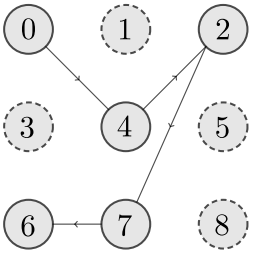
difficile | à trous | graphe | pile

Motifs de déverrouillage

Certains téléphones proposent de déverrouiller l'appareil en dessinant un motif à l'écran. Ce motif est dessiné sur une grille contenant 9 cellules réparties sur 3 lignes de 3. Les cellules sont numérotées de 0 à 8 comme sur la figure ci-dessous qui illustre un motif issu de la cellule 0 et passant par 5 cellules au total.

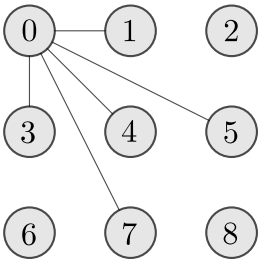


Dans cet exercice, on appelle « motif » un chemin passant par au moins deux cellules. Depuis une cellule donnée, les **seuls** déplacements possibles sont :

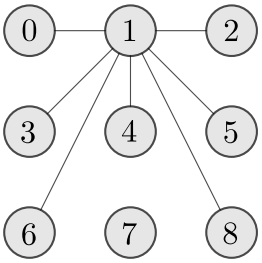
- passer à une cellule directement adjacente (horizontalement, verticalement ou en diagonale) ;
- passer à une cellule située deux lignes et une colonne plus loin ;
- passer à une cellule située deux colonnes et une ligne plus loin.

Les figures ci-dessous illustrent les déplacements possibles.

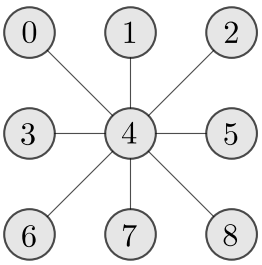
Cellules accessibles depuis 0



Cellules accessibles depuis 1



Cellules accessibles depuis 4



Un motif peut passer plusieurs fois par une cellule donnée mais pas consécutivement. Ainsi, le motif $5 \longrightarrow 6 \longrightarrow 6$ est invalide car il passe deux fois de suite sur la cellule 6.

Enfin, les déplacements passant *par-dessus* une cellule, tels que $0 \longrightarrow 2$ ou $0 \longrightarrow 8$, sont interdits.

On représente un motif en machine par un tuple contenant les numéros des cellules traversées. Ces valeurs sont données dans l'ordre du parcours. Ainsi, le motif illustré plus haut est représenté par le tuple `(0, 4, 2, 7, 6)`.

Écrire la fonction `motifs` qui prend en paramètres les entiers `source` et `longueur` désignant respectivement la cellule de départ et la longueur des motifs souhaités. Cette fonction renvoie la liste des motifs issus de la cellule indiquée et de longueur donnée.

On garantit que la valeur de `source` est un entier entre 0 et 8 (inclus) et que la longueur des motifs cherchés est toujours comprise entre 2 et 6 (inclus).

Exemples

```
>>> # motifs issus de 0 et de longueur 2
>>> motifs(0, 2)
[(0, 1), (0, 3), (0, 4), (0, 5), (0, 7)]
>>> # motifs issus de 1 et de longueur 3
>>> motifs(1, 3)
[(1, 0, 1), (1, 0, 3), (1, 0, 4), (1, 0, 5), (1, 0, 7), (1, 2, 1), (1, 2, 3), (1, 2, 4), (1, 2,
5), (1, 2, 7), (1, 3, 0), (1, 3, 1), (1, 3, 2), (1, 3, 4), (1, 3, 6), (1, 3, 7), (1, 3, 8), (1,
4, 0), (1, 4, 1), (1, 4, 2), (1, 4, 3), (1, 4, 5), (1, 4, 6), (1, 4, 7), (1, 4,
8), (1, 5, 0), (1, 5, 1), (1, 5, 2), (1, 5, 4), (1, 5, 6), (1, 5, 7), (1, 5, 8), (1, 6, 1), (1,
6, 3), (1, 6, 4), (1, 6, 5), (1, 6, 7), (1, 8, 1), (1, 8, 3), (1, 8, 4), (1, 8, 5), (1, 8, 7)]
```

Aides

Plusieurs approches sont possibles. L'une d'elles consiste à effectuer le parcours en profondeur d'un graphe, à l'aide d'une pile par exemple. On empile les motifs construits jusqu'à atteindre la longueur souhaitée.

On rappelle qu'il est possible de créer un tuple contenant un unique élément en faisant `(element,)`.

Enfin, si `a` et `b` sont deux tuples, leur somme `a + b` est un nouveau tuple contenant les valeurs de `a` suivie par celles de `b`. Par exemple `(0, 1) + (5, )` est égal à `(0, 1, 5)`.