TP Programmation dynamique

Introduction.

L'objectif de ce TP est d'implémenter l'algorithme de recherche du chemin de poids minimum, en utilisant un paradigme programmation dynamique.

Dans le TD réalisé précédemment, cet algorithme a été appliqué sur la liste de liste ci-contre que l'on nommera « ***pixel*** » indiquant le poids des 4x4 pixels composant une image.

1	2	8	9
7	2	2	1
2	1	4	5
10	8	6	2

Pour rechercher le chemin optimal, un **tableau intermédiaire** nommé « ***tab*** » a été constitué ligne par ligne :

1	2	8	9

⇨

1	2	8	9
8	3	4	9

⇨

1	2	8	9
8	3	4	9
5	4	7	9

⇨

1	2	8	9
8	3	4	9
5	4	7	9
14	12	10	9

On utilise ensuite ce tableau pour rechercher les indices des pixels qui composent ce chemin optimal. Pour cela, on commence par rechercher la valeur minimum sur la ligne du bas. Ensuite on remonte les lignes en sélectionnant le minimum parmi les 2 ou 3 voisins de la ligne supérieure. On constitue ainsi une liste que l'on nommera « *resultat* » qui contient les indices de colonnes des pixels du chemin trouvé. Ici : *resultat = [0 , 1 , 2 , 3]* car en allant du haut vers le bas, les indices colonnes des pixels traversés sont 0 sur la 1ère ligne, puis 1 sur la 2ième ligne, puis 2 et enfin on termine sur l'indice 3 pour la ligne du bas.

Ainsi sur le tableau « ***tab*** », le chemin qui conduit au poids minimum est celui ci-contre et la somme des poids rencontrés est de 9 .

1	2	8	9
8	**3**	4	9
5	4	**7**	9
14	12	10	**9**

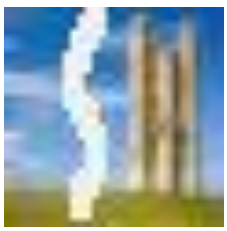
Sur le tableau « ***pixel*** » de départ, ce même chemin donne :
En additionnant les poids des cases rencontrées, on retrouve bien la somme de 9.

1	2	8	9
7	**2**	2	1
2	1	**4**	5
10	8	6	**2**

La mise au point du code complet pouvant être fastidieuse, on part d'une base qui offre l'avantage de proposer une interface Tkinter permettant de visualiser les données initiales et les résultats.

⇨ Télécharger à partir de *nsibranly.fr*, le dossier *redim.zip* . Extraire les 3 fichiers qu'il contient dans votre zone de travail :

- *bibliRedim.py* contient le code de la classe de la classe *Redim()*. Cette classe permet entre autres de générer aléatoirement la liste de listes « ***pixel*** » et ensuite de visualiser sur l'interface Tkinter, le chemin de poids minimum. Ce fichier n'est pas à modifier.



- *redimensionnement.py* contient le code de la classe *RedimTp()* qui hérite des méthodes de la classe *Redim()*. C'est uniquement dans ce fichier que vous rajouterez du code.

- *image.jpg* qui est une image de 30px par 30px.

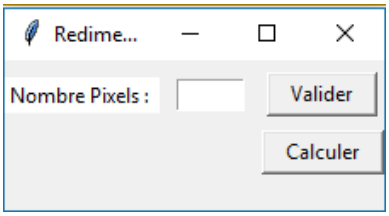
TP Programmation dynamique

L'objectif de ce Tp est de rajouter des méthodes dans la classe *RedimTp()* afin de pouvoir déterminer le chemin de poids minimum.

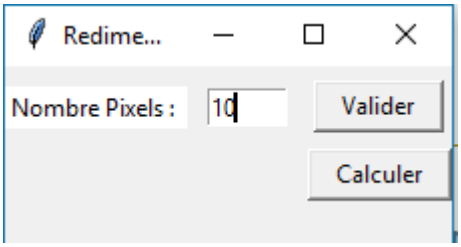
⇨ Ouvrir le fichier *redimensionnement.py* .

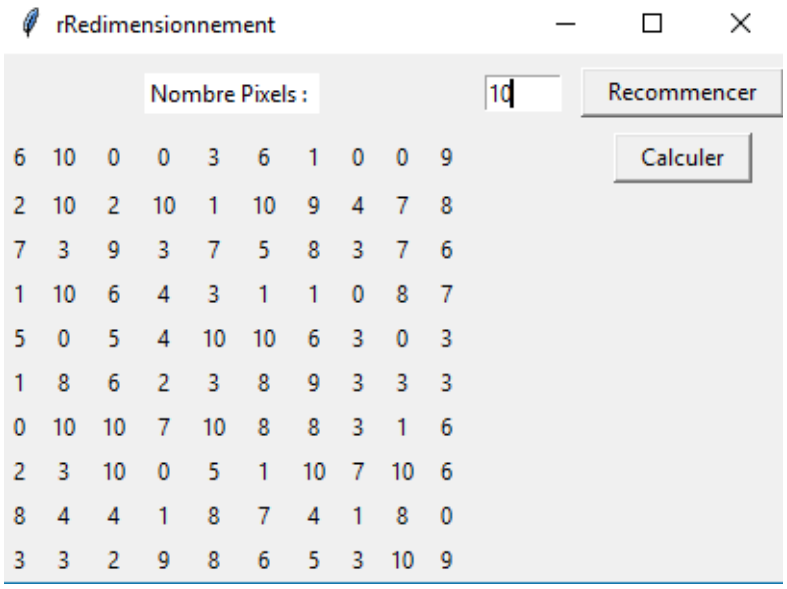
⇨ Exécuter le code

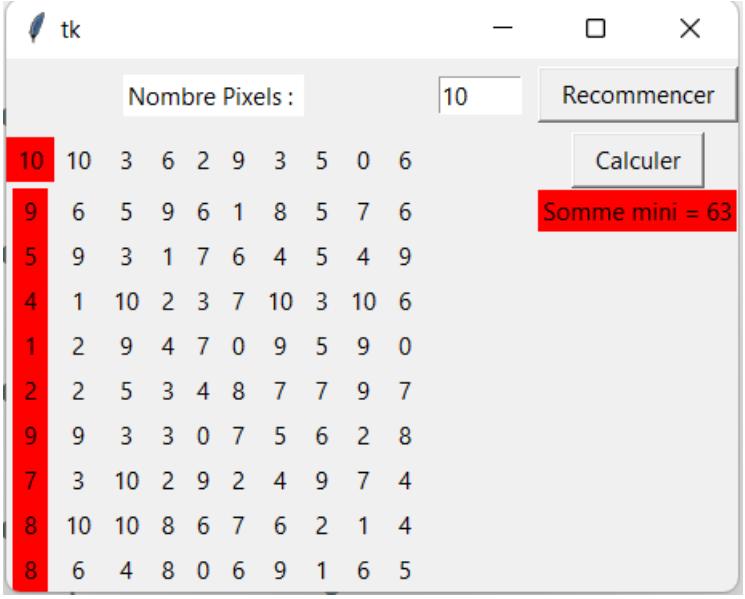
….. la fenêtre Tkinter suivante apparait :



En saisissant une valeur de 10 et en cliquant sur *Valider*, un tableau de 10 lignes et 10 colonnes rempli de nombres entiers aléatoires compris entre 0 et 10 apparait :







En cliquant sur *Recommencer* , en ayant modifié ou pas le nombre de pixels, un nouveau tableau est généré.

En cliquant sur *Calculer* , la colonne de gauche devient rouge pour l'instant.

1. Prise en main du code fourni :

Les différentes parties du code déjà écrit sont détaillées ci-dessous :

```
from bibliRedim import Redim

class RedimTp(Redim) :
    def __init__(self):
        Redim.__init__(self)

    def getPixel(self) :
        return self.pixel

    def setResultat(self,liste) :
        self.resultat = liste

    def redimensionnement(self) :
        pix = self.getPixel()
        n = len(pix)
        resultat = [0 for i in range(n)]


        self.setResultat(resultat)
        self.redWay()


# Main
R = RedimTp()
R.mainloop()
```

La classe RedimTp hérite des méthodes de la classe Redim

En créant une instance de la classe RedimTp, le constructeur de la classe Redim est exécuté. Celui-ci crée entre autres, la fenêtre Tkinter de départ.

Cette méthode retourne la liste de listes « **pixel** » qui est générée aléatoirement et affichée sur l'interface Tkinter. Cette liste est contenue dans l'attribut *pixel* de l'objet

Cette méthode permet de modifier l'attribut *resultat* de l'objet qui contient la liste « ***resultat*** ». Cette liste permet de colorier en rouge le chemin de poids minimum sur l'interface Tkinter.

La méthode *redimensionnement()* est exécutée au clic sur le bouton ***Calculer***.

Pour l'instant, elle permet simplement :

- De récupérer la liste de liste « ***pixel*** » nommée ici *pix[ ]* et de l'afficher dans la console
- De générer une liste *resultat[ ]* initialisée à 0
- De modifier l'attribut *resultat* sur l'objet
- De colorer le chemin en rouge sur l'interface Tkinter (méthode *redWay()*)

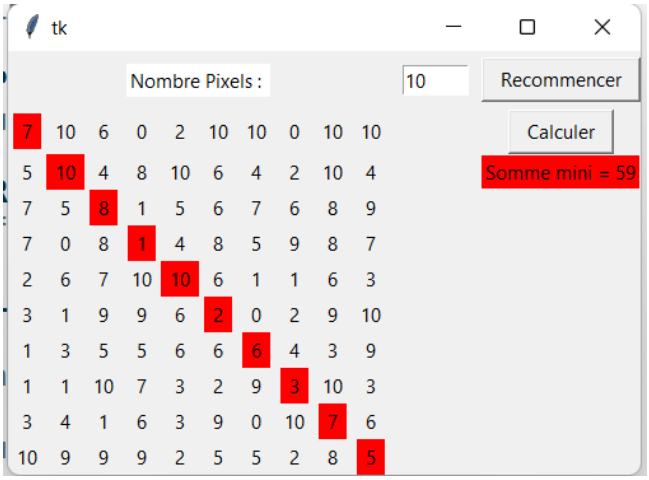
```
def redimensionnement(self) :
    pix = self.getPixel()
    print(pix)
    n = len(pix)
    resultat = [0 for i in range(n)]


    self.setResultat(resultat)
    self.redWay()
```

⇨ Pour prendre en main ce code, modifier par exemple les valeurs d'initialisation de la liste *resultat* :

```
resultat = [i for i in range(n)]
```

En exécutant alors le script, après clic sur le bouton *Calculer*, on obtient :



2. Création du code :

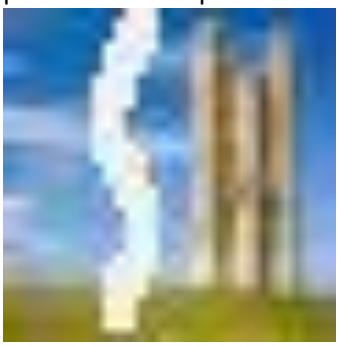
⇨ Compléter la classe *RedimTP* afin qu'elle puisse afficher le chemin de poids minimum. Il s'agit de compléter la méthode *redimensionnement()* et d'en créer d'autres qui permettront au final, de générer une liste de listes « ***tab*** » et à partir de celle-ci, de générer la liste «***resultat*** ».

3. Tests du code obtenu

⇨ Tester le code pour différentes valeurs de n : n = 2, n = 8, n = 20, n = 40. La valeur maximale de n a été bridée à 40.

4. Tests du code sur le cas d'une image

On se propose ici d'utiliser le code réalisé sur une liste de listes *pixel* composés de poids obtenus à partir de l'image ci-contre, de dimension carrée.

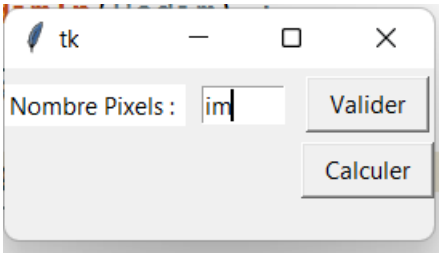


L'interface Tkinter ne supportant pas un traitement pour un nombre n important de pixels, on travaillera ici sur la seconde image donnée à gauche et de taille 30 x 30 pixels uniquement.

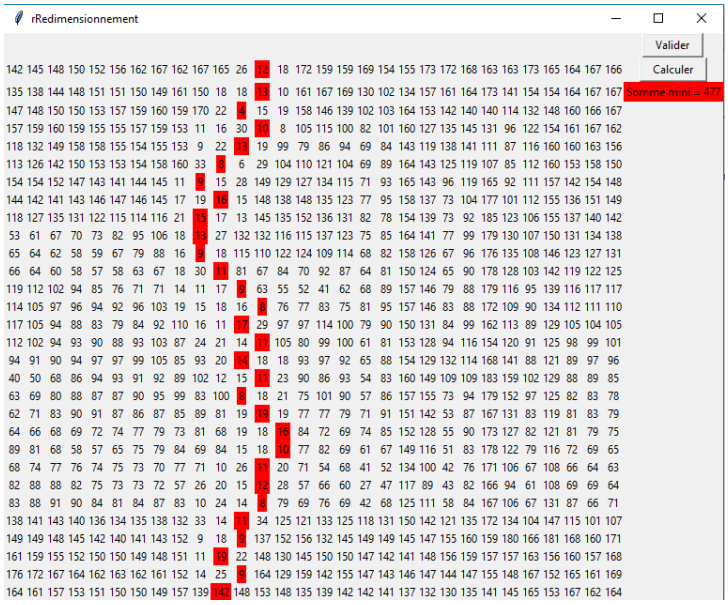
Les poids utilisés correspondront au complémentaire à 216 de la moyenne des valeurs R V B de chaque pixel. Ces poids correspondent ainsi au complémentaire de l'intensité en niveau de gris de chaque pixel. De plus un trait clair a été tracé à main levée à travers l'image afin d'obtenir des poids plus faibles dans le tableau.

⇨ Ouvrir le fichier *image.jpg* avec un visionneur d'image pour vérifier qu'il contient bien l'image ci-dessus.

⇨ Exécuter votre code et en saisissant cette fois ci, non plus le nombre n de pixels de la liste « ***pixel*** », mais le mot clé *im*. En indiquant ce mot clé, la liste « **pixel** » est créée par lecture des valeurs *rgb* de chacun des 30x30 pixels contenus dans le fichier *image.jpg.*



⇨ Cliquer sur *Valider*, puis sur *Calculer* ….



… on retrouve ainsi la trace du trait réalisé dans l'image.

Conclusion.

Ce TP a permis d'implémenter l'algorithme de recherche du chemin de poids minimum, en utilisant un paradigme de programmation dynamique. Il a été utilisé pour trouver un chemin sur une image au format .jpg .
Le paradigme de programmation dynamique utilisé permet de réduire fortement le volume de calculs et donc la complexité du problème, tout en permettant d'obtenir une solution exacte.

Il serait intéressant d'utiliser ce principe pour l'appliquer sur les problèmes de redimensionnement intelligent d'images. Mais cela sort du cadre de ce Tp.