Informatique - Master 1
Module IA
TP 2 - Jeu du taquin

Tassadit BOUADI

2013/2014

Lors de ce TP, vous allez vous familiariser avec l'algorithme A*. Vous allez appliquez cette méthode à la recherche d'une solution d'un jeu de taquin.

Il s'agit d'un jeu très connu dans lequel 8 cases numérotées de 1 à 8 sont disposées sur un damier 3×3. Le but du jeu est de rétablir une situation objectif à partir de la situation initiale, sachant que les seuls opérateurs de transformation d'un état du jeu à un autre consiste à échanger la case vide avec un de ses cases adjacentes. Les 2 états finaux (ou objectifs) sont les suivants :

1	2	3
8	0	4
7	6	5

(a)

1	2	3
4	5	6
7	8	0

(b)

Recopier les sources du TP. Vous devez ajouter les .jar fournis, *utilsTaquin.jar*, dans votre projet Eclipse. Vous disposez également de la documentation des classes *FileTaquin* et *SetTaquin*.

1 Définition des états successeurs

Dans la classe Etat, définissez la fonction*getSuccesseurs* qui permet d'obtenir les états successeurs de l'état courant, en considérant tous les déplacements possibles de la case vide et en utilisant l'heuristique en paramètre, pour calculer les valeurs associées à chacun de ces états successeurs. Pour cela, vous utiliserez les fonctions *deplacementPossible* et *etendEtat* qui sont également à définir.

2 Recherche d'une solution par l'algorithme A*

Définissez la fonction *algoAEtoile* qui, à partir d'un état initial et d'une heuristique, donne un des 2 états finaux pouvant être obtenus, si un état final est accessible.

Vous comparerez ensuite les 2 heuristiques utilisées, en appliquant l'algorithme A*, à partir de chacune des 2 configurations initiales. Pour un état donné, les heuristiques sont les suivantes :

- h_1 : le nombre de pièces mal placées ;
- h_2 : la somme des distances de chaque pièce à sa position finale (somme des distances de Manhattan).

Quelle est la meilleure heuristique (en terme de nombre de nœuds générés) ?

Algorithme A* appliqué au jeu du taquin

```
booléen algoAEtoile(Etat étatInit, FonctionHeuristique heurist)
   OUVERTS = {etatInit};
   FERMES = {};

   tant que OUVERTS non vide et étatFinal non défini faire
      état = premier élément de OUVERTS;
      si état est un état final
         alors étatFinal = état;
      sinon ajouter état dans FERMES;
         pour tous les états successeur de état faire
            si succ est dans FERMES
               soit succ_fermé l'état de FERMES égal à succ;
               alors si f(succ) < f(succ_fermé)
                        alors ôter succ_fermé de FERMES;
                              ajouter succ dans OUVERTS;
                     fsi
            sinon si succ est dans OUVERTS
               soit succ_ouvert l'état de OUVERTS égal à succ;
               alors si f(succ) < f(succ_ouvert)
                        alors ôter succ_ouvert de OUVERTS;
                              ajouter succ dans OUVERTS;
                     fsi
            sinon ajouter succ dans OUVERTS;
            fsi
         fait
      fsi
   fait
```