

Intelligence Artificielle

Atelier N°1

Exercice 1

Le but de ce TP est de résoudre le problème du taquin à 8 en utilisant l'exploration non informée.

Figure 1: Etat Final

	1	2
3	4	5
6	7	8

1. Créer une classe *Nœud* avec les attributs suivants:
 - Un attribut état qui consiste en un tableau d'entiers 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.
2. Implémenter un constructeur avec un paramètre de type `int[]`.
3. Implémenter les méthodes suivantes :
 - Une méthode *estBut()* permettant de tester si le nœud en cours contient l'état but.
 - Ajouter les *Getters* et les *Setters*.
 - Une méthode *memeEtat(Nœud n)* qui permet de tester si le nœud en cours et le nœud passé en paramètre en le même état.
 - Une méthode *dupliquer()* qui permet de faire une copie du nœud en cours.
 - Une méthode *afficher()* qui permet d'afficher l'état d'un nœud.

- Une méthode *développer()* qui permet de générer les fils du nœud en cours en faisant appel à aux méthode suivante (chacune de ces méthodes prend en paramètre la position de la case vide (0)) :
 - *enBas(int i)* qui permet de déplacer la case vide en bas.
 - *enHas(int i)* qui permet de déplacer la case vide en haut.
 - *aGauche(int i)* qui permet de déplacer la case vide à gauche.
 - *aDroite(int i)* qui permet de déplacer la case vide à droite.

Indication! prenez l'exemple de la méthode *aGauche()*. Notre état est représentée par un tableau d'une seule dimension.

1	2	4	3	6	5	7	0	8
---	---	---	---	---	---	---	---	---

qui est équivalente à la représentation suivante (les exposants représentent les positions des éléments dans le tableau à 1d):

$$\begin{array}{ccc} {}^0_1 & {}^{12}_2 & {}^{24}_4 \\ {}^{33}_3 & {}^{46}_6 & {}^{55}_5 \\ {}^{67}_7 & {}^{70}_0 & {}^{88}_8 \end{array}$$

Si $i \% 3 > 0$ avec i est la position actuelle de la case vide (0), alors on peut alors effectuer le déplacement de 0 à gauche par les instructions suivante :

$etat[i] = etat[i - 1];$
 $etat[i - 1] = 0;$

Vous pouvez aussi représenter l'état comme une matrice.

4. Créer une classe *Exploration* pour mettre en œuvre l'exploration en largeur d'abord contenant les méthodes suivantes:
 - Une méthode *contient(Nœud nœud, ArrayList<Nœud> list)* pour tester si un nœud existe dans une liste de nœuds .
 - Une méthode *rechercher(Nœud racine)* qui implémente l'exploration en largeur d'abord et qui retourne le chemin vers le but.
 - Une méthode *ArrayList<Nœud> construireChemin(Nœud n)* qui permet de reconstruire le chemin.
5. Créer une classe *Test* pour tester le programme.
6. Modifier la méthode *rechercher* pour implémenter l'exploration en profondeur d'abord.