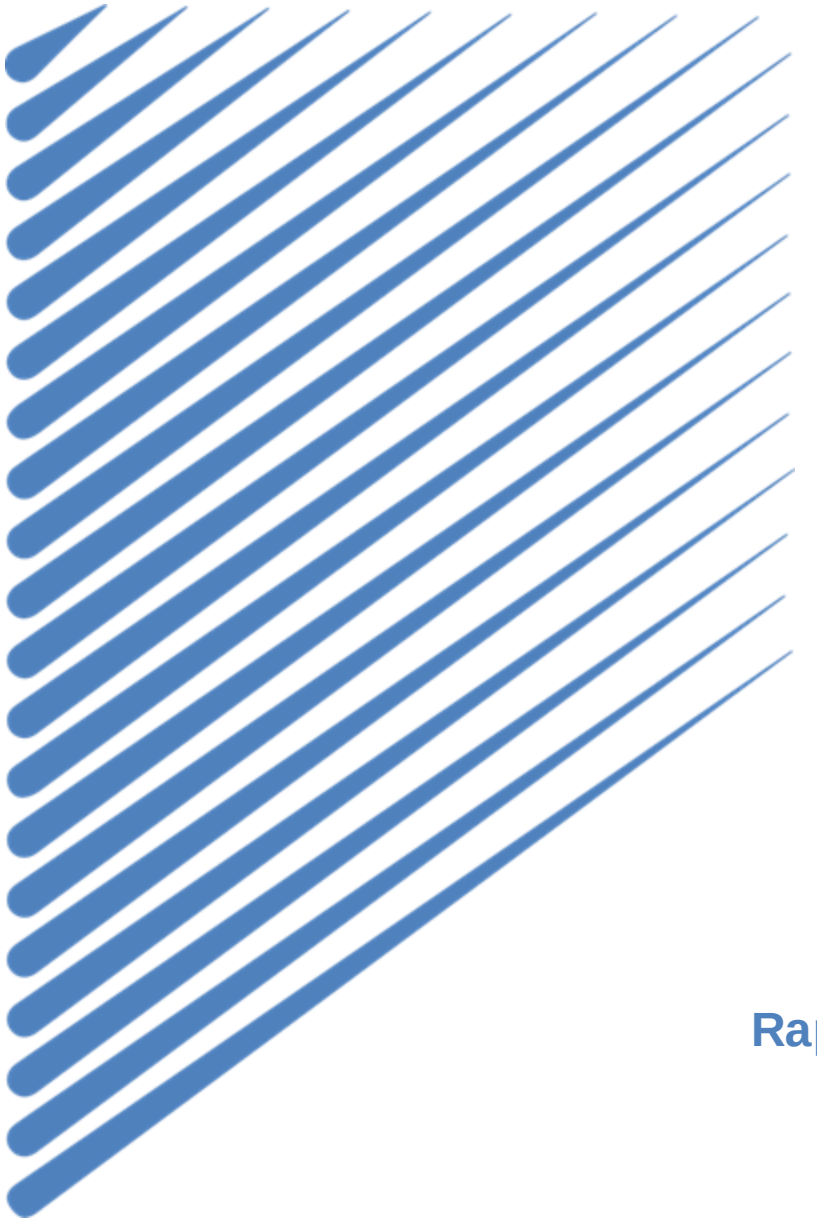


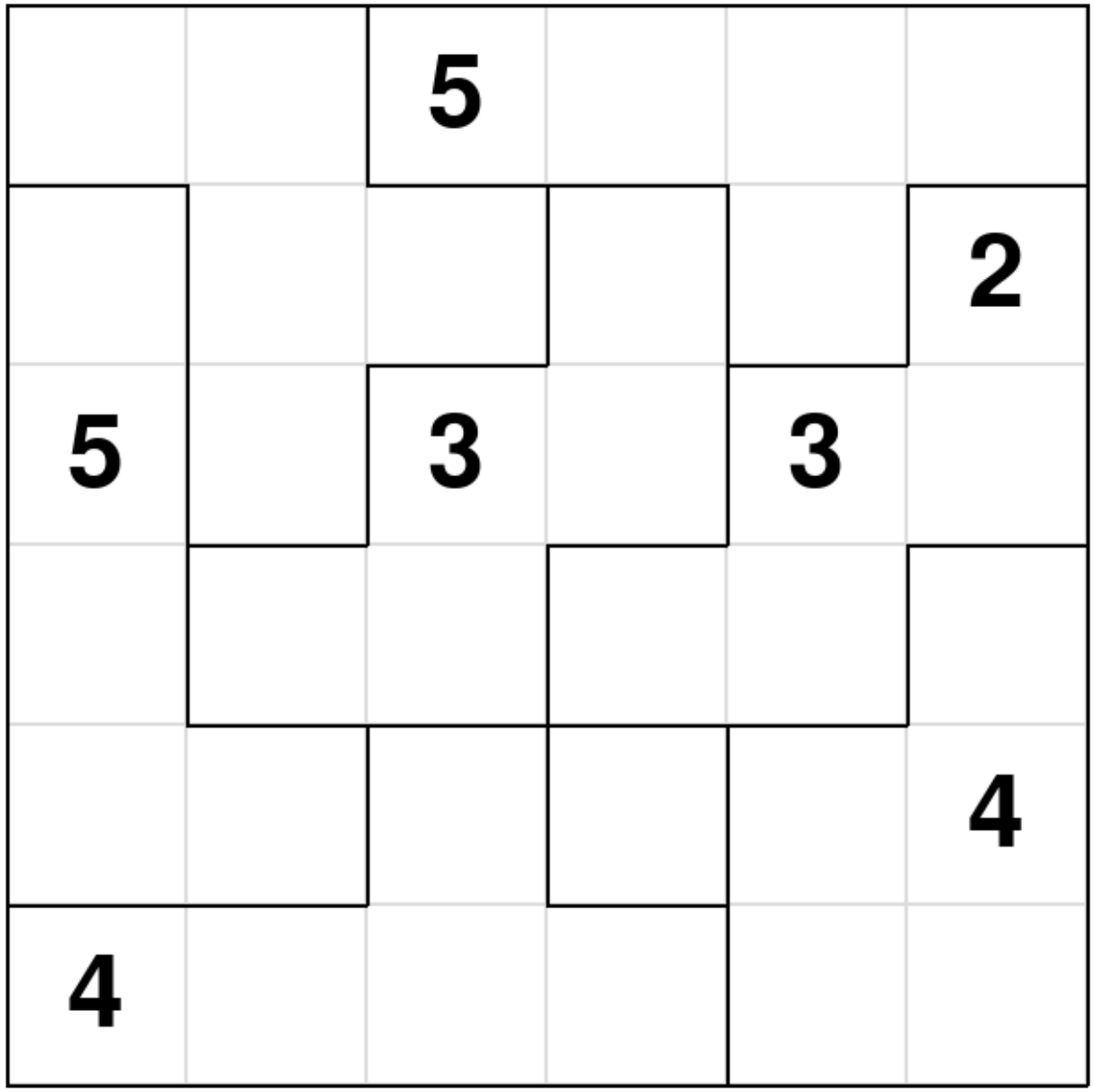
MINES PARIS | PSL

Soutenance projet Kemaru

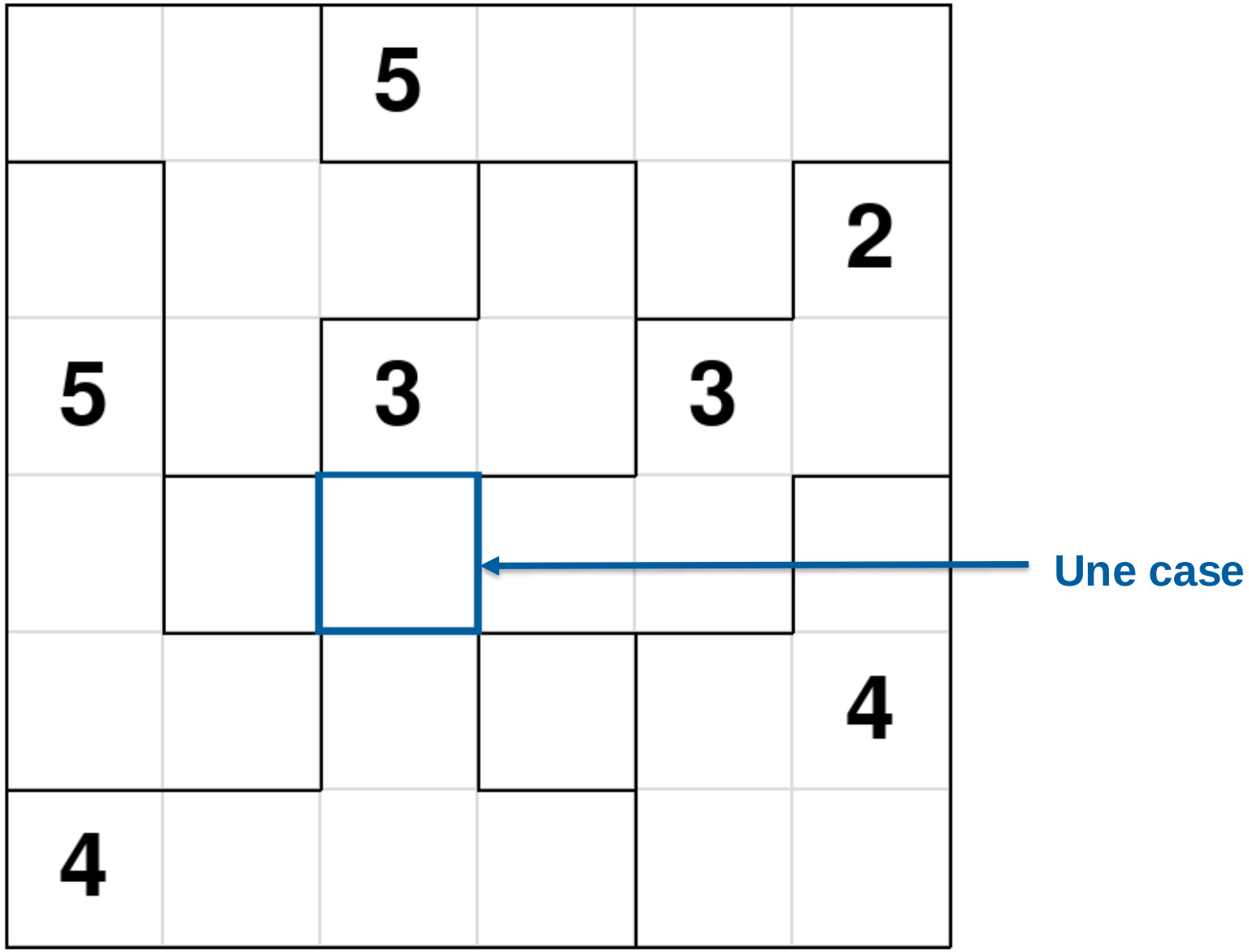
Raphaël Oculi, Gabriel Party, Raphaël Poux, Mathis Verdan
Encadrant : **Nikolas Stott**

Sommaire

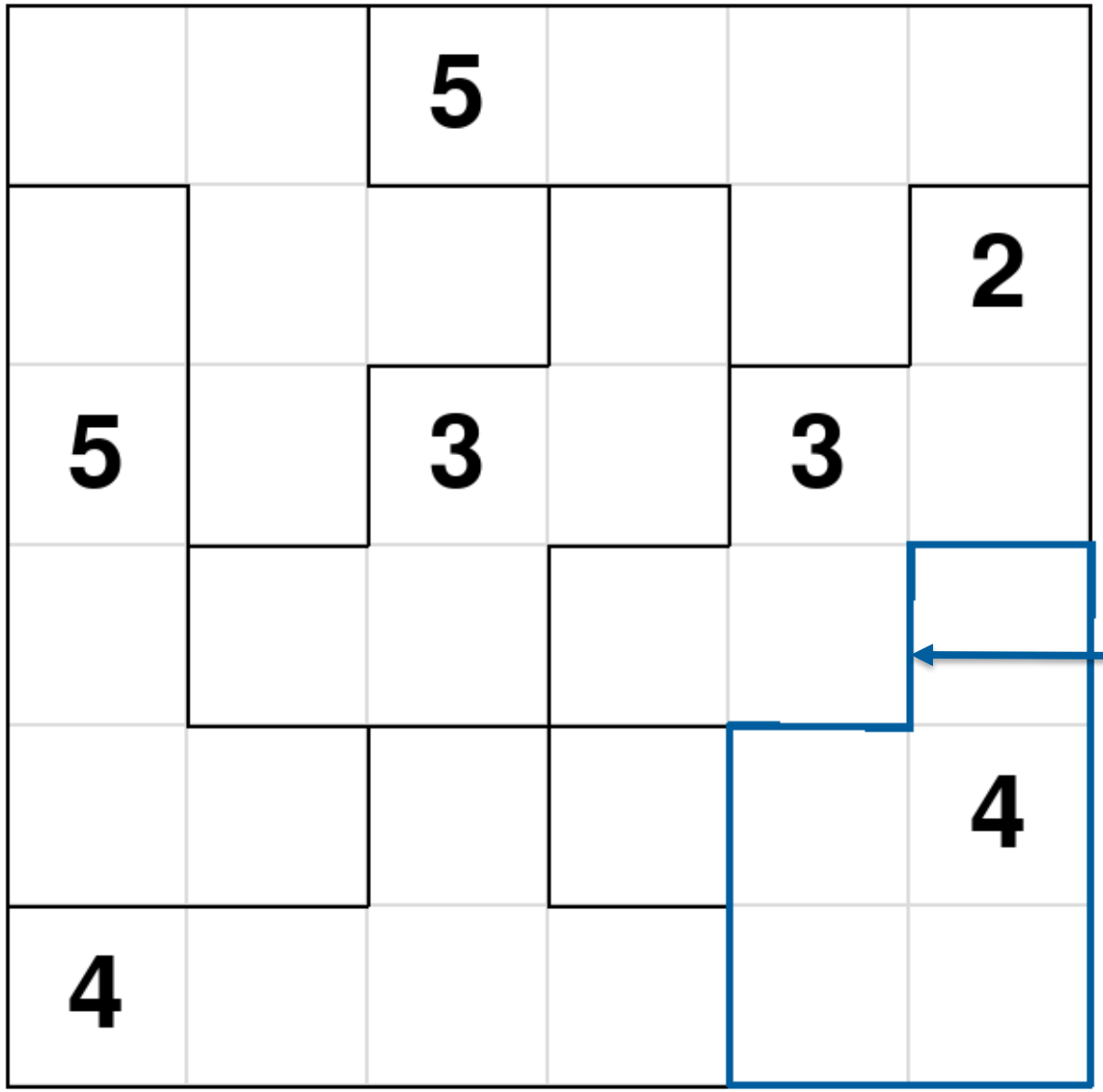
I-a) - Les règles du Kemaru



I-a) - Les règles du Kemaru



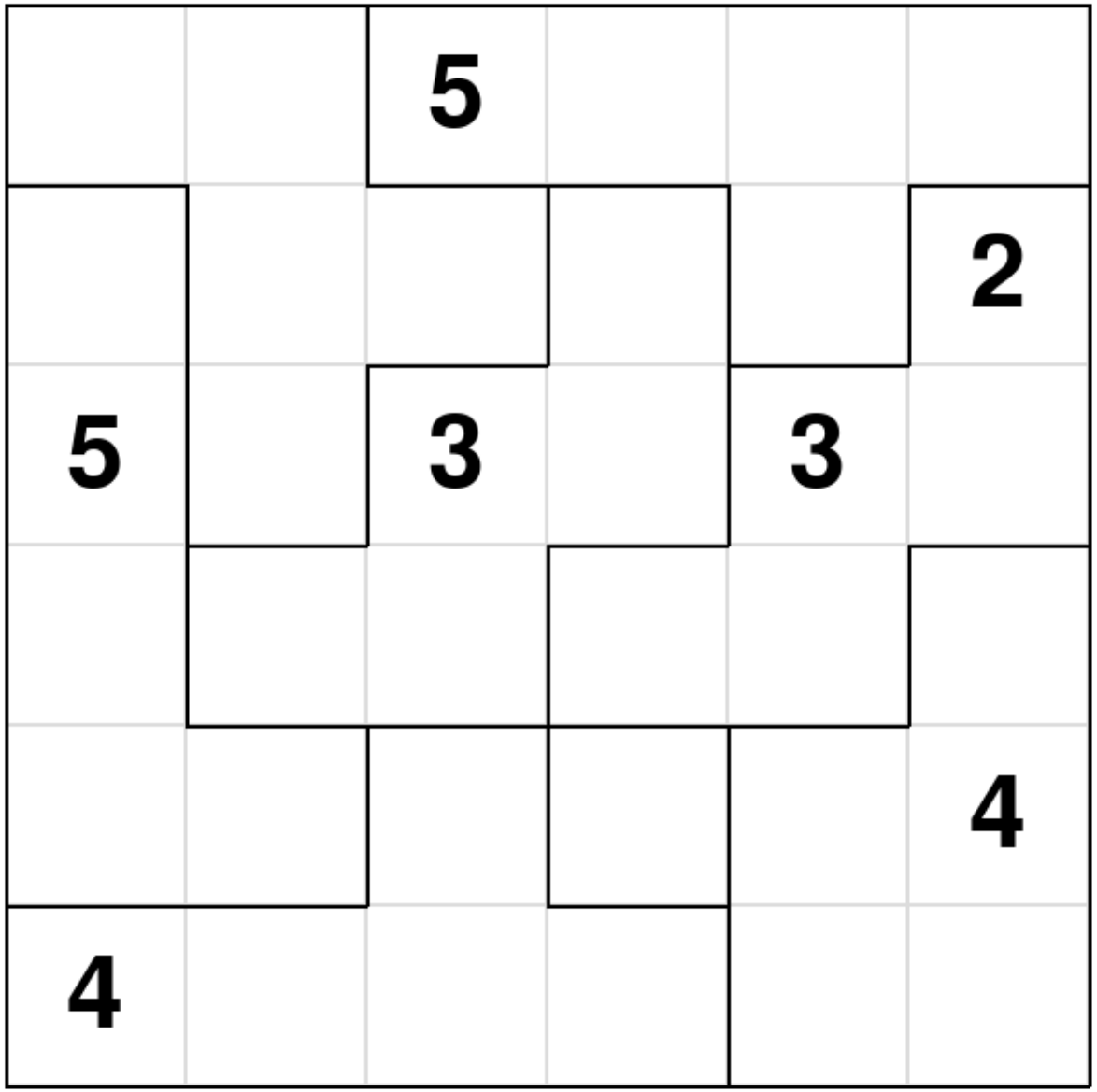
I-a) - Les règles du Kemaru



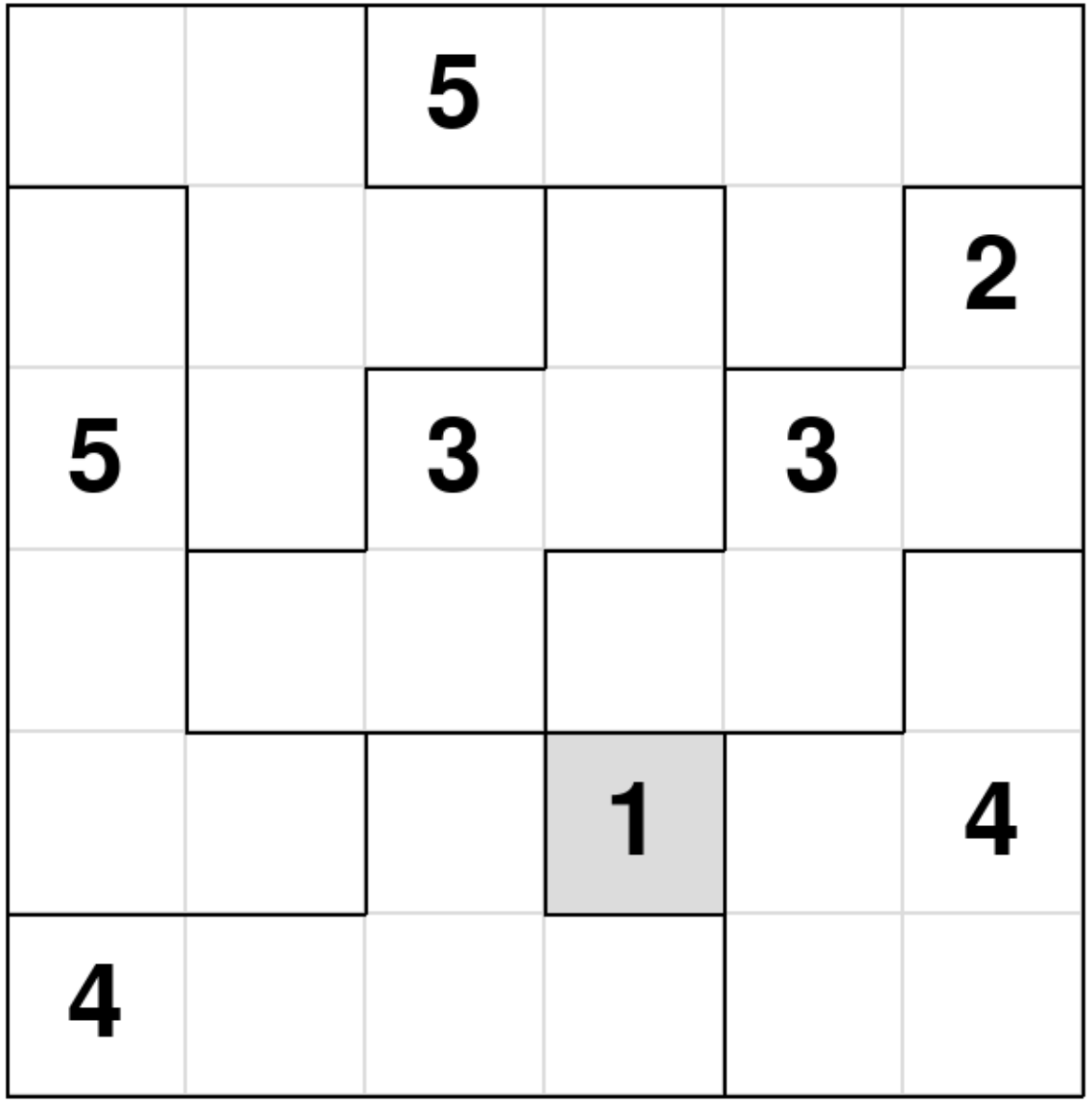
I-a) - Les règles du Kemaru

Règles :

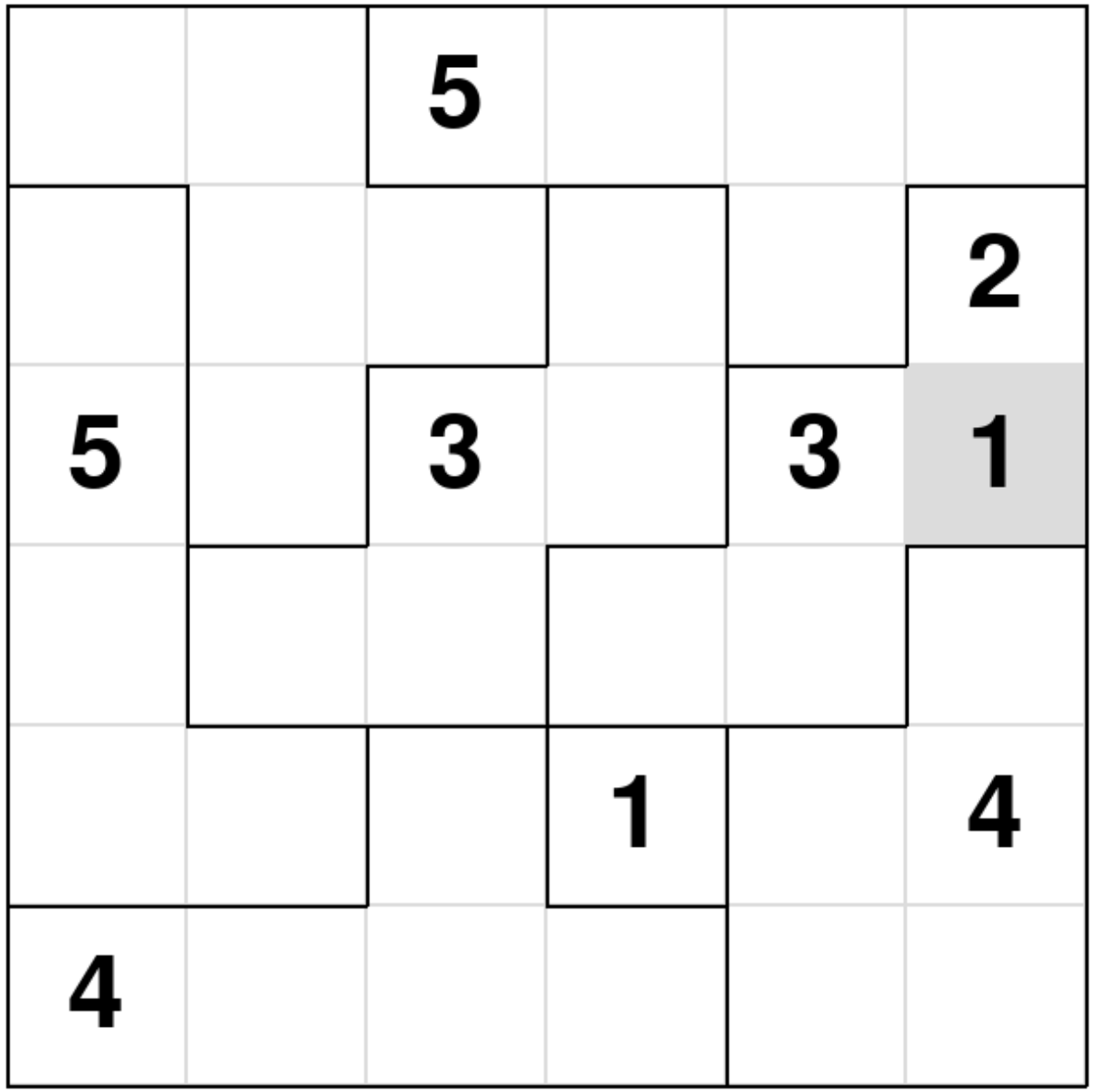
- Interdiction d'avoir deux chiffres identiques à côté
- Une cage de taille N doit contenir les chiffres de 1 à N



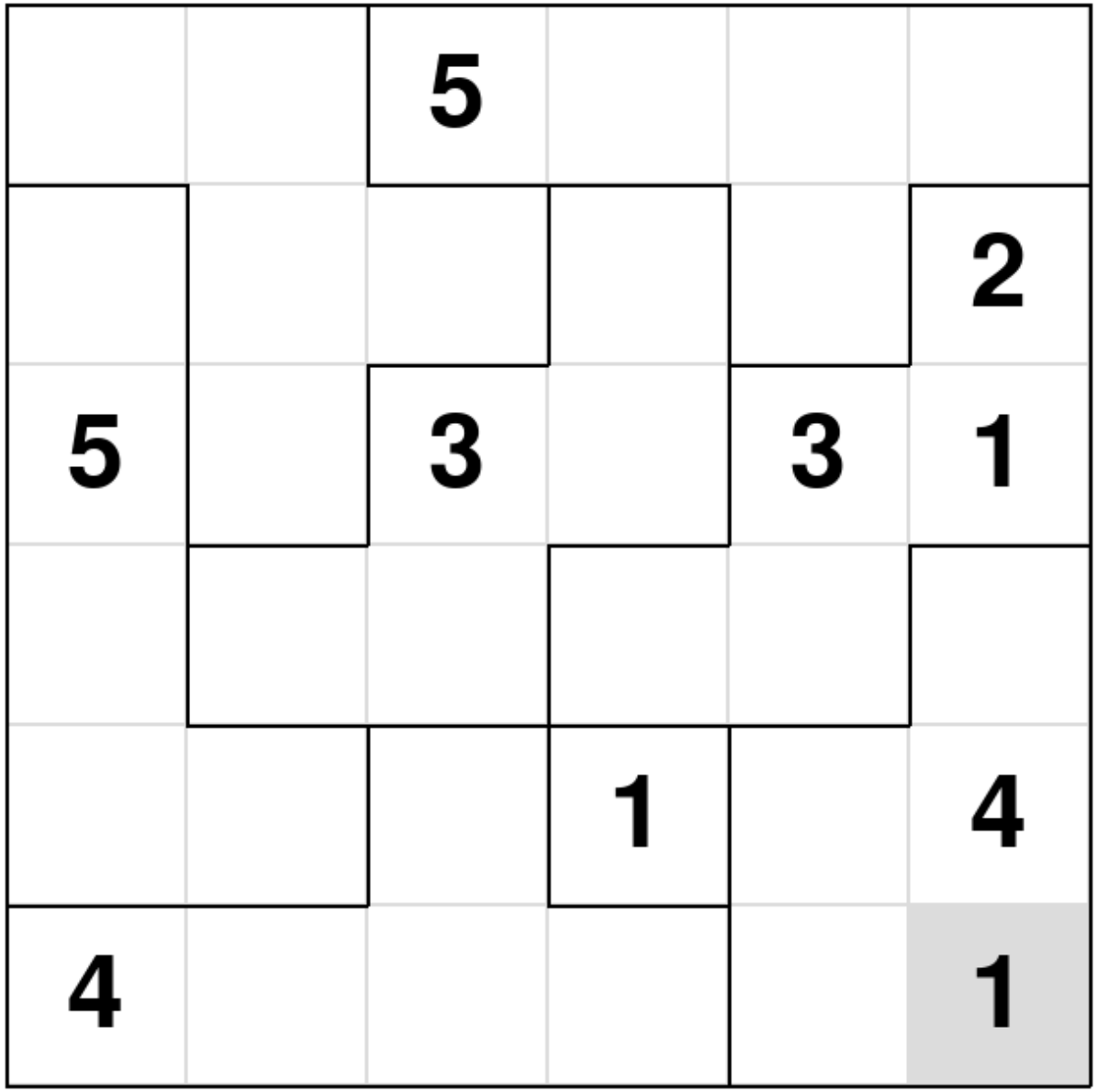
I-a) - Les règles du Kemaru



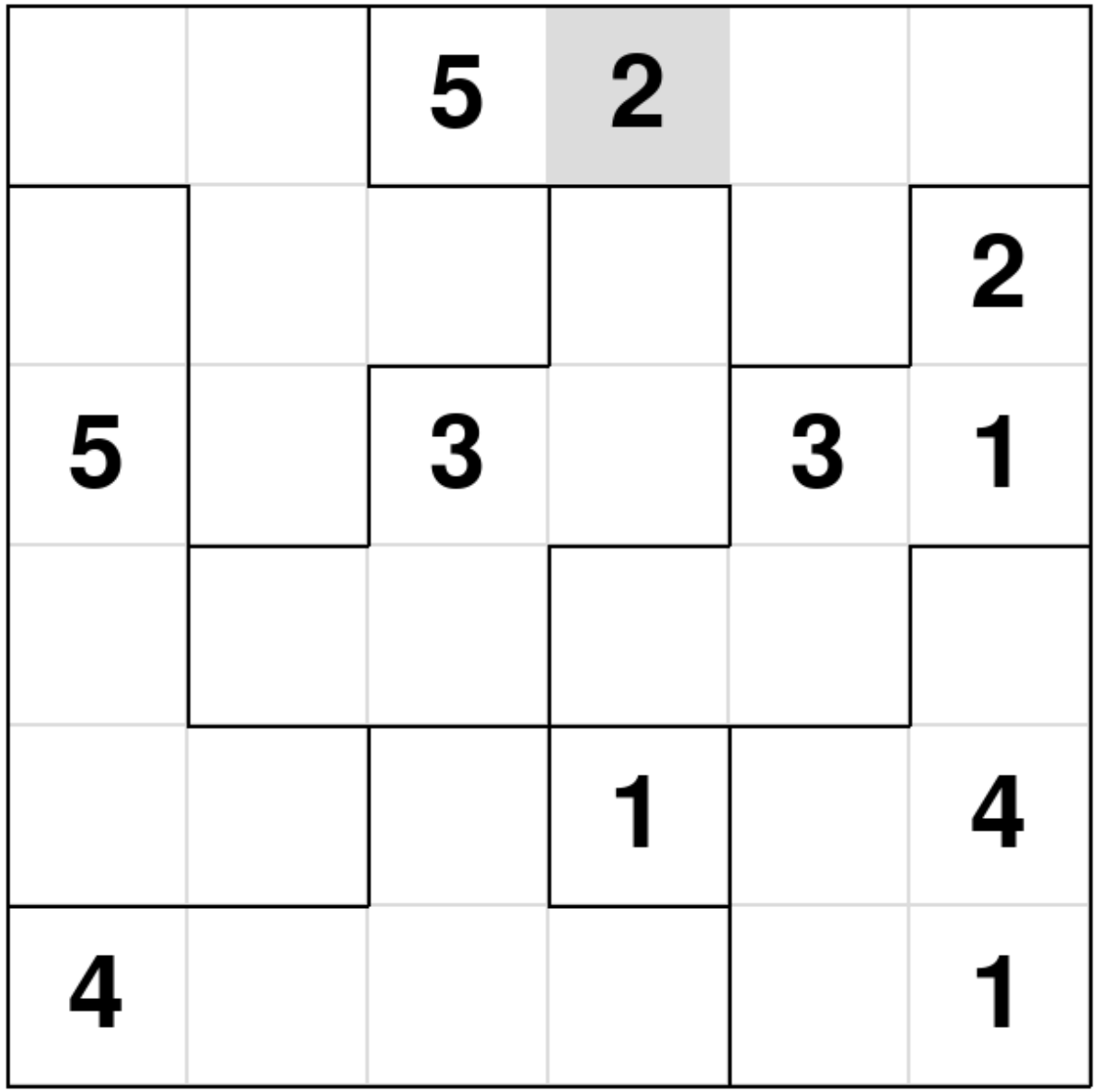
I-a) - Les règles du Kemaru



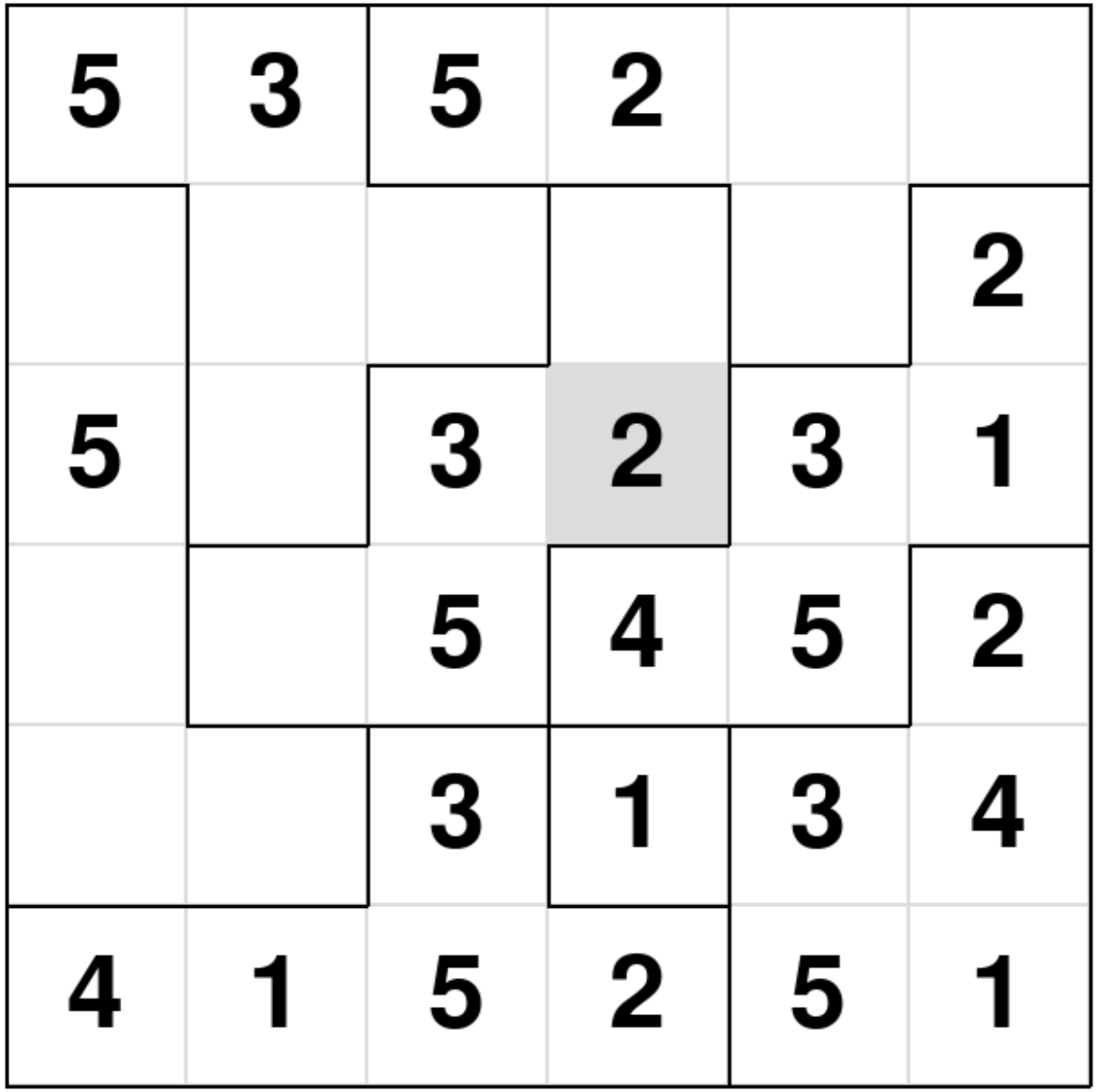
I-a) - Les règles du Kemaru



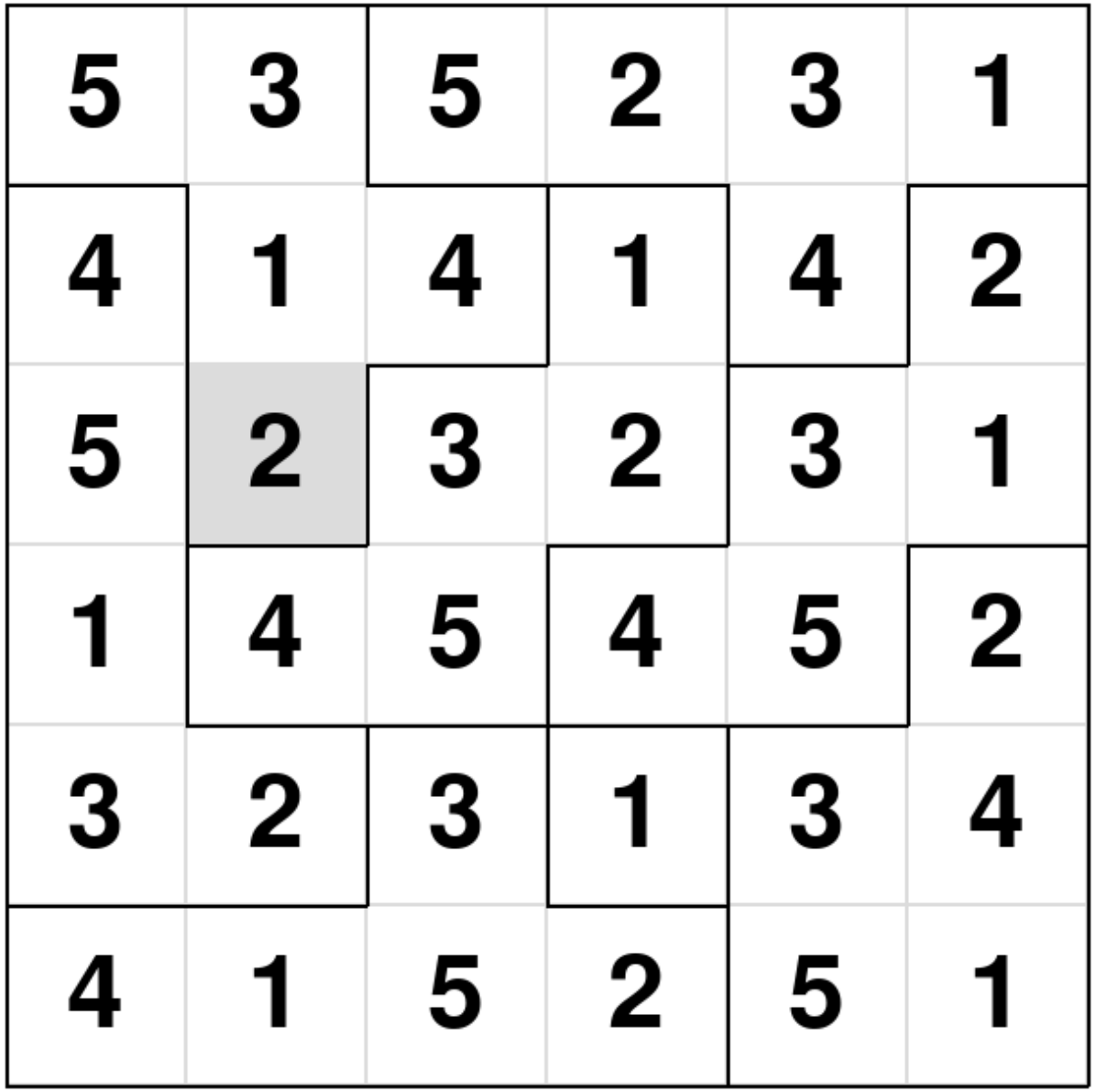
I-a) - Les règles du Kemaru



I-a) - Les règles du Kemaru



I-a) - Les règles du Kemaru



I-b) Objectifs du projet

- Résoudre toutes les grilles de Kemaru

-Trouver la solution optimale avec le moindre coût et expliquer comment y arriver

-La représenter avec une interface graphique

-Résoudre le problème avec la plus faible complexité

I-c) Nos choix et répartition du travail

Nos choix :

-Implémentation d'un algorithme de résolution basé sur une recherche de solution de niveau 0, 1 et 2

-Implémentation d'un algorithme de Dijkstra optimisé

-Interface graphique implémentée avec la librairie Pygame

Répartition du travail :

-Implémentation de l'algorithme de résolution pendant les séances de travail en utilisant Liveshare

-Division en deux groupes de travail pour le plus court chemin de résolution et l'interface graphique

-Utilisation de GitHub pour une évolution simultanée des 2 groupes sur deux branches différentes

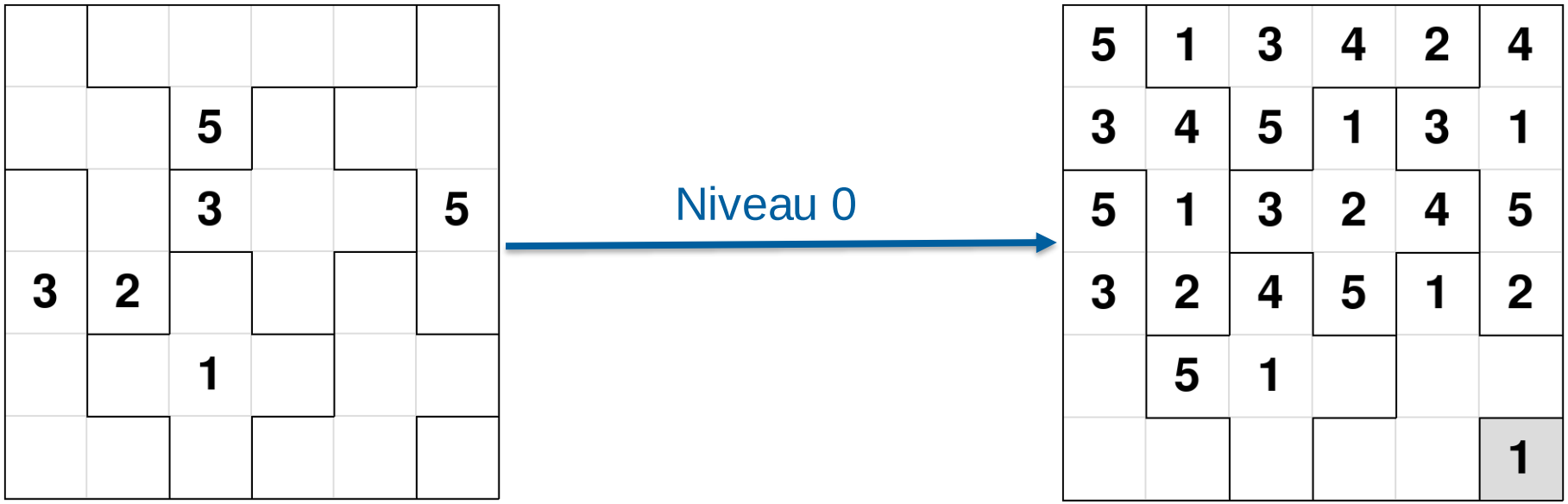
```
* e71c576 (origin/tests) Add level 2 to tree search: don't do it when level 1 su
ffices, and commit to a new search tree after doing a level 2, rather than keepi
ng the old one around
* de4194f Fixup iteration variable
* ff66742 Compute hash and filter nodes based on hash
*   b6a219a Merge branch 'main' into tests
|\
| * 848f85a (HEAD -> main, origin/main) remove dico_est_trouve from level 0
* | 672c76b Remove cages_valeurs and recompute it on the fly when needed
* | 6a5b187 don't do level 0 within the tree search + don't add leaves that don'
t change the amount of information
* | 0679e50 add test instance and comment out some code
|/
* 6f6eb0d en fait je mété trompé
* 0ddbee9 ajout de la recherche du plus court chemin
* b289050 démo de la résolution 1.690
* 5e9d109 ajout  du niveau 1 interface
* 08844e6 Add files via upload
* 89abcd2 plus court chemin de résolution
| * a4fe939 (origin/test_interface, test_interface) fin de séance(et presque bie
n pour niveau 0)
| * 5eafb3e don't follow
| * 3bfeee6 interface sommaire+ peu de détail des étapes
|/
* fe4068c impélmentation des couts
* 783eb32 niveau 1 qui marche (lentement)
* e4c115c niveau 1 qui ne marche pas trop
* 3381880 session collaborative 1
* 0ee3ecd ajout du solveur
* f3b2afa Add files via upload
* 72c736a Delete Photos directory
* 52cab6a Add files via upload
* c69896f Add files via upload
```

II-a) Implémentation des différents raisonnements et tests

Algorithme de résolution d'une grille :

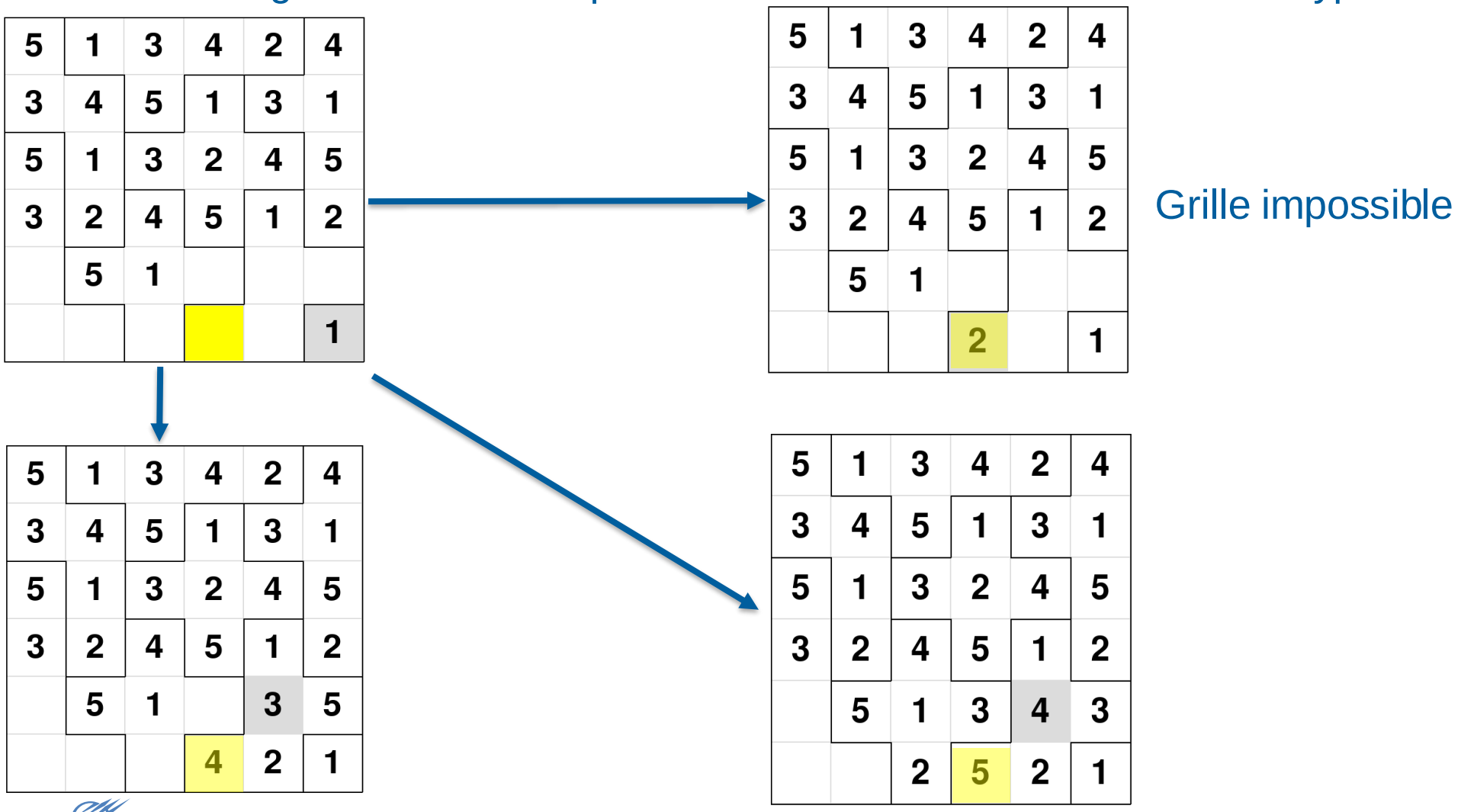
- Grille = tableau 3D où grille[i, j] = [valeur dans la case, numéro de la cage]
- On utilise de plus un dictionnaire qui donne les valeurs possibles dans chaque case
- On applique ensuite des algorithmes de "niveau 0", "niveau 1 ", "niveau 2" …
- On met à jour la grille et le dictionnaire au fur et à mesure

Niveau 0 : application des règles du jeu à l'ensemble de la grille



II-a) Implémentation des différents raisonnements et tests

Niveau 1 : on regarde les valeurs possibles dans une case et on fait des hypothèses



Grille impossible

II-a) Implémentation des différents raisonnements et tests

Niveau 1 : on regarde les valeurs possibles dans une case et on fait des hypothèses

5	1	3	4	2	4
3	4	5	1	3	1
5	1	3	2	4	5
3	2	4	5	1	2
	5	1			
					1

5	1	3	4	2	4
3	4	5	1	3	1
5	1	3	2	4	5
3	2	4	5	1	2
	5	1			
			2		1

Grille impossible, de même en supposant que la case vaut 3

5	1	3	4	2	4
3	4	5	1	3	1
5	1	3	2	4	5
3	2	4	5	1	2
	5	1		3	5
			4	2	1

5	1	3	4	2	4
3	4	5	1	3	1
5	1	3	2	4	5
3	2	4	5	1	2
	5	1	3	4	3
		2	5	2	1

Dans tous les cas, le 2 est à la même place !

II-a) Implémentation des différents raisonnements et tests

Niveau 2 : pareil que le niveau 1 mais on suppose 2 cases.

On pourrait généraliser jusqu'au niveau n mais cela est inutile car trop compliqué pour un humain.

Tests des différents algorithmes :

Nombre total de grilles testées	23 580		
Nombre de grilles résolues	Niveau 0	Niveau 1	Niveau 2
23 580	1 559	21 661	360

II-b) Implémentation d'un algorithme de plus court chemin

Initialisation:
On applique le niveau 0 et tous les niveaux 1 et 2 possibles à partir de la grille initiale et on obtient un arbre de hauteur 1 et la racine a 1+n+n*(n-1)/2 où n est le nombre de cases manquantes dans la grille initiale

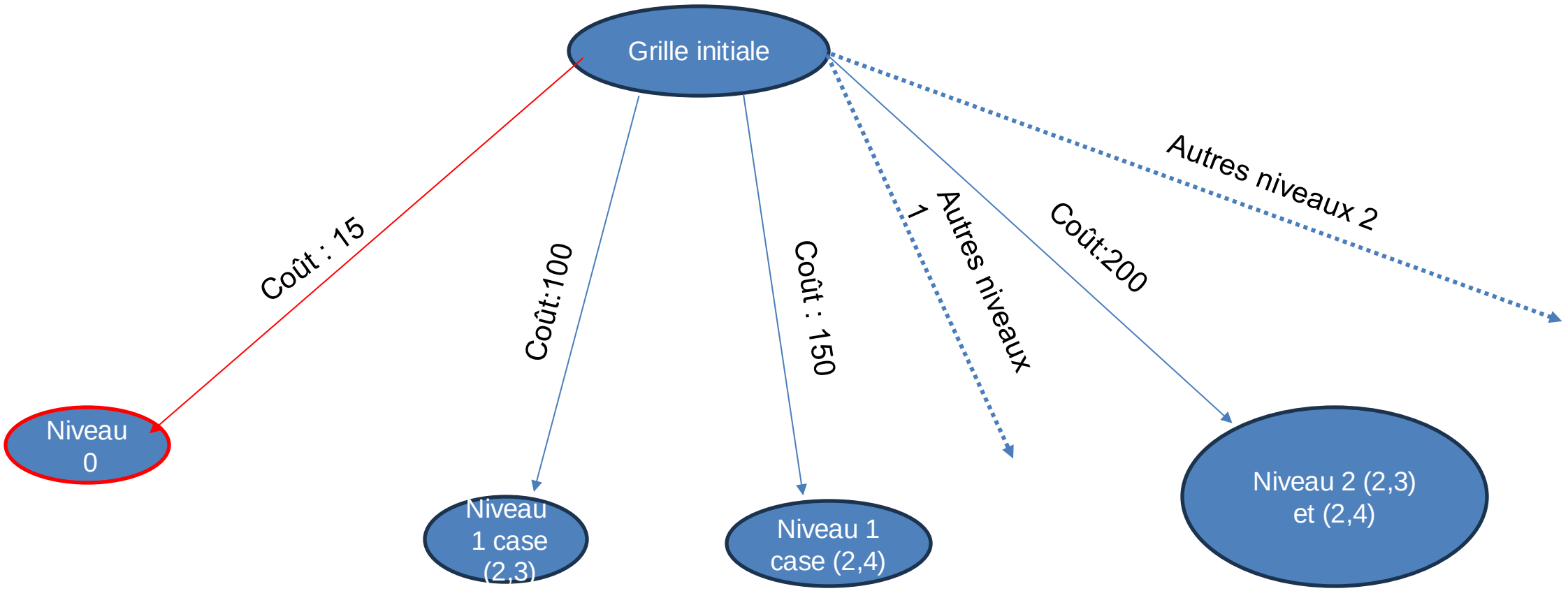
Hérédité:
on prend le chemin avec le plus petit coût:

-Si la grille est complète on s'arrête et on renvoie cette grille

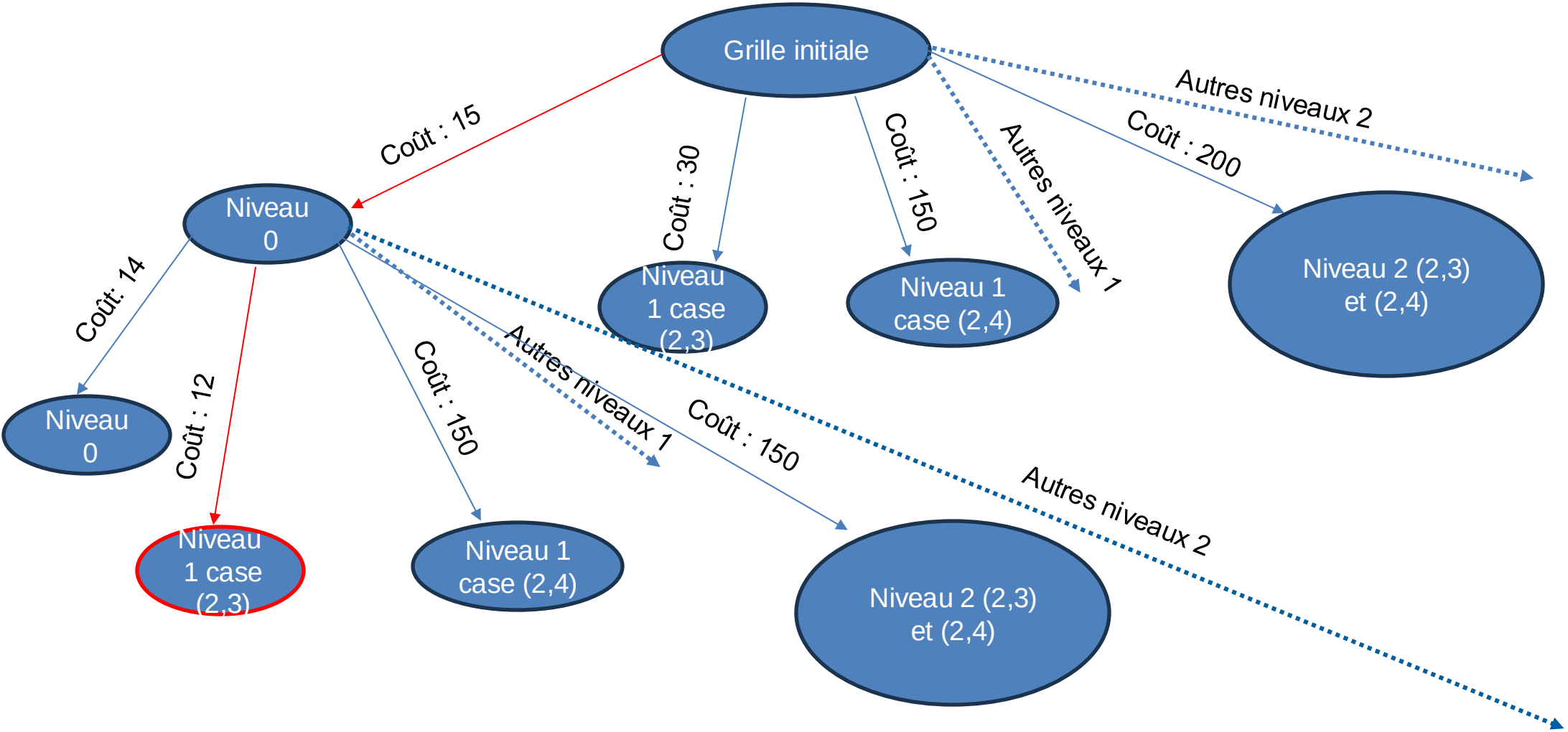
-Sinon on teste tous les niveaux possibles et on complète l'arbre: on réalise tous les niveaux 1,2 et 0 à partir de ce niveau et ce nœud a donc 1+d + d*(d-1)/2 où d est le nombre de cases manquantes dans la grille sur laquelle on fait l'hérédité

.

II-b) Implémentation d'un algorithme de plus court chemin



II-b) Implémentation d'un algorithme de plus court chemin



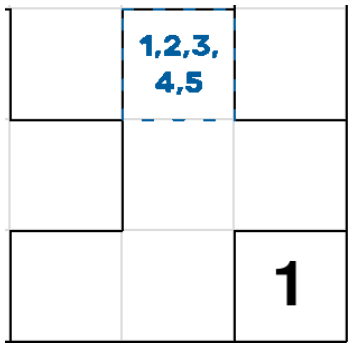
II-c) Difficultés et nouvel algorithme

Difficultés :

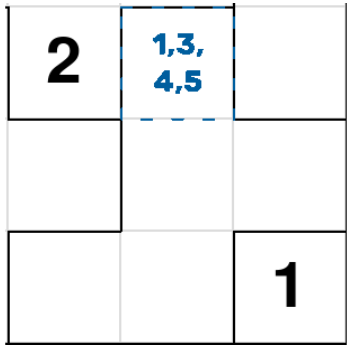
- L'algorithme présenté peut être (très) long : pour trouver le plus court chemin, il faut d'abord parcourir tous les chemins plus courts
- Une étape de faible coût ne nous apprend pas forcément grand-chose
- Plus l'arbre des possibilités est grand, plus il grandit vite

Autre méthode :

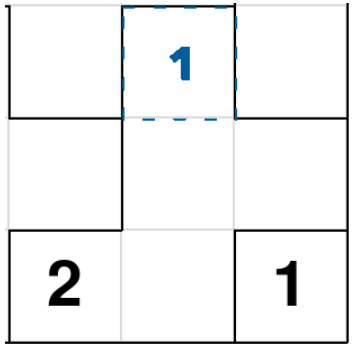
- Quantité d'informations = nombre de valeurs impossibles dans la grille
- On reprend le même algorithme mais on maximise la quantité de nouvelles informations de chaque étape
- Cela privilégie des coups stratégiques et améliore grandement la rapidité



Informations : 0



Informations : 1



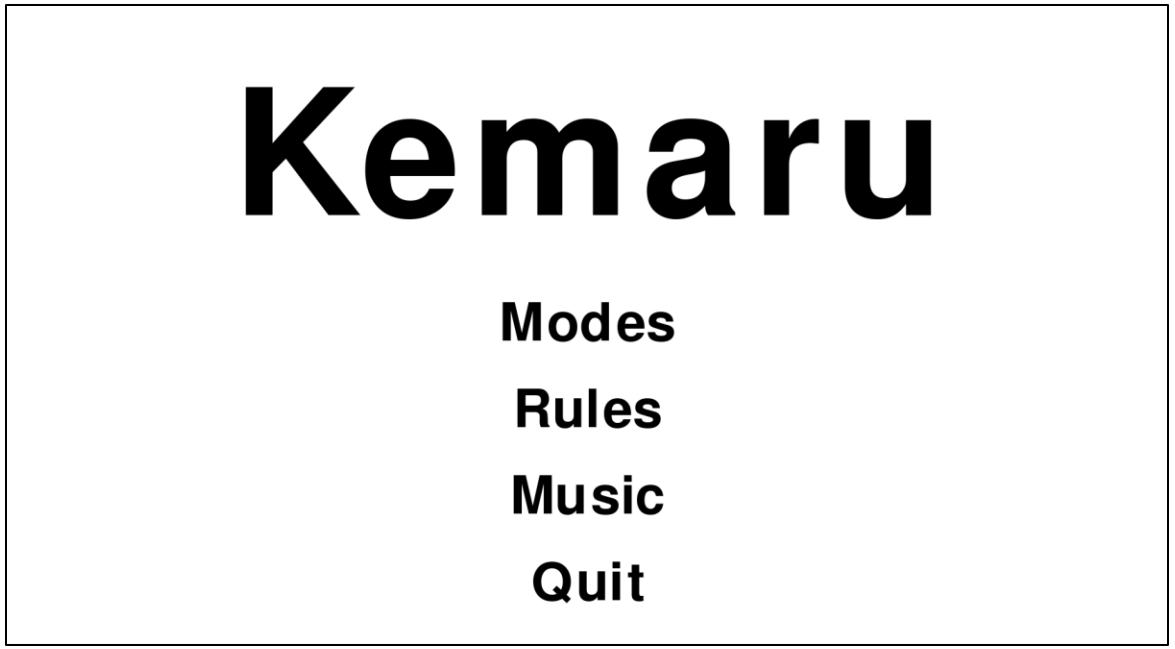
Informations : 4

II-d) Astuces pour accélérer l'algorithme précédent

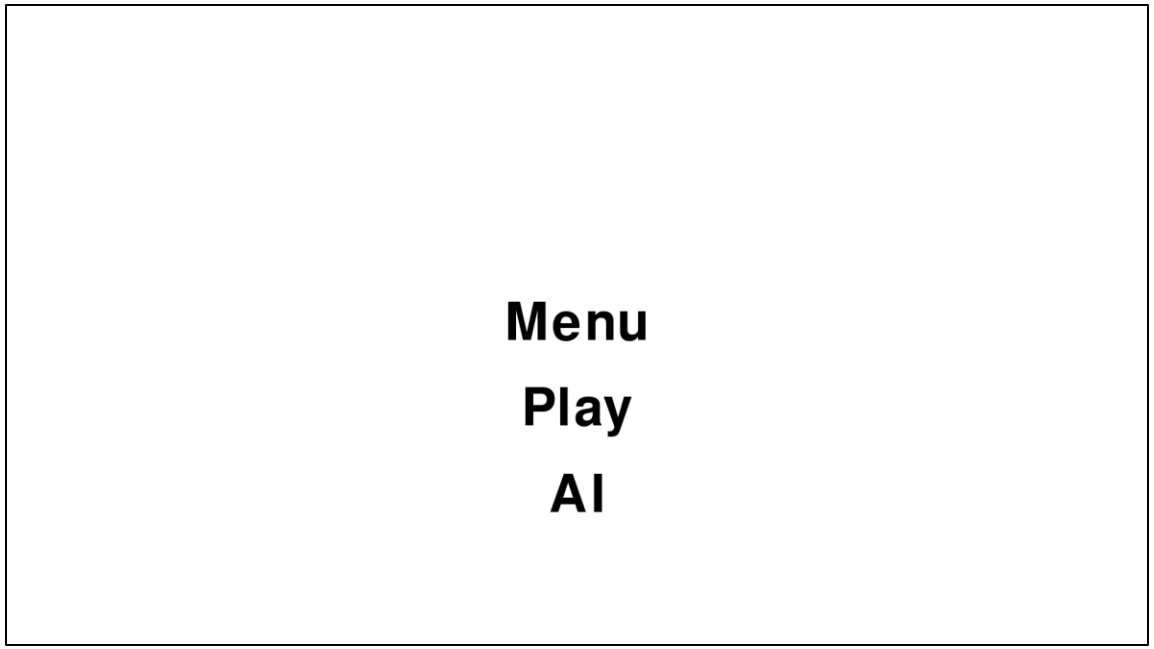
- Eliminer les niveaux 0 inutiles
- On associe à chaque grille un nombre h tel que si deux grilles donnent le même nombre h alors cela signifie que les deux grilles sont identiques->on en supprime une
- On associe à chaque grille son information (nombre de possibilités totales) et si l'information n'augmente pas on supprime la branche

III- a) Interface de jeu

-Vrai jeu indépendant



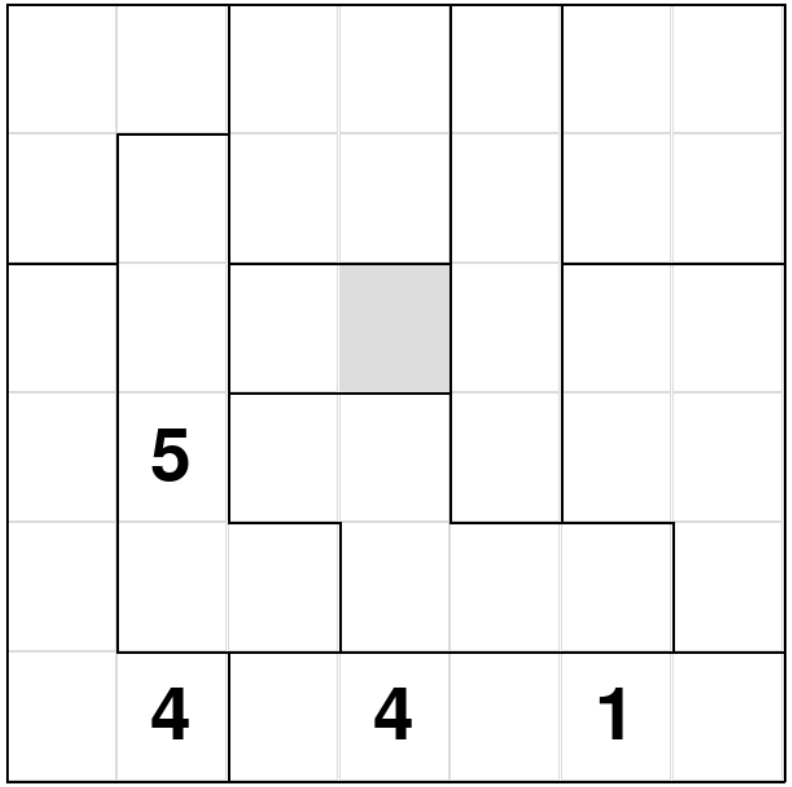
Ecran d'accueil du Kemaru



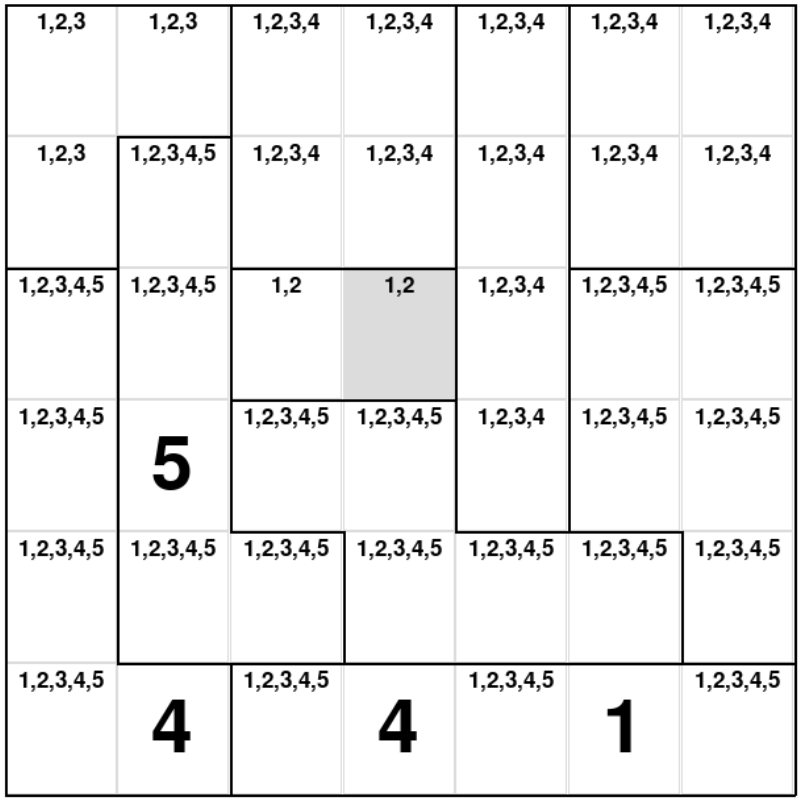
2 modes de jeu : Play et AI

III- a) interface de jeu

Mode de jeu : Play



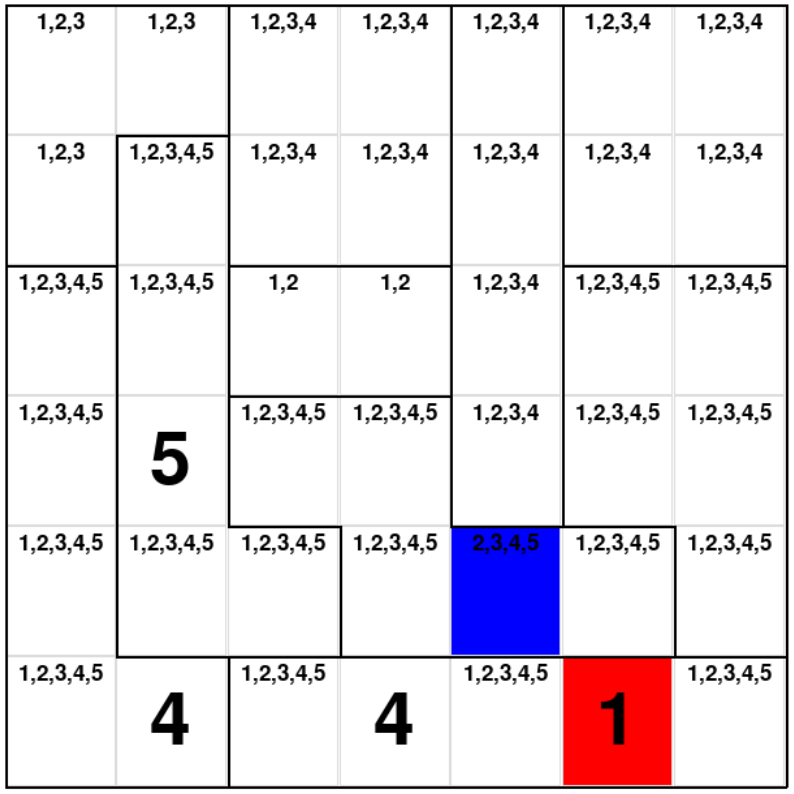
Ecran de jeu (Remarque : la grille est carrée)



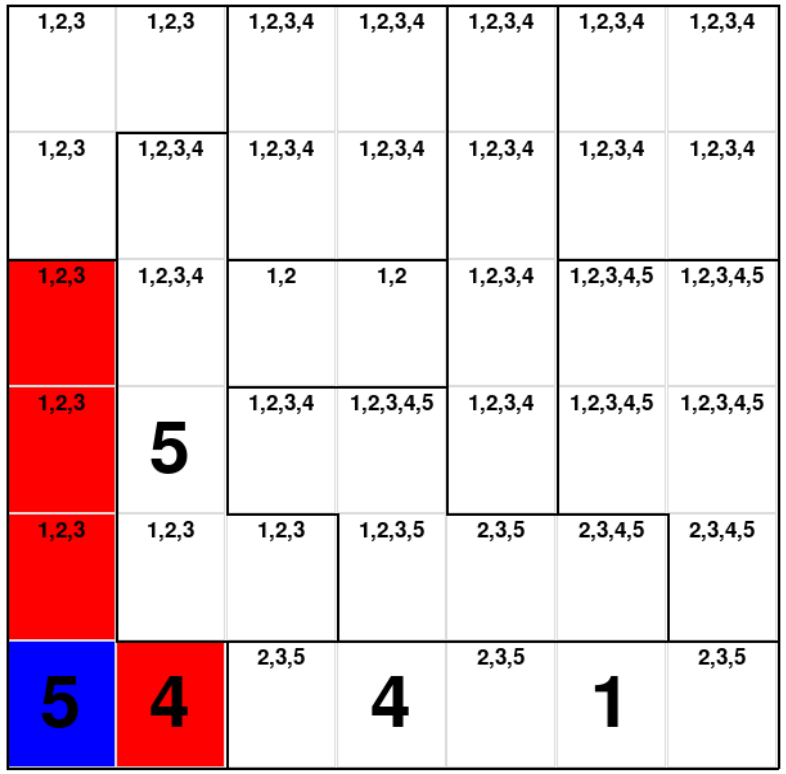
Avec ou sans les possibilités

III- b) Visualisation du niveau 0

-Affichage des étapes de résolution
-Une étape : une possibilité éliminée
-On appuie sur la touche S pour avancer d'une étape



Case en rouge : case servant à éliminer une valeur dans la case en bleu



Cage en rouge : cage servant à éliminer une valeur dans la case en bleu

III- c) Visualisation du niveau 1

-Niveau 1 : plusieurs niveaux 0 en parallèles
-Affichons ces différents univers !

1,2,3	1,2,3	1,2,3,4	1,2,3,4	1,2,3,4	1,2,3,4	1,2,3,4
1,2,3	3,4	1,2,3	3,4	1,2,3,4	1,2,3,4	1,2,3,4
1,2,3	1,2,3,4	1	1,2	1,2,3,4	1,2,3,4,5	1,2,3,4,5
1,2,3	5	3,4	2,3,4,5	3,4	1,2,3,4,5	1,2,3,4,5
1,2,3	1,2,3	1,2,3	1,2,3,5	2,3,5	2,3,4,5	2,3,4,5
5	4	2,3,5	4	2,3,5	1	2,3,5

On affiche autant de grille que de possibilités dans la case en rouge au début du niveau 1

1,2,3	1,2,3	1,2,3,4	1,2,3,4	1,2,3,4	1,2,3,4	1,2,3,4
1,2,3	3,4	1,2,3	3,4	1,2,3,4	1,2,3,4	1,2,3,4
1,2,3	1,2,3,4	2	1,2	1,2,3,4	1,2,3,4,5	1,2,3,4,5
1,2,3	5	3,4	1,3,4,5	3,4	1,2,3,4,5	1,2,3,4,5
1,2,3	1,2,3	1,2,3	1,2,3,5	2,3,5	2,3,4,5	2,3,4,5
5	4	2,3,5	4	2,3,5	1	2,3,5

On affiche les étapes du niveau 0 jusqu'à obtention (ou pas) d'information

IV-Démonstration

Remerciements

Nikolas Stott, pour son encadrement ….

…. et **vous**, pour votre écoute !