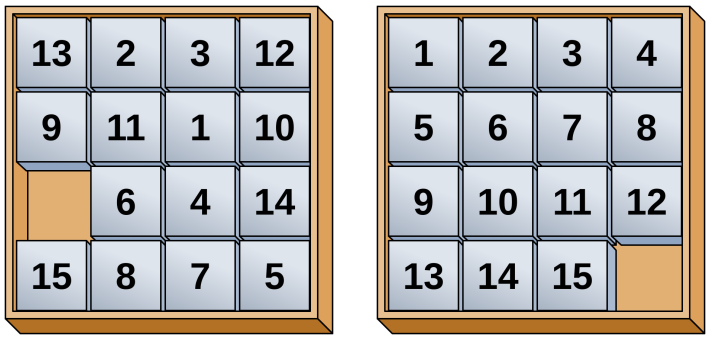
Jeu de taquin : résolution heuristique

Nous avons obtenu avec le parcours en largeur une résolution exacte du jeu de Taquin. Le problème est le nombre trop important de sommets explorés par ce parcours qui limite rapidement la résolution pratique dès que le taquin devient plus grand. Pour résoudre cette difficulté, on fait appel à des méthodes heuristiques.



Taquin à résoudre Taquin résolu

Figure 1 Jeu de Taquin (sources: (1) et (2))

1 Approche heuristique

Question 1

a. Rappeler la définition de distance de Manhattan entre deux cases d'une grille.

b. Proposer une heuristique h : *grille* $\rightarrow$ *int* qui estime le nombre de déplacements restants pour résoudre un jeu de taquin. Cette heuristique reposera sur la distance de Manhattan.

c. L'heuristique est-elle admissible ? Justifier.

d. Implémenter la fonction heuristique.

Question 2

À l'aide de la résolution obtenue précédemment, afficher à l'écran pour une 20aine de taquins alétoire, la plus courte solution et la valeur de l'heuristique. Comparer.

2 Algorithme best-first search

Question 3

Programmer un petit module `Fileprio` qui représente une file de priorité min et dont l'interface sera

```
type ('a, 'b) t (* a: type de la clef, b: type de la valeur *)
val make : unit -> ('a, 'b) t
val push : 'a -> 'b -> ('a, 'b) t -> unit
val popmin : ('a, 'b) t -> 'a
val size : ('a, 'b) t -> int
```

on pourra se satisfaire initialement d'une implémentation simple à l'aide d'une liste triée.

Question 4

L'algorithme *best-first* consiste à parcourir le graphe des configurations depuis le taquin de départ, en choisissant parmi les sommets à explorer un nouveau sommet de meilleure heuristique. Autrement dit, la file du parcours en largeur est remplacée par une file de priorité ordonnée selon les valeurs de l'heuristique.

a. Implémenter cet algorithme.

b. Comparer sur plusieurs exemples la longueur des solutions obtenues entre les deux méthodes.

c. Comparer également le nombre de nœuds explorés dans chaque cas.

3 Algorithme A*

On rappelle que l'algorithme A* est similaire à l'algorithme *best-first* mais au lieu de choisir le sommet qui minimise h, on choisit le sommet qui minimise h + longueur chemin deja construit.

Question 5

On souhaite implémenter l'algorithme A*

a. Quelle est la différence importante lorsqu'on redécouvre un sommet dans A* qui n'existe pas avec *best-first* ?

b. Quelle opération doit-on ajouter à la structure de file de priorité ?

c. Quelle information supplémentaire peut-on ajouter dans le dictionnaire ?

d. Implémenter l'algorithme A*.

e. Comparer les solutions obtenues avec les deux méthodes précédentes : à la fois la longueur des solutions et la taille de l'espace de recherche.

4 Sources

1. Par Booyabazooka — http://en.wikipedia.org/wiki/Image:15-puzzle.svg, Domaine public, https://commons.wikimedia.org/w/index.php?curid=1059593
2. Par Theon — own work, modified from the original file Image:15-puzzle.svg, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=3759824