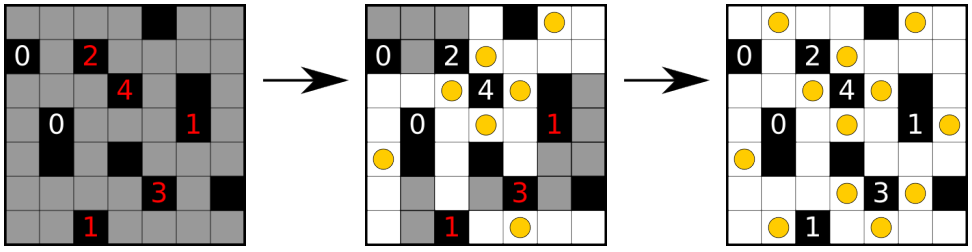
# IPIPIP : Akari

Sophie.Demassey@emn.fr



## 1 Le jeu

Akari se joue sur une grille rectangulaire composée de cases blanches et de cases noires. Le jeu consiste à placer des ampoules lumineuses sur les cases blanches de façon à éclairer toutes les cases blanches de la grille. Une ampoule émet des rayons lumineux horizontalement et verticalement : elle éclaire toutes les cases blanches contigues de la ligne ou de la colonne jusqu'à ce qu'une case noire ne bloque la lumière. Une ampoule ne doit pas éclairer une autre ampoule. Certaines cases noires indiquent un chiffre compris entre 0 à 4 : il s'agit du nombre total d'ampoules à placer dans les quatres cases adjacentes (gauche, droite, dessus, dessous : on ne considère pas les diagonales).

## 2 Techniques de résolution

D'après les règles du jeu, il est possible de déterminer a priori si certaines cases doivent nécessairement contenir une ampoule ou, au contraire, ne pas en contenir. Par exemple, les cases voisines d'une case numérotée 4 doivent toutes contenir une ampoule, tandis qu'aucune case voisine d'un 0 ne peut en contenir. Dès qu'une ampoule est placée, il devient également impossible de placer une nouvelle ampoule sur les cases adjacentes de la ligne et de la colonne. Enfin, pour éclairer une case donnée, il n'existe parfois qu'un seul placement possible.

Ainsi, simplement en raisonnant sur l'ensemble des règles du jeu et en les faisant interagir, il est souvent possible de résoudre entièrement une grille de manière automatique, sans jamais avoir à faire de supposition.

## 3 Le programme de base

L'objectif premier du projet est de programmer une version en console du jeu Akari, jouable par un utilisateur. Partant d'une grille vide, le joueur choisit, à chaque tour, d'ajouter une nouvelle ampoule ou de supprimer une ampoule déjà placée. Le programme vérifie chaque action :

- une action d'ajout est validée et effectuée seulement si la case choisie par le joueur est une case blanche qui n'est pas déjà éclairée par une autre ampoule ;
- une action de suppression est validée et effectuée seulement si la case choisie contient une ampoule ;
- une action non validée n'est pas effectuée par le programme.

Le programme s'arrête automatiquement, en indiquant le nombre d'actions effectuées, dès que toutes les cases blanches sont éclairées et que, pour chaque case numérotée, le nombre d'ampoules voisines est égal au numéro indiqué.
Il est demandé de réaliser un programme possédant a minima les fonctions suivantes :

- chargement d'une grille vide depuis un fichier texte (*NB : on trouve sur le web de nombreuses instances du jeu Akari, de différentes tailles et de différents niveaux de difficulté*)
- affichage de la grille en console
- saisie en console d'une action du joueur
- validation et exécution de l'action
- test de fin de jeu
- affichage du score (nombre d'actions)

Pour aider l'utilisateur, on affichera également à chaque tour un message indiquant : le nombre d'actions effectuées, le nombre de cases blanches non encore éclairées, le nombre de cases numérotées non satisfaites et, le cas échéant, les raisons d'invalidation d'une action.

Dans un second temps, on cherchera à équiper le programme d'une intelligence artificielle. La version intelligente du programme doit être capable de résoudre le jeu (du moins les instances les plus faciles du jeu) de manière automatique (sans joueur). Il s'agit ici d'implémenter un maximum de techniques de raisonnement sur les règles du jeu, telles que celles évoquées ci-dessus. À vous d'identifier de nouvelles règles de déduction et de les mettre en œuvre.

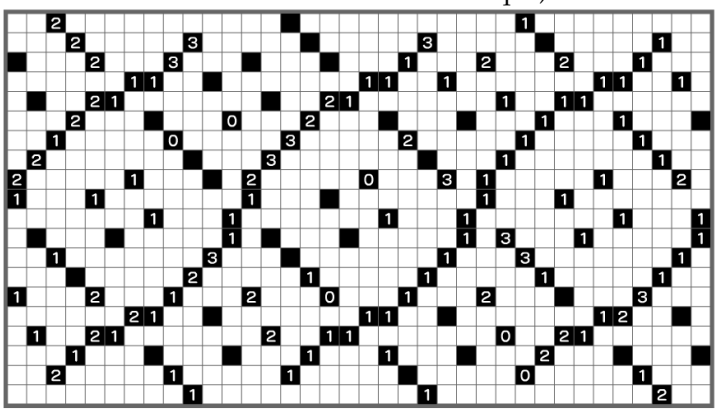
# 4 L'implémentation

Le code devra s'articuler autour d'un minimum de classes :

- un objet principal servant à l'exécution du jeu et les interactions avec l'utilisateur
- un objet `Grille` représentant l'état des cases de la grille à tout instant (case blanche/noire/numérotée, case éclairée/éteinte, ampoule interdite/obligatoire/indécis),
- un objet `Contrainte` pour chaque règle du jeu à respecter. Une méthode `estValide()` indiquera si l'état courant de la grille respecte la règle. Dans le programme intelligent, une méthode `deduction()` modifiera l'état de la grille en déduisant des placements obligatoires ou interdits à partir d'un raisonnement sur la règle en fonction de l'état courant de la grille. *NB : pensez à l'héritage !*

Enfin, toutes les méthodes devront être les plus élémentaires possibles : découpez-les !

Remarque : **il n'est pas demandé** de développer une IHM autre que la saisie/affichage en console. Vous pouvez bien-sûr en développer une, mais seulement après avoir implémenté et testé les fonctionnalités requises (jeu interactif en console et résolution automatique).



# 5 La difficulté

Algorithmique : ***
Programmation : **
Modélisation du problème : *
Modélisation objet : ***