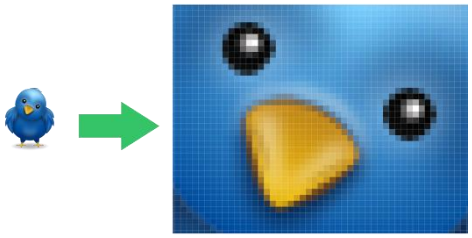
TD – REDIMENSIONNEMENT IMAGE INTELLIGENT

Table des matières

1. Rappels image numérique matricielle

Sur un écran ou une image le pixel est le plus petit élément constitutif d'une image :

- Chaque pixel va être codé avec un code couleur.
- Si on l'on regarde de nouveau cette image et que l'on zoom. On remarque qu'elle est composée de carrés de couleurs.

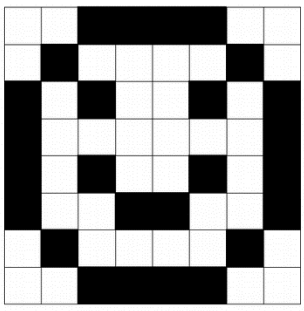


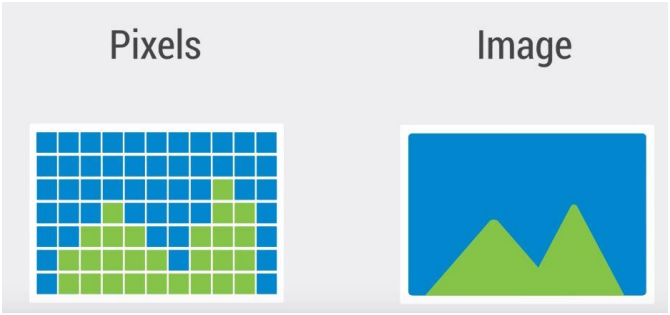
C'est l'organisation de ces pixels qui va former une image matricielle.

Chaque pixel est organisé dans un quadrillage, ou chaque case va contenir un code couleur

Au lieu de pixel, on parle aussi de « point » d'où le second nom de cette méthode de codage : la représentation **Bitmap** (pour « Carte de points »).

Une image matricielle est donc un tableau de valeurs (ou matrice) qui sera encodé dans un format particulier (PBM, TIFF, PNG, JPG)





2. Principe du redimensionnement

Une image étant une matrice de pixels. On peut alors affecter un « poids » à chaque pixel en fonctions de critères déterminés :

- même couleur que ses voisins = un poids petit (couleur du ciel)
- en fonction de sa couleur
- en fonction des caractéristiques de ses voisins

Au final chaque case du tableau de pixel aura un certain poids ou une certaine « énergie »

TD – REDIMENSIONNEMENT IMAGE INTELLIGENT

Exemple de redimensionnement « intelligent »

Image initiale	*Image rognée*	*Image redimensionnée*
		

Sur cet exemple l'image simplement rognée perd des informations importantes (la forme de la folie – maison de plaisance de l'image la position du personnage dans l'image).

L'image redimensionnée intelligemment garde les informations essentielles et supprime celles de moindre importance (partie du ciel et de la pelouse)

Exemples de traitement d'image par cette méthode :

https://youtu.be/6NcIJXTlugc

3. Objectif de ce Td

L'objectif de ce TD est de définir un algorithme qui permet de trouver des lignes descendantes traversant l'image de haut en bas et pour lesquelles la somme des poids des pixels traversés sera minimale.

4. Redimensionnement naïf

Pour illustrer un principe de redimensionnement naïf nous allons prendre un exemple modeste.

Soit une image de 4 x 4 pixels avec les poids suivants :

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

4.1. Solution optimale

En analysant cette grille, on peut visuellement voir que la ligne « de poids minimal » sera la suivante :

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

Il s'agit à présent de définir un algorithme qui permette de trouver ces lignes automatiquement.

4.2. Algorithme glouton

Le premier algorithme qui vient à l'esprit consiste à retenir pour chaque ligne traversée le pixel de poids minimal. On obtient ainsi un algorithme de type Glouton. En informatique, un algorithme glouton (greedy algorithm en anglais, parfois appelé aussi algorithme gourmand) est un algorithme qui suit le principe de faire, étape par étape, un choix optimum local. Dans certains cas cette approche permet d'arriver à un optimum global, mais dans le cas général c'est une heuristique/règle empirique pris telle-quelle.

4.2.1. Trouver par cet algorithme pour l'exemple le chemin des pixels de « moindre poids » en entourant les pixels sur le tableau suivant :

1	2	8	9
7	2	2	1
2	1	3	5
10	8	6	2

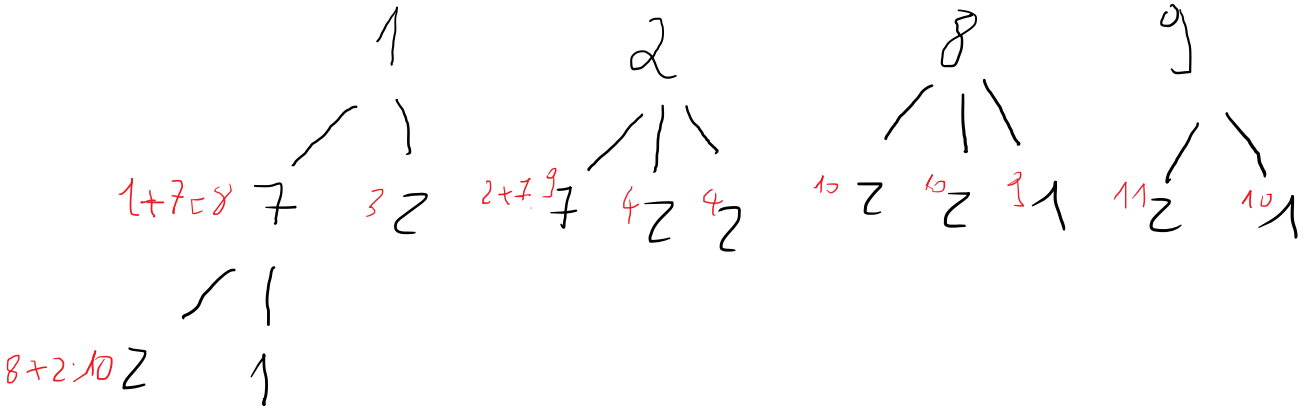
4.2.2. Est-ce la somme minimale que l'on puisse trouver sur ce tableau ?

4.2.3. Donner un ordre de grandeur du nombre d'opérations nécessaires dans le pire des cas.

4.2.4. Conclure :

4.3. Algorithme de recherche du chemin avec le moins de poids

On désire obtenir le « meilleur « chemin » celui avec le moins de poids. On peut représenter les décisions sous la forme d'un arbre :



4.3.1. Compléter cet arbre sur le document réponse

4.3.2. Evaluation de la complexité :

Donner deux réponses une première pour la matrice de l'exemple une deuxième pour une image matricielle de n x n pixels

- Combien y a-t-il de colonnes au départ ? :
- Combien de chemins sont-ils possibles après chaque pixel (on admet le même nombre pour les pixels au bord de l'image) :
- Evaluer le nombre de chemin possible des premiers pixels en haut de l'image jusqu'en bas

4.3.3. Conclure :

5. Redimensionnement dynamique

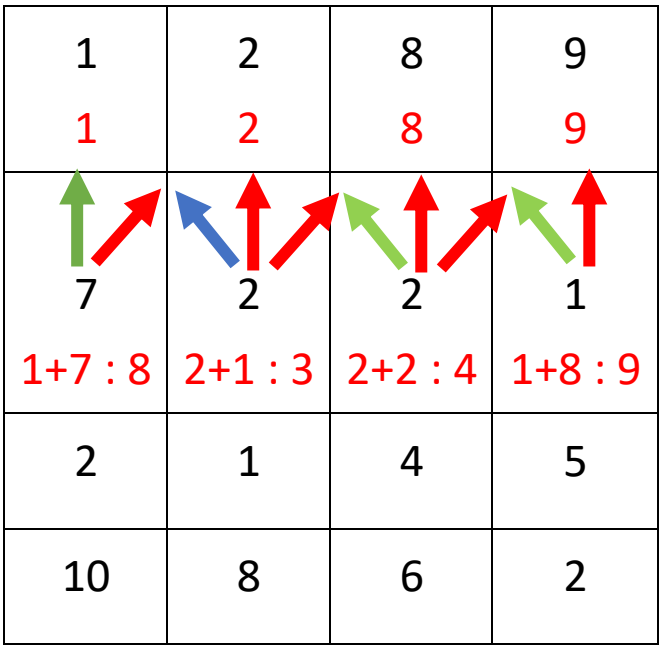
Le gros inconvénient de la méthode précédente est sa complexité et aussi le besoin de recalculer les différents poids intermédiaires.

L'idée pour obtenir une exécution plus rapide est de décomposer le problème initial (calculer le chemin des moindres poids) en sous problèmes (calcul de chemins intermédiaires) tout en stockant ces calculs intermédiaires pour éviter d'avoir à les réévaluer.

C'est ce que l'on appelle la programmation dynamique . celle -ci se traduit par une relation de récurrence associée à une technique de mémoïsation.

Reprenons l'exemple précédent

Déroulement de l'algorithme



Etablissement du tableau des poids des chemin intermédiaires

- la valeur en **noire** en haut à gauche représente **le poids du pixel correspondant**,
- la valeur en **rouge** représente la **somme cumulative** des poids **qui ont mené à ce pixel** (en incluant ce dernier).

Les pixels de la **première ligne** ne peuvent pas avoir de **provenance antérieure**, puisqu'ils sont au **sommet de l'image** : ainsi leur **somme cumulative** (en rouge) est simplement la valeur de **leur poids** (en noir).

Pour la **deuxième ligne**, considérons par exemple le **second pixel** : son **poids** est 2 (en noir). En regardant au niveau de la **ligne précédente**, nous avons le choix entre **trois provenances** :

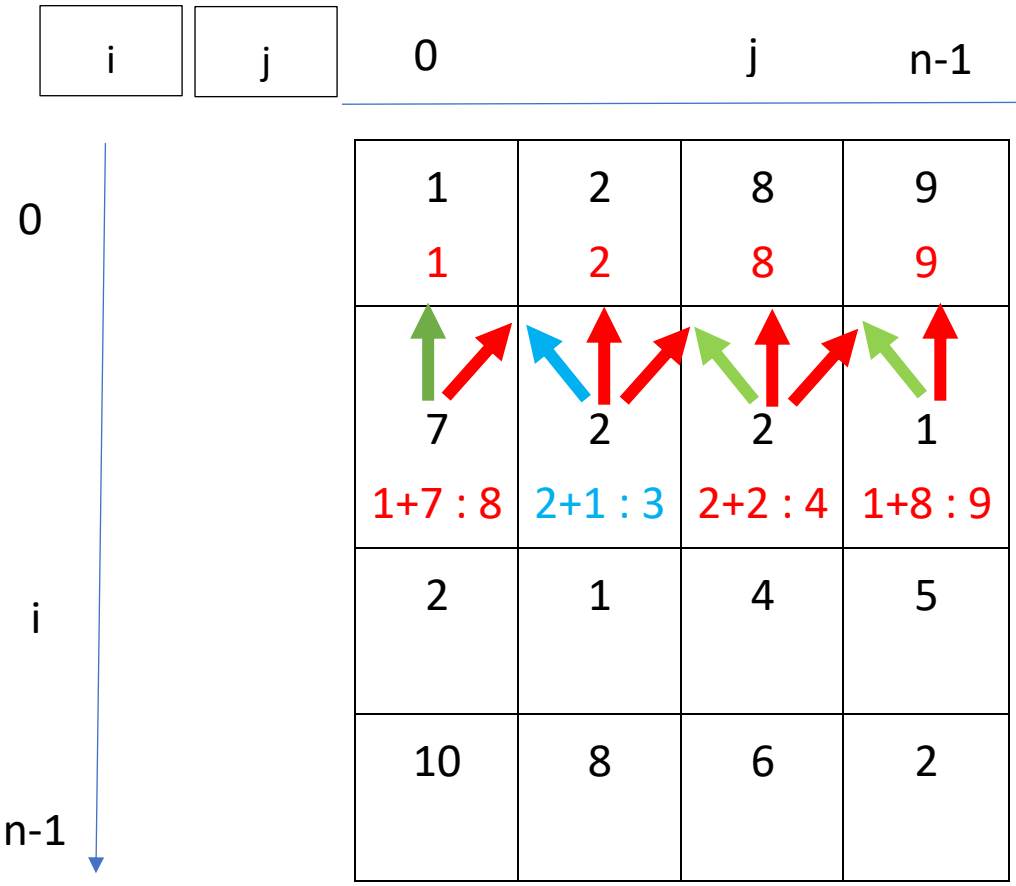
- le chemin provenant de la gauche et de **somme 3 (2+1)**,
- celui provenant du centre et de valeur **4** (2+2) ou
- celui provenant de la droite et de valeur **10 (8+2)** .

Puisque **3 est la plus petite** des ces trois valeurs, l'algorithme **ignore** les deux chemins et nous pouvons ainsi déterminer que la **somme cumulative** des poids des **pixels "traversés"** pour arriver jusqu'à celui-ci est de poids **2** (son poids) **+ 1** (la somme du chemin de provenance minimale), **soit 3** (en bleu).

Il s'agit ensuite simplement de **répéter l'opération** pour chaque pixel de chaque ligne jusqu'à avoir rempli notre image, pour finalement obtenir le tableau des résultats intermédiaires.

Au final, en partant du **bas de l'image**, nous pouvons voir apparaître les ***chemins de moindres poids*** : ce sont les chemins que nous pouvons parcourir en **suivant les flèches vertes** depuis les pixels dont la **somme cumulative** (valeur en bleu dans la dernière ligne) est la **plus faible**.

5.1. Compléter le tableau des calculs intermédiaires pour notre exemple



5.2. Formules de récurrence

On note i l'indice des lignes et j celui des colonnes du tableau des calculs intermédiaires appelé « Tab ». Donner la formule de récurrence qui donne le calcul à effectuer pour la case i x j.

- au milieu du tableau

  condition :

  formule :

- à la gauche du tableau
  condition :

  formule :

- à la droite du tableau

  condition :

  formule :

5.3. Principe pour identifier le chemin de moindre poids

Pour remonter le tableau et identifier les pixels :

- Identifier pour la ligne du bas le pixel de moindre poids cumulé

  numéro de ligne :

  identification de la colonne :

- Identifier pour la ligne juste au-dessus la colonne j du poids cumulé le plus bas

  numéro de ligne :

  identification de la colonne :

pour un pixel en ne se trouvant pas sur un des bords

pour le bord gauche si $j_{n-1} = 0$

pour le bord droit si $j_{n-1} = n-1$

Généraliser d'une ligne à l'autre en remontant

Bien pensé à stocker les différents indices j pour chaque ligne.