LU3IN003 - PROJET
Un problème de tomographie discrète

Esther CHOI (3800370) et Vinh-Son PHO ()

November 21, 2020

Sommaire

Les tests de temps de résolution ont été réalisés sur un processeur Intel©Core™i7-8550U 1.80GHz.

1 Méthode incomplète de résolution

1.1 Première étape

Q1 Il suffit de regarder s'il existe $j \in \{1, ..., M-1\}$ tel que $T(j,k) =$ vrai. En effet cela signifierait qu'il existe un coloriage possible des $j+1$ premières cases avec la séquence complète $(s_1, ..., s_k)$.

Q2 Commençons par remarquer que $\forall j \in \{0, ..., M-1\}$ et $\forall l \in \{1, ..., k\}$, pour que $T(j,l)$ soit vrai, il faut que les $j+1$ premières cases contiennent au moins les l premiers blocs noirs en entier en plus des $l-1$ cases blanches pour séparer chaque bloc noir, c'est-à-dire qu'il faut :

$$j+1 \geq l-1+\sum_{i=1}^{l} s_i = l-1+s_l+\sum_{i=1}^{l-1} s_i \implies \boxed{j \geq l-1+s_l-1+\sum_{i=1}^{l-1} s_i}$$

- cas de base 1 : si $l=0$, cela signifie qu'il n'y a pas de blocs à placer. Donc il existe un coloriage possible pour les $j+$ premières cases : il suffit qu'elles soient toutes blanches.
- cas de base 2a : supposons $j < s_l - 1$
  - si $l=1$: alors $l-1+s_l-1+\sum_{i=1}^{l-1} s_i = s_1 - 1$
    D'après la remarque que l'on a faite, pour avoir $T(j,l) =$ vrai, il faudrait avoir $j \geq s_1 - 1$.
    Or, on a supposé $j < s_l - 1 = s_1 - 1$
    Donc pour $l=1$, $T(j,l) =$ faux
  - si $l \geq 2$: alors $l-1+s_l-1+\sum_{i=1}^{l-1} s_i > s_l - 1$ car $l-1>0$
    Pour avoir $T(j,l) =$ vrai, il faudrait avoir $j > s_l - 1$
    Or, on a supposé $j < s_l - 1$
    Donc pour $l \geq 2$, $T(j,l) =$ faux

  Conclusion : $\boxed{\forall l \geq 1, \text{ si } j < s_l - 1, \text{ alors } T(j,l) = \text{faux}}$
- cas de base 2b : supposons $j = s_l - 1$
  - si $l=1$: alors de même, $l-1+s_l-1+\sum_{i=1}^{l-1} s_i = s_1 - 1$
    D'après la remarque que l'on a faite, pour avoir $T(j,l) =$ vrai, il faudrait avoir $j \geq s_1 - 1$.
    Donc, en particulier pour $j = s_l - 1$, $T(j,l) =$ vrai
  - si $l \geq 2$: alors de même, $l-1+s_l-1+\sum_{i=1}^{l-1} s_i > s_l - 1$ car $l-1>0$
    Pour avoir $T(j,l) =$ vrai, il faudrait avoir $j > s_l - 1$

Or, on a supposé $j = s_l - 1$
Donc pour $l \geq 2$, $T(j,l) = \text{faux}$

Conclusion : $\boxed{\text{si } j = s_l - 1, \text{ alors } T(j,l) = \begin{cases} \text{vrai} & \text{si } l = 1 \\ \text{faux} & \text{si } l \geq 2 \end{cases}}$

Q3 On a la relation de récurrence suivante : $\boxed{T(j,l) = T(j-1,l) \vee T(j - s_l - 1, l-1)}$
Avec comme cas de base : $\forall j \in \{0, ..., M-1\}, T(j,0) = \text{vrai}$, et $T(s_1 - 1, 1) = \text{vrai}$.
En effet :

- si on arrive à faire rentrer les l premiers blocs dans les j premières cases (c'est-à-dire si $T(j-1,l) = \text{vrai}$), alors on arrivera à les faire rentrer dans les $j+1$ premières cases (c'est-à-dire $T(j,l) = \text{vrai}$) en coloriant la j-ème case en blanc.

- si on arrive à faire rentrer les $l-1$ premiers blocs sur un certain nombre de cases j' (c'est-à-dire si $T(j', l-1) = \text{vrai}$), alors on pourra faire rentrer le bloc l si et seulement si $j \geq j' + s_l + 1$ (le +1 venant de la case blanche séparant le bloc $l-1$ du bloc l), donc en particulier si $j = j' + s_l + 1$, c'est-à-dire si $j' = j - s_l - 1$. Ainsi, on a $T(j,l) = \text{vrai}$, et la j-ème case est noire et correspond à la dernière case du bloc l.

- il suffit que l'une des deux conditions précédentes soit vraie pour que $T(j,l)$ soit égale à vrai, d'où le $\vee$.

- les cas de base sont les cas de base 1 et 2b pour $l = 1$ de la question précédente.

Q4 Il s'agit de la fonction *ColoriagePossibleRec* dans le fichier *src/partie1*. L'algorithme en pseudo-code est le suivant :

Algorithme 1 ColoriagePossibleRec

Entrée : T matrice vide de taille $M \times k$, $s = (s_1, ..., s_k), j, l$

Sortie : Retourne la matrice T telle que $T[j][l] = T(j, l)$. On suppose que pour le premier appel, $j = M - 1$ et $l = k$

1: **Si** $T[j][l] \neq$ VIDE **alors**
2:     **Retourner** $T[j][l]$
3: **Sinon si** $j < s_l - 1$ **alors**
4:     $T[j][l] \leftarrow$ faux
5:     **Retourner** faux
6: **Sinon si** $j = s_l - 1$ **alors**
7:     **Si** $l = 1$ **alors**
8:         $T[j][l] \leftarrow$ vrai
9:         **Retourner** vrai
10:     **Sinon**
11:         $T[j][l] \leftarrow$ faux
12:         **Retourner** faux
13:     **Fin Si**
14: **Sinon**
15:     $T[j][l] \leftarrow (ColoriagePossibleRec(T, s, j-1, l)$ ou $ColoriagePossibleRec(T, s, j - s_l - 1, l-1))$
16:     **Retourner** $T[j][l]$
17: **Fin Si**

- ligne 1 : initialisation de la matrice
- lignes 2-5 : cas de base du cas 2c
- lignes 8-10 : cas 1
- lignes 11-18 : cas 2a
- lignes 19-22 : cas 2c

En réalité, l'algorithme ne calcule pas nécessairement toutes les cases du tableau T : il ne calcule que les valeurs utiles, celles qui répondent à la question 1. En effet, si le résultat voulu est calculé, on le retourne ; sinon, on stocke progressivement les valeurs qui permettent le calculer. C'est le principe de la programmation dynamique.

1.2 Généralisation

Q5

- cas de base 1 : si $l = 0$, $T(j, l) =$ vrai s'il y a pas de case noire parmi toutes les cases précédentes

- cas de base 2a : on a toujours $T(j,l) =$ faux
- cas de base 2b :
  - si $l = 1$ et donc $j = s_1 - 1$, alors $T(j,l) =$ vrai s'il n'y a pas de case blanche parmi les case précédentes, puisque les j+1 premières cases contiennent le bloc 1
  - si $l > 1$, on a toujours $T(j,l) =$ faux
- cas de base 2c : il y a deux cas possibles.
  - la case (i,j) est blanche : il faut regarder si les l premiers blocs rentrent dans les j premières cases, c'est-à-dire qu'il faut faire un appel récursif sur la fonction pour calculer $T(j-1,l)$
  - la case (i,j) est noire : on est sur la dernière case du bloc l donc $T(j,l) =$ vrai s'il n'y a pas de case blanche parmi les cases que constituent le bloc l et si la case d'indice $j - s_l$ n'est pas noire (car si elle l'était, la case $(i,j-1)$ contiendrait la dernière case du bloc l et (i,j) ne pourrait être noire). Si l'on a les deux conditions, alors il faut regarder s'il est possible de faire rentrer les $l-1$ premiers blocs dans les $j - (s_l -)$ premières cases

Q6 La matrice qui contient la valeur des $T(j,l)$ est de taille $M \times k$. Or $k \leq \lfloor \frac{M+1}{2} \rfloor$: au maximum, sur une ligne de taille M, on peut faire rentrer $\lfloor \frac{M+1}{2} \rfloor$ blocs noirs (ils sont alors tous de longueur 1). Donc il y a au plus $M \times \lfloor \frac{M+1}{2} \rfloor$ valeurs à calculer.
Pour calculer une valeur de $T(j,l)$, on fait appel à la fonction *TestVal* qui est de complexité $O(M)$ (dans le pire cas, on cherche une occurrence d'une valeur dans la ligne entière).
Conclusion : l'algorithme est de complexité $O(M \times M \times \lfloor \frac{M+1}{2} \rfloor) = \boxed{O(M^3)}$

Q7 Il s'agit de la fonction *ColoriagePossibleRec2* dans le fichier *src/partie1.py*.

1.3 Propagation

Q8 La fonction *ColoreLig* comporte une boucle *for* de M itérations, et chaque itération fait appel à la fonction *ColoriagePossibleRec2* de complexité $O(M^3)$, donc elle est de complexité $O(M^4)$. De même, *ColoreCol* est de complexité $O(N^4)$.
La fonction *Coloration* comporte deux boucles *for*, l'une sur *lignesAVoir* et l'autre sur *colonnesAVoir* de tailles au plus N et M respectivement, qui sont exécutées tant que *lignesAVoir* et *colonnesAVoir* ne sont pas vides, c'est-à-dire tant que l'on peut colorier des nouvelles cases. La boucle *while* fait donc au plus MN itérations, et les boucles *for* au plus N et M itérations.
Pour la boucle sur *lignesAVoir*, on fait appel à la fonction *ColoreLig* qui est de complexité $O(M^4)$.
Pour la boucle sur *colonnesAVoir*, on fait appel à la fonction *ColoreCol* qui est de complexité $O(N^4)$.

Conclusion : la fonction *Coloration* est de complexité $O(MN(N \times M^4 + M \times N^4)) = \boxed{O(N^2 \times M^5 + M^2 \times N^5)}$. On a donc bien une complexité polynomiale en N et M

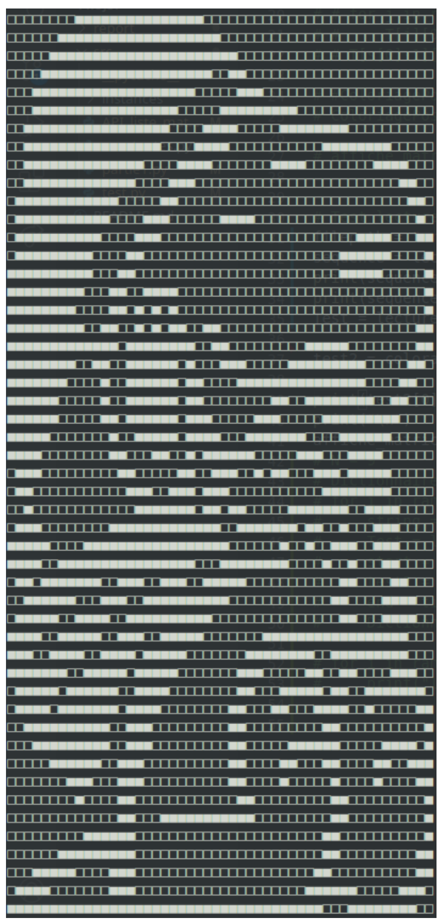
Q9 Il s'agit des fonctions *lecture, ColoreLig, ColoreCol, Coloration* dans le fichier *src/partie1.py*

1.4 Tests

Q10 Le tableau suivant donne les temps de résolution de chacune des instances 0 à 10.

instance	1	2	3	4	5	6	7	8	9	10
temps de résolution (s)	0.00117	0.17	0.24	0.71	0.21	1.06	0.55	1.11	205.01	226.13

La grille obtenue pour l'instance 9.txt est la suivante :



Q11 L'algorithme implémenté dans cette partie ne sait pas résoudre l'instance 11.txt. En effet :