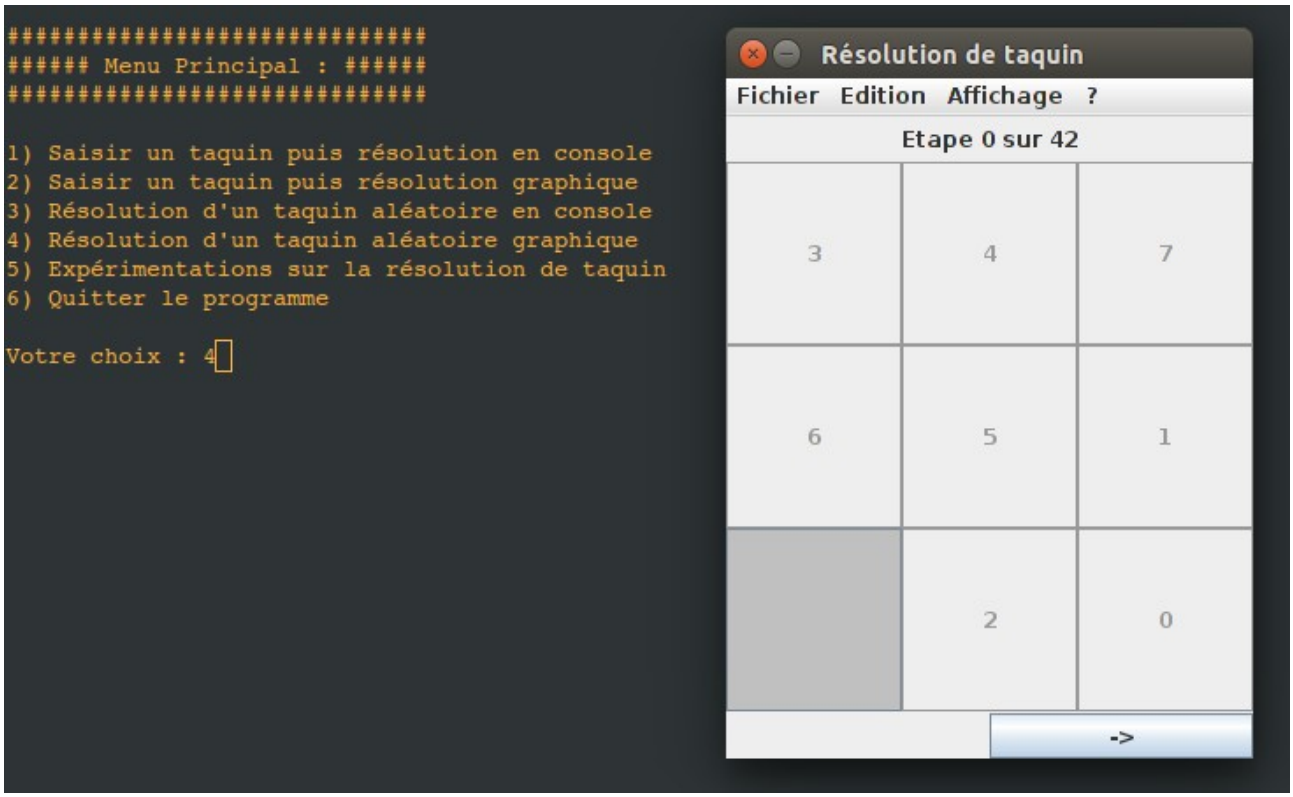
Projet d'Intelligence Artificielle :

*Résolution de taquin avec A**

par Théo CHELIM et Maxime KERMARQUER

I) Résolution du taquin avec A*

Le taquin est un puzzle carré constitué de NxN cases et le but consiste à remettre dans l'ordre les carreaux dans l'ordre à partir d'une configuration initiale quelconque.
Le jeu du Taquin est un exemple très utilisé comme problème de recherche, pour pouvoir résoudre ce « puzzle » nous avons implémenté l'algorithme A qui trouve toujours une solution quand le problème en a une.*
L'algorithme A utilise une heuristique dans sa résolution du problème de recherche. Pour la résolution du Taquin, nous utilisons des distances de Manhattan d1, d2, … , d6 qui représente la somme des distances élémentaires de chaque case puis pondéré respectivement par des poids donné* $\prod 1, \prod 2, \dots, \prod 6$.
Nous étudierons les solutions données par A par rapport aux différentes distances (heuristiques) utilisées.*
Pour éviter d'essayer de résoudre un taquin dans une configuration initiale insoluble, nous avons rejeter tout état initial qui ne permet pas d'arriver jusqu'à l'état final en cherchant la condition pour laquelle le taquin est soluble. En effet, la configuration initiale d'un taquin est une permutation de sa configuration finale. Cette permutation est dite paire si elle peut être obtenue par un nombre pair d'échanges successifs de deux cases, et impaire sinon.
Le problème sera soluble si la parité de la permutation est identique à la parité de la distance élémentaire de la case vide.
Cela permet de rejeter la moitié des états initiaux possibles.



II) Implémentation du problème

Nous avons implémenté ce problème dans le langage Java.

Les classes du projet :

- Fenetre.java : *Pour l'interface graphique de notre projet.*
- Taquin.java : *Implémente l'algorithme de résolution d'un taquin*
- Etat.java : *Représente un état du taquin.*
- Chemin.java : *Représente un chemin c'est-à-dire un ensemble d'état.*
- CheminPriorite.java : *Hérite de chemin avec une priorité en plus.*
- FileAttente.java : *Représente la file d'attente de chemins.*
- Dictionnaire.java : *Dictionnaire des états déjà visités.*
- Noeud.java : *Représente un nœud du dictionnaire d'états.*
- Solution.java : *Toutes les données relatives à une solution du problème.*

Les structures de données :

- Un chemin a été représenté comme un couple (pile d'états, longueur pile) pour pouvoir accéder à la longueur de la pile en temps O(1).
- Le dictionnaire est représenté par un arbre binaire de recherche.
- La file d'attente de chemins est une liste de CheminPriorite trié dans l'ordre croissante de priorité. L'ajout d'un chemin se fait donc en gardant le trie de cette liste.

Fonctionnalités du projets :

- Saisir un état initial pour le taquin en console ou avec l'interface graphique.
- Sélection d'un état initial aléatoire en console ou avec l'interface graphique.
- Résolution du taquin pour cet état initial.
- Sélection de la distance de Manhattan à utiliser (heuristique).
- Affichage de toutes les informations sur la solution du problème (durée, longueur, …).
- Représentation graphique du taquin avec une image.
- Expérimentations de résolutions d'un grand nombre de taquins pour étudier les solutions.
- En console le problème a été généralisé pour le cas NxN.

III) Expérimentations

Nous avons instancié un nombre de 1000 taquins aléatoires dans un tableau d'État puis nous avons lancé la résolution de ces taquins avec les distances d1, d2, …, d6. Pour faire des moyennes sur les données des solutions et ainsi comparer les différentes heuristiques entre-elles.
Voici le résultat de ces expérimentations :

```
* Résolution de taquins pour la distance d1:
        Temps moyen de résolution d'un taquin pour d1 (1000 taquins résolus): 1ms.
        Longueur moyenne de la solution d'un taquin pour d1 (1000 taquins résolus): 27.
        Nombre d'etats visites en moyenne pour d1 (1000 taquins résolus): 727.
        Nombre de chemins sortis de la file en moyenne pour d1 (1000 taquins résolus): 437.

* Résolution de taquins pour la distance d2:
        Temps moyen de résolution d'un taquin pour d2 (1000 taquins résolus): 0ms.
        Longueur moyenne de la solution d'un taquin pour d2 (1000 taquins résolus): 32.
        Nombre d'etats visites en moyenne pour d2 (1000 taquins résolus): 497.
        Nombre de chemins sortis de la file en moyenne pour d2 (1000 taquins résolus): 299.

* Résolution de taquins pour la distance d3:
        Temps moyen de résolution d'un taquin pour d3 (1000 taquins résolus): 3ms.
        Longueur moyenne de la solution d'un taquin pour d3 (1000 taquins résolus): 22.
        Nombre d'etats visites en moyenne pour d3 (1000 taquins résolus): 1554.
        Nombre de chemins sortis de la file en moyenne pour d3 (1000 taquins résolus): 976.

* Résolution de taquins pour la distance d4:
        Temps moyen de résolution d'un taquin pour d4 (1000 taquins résolus): 0ms.
        Longueur moyenne de la solution d'un taquin pour d4 (1000 taquins résolus): 32.
        Nombre d'etats visites en moyenne pour d4 (1000 taquins résolus): 456.
        Nombre de chemins sortis de la file en moyenne pour d4 (1000 taquins résolus): 274.

* Résolution de taquins pour la distance d5:
        Temps moyen de résolution d'un taquin pour d5 (1000 taquins résolus): 2ms.
        Longueur moyenne de la solution d'un taquin pour d5 (1000 taquins résolus): 22.
        Nombre d'etats visites en moyenne pour d5 (1000 taquins résolus): 1338.
        Nombre de chemins sortis de la file en moyenne pour d5 (1000 taquins résolus): 834.

* Résolution de taquins pour la distance d6:
        Temps moyen de résolution d'un taquin pour d6 (1000 taquins résolus): 8ms.
        Longueur moyenne de la solution d'un taquin pour d6 (1000 taquins résolus): 21.
        Nombre d'etats visites en moyenne pour d6 (1000 taquins résolus): 2473.
        Nombre de chemins sortis de la file en moyenne pour d6 (1000 taquins résolus): 1605.

Durée du programme pour 6x1000 taquins à résoudre: 0min17s.
```

Observations

- La distance d6 visite le plus d'états, et a le temps de résolution moyen le plus long.
- Les distances d6, d5, d3 ont les chemins moyens les plus courts.
- La distance d4 semble être la plus rapide à trouver une solution, au vu de son nombre d'états visités. Elle est suivie de d2 puis d1.