TNSI	Algorithmique Programmation dynamique	TP – Seam carving

Objectifs :

- ⇨ Mettre en œuvre un algorithme de programmation dynamique
- ⇨ Découvrir la technique du « Seam carving »

D'après img.univ-mlv.fr

I – Principe du rétrécissement intelligent

Le « *Seam carving* » est une technique de rétrécissement d'image permettant de réduire ses dimensions tout en conservant les parties « importantes » de l'image.

Question 1 :
Lire attentivement les explications de la page suivante :
http://igm.univ-mlv.fr/~dr/XPOSE2012/La%20programmation%20dynamique/seamcarving.html

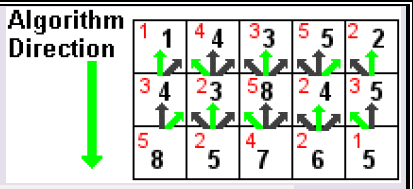
II – Programmation de l'algorithme

On cherche à mettre en œuvre la technique du rétrécissement intelligent et pour ce faire nous allons d'abord implémenter l'algorithme de détection du chemin minimal.

Pour cette partie, on peut utiliser le fichier « TP_Seam_carving.py » dans le répertoire du TP où on complètera les `# TODO`.

1) Détermination des chemins minimaux

Question 2 :



1) Ecrire une fonction `chemin_minimaux(t)` qui renvoi un tableau contenant les sommes cumulées des chemins minimaux dans le tableau `t`.

Si `t=[[1,4,3,5,2],[3,2,5,2,3],[5,2,4,2,1]]` (tableau de départ avec les nombres en rouge de l'exemple ci-contre), alors la fonction renverra `[[1,4,3,5,2],[4,3,8,4,5],[8,5,7,6,5]]`.

2) Vérifier qu'avec `t = [[9, 5, 9, 4, 7, 5, 4], [8, 6, 3, 7, 3, 6, 3], [1, 7, 3, 5, 7, 1, 7], [7, 1, 6, 2, 2, 4, 6], [3, 7, 8, 4, 7, 9, 4], [4, 2, 7, 8, 7, 7, 9]]`
on obtient bien : `[[9, 5, 9, 4, 7, 5, 4], [13, 11, 7, 11, 7, 10, 7], [12, 14, 10, 12, 14, 8, 14], [19, 11, 16, 12, 10, 12, 14], [14, 18, 19, 14, 17, 19, 16], [18, 16, 21, 22, 21, 23, 25]]`.

3) Quelle est la complexité de cet algorithme (en posant *n* le nombre de cases du tableau) ?

2) Reconstitution du plus petit chemin

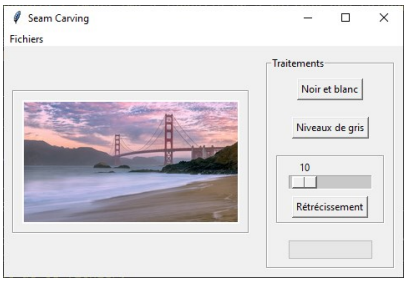
Question 3 :

1) Ecrire une fonction `determine_chemin_minimal(t,chemins)` qui prend en argument le tableau `t` ainsi que le tableau `chemins` qui contient les sommes cumulées des chemins du tableau `t` (c'est le tableau renvoyé par la fonction précédente). Cette fonction doit renvoyer un tableau des abscisses des cases du chemin minimal pour chaque ligne du tableau t.

Sur l'exemple du site, la fonction devrait renvoyer `[0,1,1]` (si on prend le premier chemin minimal rencontré) ou `[4, 3, 4]` si on prend le dernier chemin minimal rencontré.

2) Vérifier que sur le deuxième exemple de la question précédente, on obtient bien `[3, 2, 2, 1, 0, 1]`.
3) Quelle est la complexité de cet algorithme (en posant *n* le nombre de cases du tableau) ?
4) *(Pour les plus rapides uniquement)* Ecrire une fonction `enleve_chemin_minimal(t, chemin_minimal)` qui renvoie le tableau `t` raccourci d'une colonne correspondant au chemin minimal (donné en argument). Ainsi la fonction, appliquée au premier exemple, doit renvoyer `[[4, 3, 5, 2], [3, 5, 2, 3], [5, 4, 2, 1]]`.

III – Mise en œuvre sur des images



Nous allons maintenant chercher à appliquer cet algorithme à des images. L'idée est de déterminer dans un premier temps l'importance des pixels de l'image. Pour cela on va se baser sur le contraste : pour chaque pixel on va calculer la différence entre sa couleur et celle de ses plus proches voisins.

Question 4 :
1) Ouvrir le fichier « Appli.py ». Ce fichier contient le code d'une application tkinter permettant d'afficher l'image, de sélectionner le nombre de colonnes à supprimer et de rétrécir l'image. Observer la structure du code.
2) Compléter le code de la méthode `retire_colonne_image`.
3) Compléter le code de la méthode `retrecissement`.
4) Tester le programme avec diverses (petites) images.

IV – Agrandissement

Pour les plus rapides, il s'agit maintenant de créer une fonction d'agrandissement de l'image en ré-utilisant le travail précédent : pour agrandir l'image sans modifier ses zones importantes, on va chercher le chemin minimal et on va insérer des pixels à droite (ou à gauche) du chemin minimal. Le pixel crée aura la couleur moyenne entre les pixels de gauche et de droite.

Question 5 :
1) Ajouter les widgets nécessaires à « Appli.py » pour qu'on puisse réaliser un élargissement d'un nombre réglable de pixels.
2) Créer une méthode `ajoute_colonne_image` sur le modèle de `retire_colonne_image`.
3) Créer une méthode `agrandissement` de la même manière.
4) Tester le programme avec diverses (petites) images.
5) Modifier l'ensemble des méthodes pour avoir un seul bouton « Agrandir/rétrécir » et un nombre de pixel pouvant donc être négatifs ou positifs.