Le fakir

Un fakir se déplace sur une planche de braises incandescentes.

- A chaque pas, il se brûle avec une intensité indiquée sur la case.
- Pour traverser, il ne peut que descendre, en choisissant soit la case en face, soit celle de gauche, soit celle de droite (s'il y en a...)
- Toutes les cases de la première ligne peuvent être choisies pour commencer le parcours.

Le fakir cherche à minimiser ses brûlures !

Sous Python, on modélisera :

- la planche par une liste de listes d'entiers représentant l'intensité de la brûlure de la case
- un chemin par une liste d'entiers représentant les colonnes successives des cases à emprunter.

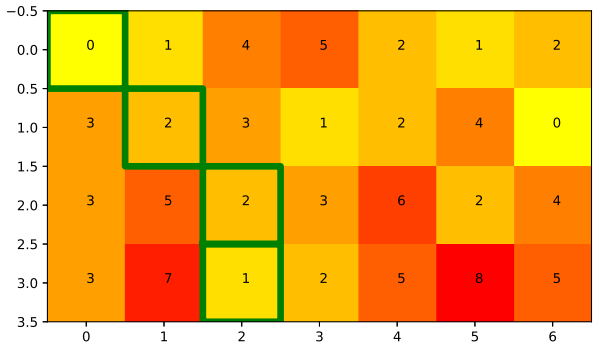
(Le code pour visualiser est donné à titre d'information en fin de sujet, mais n'est pas utile pour répondre aux questions).

Par exemple :

```
planche = [
    [0, 1, 4, 5, 2, 1, 2],
    [3, 2, 3, 1, 2, 4, 0],
    [3, 5, 2, 3, 6, 2, 4],
    [3, 7, 1, 2, 5, 8, 5]]

chemin = [0,1,2,2]
```

Représente la situation :



Exercice 1 Écrire une fonction `verif(P,C)` qui reçoit une planche `P` et renvoie un booléen indiquant si la liste d'entiers `C` représente un chemin respectant les règles de déplacement.. Par exemple :

```
>>> verif(planche, [1])
False
>>> verif(planche, [1,2,2,1])
True
>>> verif(planche, [1,3,2,1])
False
>>> verif(planche, [5,6,7,8])
False
```

Exercice 2 Écrire une fonction `brulure(P,C)` qui reçoit une planche `P` et un chemin valide et renvoie le total des brulures en empruntant ce chemin. Par exemple :

```
>>> brulure(planche, [0,1,2,2])
5
```

Exercice 3 Première solution récursive :

1. Écrire une fonction **récursive** (parcours en profondeur) `brulures_rec(P,i,j)` qui reçoit une planche ainsi que deux indices i et j et retourne le total minimal de brulure qu'il est possible d'obtenir en partant de la case à la ligne `i` et colonne `j`.

```
>>> brulures_rec(planche,1,3)
4
```

2. En déduire une fonction `brulures_mini1(P)` qui reçoit une planche et retourne le minimum de brulures envisageables.

```
>>> brulues_mini1(planche)
5
```

Exercice 4 Modifier les fonctions de l'exercice précédent pour obtenir une fonction `brulures_mini2(P)` renvoyant la brulure minimale ainsi que le chemin à prendre.

```
>>> brulues_mini2(planche)
(5, [0, 1, 2, 2])
```

Exercice 5 Dans l'exercice 3, pour une planche de taille $n \times p$, combien d'appels à la fonction `brulures_rec` sont effectués ?

Exercice 6 Améliorer la version 2 en une version `brulures_mini3(P)` qui utilise une solution récursive Top-Down avec mémoïsation (on pourra utiliser un dictionnaire à cet effet).

A la recherche d'une solution dynamique...

Exercice 7 A l'aide d'une programmation dynamique, écrire une fonction `brulures_mini4(P)` qui ne renvoie dans un premier temps que le total minimal de brulure : On calcule ligne par ligne en partant du haut. A chaque case, on remplace la valeur par la valeur minimale de brulure pour arriver jusqu'à cette case.

Exercice 8 Améliorer enfin en une fonction `brulures_mini5(P)` qui renvoie aussi le chemin à parcourir.

```
from matplotlib import pyplot as plt

def dessine(T, L) :
    plt.imshow(T, cmap='autumn_r')
    for i in range(len(T)) :
        for j in range(len(T[0])) :
            plt.text(j,i,str(T[i][j]))
    for i in range(len(L)) :
        X = [L[i]-0.5,L[i]+0.5,L[i]+0.5,L[i]-0.5,L[i]-0.5]
        Y = [i-0.5, i-0.5, i+0.5, i+0.5, i-0.5 ]
        plt.plot(X,Y,'g-', lw=5)
    plt.show()
```