Intelligence Artificielle

Atelier 2

Exercice 1

Le but de ce TP est de résoudre le problème du taquin à 8 en utilisant l'algorithme A*. Télécharger le code source incomplet à partir de la plateforme e-service de l'Ensah et compléter les parties demandées.

Figure 1: Etat Final

1. Le code source contient une classe Nœud avec les attributs suivants:
   - Un attribut état qui consiste en une matrice pour stocker l'état (la case vide est représentée par 0).
   - Un tableau *ArrayList* de Nœuds pour contenir les nœuds fils du nœud en cours.
   - Un attribut parent de type Nœud pour stocker le parent du Nœud en cours. Cet attribut est nécessaire pour reconstruire le chemin à la fin de la recherche.
   - Un attribut h de type int pour stocker la valeur de l'heuristique.
   - Un attribut g de type int pour stocker le coût nécessaire pour arriver au nœud en cours à partir de l'état initial.
   - Un attribut F de type int pour stocker la valeur de la fonction $F(n) = h(n) + g(n)$.
2. Plusieurs méthodes ont été déjà implémentées. Pour plus de détails sur chaque méthode, veuillez consulter le code source.
   Pour cette classe, on vous demande d'implémenter les méthodes suivantes :

- *enBas(int i)* qui permet de déplacer la case vide en bas.
- *aGauche(int i)* qui permet de déplacer la case vide à gauche.
- *aDroite(int i)* qui permet de déplacer la case vide à droite.

La méthode *enHaut(int i)* est déjà implémentée pour vous aider à comprendre le principe.
A titre d'indication, l'état est représentée sous forme d'une matrice (et non pas un tableau de 1 dimension comme dans le TP précédent.)

1	2	4
3	6	5
7	0	8

3. Implémenter la méthode *public static int calculerH1(Nœud noeudCourant, Nœud solution)* qui permet de calculer l'heuristique h_1 (une estimation de la distance entre le nœud en cours et l'état solution), où h_1 est le nombre de cases mal placée.
   Pour l'état ci-dessous, les cases 1,2,4,6,7 et 0 sont mal placées. Donc $h_1 = 6$.

1	2	4
3	6	5
7	0	8

4. Dans la classe RechercheInformee, compléter l'algorithme A*.

5. Tester votre programme.

6. Modifier la méthode *public static int calculerH1(Noeud noeudCourant, Noeud solution)* pour calculer l'heuristique h_2, où h_2 est la somme des distances de manhattan entre chasue tuile mal placée et sa position dans l'état but (Le nombre de mouvements nécessaires pour qu'une tuile atteint sa position correcte, en supposant que chaque tuile peut être déplacée à une case adjacente même si celle-ci n'est pas vide).
   Prenant l'état ci-dessous,
   pour atteindre l'état but (Figure 1) Donc $h_2 = 1 + 1 + 2 + 0 + 2 + 0 + 1 + 3 + 0 = 10$

1	2	4
3	6	5
7	0	8

7. Tester votre programme et comparer le temps d'exécution entre les deux heuristiques.

8. Quelle est la meilleur heuristique pour résoudre ce problème (Justifier)?.