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 le tableau ci-contre 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 la 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.

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

Ainsi sur le tableau de départ, le chemin qui conduit au poids minimum est celui-ci :

et la somme des poids rencontrés est de 9 .

On reprend ces différentes étapes dans le code que vous compléterez dans ce TP.

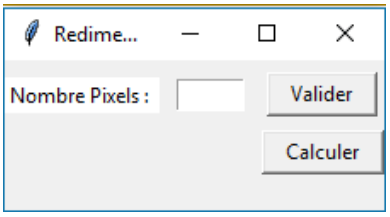
1. Découverte du code de départ avec une interface Tkinter :

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 et les résultats.

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

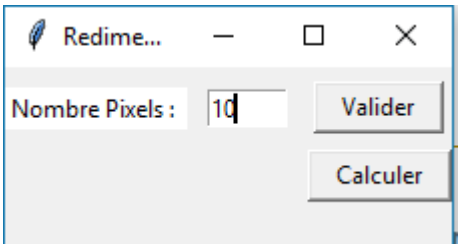
⇨ Ouvrir le fichier nommé « *redimensionnement.py* »

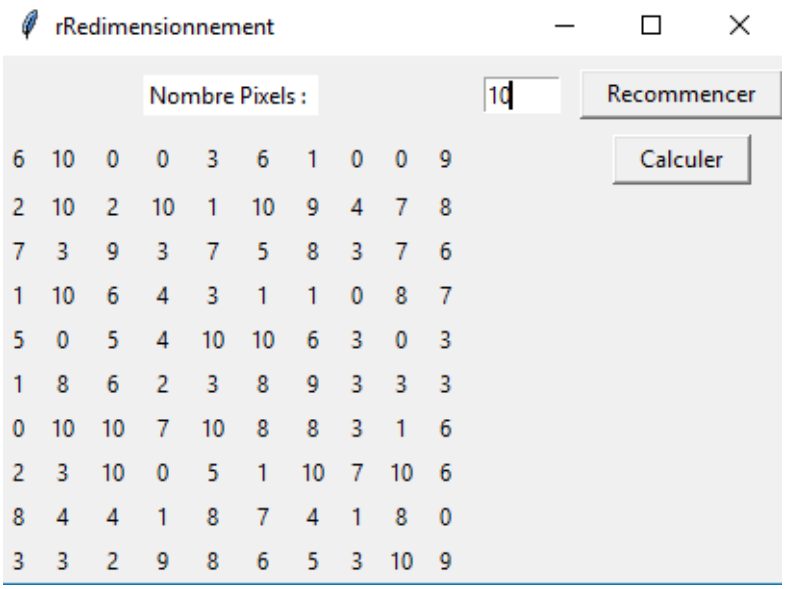
⇨ Exécuter ce code ….. la fenêtre Tkinter suivante apparait :

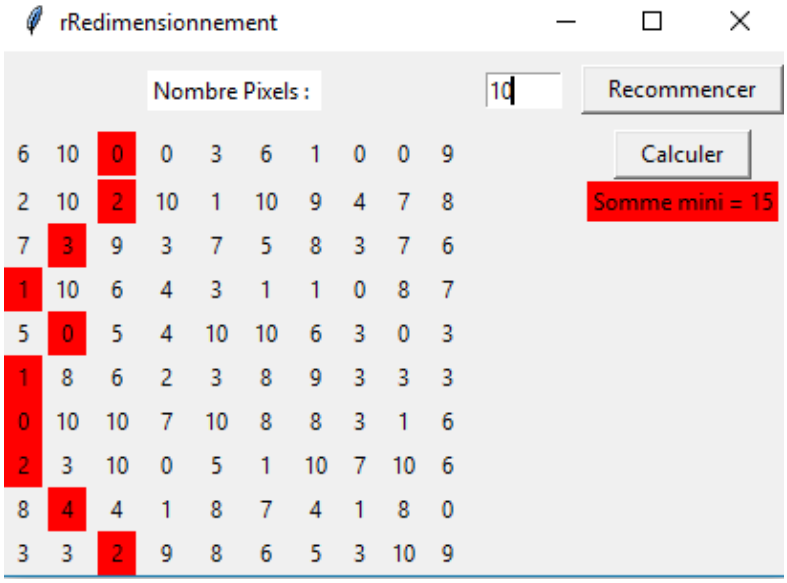


TP Programmation dynamique

En saisissant une valeur de 10 Pixels 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.
L'objectif de ce TP est de réaliser le code qui permettra d'obtenir le chemin de poids minimum, comme indiqué sur la figure ci-contre.

Le code donné au départ est constitué de plusieurs parties.
Seule la partie ------ *Fonctions* ------- sera à compléter. Elle contient en particulier une fonction nommée « *pix()* » qui est appelée au click du bouton *Valider*. Cette fonction ne sera pas modifiée dans ce Tp. Elle permet de générer le tableau précédent nommé « *pixel* » et de gérer son affichage dans la fenêtre Tkinter.

En cliquant sur le bouton *Calculer*, la fonction nommée « *redimensionnement(n,pixel)* » est appelée. Cette fonction passe en argument la valeur *n* du nombre de pixels saisi et la liste *pixel* générée.
L'objectif de ce TP sera de compléter le code lié à cette fonction « *redimensionnement(n,pixel)* ».

2. Création de la table intermédiaire nommée « *tab* » :

La fonction nommée « *redimensionnement(n,pixel)* » est appelée au click sur le bouton *Calculer*. Cette fonction passe en argument la valeur *n* du nombre de pixels saisi et la liste *pixel* générée :

```
def redimensionnement(n,pixel) :
    '''
        Fonction qui gère le calcul de recherche du chemin de poids minimum
    '''
    # création du tableau tab à partir du tableau pixel
    tab = creation_tab(n,pixel)
```

3. Recherche dans la table intermédiaire « *tab* » du chemin de poids minimum :

Pour rechercher le chemin de poids minimum, on appelle la fonction nommée « *recherche_chemin(n,tab)* ». Elle retourne la liste *resultat[]* qui contient les indices du chemin identifié.

```
def redimensionnement(n,pixel) :
    '''
        Fonction qui gère le calcul de recherche du chemin de poids minimum
    '''
    # création du tableau tab à partir du tableau pixel
    tab = creation_tab(n,pixel)
    # création du tableau resultat à partir du tableau tab
    resultat = recherche_chemin(n,tab)
```

4. 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 100.

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

On se propose ici d'utiliser le code réalisé sur un tableau *pixel* composés de poids obtenu à partir de l'image de droite ci-contre de dimension carrée.

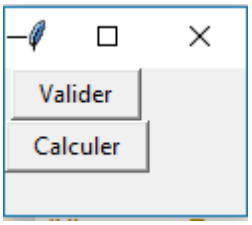
L'interface Tkinter ne supportant pas un traitement pour un nombre n important de pixels, on travaillera ici sur la seconde image donnée à gauche et de taille 30 x 30 pixels uniquement.

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.

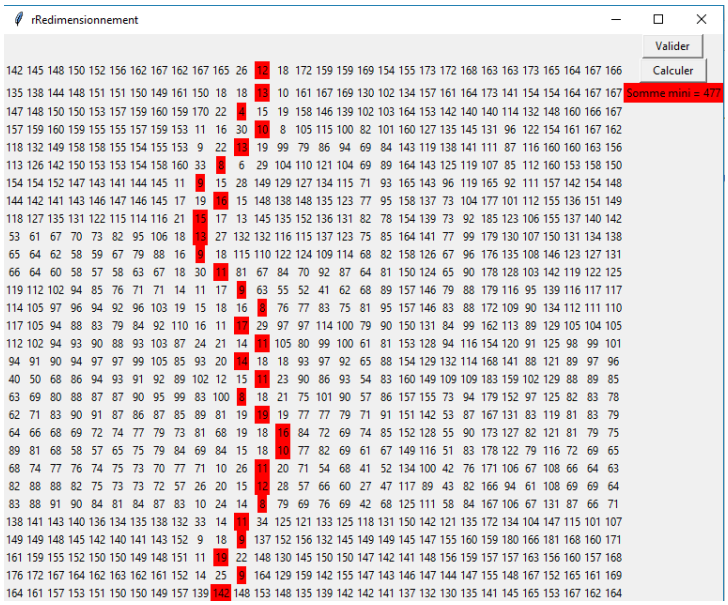
Le fichier nommé « *redimensionnement_image .py*» donné, contient le code que vous avez réalisé. Par contre la génération de la liste *pixel* n'est plus aléatoire puisqu'elle est obtenue à partir des pixels de l'image précédente, enregistrée dans le fichier *image.jpg* .

```
def pixel() :
    '''
        fonction appelée après click sur le bouton Valider ou Recommencer. Permet
        la création et l'affichage du tableau pixel et permet le lancement des calculs
        de recherche du chemin de plus faible poids (ftc "redimensionnement()")
    '''
    global compteur , n_precedent
    somme.configure(text = '' , bg='#f0f0f0' )
    # Création du tableau des pixels
    im = Image.open('image.jpg')
    l, h = im.size
    n = l
    pixel = [[0 for i in range(n)] for j in range(n)]
    for x in range(l):
        for y in range(h):
            r, v, b = im.getpixel((x, y))
            g=(r+v+b)//3
            pixel[y][x] = 256 - g
```

⇨ Ouvrir le fichier *redimensionnement_image .py*



⇨ Exécuter le code : cliquer sur Valider, puis sur calcul ….



… 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.