Tomographie

Le projet est l'implémentation d'un algorithme de recherche tomographique efficace. Les points significatifs sont dans les sections 3.4 et 3.5.

Dans la suite, n désigne le nombre de lignes et le nombre de colonnes. On peut bien sûr envisager des images non carrées.

1 Quelques outils de base

1.1 La bibliothèque `MaBibTomo` contient quelques fonctions

♯ `print_mat(T)` affiche lisiblement un tableau 2D

♯ `somme_liste(L)` retourne la somme des entiers constituant la liste `L`

♯ `extract_col(T,j)` retourne la liste `L` correspondant à la colonne j du tableau `T`

♯ `somme_li(T)` retourne la liste des sommes par lignes

♯ `somme_col(T)` retourne la liste des sommes par colones

♯ `compatl(L1,L2)` retourne vrai quand les sommes des entiers des listes sont égales

♯ `MGC(L)` retourne un tableau dont `L` est la projection tomographique horizontale, dont les 1 sont à gauche des 0

♯ `permut(L)[0]` retourne la liste triée par ordre décroissant

♯ `permut(L)[1]` retourne la permutation correspondante

1.2 Test de compatibilité (le faire pour s'entrainer OU utiliser `MaBibTomo`)

Entrée : $(v_1, ..., v_n)$ et $(h_1, ..., h_n)$
Sortie : Vrai quand $v_1 + ... + v_n = h_1 + ... + h_n$

1.3 Calcul des projections tomographiques (idem)

Entrée : Tableau bidimensionnel `T`
Sortie : sommes par lignes

et

Entrée : Tableau bidimensionnel `T`
Sortie : sommes par colonnes

2 Algorithme exhaustif

Quelqu'un n'ayant jamais programmé d'algorithme de résolution d'un problème tomographique pourra trouver quelque intérêt à programmer un algorithme de recherche exhaustif.

3	.	.	.	.	.
2	.	.	.	.	.
2	.	.	.	.	.
2	.	.	.	.	.
3	.	.	.	.	.
	1	4	5	1	1

3 Algorithme de Ryser

3.1 Principe : les algorithmes efficaces proviennent de la mise en synergie de deux (au moins) idées.

3.1.1 Première idée :

En permuttant les colones d'une configuration tomographique correcte (l'image ET les valeurs des projections verticales), on obtient une configuration tomographique correcte. On peut donc permuter les colonnes pour avoir des sommes par colonnes de valeurs décroissantes. Si on résoud ce nouveau problème, la permutation réciproque permettra de résoudre le problème initial.
Le problème devient donc :

3	.	.	.	.	.
2	.	.	.	.	.
2	.	.	.	.	.
2	.	.	.	.	.
3	.	.	.	.	.
	5	4	1	1	1

3.1.2 Deuxième idée : satisfaire la contrainte tomographique horizontale en poussant tous les 1 à gauche puis pratiquer des échanges horizontaux

3	1	1	0	0	1
2	1	1	0	0	0
2	1	1	0	0	0
2	1	1	0	0	0
3	1	1	1	0	0
	5	4	1	**1**	1

Dans la colonne 4, la somme par colonne du tableau à gauche en cours de traitement est inférieure à la somme par colonne visée, alors que dans la colonne 5, le compte est bon.
On cherche le premier 1 disponible dans les colonnes à gauche de la colonne 4 pour le déplacer vers la droite.

3	1	1	0	0	1
2	1	1	0	0	0
2	1	1	0	0	0
2	1	1	0	0	0
3	1	1	0	1	0
	5	4	**1**	1	1

On a ainsi comblé le déficit de la colonne 4 au prix d'un déficit de la colonne 3. On recommence pour le combler....

3.2 Construction de la matrice gauche compatible avec les projection tomographiques horizontales (le faire pour s'entrainer OU utiliser `MaBibTomo`)

Entrée : $(v_1, ..., v_n)$
Sortie : Tableau bidimensionnel `TR` à valeur dans $\{0; 1\}$ tel que
`TR[1][k]+TR[2][k]+...+TR[n][k]=`h_k
et
(`TR[i][k]`$\neq 0$ implique `TR[i][k-1]`$\neq 0$)

Dans la suite, je conserve la notation `TR` pour ce tableau ; $\bar{v}_i$ ses sommes par colonnes

3.3 Tri décroissant (le faire pour s'entrainer OU utiliser `MaBibTomo`)

Entrée : un tableau unidimensionnel contenant les valeurs $(v_1, ..., v_n)$
Sortie : une permutation σ telle que $v_{\sigma(i)} \geq v_{\sigma(i+1)}$ et la liste des valeurs triées par ordre décroissant.

Dans la suite, on conserve la notation σ pour cette permutation.

3.4 Test de Ryser : à faire

Entrée : $(v_1, ..., v_n)$ et $(h_1, ..., h_n)$
Sortie : vrai quand $v_1 + ... + v_n = h_1 + ... + h_n$
et
pour tous les k compris entre 1 et n, on a $v_{\sigma(k)} + v_{\sigma(k+1)} ... + v_{\sigma(n)} \geq \bar{v}_k + \bar{v}_{k+1} + ... + \bar{v}_n$

Le théorème de Ryser énonce que le problème tomographique est résoluble si les listes satisfont cette condition.

En `Python`, `L[a:b]` extrait la sous-liste de `L` entre entre les indices a et b.

3.5 Algorithme de Ryser : à faire

Entrée : $(v_1, ..., v_n)$ et $(h_1, ..., h_n)$ satisfaisant le test de Ryser
Sortie : un tableau solution du problème tomographique

Initialisation : tableau `T = TR` et `k = n`
Tant que `(k > 1)` faire :

Rédiger l'algorithme réalisant la deuxième idée

fin tant que
`k = k - 1`
Fin tant que
Retourner la solution : **rédiger l'algorithme réalisant la première idée**

Implémenter l'algorithme.

En `Python`, `[i for i in range (n)]` créé la liste des entiers de 0 à $n - 1$.