DS 2 : lundi 16 décembre

2h sans calculatrice

Le candidat encadrera ou soulignera les résultats.
N.B. : le candidat attachera la plus grande importance à la clarté, à la précision et à la concision de la rédaction. Si un candidat est amené à repérer ce qui peut lui sembler être une erreur d'énoncé, il le signalera sur sa copie et devra poursuivre sa composition en expliquant les raisons des initiatives qu'il a été amené à prendre.

Redimensionnement d'image : algorithme de Seam carving

L'objectif de ce sujet est de réduire la largeur d'une image, pour cela on introduira l'algorithme de Seam carving (ou recadrage intelligent, développé par Shai AVIDAN et Ariel SHAMIR en 2007).
L'article se trouve ici : `https://perso.crans.org/frenoy/matlab2012/seamcarving.pdf`

I] Préliminaires

A. SQL

On s'intéresse à une base de données de photographies qui contient trois tables. La première table, `photos`, a cinq colonnes dont `id`, un identifiant unique pour chaque photo. Quelques lignes sont données ci-après.

id	titre	largeur	hauteur	lieu
1	Les goélands sur la plage	640	480	Dunkerque
2	Le petit Quinquin	1024	768	Lille
3	Le chat voyeur	800	600	Paris
⋮	⋮	⋮	⋮	⋮

La seconde table, `animaux`, a 3 colonnes, notamment un identifiant unique pour chaque animal (`id`).

id	nom	type
1	goéland	oiseau
2	chat	mammifère
⋮	⋮	⋮

Et enfin la troisième `estsur`, a 4 colonnes et fait la jonction entre les deux autres tables, elle a un identifiant unique `id`, elle permet de savoir que sur la photo dont l'`id` est `idphoto` il y a un animal dont l'`id` est `idanimal` et en précise le nombre (par exemple il y a 2 goélands sur la première photo et 1 chat sur la troisième, mais aussi un goéland sur la troisième).

id	idphoto	idanimal	nombre
1	1	1	2
2	3	2	1
3	3	1	1
⋮	⋮	⋮	⋮

Les questions suivantes demandent d'écrire des requêtes SQL.

Q1. Écrire une requête SQL renvoyant les titres des photos prises à "Dunkerque".

Q2. Écrire une requête SQL renvoyant les titres et les lieux des photos qui ont une largeur au moins égale à 1024 et une hauteur au moins égale à 768.

Q3. Écrire une requête SQL renvoyant le nombre de photos de chaque ville

Q4. Écrire une requête SQL renvoyant les lieux où il y a une photo contenant un chien.

Q5. Écrire une requête SQL renvoyant les titres des photos où il y a des animaux différents dessus.

B. Fonction basique

La suite du devoir se réalisera en utilisant le langage python.

Q6. Écrire une fonction `imin` qui prend en argument une liste et qui retourne la position de la première occurrence du minimum.
Par exemple `imin([132,128,30,250,30])` doit retourner 2.

C. Traitement d'images

Dans ce sujet on considère qu'une image en niveau de gris est une matrice I d'entiers compris entre 0 (pour noir) et 255 (pour blanc), on note h le nombre de lignes (ie la hauteur de l'image) et w le nombre de colonnes (ie la largeur de l'image). Cette matrice sera représentée par la liste `img` de ses lignes (une ligne étant une liste d'entiers). Ainsi pour $i \in [\![0, h-1]\!]$ et $j \in [\![0, w-1]\!]$ le pixel en position (i, j) de l'image, que l'on notera $I_{i,j}$ correspondra en `Python` à `I[i][j]`.

Q7. Écrire une fonction `dim` qui prend en argument une image `img` et qui retourne le couple d'entiers `(w,h)` où `h` est la hauteur et `w` est la largeur de l'image.

Q8. Écrire une fonction d'entête `nulle(w,h)` qui retourne une liste de liste qui correspond à la matrice nulle de $\mathcal{M}_{h,w}(\mathbb{R})$. Dit autrement, si on voit la matrice comme une image, elle retourne l'image complètement noire de largeur w et de hauteur h.

Q9. Que fait la fonction suivante ?

```
def mystere(img):
    (w,h)=dim(img)
    res=nulle(w,h)
    for i in range(h):
        for j in range(w):
            res[i][j]=img[i][w-1-j]
    return res
```

II] Redimensionnement naïf

Pour réduire la largeur de moitié on peut prendre un pixel sur deux dans chaque ligne, par exemple :

4	3	1	25
12	1	15	22
28	20	20	23
132	128	32	250

$\longmapsto$

4	1
12	15
28	20
132	32

On supposera que la largeur de l'image est paire.

Q10. Écrire une fonction `reduc_moitie` qui prend en argument une image `img` et qui retourne une image deux fois moins large avec la méthode naïve qui vient d'être présentée.

Pour éviter de perdre complètement les pixels qu'on supprime on peut fusionner les pixels $I_{i,2k}$ et $I_{i,2k+1}$ en un unique pixel contenant comme valeur $(I_{i,2k} + I_{i,2k+1})//2$, ce qui donne sur notre exemple :

4	3	1	25
12	1	15	22
28	20	20	23
132	128	32	250

$\longmapsto$

3	13
6	18
24	21
130	141

Q11. Écrire une fonction `reduc_moitiev2` qui prend en argument une image `img` et qui retourne une image deux fois moins large avec la méthode naïve qui vient d'être présentée.

III] Redimensionnement "intelligent"

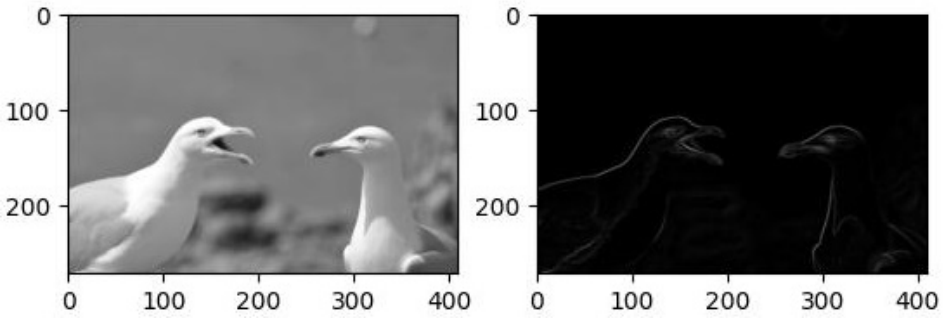
A. Énergie d'un pixel

Tous les pixels n'ont pas la même importance dans une image, en effet un si on enlève pixel entouré de pixels de la même couleur on ne le remarquera pas trop. Pour cela on introduit l'*énergie* du pixel (i, j) (qui n'est pas au bord de l'image) comme étant le nombre $e_{i,j}$ défini par :

$$e_{i,j} = \left|\frac{I_{i-1,j} - I_{i+1,j}}{2}\right| + \left|\frac{I_{i,j-1} - I_{i,j+1}}{2}\right|$$

Si $i = 0$ on mettra $I_{i,j} - I_{i+1,j}$ à la place de $\frac{I_{i-1,j}-I_{i+1,j}}{2}$, si $i = h-1$ on mettra $I_{i-1,j} - I_{i,j}$ à la place de $\frac{I_{i-1,j}-I_{i+1,j}}{2}$, idem avec la deuxième valeur absolue si $j = 0$ ou $j = w-1$.
La matrice $(e_{i,j})$ obtenue est appelée *matrice d'énergie* de l'image I.

Remarque : Cette quantité n'est rien d'autre qu'une discrétisation de la norme 1 du gradient ($\|\nabla I\|_1 = \left|\frac{\partial I}{\partial x}\right| + \left|\frac{\partial I}{\partial y}\right|$), on remarquera que cette quantité est petite quand le pixel est dans une zone uniforme de l'image, et est grande sinon (par exemple au niveau des contours). Par exemple :



Q12. Écrire une fonction `energie` qui prend en argument une image `img` et qui retourne une image correspondant à son énergie (les valeurs des "pixels" ne seront plus nécessairement des entiers).

B. Présentation de l'algorithme

Pour avoir une image d'une colonne de moins il suffit de supprimer un pixel par ligne, si on enlève le pixel d'énergie minimale de chaque ligne il risque d'y avoir un effet de décalage pas très gracieux, on pourrait aussi calculer la somme des énergies de chaque colonne et enlever la colonne de plus basse énergie. Soyons plus fin.

On définit un *chemin de pixels* comme étant une suite de pixels se "touchant" (verticalement ou en diagonale) dont le premier est sur la première ligne de l'image et le dernier est sur la dernière ligne de l'image et qui possède exactement un pixel par ligne de l'image.
On définit l'énergie d'un chemin comme étant la somme des énergies des pixels de ce chemin. Par exemple voici un chemin représenté dans une matrice d'énergie E (ce chemin est d'énergie 10) :

$E=$

4	3	0	1
1	2	2	1
1	1	3	3
0	4	1	0

On représentera un chemin par une liste `ae` de h nombres où `ae[i]` est l'abscisse du pixel du chemin à la i-ième ligne de la matrice, ainsi le chemin donné en exemple correspond à la liste `ae=[1,2,1,1]`

L'objectif est donc de déterminer un chemin d'énergie minimale pour ensuite supprimer les pixels correspondants de l'image.

1/ Approche gloutonne

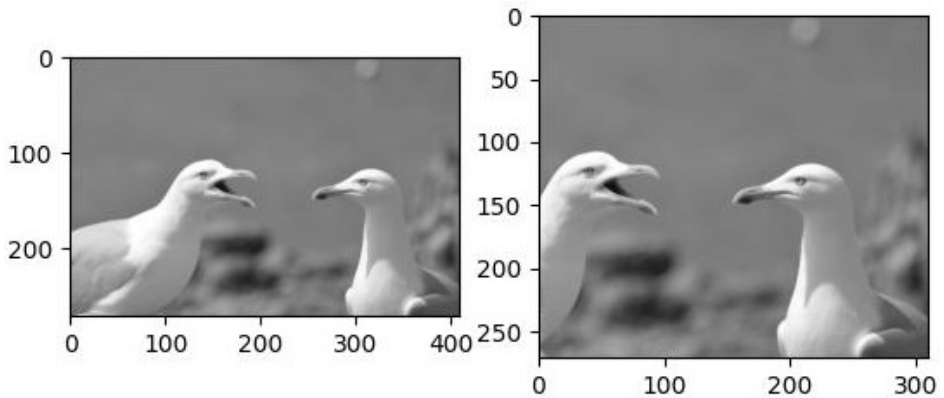
On peut appliquer une approche gloutonne (ie d'optimisation locale), pour cela on détermine le pixel d'énergie minimale de la première ligne, c'est le premier pixel de notre chemin, ensuite on choisit parmi les trois (ou deux) pixels en dessous celui d'énergie minimale (le plus à gauche en cas d'égalité), c'est notre second pixel du chemin, etc. jusqu'à arriver en bas de l'image.

Q13. Donner le chemin obtenu si on applique la stratégie gloutonne à la matrice E donnée juste au dessus.

Q14. Mettre en œuvre la stratégie gloutonne en programmant une fonction `glouton(img)` qui prend en argument une matrice d'énergie et qui retourne une liste `ae`.
Remarque : On pourra utiliser des fonctions présentées lors des précédentes questions.

Q15. La solution obtenue par la méthode gloutonne est-elle optimale ? On justifiera sa réponse (par exemple en donnant un exemple qui montre que la stratégie gloutonne n'est clairement pas optimale, par exemple sur une matrice 3×3).

Voici ce que cela donne quand on réduit de 100 colonnes :



2/ Approche dynamique

Pour le pixel de coordonnée (i, j) on considère l'ensemble des chemins qui vont de la première ligne à ce pixel de coordonnée (i, j), parmi ceux ci un possède une énergie plus basse que tous les autres, on note $f(i, j)$ cette énergie.

Q16. Notons e la matrice d'énergie de l'image, justifier que l'on a :

$$f(i,j) = \begin{cases} e_{i,j} & \text{si } i = 0 \\ e_{i,j} + \min(f(i-1,j), f(i-1,j+1)) & \text{si } i > 0 \text{ et } j = 0 \\ e_{i,j} + \min(f(i-1,j-1), f(i-1,j)) & \text{si } i > 0 \text{ et } j = w-1 \\ e_{i,j} + \min(f(i-1,j-1), f(i-1,j), f(i-1,j+1)) & \text{si } i > 0 \text{ et } 0 < j < w-1 \end{cases}$$

Q17. Écrire une fonction d'entête `f_rec(i,j)` qui calcule $f(i, j)$, on procèdera de manière récursive et sans mémoïsation. Pour gagner du temps, vous pouvez sauter cette question (les points de cette question seront alors intégrées à la question **Q19.**).

Q18. Expliquer en quoi la fonction précédente `f_rec(i,j)` est problématique.

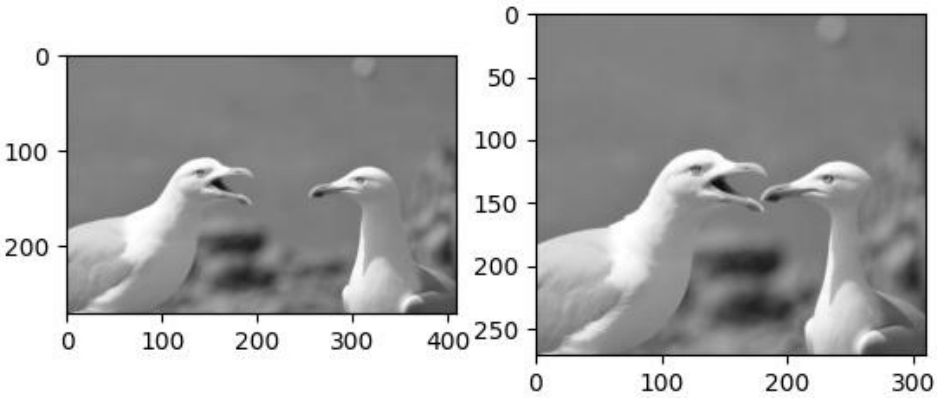
Q19. Écrire une fonction d'entête `f(i,j)` qui calcule $f(i, j)$, on procèdera de manière récursive et avec mémoïsation.

Q20. Compléter la fonction précédente pour qu'on puisse avoir le chemin minimal qui va jusqu'au pixel (i, j).

Q21. Comment déterminer le chemin minimal `ae` recherché ?

Q22. Écrire les lignes de code en python pour répondre à la question précédente

Voici ce que cela donne quand on réduit de 100 colonnes :



Remarque : Pour obtenir cette image il faut (sur mon ordinateur) environ 50 secondes, si on utilise plutôt une méthode de bas en haut (ie une méthode itérative) le temps de calcul est d'environ 20 secondes.

Q23. Écrire une fonction `seamcarving` qui prend en argument une matrice d'énergie `e` et qui retourne le chemin minimal `ae` recherché, pour cella elle utilisera une méthode de bas en haut : on stockera dans une matrice de même taille que `e` les valeurs de $f(i, j)$ en commençant par $i = 0$, et on fera en sorte de garder en mémoire les chemins (ou si on ne le fait pas on le retrouvera à la fin à partir de cette matrice)