X
2		O	O	X

	0	1	2	3
0		O	O	X
1	O	X	O	X
2	O	X	O	X



	0	1	2	3
0	17	16	16	17
1	14	5	5	14
2	17	16	16	17

# Plan

- 1 Les structures de données
  - Constantes de direction
  - Structure de la grille
  - Structure des joueurs
- 2 Algorithme de résolution
  - Principe général
  - Tirs en aveugle
  - **Tirs ciblés**
  - Algorithme complet de résolution
  - Étude statistique
- 3 Le reste dont je n'aurai pas le temps de parler
- 4 Conclusion

# Généralités

Gestion d'une file d'attente sous la forme de liste : `Ordre.queue`

# Généralités

Gestion d'une file d'attente sous la forme de liste : `Ordi.queue`

Mémorisation de la première case touchée dans `Ordi.case_touchee`  
et de la liste des cases touchées sur le bateau dans  
`Ordi.liste_touchee`

# Création de la file d'attente

Création de la file d'attente après la première case touchée lors de la phase de tirs en aveugle

# Création de la file d'attente

Création de la file d'attente après la première case touchée lors de la phase de tirs en aveugle

Ajout des cases adjacentes valides à `case_touchee`

# Création de la file d'attente

Création de la file d'attente après la première case touchée lors de la phase de tirs en aveugle

Ajout des cases adjacentes valides à `case_touchee`

N'ajoute que les cases dans les directions dans lesquelles le plus petit bateau peut rentrer.

`Ordi.add_adjacentes_premiere(self) :`

adjacent est la liste des cases vides adjacentes à case\_touchee

taille\_min est la taille minimale des bateaux restants

Pour direction dans [HORIZONTAL, VERTICAL] :

direction[0]→k

Si l'espace maximum dans direction sur case touché  $\geq$  taille\_min :

Pour case dans adjacent :

Si case[k]==case\_touchee[k] :

Ajouter case à la file d'attente

Sinon afficher que cette direction ne convient pas



# Optimisation de la file d'attente

Détermination du nombre de bateaux possibles contenant  
`case_touchee` sur les cases de la file d'attente

Tri décroissant de la file d'attente selon ce critère

```
Grille.case_max_touchee(self, case_touchee) :
```

Pour des raisons de mise en page, notons ct la case touchée.

On marque temporairement case\_touchee comme vide

On récupère la liste placements possibles pour chaque bateau

(dans le dictionnaire possibles)

probas est un dictionnaire indexé sur les cases

Pour chaque case,  $0 \rightarrow \text{probas}[\text{case}]$

Pour chaque taille de bateau restant :

Pour chaque direction dans [HORIZONTAL, VERTICAL] :

#Bateau qui se termine sur case\_touchee

Si  $ct[direction[1]] - (taille - 1) * direction[direction[1]] \geq 0$

et  $((ct[0] - (taille - 1) * direction[0],$   
 $ct[1] - (taille - 1) * direction[1]), direction)$

est dans possibles :

$probas[(ct[0] - direction[0], ct[1] - direction[1])] += 1$

#Bateau à cheval sur sur case\_touchee

Pour k allant de 1 à taille-2 :

Si  $ct[direction[1]] - k * direction[direction[1]] \geq 0$

et  $((ct[0] - k * direction[0], ct[1] - k * direction[1]),$   
 $direction)$  est dans possibles :

$probas[(ct[0] - direction[0], ct[1] - direction[1])] += 1$

$probas[(ct[0] + direction[0], ct[1] + direction[1])] += 1$

#Bateau qui démarre sur case\_touchee

Si ((ct[0], ct[1]), direction) est dans possibles :

    probas[(ct[0]+direction[0],ct[1]+direction[1])] += 1

On remet l'état de case\_touchee à touché

On trie probas dans l'ordre décroissant du nombre de possibilités

On renvoie cette liste

Sur la grille suivante, la case (6,2) a été manquée et la première case touchée est la case (3,2) :

## Bateaux horizontaux de taille 5

	0	1	2	3	4	5	6	7	8	9
0										
1				0						
2			1	X	1		O			
3				0						
4										
5										
6										
7										
8										
9										

## Bateaux horizontaux de taille 5

	0	1	2	3	4	5	6	7	8	9
0										
1				0						
2			2	X	2		O			
3				0						
4										
5										
6										
7										
8										
9										

## Bateaux horizontaux de taille 4

	0	1	2	3	4	5	6	7	8	9
0										
1				0						
2			3	X	2		O			
3				0						
4										
5										
6										
7										
8										
9										



## Bateaux horizontaux de taille 4

	0	1	2	3	4	5	6	7	8	9
0										
1				0						
2			4	X	3		O			
3				0						
4										
5										
6										
7										
8										
9										

## Bateaux horizontaux de taille 4

	0	1	2	3	4	5	6	7	8	9
0										
1				0						
2			5	X	4		O			
3				0						
4										
5										
6										
7										
8										
9										

## Bateaux horizontaux de taille 3 (×2)

	0	1	2	3	4	5	6	7	8	9
0										
1				0						
2			7	X	4		O			
3				0						
4										
5										
6										
7										
8										
9										

## Bateaux horizontaux de taille 3 (×2)

	0	1	2	3	4	5	6	7	8	9
0										
1				0						
2			9	X	6		O			
3				0						
4										
5										
6										
7										
8										
9										

## Bateaux horizontaux de taille 3 (×2)

	0	1	2	3	4	5	6	7	8	9
0										
1				0						
2			9	X	8		O			
3				0						
4										
5										
6										
7										
8										
9										

## Bateaux horizontaux de taille 2

	0	1	2	3	4	5	6	7	8	9
0										
1				0						
2			10	X	8		O			
3				0						
4										
5										
6										
7										
8										
9										

## Bateaux horizontaux de taille 2

	0	1	2	3	4	5	6	7	8	9
0										
1				0						
2			10	X	9		O			
3				0						
4										
5										
6										
7										
8										
9										

## Bateaux verticaux de taille 5

	0	1	2	3	4	5	6	7	8	9
0										
1				1						
2			10	X	9		O			
3				1						
4										
5										
6										
7										
8										
9										



## Bateaux verticaux de taille 5

	0	1	2	3	4	5	6	7	8	9
0										
1				2						
2			10	X	9		O			
3				2						
4										
5										
6										
7										
8										
9										

## Bateaux verticaux de taille 5

	0	1	2	3	4	5	6	7	8	9
0										
1				2						
2			10	X	9		O			
3				3						
4										
5										
6										
7										
8										
9										

## Bateaux verticaux de taille 4

	0	1	2	3	4	5	6	7	8	9
0										
1				3						
2			10	X	9		O			
3				4						
4										
5										
6										
7										
8										
9										

## Bateaux verticaux de taille 4

	0	1	2	3	4	5	6	7	8	9
0										
1				4						
2			10	X	9		O			
3				5						
4										
5										
6										
7										
8										
9										

## Bateaux verticaux de taille 4

	0	1	2	3	4	5	6	7	8	9
0										
1				4						
2			10	X	9		O			
3				6						
4										
5										
6										
7										
8										
9										

## Bateaux verticaux de taille 3 (x2)

	0	1	2	3	4	5	6	7	8	9
0										
1				6						
2			10	X	9		O			
3				6						
4										
5										
6										
7										
8										
9										

## Bateaux verticaux de taille 3 (x2)

	0	1	2	3	4	5	6	7	8	9
0										
1				8						
2			10	X	9		O			
3				8						
4										
5										
6										
7										
8										
9										

## Bateaux verticaux de taille 3 (x2)

	0	1	2	3	4	5	6	7	8	9
0										
1				8						
2			10	X	9		O			
3				10						
4										
5										
6										
7										
8										
9										



## Bateaux verticaux de taille 2

	0	1	2	3	4	5	6	7	8	9
0										
1				9						
2			10	X	9		O			
3				10						
4										
5										
6										
7										
8										
9										

## Bateaux verticaux de taille 2

	0	1	2	3	4	5	6	7	8	9
0										
1				9						
2			10	X	9		O			
3				11						
4										
5										
6										
7										
8										
9										

Au final la file d'attente sera ordonnée de la façon suivante :

[ (3,3) , (2,2) , (3,1) , (4,2) ]

Au final la file d'attente sera ordonnée de la façon suivante :

$[(3,3) , (2,2) , (3,1) , (4,2) ]$

La liste `probas_liste` contient les tuples (case, proba).

Au final la file d'attente sera ordonnée de la façon suivante :

`[(3,3) , (2,2) , (3,1) , (4,2) ]`

La liste `probas_liste` contient les tuples (case, proba).

On la trie avec l'instruction suivante :

```
sorted(probas_liste, key=lambda proba: proba[1],  
reverse = True)
```

## Deuxième tir touché

Détermination de la direction du bateau

## Deuxième tir touché

Détermination de la direction du bateau

On compare les coordonnées de `case_touchee` avec celles de `case_courante`

## Deuxième tir touché

Détermination de la direction du bateau

On compare les coordonnées de `case_touchee` avec celles de `case_courante`

On enlève de la file d'attente les cases qui ne sont pas dans la bonne direction



## Deuxième tir touché

Détermination de la direction du bateau

On compare les coordonnées de `case_touchee` avec celles de `case_courante`

On enlève de la file d'attente les cases qui ne sont pas dans la bonne direction

On ajoute à la file d'attente la case à l'extrémité (si elle est valide)

`Ordi.update_queue_touche(self) :`

Si `case_courante[1] == case_touchee[1] :`

`direction=HORIZONTAL`

Sinon :

`direction=VERTICAL`

Si on vient de découvrir la direction du bateau (`len(liste_touchee)==1`) :

    On affiche cette direction

    On enlève les cases

        (`case_touchee[0]-direction[1], case_touchee[1]-direction[0]`)

        et (`case_touchee[0]+direction[1], case_touchee[1]+direction[0]`)

        de la file d'attente

nv\_case est la case

$$\begin{aligned} &(\text{case\_courante}[0] + \\ &\text{direction}[0] * \text{signe}(\text{case\_courante}[0] - \text{case\_touchee}[0]) , \\ &\text{case\_courante}[1] + \\ &\text{direction}[1] * \text{signe}(\text{case\_courante}[1] - \text{case\_touchee}[1])) \end{aligned}$$

Si nv\_case est dans la grille et est vide :

On ajoute nv\_case à la file d'attente

On ajoute case\_courante à liste\_touches

## Deuxième coup manqué

Élimination d'une direction éventuelle

## Deuxième coup manqué

Élimination d'une direction éventuelle

On détermine la direction dans laquelle on vient de tirer et on regarde si le plus petit bateau rentre en face dans cette direction

Le plus petit bateau à trouver est de taille 4 et la première case touchée est la case (3,0).

Le plus petit bateau de taille 4 rentre horizontalement

	0	1	2	3	4	5	6	7	8	9
0	O			X						

Le plus petit bateau à trouver est de taille 4 et la première case touchée est la case (3,0). Nous venons de manquer la case (4,0).

Le plus petit bateau de taille 4 ne rentre plus horizontalement

	0	1	2	3	4	5	6	7	8	9
0	O			X	O					

```
Ordi.update_queue_manque(self) :
```

taille\_min est la plus petite taille de bateau restant

delta =

```
(case_courante[0]-case_touchee[0],  
 case_courante[1]-case_touchee[1])
```

direction =

```
(abs(case_courante[0]-case_touchee[0]),  
 abs(case_courante[1]-case_touchee[1]))
```

case\_face =

```
(case_touchee[0]-delta[0],  
 case_touchee[1]-delta[1])
```

Si l'espace vide sur case\_face dans direction < taille\_min-1 :

Enlever case\_face de la file d'attente



# Tirs suivants

À chaque tir touché suivant, on ajoute la case adjacente à la file d'attente si elle est vide et est dans la grille

# Tirs suivants

À chaque tir touché suivant, on ajoute la case adjacente à la file d'attente si elle est vide et est dans la grille

Le bateau est coulé lorsque la file d'attente est vide ou on a touché autant de cases que le plus grand bateau restant

On repart alors dans une phase de tirs en aveugles

On repart alors dans une phase de tirs en aveugles

On sait qu'un bateau vient d'être coulé si la file d'attente est vide et la liste des cases touchées ne l'est pas

On repart alors dans une phase de tirs en aveugles

On sait qu'un bateau vient d'être coulé si la file d'attente est vide et la liste des cases touchées ne l'est pas

Dans ce cas on enlève le bateau de la liste et on élimine ses case adjacentes

# Plan

- 1 Les structures de données
  - Constantes de direction
  - Structure de la grille
  - Structure des joueurs
- 2 Algorithme de résolution
  - Principe général
  - Tirs en aveugle
  - Tirs ciblés
  - Algorithme complet de résolution
  - Étude statistique
- 3 Le reste dont je n'aurai pas le temps de parler
- 4 Conclusion

La file d'attente est une liste vide

liste\_touches est une liste vide

Tant que le grille n'est pas résolue :

    Si la file d'attente est vide (tir en aveugle) :

        Si liste\_touches n'est pas vide :

            On enlève le bateau de taille len(liste\_touches)

            On élimine les cases adjacentes à celles de liste\_touches

            On vide liste\_touches

        On élimine les zones trop petites

        case\_courante reçoit une case en aveugle (suivant le niveau)

Sinon (tir ciblé) :

case\_courante reçoit le premier élément de la file d'attente

On enlève cette case de la file d'attente

On tire sur case\_courante

Si on a touché :

Si liste\_touchees est vide (1ère case du bateau) :

On ajoute case\_courante dans liste\_touchees

case\_courante → case\_touchee

On ajoute ses cases adjacentes dans la file d'attente

(en testant également les directions impossibles éventuelles)



Sinon :

Si  $\text{len}(\text{liste\_touches}) == 1$  (2<sup>ème</sup> case du bateau) :

On détecte la direction du bateau

On met à jour la file d'attente

(avec la case adjacente à `case_courante` dans la bonne direction si elle est vide)

Si le bateau touché est le plus grand restant :

On vide la file d'attente

Sinon : (on a manqué)

Si `len(liste_touchees) == 1` :

On met à jour la file d'attente

(on élimine éventuellement la case en face de `case_touchee`)

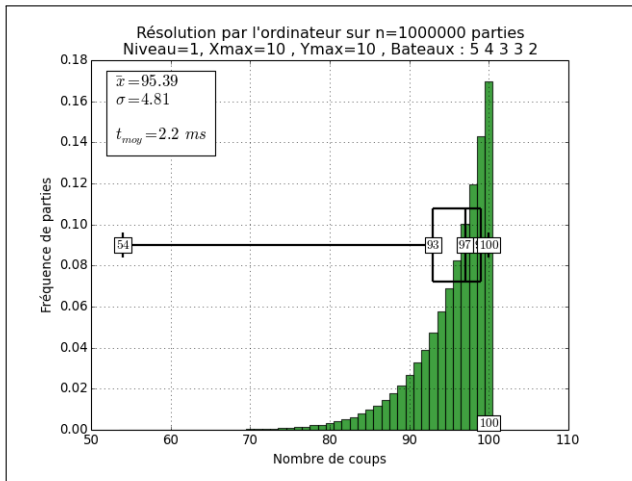
On affiche enfin le nombre de coups

# Plan

- 1 Les structures de données
  - Constantes de direction
  - Structure de la grille
  - Structure des joueurs
- 2 Algorithme de résolution
  - Principe général
  - Tirs en aveugle
  - Tirs ciblés
  - Algorithme complet de résolution
  - Étude statistique
- 3 Le reste dont je n'aurai pas le temps de parler
- 4 Conclusion

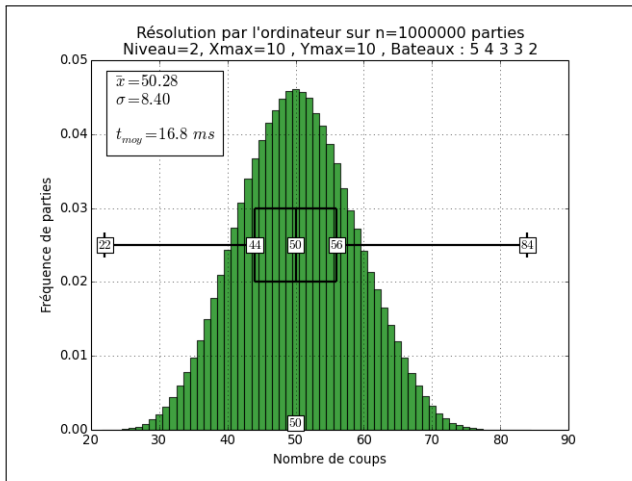
# Niveau 1 : que des tirs aléatoires

Résultats sur  $n = 1\,000\,000$  parties :



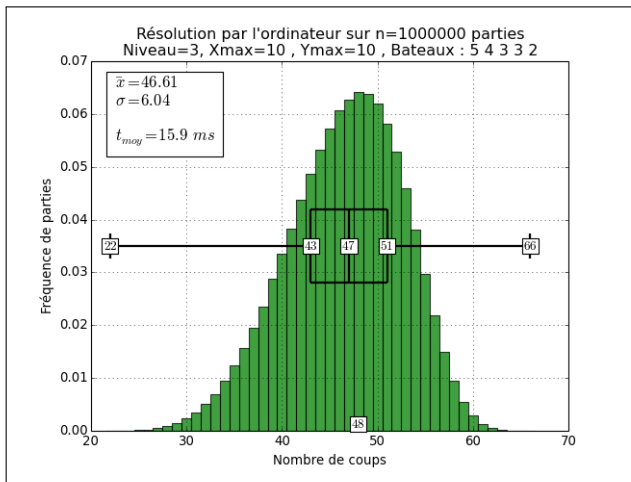
## Niveau 2 : tirs aléatoires + file

Résultats sur  $n = 1\,000\,000$  parties :



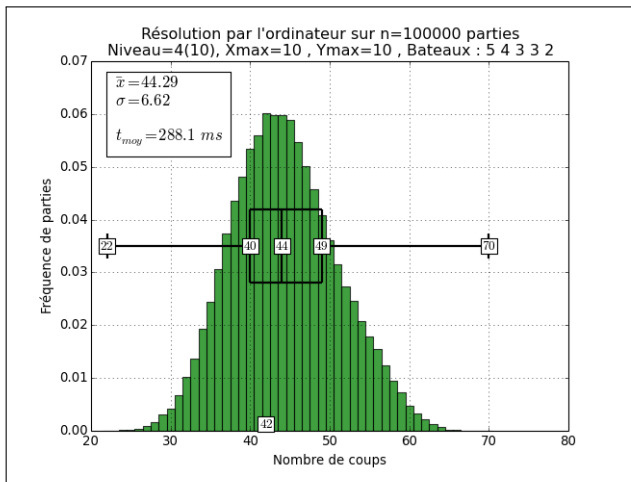
## Niveau 3 : cases noires + file

Résultats sur  $n = 1\,000\,000$  parties :



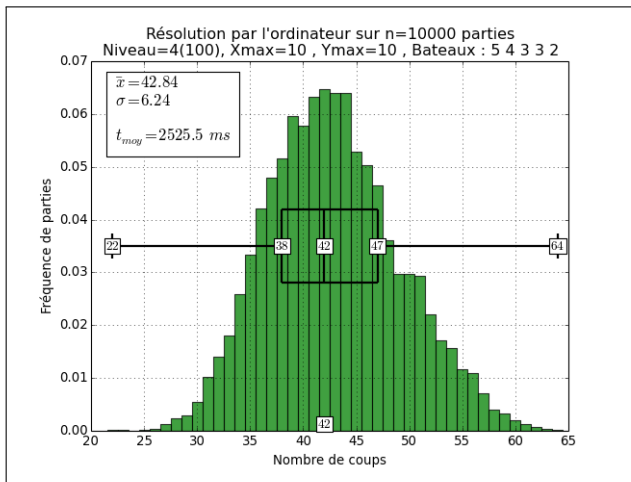
## Niveau 4(10) : échantillons de taille 10

Résultats sur  $n = 100\,000$  parties :



## Niveau 4(100) : échantillons de taille 100

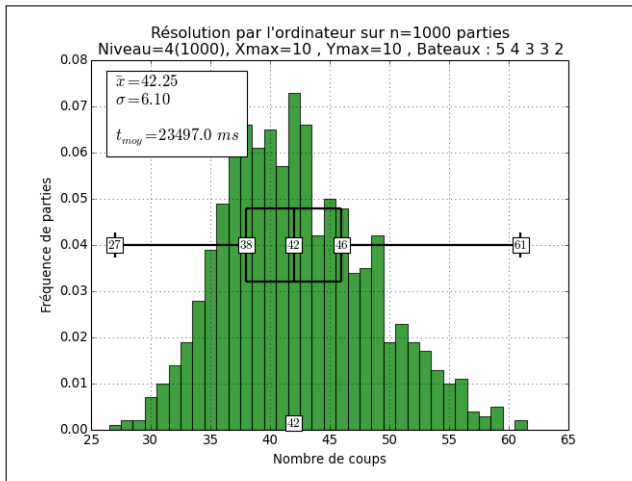
Résultats sur  $n = 10000$  parties :





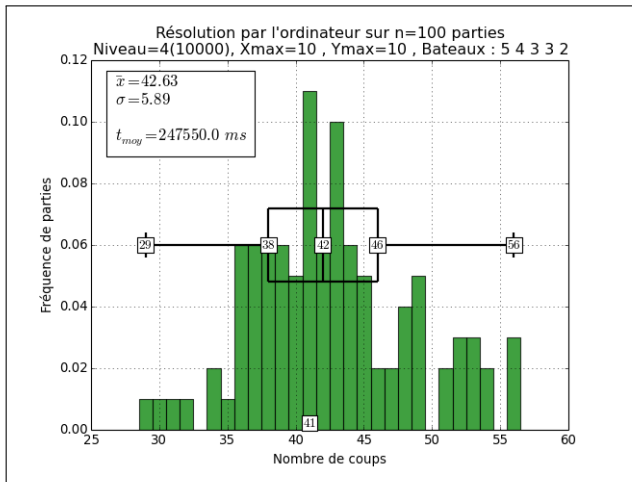
## Niveau 4(1000) : échantillons de taille 1000

Résultats sur  $n = 1\,000$  parties :



## Niveau 4(10000) : échantillons de taille 10000

Résultats sur  $n = 100$  parties :



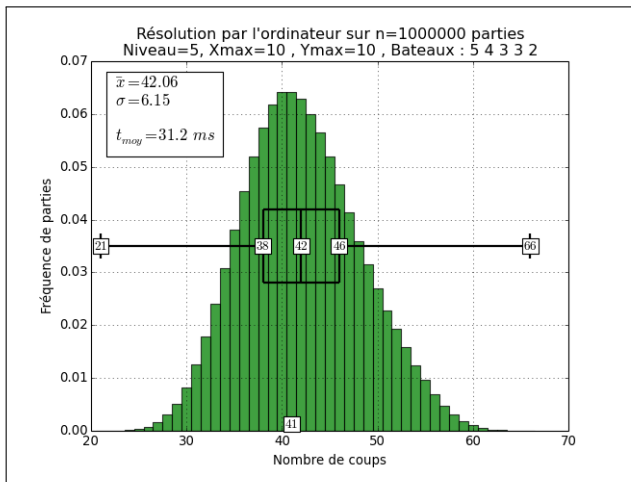
## Conclusion niveau 4

$n^{\circ}$	1	2	3	4
nb_echantillons	10	100	1 000	10 000
$n$	100 000	10 000	1 000	100
$\bar{x}$	44,29	42,84	42,25	42,63
$\sigma$	6,62	6,24	6,10	5,89
$t_{moy}$ (en secondes)	0,29	2,5	23,5	247

n°	nb_echantillons	Intervalle de confiance de $\mu$
1	10	[44,25 ; 44,33]
2	100	[42,72 ; 42,96]
3	1 000	[41,87 ; 42,63]
4	10 000	[41,48 ; 43,78]

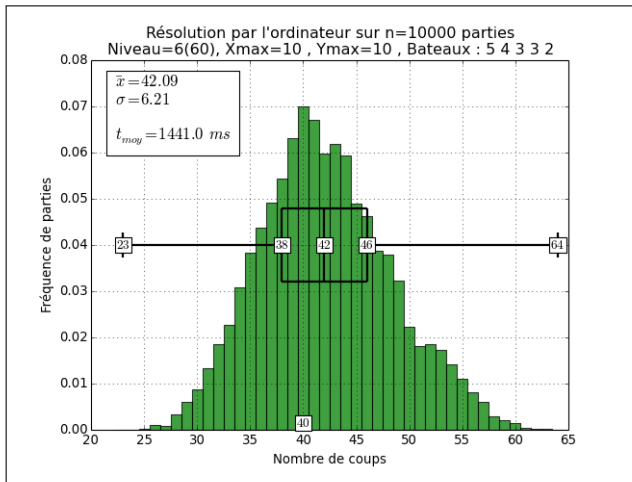
## Niveau 5 : nombre local de bateaux

Résultats sur  $n = 1\,000\,000$  parties :



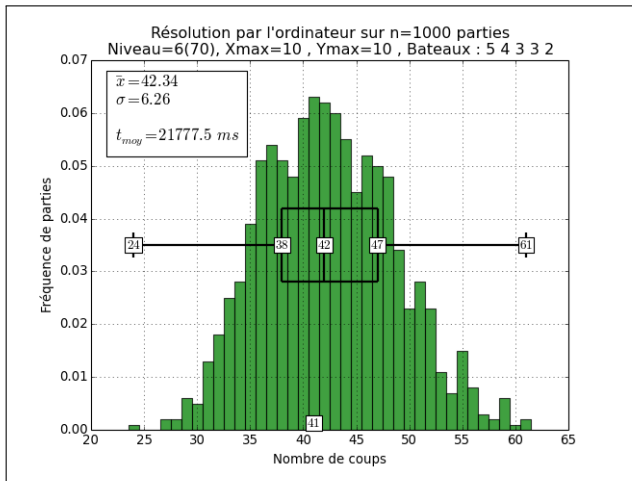
## Niveau 6(60) : tous les arrangements, seuil=60

Résultats sur  $n = 10\,000$  parties :



## Niveau 6(70) : tous les arrangements, seuil=70

Résultats sur  $n = 1\,000$  parties :



## Conclusion niveau 6

n°	Niveau	$n$	$\bar{x}$	$\sigma$	Intervalle de confiance
1	5	1 000 000	42,06	6,15	[42,05 ; 42,07]
2	6(60)	10 000	42,09	6,21	[41,97 ; 42,21]
3	6(70)	1 000	42,34	6,26	[41,95 ; 42,73]



# Plan

- 1 Les structures de données
  - Constantes de direction
  - Structure de la grille
  - Structure des joueurs
- 2 Algorithme de résolution
  - Principe général
  - Tirs en aveugle
  - Tirs ciblés
  - Algorithme complet de résolution
  - Étude statistique
- 3 Le reste dont je n'aurai pas le temps de parler
- 4 Conclusion

# Interface console

- Menu de choix de mode de jeu

# Interface console

- Menu de choix de mode de jeu
- Étude statistique des algorithmes

## Interface console

- Menu de choix de mode de jeu
- Étude statistique des algorithmes
- Affichages de la grille, et surlignage des bateaux

# Interface web

- Script cgi

# Interface web

- Script cgi
- Gestion d'un canvas et de la souris en JavaScript

# Interface web

- Script cgi
- Gestion d'un canvas et de la souris en JavaScript
- Sauvegarde des paramètres de la partie

# Interface en tkinter

- Différentes fenêtres



# Interface en tkinter

- Différentes fenêtres
- Placement des bateaux

# Interface en tkinter

- Différentes fenêtres
- Placement des bateaux
- Illustration graphique de l'algorithme de résolution

# Plan

- 1 Les structures de données
  - Constantes de direction
  - Structure de la grille
  - Structure des joueurs
- 2 Algorithme de résolution
  - Principe général
  - Tirs en aveugle
  - Tirs ciblés
  - Algorithme complet de résolution
  - Étude statistique
- 3 Le reste dont je n'aurai pas le temps de parler
- 4 Conclusion

Un projet passionnant et prenant

Un projet passionnant et prenant

Réalisé entièrement en solo

Un projet passionnant et prenant

Réalisé entièrement en solo

Un travail conséquent :

Un projet passionnant et prenant

Réalisé entièrement en solo

Un travail conséquent :

- 4 mois et plus de 200 heures de travail intensif

Un projet passionnant et prenant

Réalisé entièrement en solo

Un travail conséquent :

- 4 mois et plus de 200 heures de travail intensif
- 3500 lignes de code Python



Un projet passionnant et prenant

Réalisé entièrement en solo

Un travail conséquent :

- 4 mois et plus de 200 heures de travail intensif
- 3500 lignes de code Python
- 2000 lignes de  $\text{\LaTeX}$  pour le rapport

Un projet passionnant et prenant

Réalisé entièrement en solo

Un travail conséquent :

- 4 mois et plus de 200 heures de travail intensif
- 3500 lignes de code Python
- 2000 lignes de  $\text{\LaTeX}$  pour le rapport
- 2000 lignes de  $\text{\LaTeX}$  pour le diaporama

Un projet passionnant et prenant

Réalisé entièrement en solo

Un travail conséquent :

- 4 mois et plus de 200 heures de travail intensif
- 3500 lignes de code Python
- 2000 lignes de  $\text{\LaTeX}$  pour le rapport
- 2000 lignes de  $\text{\LaTeX}$  pour le diaporama

Et quelques nuits blanches de réflexion...

# Merci de votre attention

Passons aux questions...