Le Taquin

Mélanie MARQUES & Guillaume COQUARD

Rendu le 18 mars 2019

Première partie

Etude Théorique

0.1 Le Problème du Taquin

Un taquin $n \times n$ est un puzzle carré, d'une largeur l et d'une taille t, telles que $l = n$, $t = n \times n$, composé de $t - 1$ tuiles numérotées de 1 à t et d'un trou.

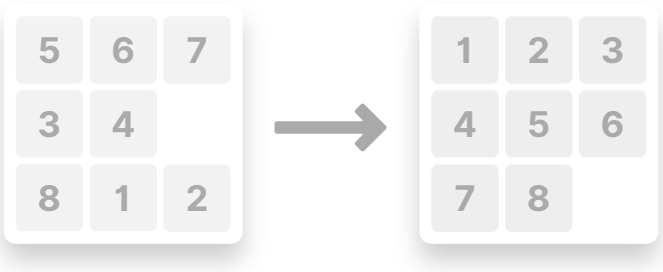

Figure 1 – Un départ possible et l'arrivée voulue

Les tuiles ne peuvent se déplacer que par glissement dans la seule case vide à un moment donné. Le jeu consiste à replacer les tuiles dans l'ordre numérique. Ainsi, par le biais de ce projet, nous allons donc, à l'aide du langage de programmation `Python`, développer ce jeu et répondre aux deux questions suivantes : Quelle est la séquence minimale de mouvements à faire sur un taquin pour obtenir la solution ? Comment trouver cette séquence ?

0.2 Etude du cas général

Pour ce faire, nous détaillerons dans cette partie l'algorithme $A*$, prononcer A star, utilisé pour résoudre le problème, la définition des états et des actions que l'agent pourra réaliser dans chaque état puis nous aborderons spécifiquement la stratégie de recherche utilisée par l'algorithme dans le but d'obtenir une solution optimale.

0.2.1 Définition d'un état

Afin de suivre précisément l'évolution de la résolution du problème, il convient de formaliser tout d'abord ce qu'est un état, et l'environnement dans lequel il évolue. Ainsi, par l'intermédiaire de la programmation orientée objet, nous pouvons définir l'environnement et les états par les attributs suivants :

Environnement

`sizes` : les dimensions du taquin, largeur et taille

`choices` : les heuristiques choisies pour une exécution, soient les identifiants de chaque pondération ou de l'heuristique de la Mauvaise Place par exemple

weightings : les pondérations utilisées pour une exécution

moves : l'historique des coups joués par l'utilisateur

end : la solution trouvée par l'algorithme

Taquin

environment : l'environnement dans lequel évolue l'état

previous : la référence au taquin précédent – parent

sequence : l'ordre des tuiles dans le taquin, représenté par une liste, le vide vaut 0 ; à l'état initial la sequence est une liste remplie aléatoirement

inv : le nombre d'inversions, c'est-à-dire, le nombre de fois pour chaque élément de la sequence où celui-ci est plus grand que chacun des éléments suivants

dis : l'abréviation de l'anglais *disorder*, désordre, soit le nombre de tuiles qui ne sont pas à la place occupée dans l'état final

man : la distance de Manhattan brute, n'ayant subie aucune pondération

path : le chemin emprunté par l'algorithme pour atteindre l'état actuel, réprésenté par une suite de lettres : **L** pour gauche (left), **R** pour droite (right), **U** pour haut (up) et **D** pour bas (down)

moves : la liste des prochains coups possibles à partir de l'état actuel

h : la somme des calculs de chaque heuristique utilisée

g : le côut d'un chemin allant de l'état initial à l'état actuel représenté par un entier

f : la fonction d'évaluation : $f(n) = g(n) + h(n)$ avec n le taquin actuel

0.2.2 Détail des heuristiques

Dans le cadre de l'utilisation de l'algorithme $A*$, le choix d'heuristiques est nécessaire. De fait, nous aurons recours à 7 heuristiques, surestimant la longueur du chemin à parcourir pour atteindre l'état final. Les 6 premières sont des distances dérivées de la distance de Manhattan, correspondant aux heuristiques **H.1**, **H.2**, **H.3**, **H.4**, **H.5**, **H.6** et pondérées par les jeux de poids suivants :

	1	2	3	4	5	6	7
8							