Programmation par contraintes : Badge F
Mini-projet

Implémentation d'applications de résolution de problèmes de satisfaction de contraintes avec la bibliothèque CHOCO. Ce projet rapporte au maximum 3 points sur votre note de contrôle continu.

Olivier Bailleux.

Objectif

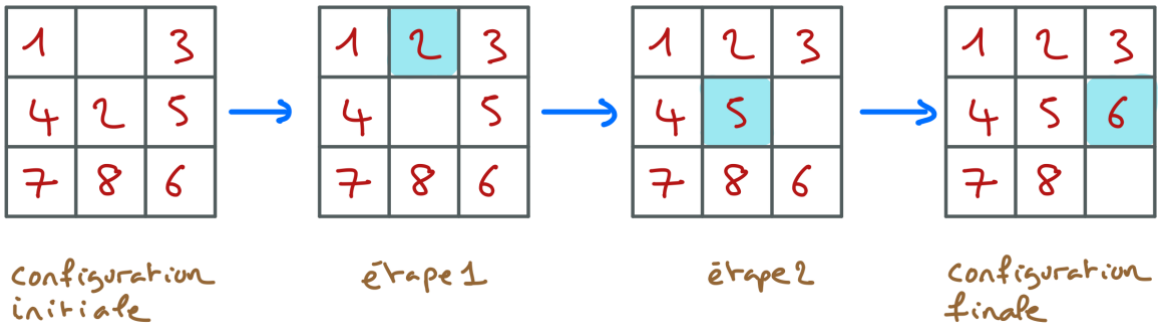
Implémenter un programme permettant de rechercher une séquence de mouvements permettant de résoudre le puzzle appelé **jeu du Taquin**.

Prérequis

Avoir achevé l'étude du document de préparation du badge F, y compris les exercices d'entraînement, et avoir réalisé et soumis le mini-projet de TP du badge F.

Travail à réaliser

Vous devez réaliser une application CHOCO qui, étant donnée une configuration initiale et une configuration finale du Taquin 3x3, trouve un moyen de passer de l'une à l'autre en un nombre N donné d'étapes. Voici un exemple avec N=3.



Les configurations initiale et finale sont données sous forme de tableaux initialisés. Voici les tableaux correspondants aux exemples ci-dessus.

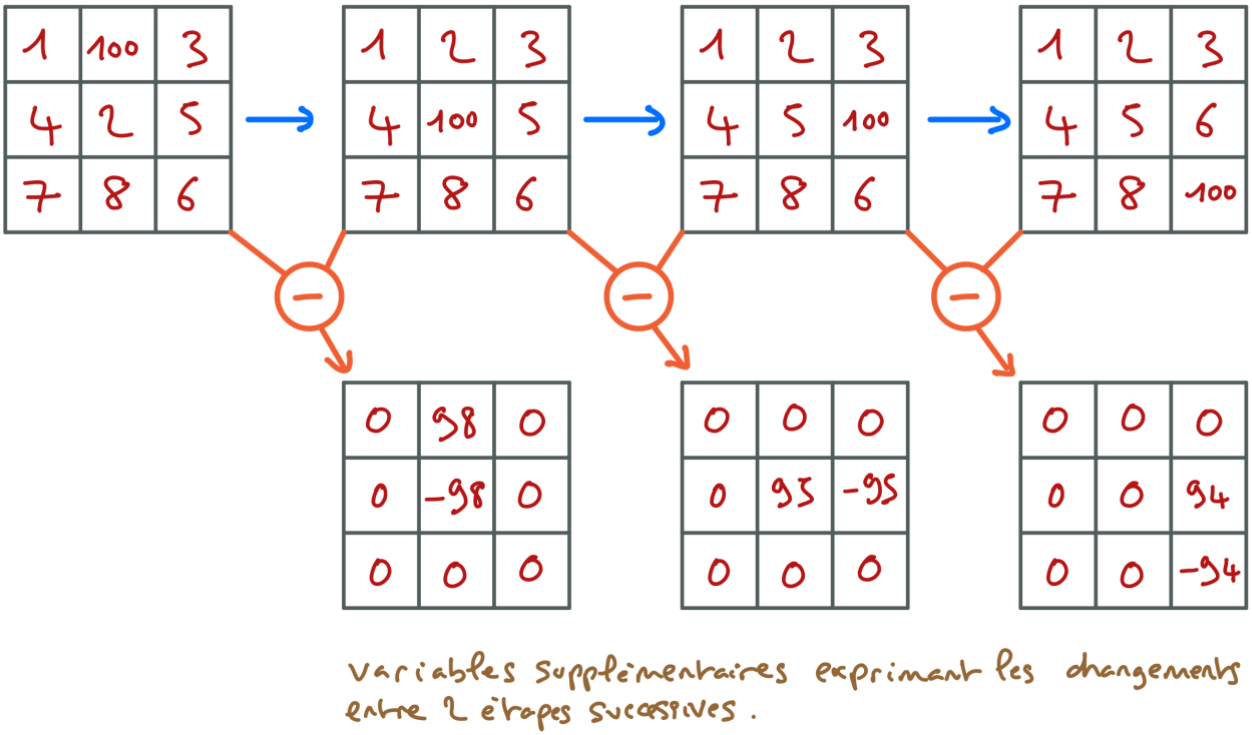
```
int[] firstDom = new int[]{1,100,3,4,2,5,6,7,8};
int[] lastDom =  new int[]{1,2,3,4,5,6,7,8,100};
```

Pour modéliser ce problème, il faut introduire des variables représentant chacune des configurations successives depuis la configuration initiale jusqu'à la configuration finale en passant par toutes les étapes. Pour N=3, comme dans l'exemple ci-dessus, Il faut donc 9 variables fois 4 configurations, incluant les configurations initiale et finale. Les domaines des configurations initiale et finale sont des singletons puisque ces configurations sont connues dès le départ.

Je préconise d'utiliser, pour les variables des configurations intermédiaires, les domaines {1, 2, 3, 4, 5, 6, 7, 8, 100}, la valeur 100 représentant la case vide.

Il « suffit » alors, pour modéliser le problème à N étapes (N déplacements d'un chiffre vers la case vide), de poser des contraintes imposant que le passage de chaque configuration à la suivante est légal, c'est-à-dire consiste à échanger la valeur 100 avec une valeur contigüe verticalement ou horizontalement.

Pour pouvoir spécifier de telles contraintes, je préconise (mais vous pouvez chercher un autre moyen) d'introduire des variables intermédiaires qui représentent les différences entre les valeurs des variables de deux étapes successives.



Des contraintes arithmétiques relient les variables de deux configurations successives et les variables intermédiaires concernées.

Le nombre de motifs possibles pour les valeurs des variables intermédiaires est assez limité. Ces motifs comportent 7 fois la valeur 0 et une paire de valeurs opposées telles que 99 et -99 (la valeur 1 a été déplacée dans la case vide), 98 et -98 (la valeur 2 a été déplacée dans la case vide), etc., situées dans des cases contigües.

Précisément, Il y a 192 motifs pour les valeurs des variables intermédiaires si la transition entre les deux configurations associées est légale. Sauf erreur de calcul de ma part, merci de me contacter si vous n'êtes pas d'accord avec ce résultat.)

Je préconise de modéliser les motifs associés aux transitions légale par des contraintes de type `Table`, c'est-à-dire des contraintes énumératives de CHOCO. Les informations permettant de construire une telle contrainte (avec la classe CHOCO `Tuple`) et de l'ajouter à un modèle (avec la méthode `Model.table`) peuvent être trouvées facilement dans la documentation CHOCO.

Attention, la résolution du problème peut prendre un certain temps si le nombre d'étapes dépasse 15. L'instance suivante ne peut être résolue en moins de 16 étapes. Sur ma machine de bureau (un iMac assez récent), sa résolution nécessite plus d'une minute de temps de calcul.

Livrables

- Un document pdf contenant la preuve que votre application fonctionne : copie d'écran d'instances solvables avec une de leurs solutions, présentées de manière lisible, et d'instances non solvables avec le nombre d'étapes indiqué. Ce document doit aussi donner toute précision utile à la compréhension de votre code.
- Un unique fichier .java contenant votre code commenté.