Rapport pour le Projet 2 d'Algorithmique

OOMS Aurélien BA1 INFO

7 avril 2011

Table des matières

1 Introduction

Dans le cadre du cours d'algorithmique, il nous a été demandé de concevoir un programme résolvant un tablier du jeu du solitaire à l'aide du backtracking.

Les sections qui vont suivre vont décrire le fonctionnement de la classe et de ses méthodes ainsi que l'interface mise en place à cet effet.

2 La classe Tablier

2.1 Attributs

Classe ayant 8 attributs (N.B. : Des méthodes get sont implémentées en conséquence) :

2.1.1 tablier

Matrice des éléments du tablier.

2.1.2 posSpeciale

Position sur laquelle le dernier pion doit se trouver (constante).

2.1.3 n

Nombre de lignes (constante).

2.1.4 m

Nombre de colonnes (constante).

2.1.5 nombrePieces

Nombre de pièces (constante).

2.1.6 deltaSauts

Nombre relatif de sauts réalisés.

2.1.7 mouvements

Sauts corrects pour arriver à la solution.

2.1.8 copietablier

Copie du tablier utilisée pour l'affichage de la solution finale.

2.2 Méthodes

2.2.1 estVide

True si la position (x,y) existe et est vide. False sinon.

2.2.2 estPiece

True si la position (x,y) existe et contient une pièce. False sinon.

2.2.3 voir

Retourne ce qui se trouve en (x,y) si cette position existe. False sinon.

2.2.4 obtenirPosSpeciale

Retourne la position où doit se trouver la seule pièce restante à la fin.

2.2.5 sauter

Fais sauter une pièce sur une case vide selon les règles suivantes :

1. La case vide doit se trouver à deux cases de la pièce horizontalement ou verticalement ;
2. La case située entre le vide et la pièce doit également contenir une pièce.

Si toutes ces conditions sont remplies, le saut est effectué : le vide devient une pièce, les deux pièces deviennent un vide et saut retourne True. False sinon.

2.2.6 antiSauter

Réalise l'opération inverse du saut. Dans le cadre de ce programme, l'antiSaut est toujours utilisé immédiatement après son saut associé de façon a détruire la solution partielle dans notre canva du backtracking.

2.2.7 jouer

Fonction récursive qui teste tous les sauts possibles jusqu'à épuisement des possibilités ou résolution du problème. Le principe est le suivant :

1. Essaye un déplacement
   - Si il est possible, il l'effectue et rejoue sur le tablier modifié. (Revient au 1 avec le nouveau tablier résultant du déplacement)
   - Détruit la solution.
2. Si il n'est pas possible, il essaye un autre déplacement.

Notez qu'une fois qu'il a essayé tous les déplacements suivant un premier déplacement sans en trouver un qui convient, il déconstruit la solution qu'il avait commencé à créer en revenant à l'étape précédent le déplacement qui nous mettait dans une situation insolvable. Une fois qu'il a trouvé la solution il s'interrompt immédiatement, sans déconstruire la solution en cours et affiche les étapes de la résolution du problème.

2.2.8 __repr__

Affiche les cases du tablier.

3 Interface

L'interface utilise la classe Tablier afin de trouver une solution à des configurations du jeu du solitaire enregistrées dans des fichiers. Elle demande à l'utilisateur le nom du fichier et les paramètres et gère les exceptions de manière à permettre à l'utilisateur de corriger ses erreurs.

Lorsqu'une solution est trouvée à un problème, elle propose un affichage dynamique (affichages successifs des différentes étapes avec des intervalles de 0.5 seconde et raffraîchissement de l'écran) ou coup par coup (affichages successifs des différentes étapes avec pause entre chaque, l'utilisateur doit appuyer sur entrer pour passer à l'étape suivante). Si il n'y a pas de solution, elle l'indique à l'utilisateur.

4 Conclusion

Une fois le principe du backtracking comprit, la réalisation du projet n'est pas compliquée.

1. Le cas où la solution est trouvée est le suivant : le nombre relatif de sauts est égal au nombre de pièces moins 1.
2. La récursivité est utilisée de la manière suivante : si le saut est possible on le fait et on repart du tablier résultant du saut, sinon on essaye un autre saut et si tous les sauts sont épuisés on revient à l'étape précédent le dernier saut à l'aide d'antiSaut.

Une fois que la solution est trouvée, on l'affiche. Si dans le cas contraire, tous les sauts sont essayés sans trouver de solution, on le signale à l'utilisateur.

Le backtracking est une technique assez intuitive qui permet de résoudre des problèmes de génération de solution où le chemin pour y arriver est un ensemble de choix successifs. Il pose un problème lorsque les possibilités de combinaison de ses choix sont trop nombreuses. Le cas s'est clairement exprimé lorsqu'on essayait de résoudre les tabliers donnés au début du projet.