Puzzle automatique

Lire tout le document avec de commencer.

1 A vous de jouer

On commence tranquillement pour comprendre ce qui vous attend : A vous de jouer

2 A vous de coder

Voici les objectifs à atteindre :

Level 1 :

1. Afficher une session de jeu. (En utilisant une programmation orienté objet)
2. Proposer au joueur de revenir un coup en arrière.
3. Proposer au joueur d'abandonner la partie.
4. Proposer au joueur qui abandonne la solution du puzzle.

Level 2 :

1. Générer un puzzle aléatoire.
2. Construire une solution du puzzle (si elle existe).

Vous trouverez ci-dessous une façon de faire. Vous n'êtes pas obligé de suivre toutes les étapes ci-dessous mais vous êtes obligé de répondre aux objectifs décrits ci-dessus et avoir un rendu semblable à ceux des annexes.

Le code doit être écrit en python et être personnel. (Certain moteur de recherche sont très très fort pour détecter le plagiat !)

2.1 Level 1

Détails des attentes du niveau 1 :

Vous trouverez en annexe une courte version d'une session de jeu où successivement on peut découvrir :

— la situation actuelle du jeu

— une invite à proposer un nouveau déplacement de la case vide

— la réponse formulée par le joueur en un seul caractère.

1. `class Tuile`

   Les tuiles sont définie par leur numéro dans la configuration du motif initial. sauf pour la tuile représentant le trou qui sera désigné par le caractère espace.

   Créer une `classe Tuile` qui comportera :

   — le constructeur

   — une méthode pour obtenir la valeur d'un tuile.

   — une méthode pour afficher la valeur d'un tuile.

   — une méthode pour savoir si une tuile est un trou.

2. `class Puzzle`

   a. Créer une `classe Puzzle`. Créer le constructeur de cette class equi prendra en paramètre la largeur, la hauteur du puzzle et une liste de tuiles (à la création le puzzle est bien rangé et le trou est en bas à droite : situation initiale)

   b. Définissez une méthode permettant d'afficher un puzzle. Par exemple :

```
+----+----+----+
|  1 |  2 |  3 |
+----+----+----+
|  4 |  5 |  6 |
+----+----+----+
|  7 |  8 |    |
+----+----+----+
```

c. Définissez les méthodes `get_largeur`, `get_longueur`, `get_listeTuiles`.

d. Définissez la méthode `get_Trou` qui fournira la position du trou.

e. Définissez la méthode `set_listeTuiles` qui prendra une liste en paramètre et qui modifiera le puzzle avec les valeurs de la liste.

f. Définissez un nouveau constructeur de la `classe Puzzle` qui permet de créer un puzzle à partir de la donnée d'un tableau de tuiles (non nécessairement rangées).

g. Définissez la méthode `nbCasesEnPlace` qui renvoie le nombre de cases qui sont à leur place par rapport à la configuration initiale.

h. Définissez la méthode `isFinish` qui renvoie un booléen informant si le puzzle est terminé.

i. Définissez la méthode `EnumereDeplacement` renvoie sous forme d'une liste les directions possibles pour le trou.

j. Définissez la méthode `deplacement` dont la documentation est :

```
# Réalise, quand il est possible, le déplacement du trou dans
# la direction de déplacement indiqué.
# param depl( type str : 'h', 'b', 'g' ou  'd') la direction du déplacement du trou à  réaliser
# exceptionDeplacementImpossibleException si le déplacement n'est pas possible
# dans la direction demandée.
# Effet de bord : Modification du puzzle
```

k. Définissez la méthode `save_undo` prenant en paramètre un déplacement et qui l'ajoute à une liste `liste_depl` où `liste_depl` est la liste des déplacements depuis le début de la partie.

l. Définissez une méthode `randomPuzzleV1` qui consiste à mélanger le puzzle en réalisant 50 déplacements du trou dans une direction aléatoire (les tentatives de déplacements impossibles ne sont pas comptabilisées dans les 50). Cette méthode retournera la liste des déplacements effectués.

m. A l'aide de ce qui précède, définissez une méthode `play` permettant de jouer et d'avoir un rendu semblable à l'annexe 1.

2.2 Level 2

Détails des attentes du niveau 2 :

1. Définissez une méthode `randomPuzzleV2` qui construit aléatoirement un puzzle. Contrairement à la version `randomPuzzleV1`. Le puzzle obtenu peut ne pas avoir de solution. Par exemple :

```
+----+----+
|    |  2 |
+----+----+
|  3 |  1 |
+----+----+
```

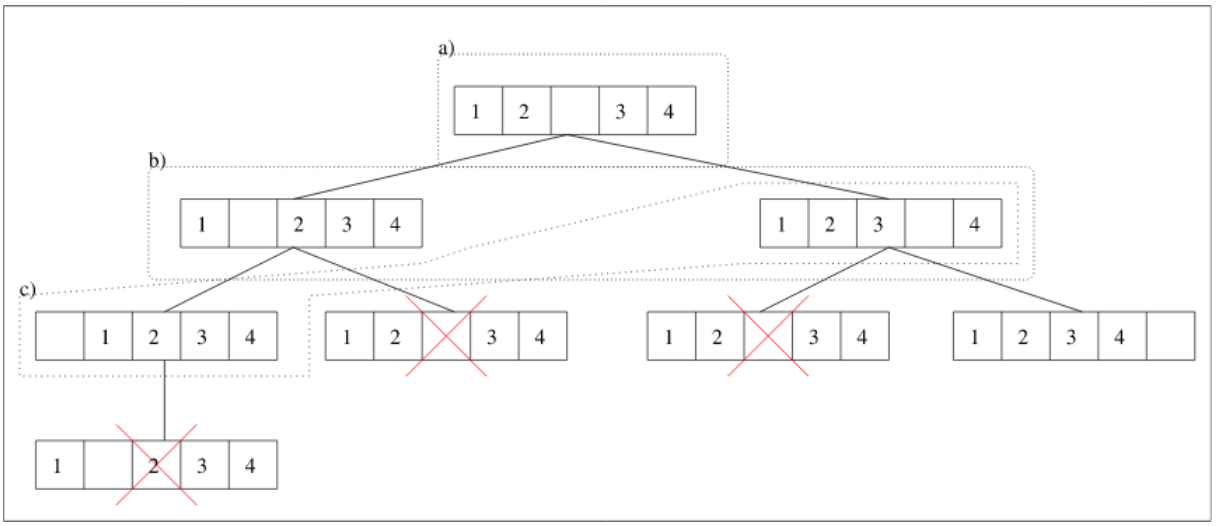
2. Définissez la méthode `neighbourList` qui renvoie la liste des objets de type `puzzle` qu'il est possible d'obtenir après un seul déplacement du trou dans chacune des directions possibles. Exemple :

```
+----+----+----+          +----+----+----+          +----+----+----+          +----+----+----+
|  1 |  2 |  3 |          |  1 |  2 |  3 |          |  1 |  2 |    |          |  1 |  2 |  3 |
+----+----+----+          +----+----+----+          +----+----+----+          +----+----+----+
|  4 |  5 |    | ------>  |  4 |  5 |  6 |          |  4 |  5 |  3 |          |  4 |    |  5 |
+----+----+----+          +----+----+----+          +----+----+----+          +----+----+----+
|  7 |  8 |  6 |          |  7 |  8 |    |          |  7 |  8 |  6 |          |  7 |  8 |  6 |
+----+----+----+          +----+----+----+          +----+----+----+          +----+----+----+
```

Le puzzle de gauche a pour voisin les trois puzzles de droite, atteignables après le déplacement du trou vers le bas, le haut et la gauche. Le déplacement du trou vers la droite n'est pas possible.

3. Avec l'aide ci-dessous, réaliser la méthode `solve` qui permet de résoudre un puzzle.

Avant de commencer le travail, nous allons illustrer cet algorithme sur un puzzle simple : prenons l'exemple d'un puzzle en une dimension à 5 cases. La position initiale (notée 1 2 * 3 4) est dessinée en haut de la figure ci-dessous :



On considère que la position gagnante est 1 2 3 4 *. Nous allons décrire l'algorithme pour résoudre le puzzle :

`candidates` et `seen` sont deux structures de données permettant de stocker des puzzles.
Initialement ils contiennent le puzzle de départ.
A chaque étape, on extrait une configuration de `candidates`, on en génère tous les puzzles voisins (methode `neighbourList`) appelé "fils" puis on ajoute à `candidates` et à `seen` les voisins qui ne sont pas dans `seen`.

Les ensembles de configurations a), b) et c), délimités en pointillés sur la figure ci-dessus, indiquent l'évolution de `candidates` lors des 3 premières étapes (au b) , c ?est la configuration 1 * 2 3 4 qui a été extraite - Nous discuterons de ce choix plus tard -). Remarquez que les configurations barrées ne sont pas ajoutées à `candidates` (ni à `seen`) puisqu'elles sont déjà présentes dans `seen` au moment où l'on essaye de les ajouter.

L'algorithme se termine lorsqu'il atteint une configuration gagnante, ou lorsque `candidates` devient vide. Ainsi on obtient une structure arborescente (ou arbre d'exploration) représentant l ?ensemble de mouvements valides obtenus à partir de la racine (puzzle de départ)

Voici le pseudo code de cet algorithme :

`canditates` est une liste qui va contenir à chaque instant un ensemble de configurations de puzzle différentes qu'il reste à examiner. Initialement cette liste contient le puzzle de départ.
`seen` est un dictionnaire où l'ensemble des clefs correspond à l'ensemble des configurations déjà examinées. Les valeurs associées correspondent au parent. (Cela va nous permettre de retrouver notre chemin).
Initialement ce dictionnaire contient le puzzle de départ avec pour valeur `None`.
`found` est un booléen égal à `False`.

Tant que la liste `candidate` n'est pas vide et que `found` n'est pas égale à `True` :
    Définir la variable `current` égale au premier élément de `canditates`.
    Supprimer `current` de candidates
    Si `current` correspond à la configuration finale
        Affecter `True` à `found`
    sinon
        Pour tous les voisins de `current` :
            Si ce voisin n'est pas dans `seen`
                Mettre ce voisin dans `seen` avec comme valeur `current`
                Ajouter ce voisin à la liste `candidate`
            Fin du si
        Fin du Pour
    Fin du Si
Fin du Tant que.
Si `found` est égale à `True` :
    Afficher(Solution trouvée :)
    executer la procédure buildSolution(current,seen)
else :
    "Il est impossible de résoudre ce puzzle"
Fin du Si.

4. Dans l'algorithme ci-dessus, la variable `canditates` a t-elle une structure de pile ou de file ? Le parcours de l'arbre ce fait-il en largeur ou en profondeur ?

5. Écrire une procédure buildSolution(cend,parent) qui permet d'afficher les puzzles menant au puzzle rangé.

6. Amélioration 1

a. Dans le pseudo code ci-dessus, expliquer pourquoi la ligne "définir la variable `current` égale au premier élément de `canditates`" n'est pas un choix très judicieux.

b. Parmi les puzzles présents dans `canditates`, lequels est-il préférable de choisir ?

c. Mettre en œuvre cette amélioration.

7. Amélioration 2

Faire en sorte d'avoir un affichage similaire à celui de l'annexe 2.

Annexe 1

```
+----+----+----+----+
|  1 |  3 |  5 |  4 |
+----+----+----+----+
| 14 |  2 |  8 | 12 |
+----+----+----+----+
|  6 | 10 |  7 |    |
+----+----+----+----+
|  9 | 13 | 11 | 15 |
+----+----+----+----+
Votre coup? ((H)aut, (B)as, (G)auche, (D)roit, (A)bandon, (R)etour) G
+----+----+----+----+
|  1 |  3 |  5 |  4 |
+----+----+----+----+
| 14 |  2 |  8 | 12 |
+----+----+----+----+
|  6 | 10 |    |  7 |
+----+----+----+----+
|  9 | 13 | 11 | 15 |
+----+----+----+----+
Votre coup? ((H)aut, (B)as, (G)auche, (D)roit, (A)bandon, (R)etour) H
+----+----+----+----+
|  1 |  3 |  5 |  4 |
+----+----+----+----+
| 14 |  2 |    | 12 |
+----+----+----+----+
|  6 | 10 |  8 |  7 |
+----+----+----+----+
|  9 | 13 | 11 | 15 |
+----+----+----+----+
Votre coup? ((H)aut, (B)as, (G)auche, (D)roit, (A)bandon, (R)etour) D
+----+----+----+----+
|  1 |  3 |  5 |  4 |
+----+----+----+----+
| 14 |  2 | 12 |    |
+----+----+----+----+
|  6 | 10 |  8 |  7 |
+----+----+----+----+
|  9 | 13 | 11 | 15 |
+----+----+----+----+
Votre coup? ((H)aut, (B)as, (G)auche, (D)roit, (A)bandon, (R)etour) b
+----+----+----+----+
|  1 |  3 |  5 |  4 |
+----+----+----+----+
| 14 |  2 |  8 | 12 |
+----+----+----+----+
|  6 | 10 |  7 |    |
+----+----+----+----+
|  9 | 13 | 11 | 15 |
+----+----+----+----+
Votre coup? ((H)aut, (B)as, (G)auche, (D)roit, (A)bandon, (R)etour) t
```

```
Votre coup? ((H)aut, (B)as, (G)auche, (D)roit, (A)bandon, (R)etour) T
Votre coup? ((H)aut, (B)as, (G)auche, (D)roit, (A)bandon, (R)etour) z
Votre coup? ((H)aut, (B)as, (G)auche, (D)roit, (A)bandon, (R)etour) g
+----+----+----+----+
|  1 |  3 |  5 |  4 |
+----+----+----+----+
| 14 |  2 |  8 | 12 |
+----+----+----+----+
|  6 | 10 |    |  7 |
+----+----+----+----+
|  9 | 13 | 11 | 15 |
+----+----+----+----+
Votre coup? ((H)aut, (B)as, (G)auche, (D)roit, (A)bandon, (R)etour) A
Avec un peu de persévérance vous auriez pu trouver :
Solution à partir de la situation donnée au début  :
['g', 'h', 'd', 'b', 'g', 'h', 'g', 'b', 'b', 'd', 'b', 'd', 'h', 'd', 'h',
 'h', 'g', 'b', 'b', 'g', 'h', 'd', 'b', 'd', 'h', 'g', 'g', 'b', 'd', 'h',
 'd', 'b', 'g', 'h', 'h', 'd', 'b', 'b', 'g', 'h', 'd', 'h', 'g', 'b', 'b',
 'g', 'g', 'h', 'h', 'd', 'b', 'g', 'b', 'd', 'h', 'h', 'g']
```

Annexe 2

```
>>> %Run Puzzle.py
  T  |   1  |   3  |   4  |
  5  |   2  |   7  |  10  |
  9  |   8  |   6  |  11  |
 13  |  14  |  15  |  12  |
Résolution en cours ...
['g', 'h', 'h', 'g', 'g', 'b', 'd', 'd', 'h', 'g', 'b', 'g', 'h',
 'd', 'd', 'b', 'g', 'g', 'h', 'd', 'b', 'd', 'h', 'g', 'g', 'h']
problème résolu en  1.16  secondes
nb de puzzles testés (len(seen)) :  413

>>> %Run Puzzle.py
  T  |   3  |   7  |   4  |
 10  |   1  |   2  |   8  |
  9  |   6  |   5  |  12  |
 13  |  14  |  11  |  15  |
Résolution en cours ...
['h', 'g', 'h', 'g', 'b', 'b', 'd', 'd', 'h', 'h', 'g', 'g', 'b', 'd', 'd', 'h', 'g', 'b', 'g',
 'h', 'h', 'g', 'b', 'd', 'd', 'h', 'g', 'b', 'g', 'h', 'd', 'd', 'b', 'b', 'b', 'd', 'h', 'h',
 'g', 'b', 'd', 'b', 'g', 'h', 'h', 'h', 'd', 'b', 'g', 'b', 'd', 'h', 'h', 'g', 'b', 'd', 'b',
 'g', 'h', 'h', 'g', 'g', 'b', 'd', 'h', 'd', 'b', 'g', 'g', 'h', 'd', 'd', 'd', 'b', 'g', 'h',
 'g', 'b', 'd', 'h', 'd', 'b', 'g', 'g', 'h', 'd', 'b', 'd', 'h', 'g', 'g', 'g']
problème résolu en  133.49  secondes
nb de puzzles testés (len(seen)) :  5842

>>> %Run Puzzle.py
  2  |  14  |   3  |   4  |
  1  |   T  |   6  |   8  |
  5  |  13  |   7  |  11  |
 15  |   9  |  10  |  12  |
Résolution en cours ...
['g', 'h', 'g', 'h', 'd', 'd', 'b', 'd', 'h', 'g', 'b', 'b', 'b',
'd', 'h', 'g', 'h', 'h', 'd', 'b', 'g', 'b', 'd', 'h', 'h', 'g',
'g', 'b', 'd', 'b', 'g', 'h', 'h', 'd', 'b', 'b', 'g', 'h', 'h',
'd', 'b', 'g', 'b', 'd', 'h', 'h', 'g', 'b', 'd', 'b', 'g', 'h',
'd', 'd', 'h', 'g', 'b', 'g', 'h', 'd', 'd', 'b', 'g', 'h', 'g',
'g', 'b', 'd', 'h', 'd', 'b', 'g', 'g', 'h']
problème résolu en  83.99  secondes
nb de puzzles testés (len(seen)) :  4286

>>> %Run Puzzle.py
  T  |   6  |   8  |
  1  |   3  |   5  |
  7  |   2  |   4  |
Résolution en cours ...
['h', 'g', 'g', 'b', 'd', 'h', 'h', 'g', 'b', 'b', 'd', 'h', 'd', 'h',
'g', 'g', 'b', 'b', 'd', 'h', 'g', 'b', 'd', 'h', 'g', 'h', 'd', 'b',
'd', 'h', 'g', 'b', 'b', 'd', 'h', 'h', 'g', 'b', 'd', 'b', 'g', 'h', 'h', 'g']
problème résolu en  14.76  secondes
nb de puzzles testés (len(seen)):  2561
```