

Retour sur trace

1 Cogitus

Cogitus est un jeu à un joueur dans lequel on doit placer 32 jetons, portant les numéros de 1 à 8 (quatre de chaque) sur une grille de taille 7×7 . Il faut par ailleurs respecter une contrainte : toute séquence contiguë de jetons, en ligne ou en colonne, ne doit pas voir sa somme dépasser strictement k (typiquement $k = 11$). Une solution est présentée ci-dessous pour $k = 11$:

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

On représentera, à tout moment, la grille par un tableau de tableaux d'entiers, où 0 désigne une case vide et un entier de 1 à 8 un jeton de la valeur correspondante.

1. Proposer une fonction `somme_h` de signature `int array array -> int -> int -> int` telle que `somme_h arr i j` renvoie la somme des jetons placés sur la grille formant une séquence contiguë de jetons sur la ligne i , à gauche de la colonne j et se terminant sur la colonne j (si cette séquence se poursuit à droite de la colonne j , ces jetons sont ignorés). Par exemple, pour la grille ci-dessus, $i = 2$ et $j = 5$, la fonction devra renvoyer 8 ($4 + 3 + 1$). On réfléchira à une possible implémentation récursive.

2. Faire de même pour une fonction `somme_v` faisant de même verticalement (avec les jetons au-dessus de la case (i, j)). Avec $i = 2$ et $j = 4$, on devra obtenir 11 ($3 + 7 + 1$).

On cherche à construire une solution pour un k donné, si elle existe. On va utiliser une méthode de retour sur trace, en remplissant les cases ligne par ligne, et de gauche à droite sur chaque ligne. On numérote les cases de 0 à 48 selon cet ordre

3. Proposer une fonction `coords` de signature `int -> int * int` prenant en argument

un entier entre 0 et 48 inclus et renvoyant un couple (i, j) représentant les coordonnées de la case dans l'ordre proposé. Par exemple, pour 18, on devra obtenir (2, 4).

Pour représenter, à tout moment, les jetons restant à poser, on utilisera un tableau de taille 9 tel que dans la case i , pour $i \in [1..8]$, on trouve le nombre de jetons restant marqués i . Pour faciliter les choses, on rangera dans la case d'index 0 la somme des autres cases.

4. Proposer une fonction `init` de signature `unit -> int array` construisant et remplissant ce tableau avec la situation initiale (on rappelle que l'on a 4 jetons de chaque valeur).

5. Proposer une fonction `retire` de signature `int array -> int -> unit` prenant en argument un tableau tel que décrit précédemment et un entier entre 1 et 8 et modifiant le tableau de façon à ce qu'il contienne un jeton de moins de la valeur fournie.

6. Proposer une fonction `ajoute` de signature `int array -> int -> unit` effectuant la transformation inverse.

7. Écrire une fonction possible de signature `int array array -> int array -> int * int -> int -> int list` prenant en argument l'état courant de la grille, le tableau des jetons disponibles, des coordonnées et la limite k et renvoyant une liste d'entiers entre 0 et 8 correspondant à ce que l'on peut placer dans la case sélectionnée (0 figurera toujours parmi les possibilités, et correspond à ne rien placer dans la case). On se servira des fonctions précédentes (on pourra supposer que la case désignée contient un 0 si c'est utile, auquel cas on en tiendra compte dans la suite).

8. En déduire une fonction `resout` de signature `int -> int array array option` prenant en argument k et renvoyant une option, `None` s'il n'y a pas de solution au problème, et `Some arr` où `arr` est une grille solution sinon.

9. Essayer la fonction pour $k = 14$, puis $k = 10$. On pourra essayer de vérifier qu'il n'y a pas de solution pour un k plus petit, mais cela prend du temps.

2 Suguru

Le jeu appelé Suguru est une variante de Sudoku dans laquelle il faut placer des entiers strictement positifs dans une grille où certains entiers sont déjà placés. La grille est divisée en plusieurs zones de tailles et formes variables. Si une zone contient n cases, alors il faut y placer chacun des chiffres de 1 à n (sans doublons donc). Par ailleurs, deux cases contenant le même chiffre ne doivent pas se toucher en 8-connexité (partager un côté ou un coin).

On encode une grille sous la forme suivante : un premier tableau de tableaux d'entiers

contenant 0 pour les cases non remplies et un entier strictement positif pour les indices, et un second tableau de tableaux d'entiers contenant, pour chaque case, un entier entre 0 et $p - 1$ indiquant à laquelle des p zones de la grille appartient la case.

Voici un exemple de problème :

3					2	
4	1		5			7
		7			6	
2	1		4			5
	7				7	1
	6					
3						3
			3			
			6		2	

Élaborer une solution à base de retour sur trace permettant de résoudre un tel problème. Vous êtes libres de vos choix d'implémentation.