JEU 2048

Vous trouverez certaines informations ici.

Vous pouvez aussi télécharger, sur votre téléphone, une application pour jouer et bien comprendre la règle du jeu.

L'objectif de ce projet est de créer un programme permettant, à un utilisateur, de jouer directement sur la console Python (input) ou via une interface graphique (tKinter).

STRUCTURE DE DONNÉES

Une grille de jeu est un tableau (numpy) à deux dimensions avec 4 lignes et 4 colonnes. Les éléments sont des puissances de 2 (au moins égales à 2) ou 0 si la tuile est vide.

La grille

2		2	2
4	4		2
	2	2	

est ainsi codée par : np.array([[0,0,0,0],[2,0,2,2],[4,4,0,2],[0,2,2,0]])

DEROULEMENT DU JEU

Au départ, on a une grille contenant 2 valeurs prises dans {2,4}.

A chaque étape,

- l'utilisateur doit déplacer les tuiles, jusqu'au « mur », vers la gauche, la droite, le haut ou le bas.
  Un tel déplacement peut alors engendrer certaines additions de tuiles (cf. ci-dessous pour plus de détails).
- A l'issue d'un déplacement, une des tuiles vides prend la valeur 2 ou 4 de manière aléatoire.

Le jeu se termine si 2048 apparaît (gagné) ou si aucun déplacement ne peut faire évoluer la grille (perdu).

ETAPE AVEC DEPLACEMENT VERS LA GAUCHE

Grille au début de l'étape :

2		2	2
4	2		2
4	4	2	2
8	4	2	2

DÉPLACER LES TUILES, JUSQU'AU « MUR », VERS LA GAUCHE

On procède ligne par ligne (elles sont toutes concernées).

D'abord, on déplace tous les entiers non nuls, jusqu'au « mur », vers la gauche.

Puis, on additionne deux tuiles, en partant de la gauche, lorsqu'elles ont la même valeur.

Ainsi,

[2 , 0 , 2 , 2] devient [2 , 2 , 2 , 0] puis [4 , 2 , 0 , 0] (après l'addition, tous les entiers non nuls sont à gauche)

[4, 2 , 0 , 2] devient [4 , 2 , 2 , 0] puis [4 , 4 , 0 , 0]

[4, 4 , 2 , 2] devient [4 , 4 , 2 , 2] puis [8 , 4 , 0 , 0]

[8, 4 , 2 , 2] devient [8 , 4 , 2 , 2] puis [8 , 4 , 4 , 0]

Grille après déplacement vers la gauche :

4	2		
4	4		
8	4		
8	4	4	

AJOUT ALÉATOIRE D'UNE NOUVELLE VALEUR PRISE DANS {2,4}

Grille à la fin de l'étape :

4	2		
4	4		4
8	4		
8	4	4	

ETAPE SUIVANTE

On poursuit l'exemple précédent.

SI ON DEPLACE VERS LA DROITE

On obtient (avant l'ajout aléatoire d'une nouvelle valeur) :

		4	2
		4	8
		8	4
		8	8

SI ON DEPLACE VERS LE HAUT

On obtient (avant l'ajout aléatoire d'une nouvelle valeur) :

8	2	4	4
16	8		
	4		

SI ON DEPLACE VERS LE BAS

On obtient (avant l'ajout aléatoire d'une nouvelle valeur) :

	2		
8	4		
16	8	4	4

MARCHE A SUIVRE

1. Déclarer une fonction **newGrille** définie de la manière suivante :
   - **Entrée** : sans paramètre
   - **Sortie** : une nouvelle grille de jeu

2. Déclarer une fonction **dptGaucheLigne** définie de la manière suivante :
   - **Entrée** : une liste de 4 entiers (0 ou puissances de 2 au moins égales à 2)
   - **Sortie** : la liste après déplacement vers la gauche

   Ainsi, dptGaucheLigne([2 , 0 , 2 , 2]) retourne [4 , 2 , 0 , 0]

   On pourra consulter l'annexe pour déclarer cette fonction.

3. Déclarer une fonction **dptGaucheGrille** définie de la manière suivante :
   - **Entrée** : une grille au début d'une étape
   - **Sortie** : la grille après déplacement vers la gauche

4. Déclarer une fonction **dptGrille** définie de la manière suivante :
   - **Entrées** : une grille au début d'une étape et une direction (G, D, H, B)
   - **Sortie** : la grille après déplacement suivant la direction

   *On pourra utiliser la fonction **dptGaucheLigne** et s'inspirer de la déclaration de **dptGaucheGrille***

5. Déclarer une fonction **ajoutValeurGrille** définie de la manière suivante :
   - **Entrée** : une grille (juste après un déplacement)
   - **Sortie** : la grille avec une tuile de plus valant 2 ou 4 choisie de manière aléatoire

6. Déclarer une fonction **evolGrille** définie de la manière suivante :
   - **Entrées** : une grille au début d'une étape et une direction (G, D, H, B)
   - **Sortie** : la grille à la fin de l'étape

7. Déclarer une fonction **finJeu** définie de la manière suivante :
   - **Entrée** : une grille (juste après l'ajout aléatoire d'une nouvelle tuile)
   - **Sortie** : la liste [True , 'gagné'] si le jeu est gagné, [True , 'perdu'] s'il est perdu et [False , ''] sinon

8. Déclarer une fonction **jouer** qui permet à l'utilisateur de faire une partie sur la console Python.

Question facultative :

9. Déclarer une fonction **jouerIG** qui permet à l'utilisateur de faire une partie via une interface graphique.

ANNEXE

On considère une ligne [a , b , c , d] juste avant les additions éventuelles (et donc juste après avoir déplacé les entiers non nuls, jusqu'au « mur », vers la gauche).

```
Si a = b
        Alors
        Si c = d
                Alors
                La ligne devient [ 2 * a , 2 * c , 0 , 0 ]
        Sinon  ( c != d )
                La ligne devient [ 2* a , c , d , 0 ]
Sinon ( cas a != b )
        Si b = c
                Alors
                La ligne devient [ a , 2 * b , d , 0 ]
        Sinon ( cas b != c )
                Si c = d
                        Alors
                        La ligne devient [ a , b , 2 * c , 0 ]
                Sinon ( cas c != d )
                        La ligne devient [ a , b , c , d ]
```