Un problème de tomographie discrète

Le sujet au format pdf ainsi que les instances sont disponibles sur le site Moodle de l'UE :
`https://moodle-sciences-23.sorbonne-universite.fr/course/view.php?id=973`

Nous considérons une grille de N lignes numérotées de 0 à $N-1$ et M colonnes numérotées de 0 à $M-1$. Chacune des $N \times M$ cases doit être coloriée en blanc ou en noir. A chaque ligne l_i, $i = 0, \ldots, N-1$, est associée une séquence d'entiers représentant les longueurs des blocs de cases noires de la ligne. De même, à chaque colonne c_j, $j = 0, \ldots, M-1$, est associée une séquence d'entiers représentant les longueurs des blocs de cases noires de la colonne.

Considérons l'exemple suivant :

	1 1	1	1 2	1	2
3					
1 1 1					
3					

Le 3 de la ligne l_0 signifie que celle-ci contient un bloc de trois cases noires consécutives. La séquence 1 2 de la colonne c_2 signifie qu'elle contient deux blocs : le premier d'une case noire, le deuxième de deux cases noires (les blocs sont séparés d'au moins une case blanche).

Une solution du jeu est :

	1 1	1	1 2	1	2
3	■	■	■		
1 1 1	■		■		■
3			■	■	■

T(2,1) = FALSE

j = 2 = sl -1 = 3 -1

Le but du projet est de construire, s'il en existe, une solution (un coloriage noir-blanc des cases) répondant aux contraintes.

Plus généralement, chaque ligne/colonne se voit donc assigner une séquence (potentiellement vide) $(s_1, s_2, \ldots, s_k)$ de nombres entiers strictement positifs. Le coloriage doit comporter sur cette ligne/colonne un premier bloc de s_1 cases noires consécutives, puis un deuxième de s_2 cases noires, ... Il n'y a pas d'autres cases noires dans la ligne/colonne. Les blocs doivent être séparés d'au moins une case blanche. Il peut y avoir des cases blanches avant le premier bloc, et/ou après le dernier.

1 Méthode incomplète de résolution

Un raisonnement simple permet de déterminer directement la couleur de certaines cases. Par exemple, supposons qu'une ligne l_i ait un unique bloc de taille $s_1 = 10$, et qu'il y ait $M = 15$ colonnes. Alors nécessairement sur cette ligne les cases (i, j) avec $j = 5, \ldots, 9$ sont noires (on rappelle qu'on numérote les indices des lignes/colonnes/cases à partir de 0).

Nous allons dans cette section colorier partiellement des grilles en généralisant ce raisonnement pour trouver la couleur de certaines cases. Dans la section 1.1, on montre comment déterminer si, pour une séquence donnée, il est possible de colorier une ligne en respectant cette séquence. Dans la section 1.2, on généralise au cas d'une ligne dont la couleur de certaines cases est imposée. Enfin, dans la section 1.3, on montre comment tirer parti de cet algorithme afin d'identifier des cases nécessairement blanches ou noires dans une ligne/colonne, et on procède par propagation pour colorier partiellement une grille.

1.1 Première étape

Considérons une ligne l_i dont la séquence associée est $(s_1, s_2, \ldots, s_k)$. Nous voulons définir un algorithme permettant de déterminer s'il existe un coloriage possible de la ligne avec cette séquence [1].

Pour cela, on définit $T(j, \ell)$ $(j = 0, \ldots, M-1,\ \ell = 1, \ldots, k)$ comme vrai s'il est possible de colorier les $j+1$ premières cases $(i, 0), (i, 1), \ldots, (i, j)$ de la ligne l_i avec la sous-séquence $(s_1, \ldots, s_\ell)$ des ℓ premiers blocs de la ligne l_i.

(Q1) *Si l'on a calculé tous les $T(j, \ell)$, comment savoir s'il est possible de colorier la ligne l_i entière avec la séquence entière ?*

Pour calculer les $T(j, \ell)$, on considère un algorithme qui procède par récurrence en distinguant les cas suivants :

1. Cas $\ell = 0$ (pas de bloc), $j \in \{0, \ldots, M-1\}$
2. Cas $\ell \geq 1$ (au moins un bloc)
   (a) $j < s_\ell - 1$
   (b) $j = s_\ell - 1$
   (c) $j > s_\ell - 1$

(Q2) *Pour chacun des cas de base 1, 2a et 2b, indiquez si $T(j, \ell)$ prend la valeur vrai ou faux, éventuellement sous condition.*

Considérons maintenant le cas 2c. Il y a deux possibilités :

— Soit la case (i, j) est blanche ;
— Soit elle est noire, et dans ce cas-là le dernier bloc de longueur s_ℓ est à la fin de la partie de l_i considérée, il se termine donc à la case (i, j).

(Q3) *Exprimez une relation de récurrence permettant de calculer $T(j, \ell)$ dans le cas 2c en fonction de deux valeurs $T(j', \ell')$ avec $j' < j$ et $\ell' \leq \ell$.*

(Q4) *Codez l'algorithme, puis testez-le.*

1. On pourrait répondre très simplement à cette question sans recourir à la récurrence proposée ici, mais il s'agit de préparer la généralisation où l'on considérera une ligne dont certaines cases sont déjà coloriées en noir ou en blanc (cases imposées).

1.2 Généralisation

En réalité, dans l'algorithme de résolution que nous présentons dans la section 1.3, nous allons colorier les cases de la grille au fur et à mesure. Nous aurons donc, à un moment donné, une ligne avec certaines cases coloriées en blanc, d'autres en noir, et d'autres non encore coloriées.

Il faut donc modifier l'algorithme précédent afin qu'étant donné une séquence $(s_1, \ldots, s_k)$ et une ligne l_i avec certaines cases déjà coloriées en blanc ou en noir, il indique si une coloration de cette ligne est possible.

(Q5) *Modifiez chacun des cas de l'algorithme précédent afin qu'il prenne en compte les cases déjà coloriées.*

(Q6) *Analysez la complexité en fonction de M de l'algorithme. Pour ce faire, on déterminera le nombre de valeurs $T(j, \ell)$ à calculer, que l'on multipliera par la complexité de calcul de chaque valeur $T(j, \ell)$.*

(Q7) *Codez l'algorithme.*

1.3 Propagation

L'algorithme de résolution consiste à partir d'une grille non coloriée, puis à tester chaque ligne et chaque colonne pour tenter de colorier des cases en se servant de l'algorithme de la section précédente. Ainsi, pour chaque case (i, j) non encore coloriée de la ligne i, on peut :

- tester si elle peut être coloriée en blanc (la colorier en blanc, et tester si la ligne l_i a une réponse positive) ;
- tester de même si elle peut être coloriée en noir ;
- si les deux tests échouent, le puzzle n'a pas de solution. Si aucun des tests n'échoue, on ne peut rien déduire. En revanche, si par exemple le test blanc échoue mais le test noir réussit, cela veut dire que la case peut être coloriée en noir.

On itère tant que des changements sont possibles. Un pseudo-code est donné en annexe (Algo. 1).

(Q8) *Montrez que cet algorithme est de complexité polynomiale en N et M.*

(Q9) *Codez l'algorithme de propagation. Votre programme prendra en entrée un fichier texte dont chaque ligne correspond à la séquence associée à une ligne ou une colonne de la grille. Le symbole # indique que l'on passe de la description des lignes à celle des colonnes. Avant le #, il y a autant de lignes dans le fichier que de lignes dans la grille, et à chaque ligne est indiquée la séquence d'entiers représentant les longueurs des blocs. Le contenu du fichier encodant l'instance de l'introduction est :*

```
3

1 1 1
3
#
1 1
1
1 2
1
2
```

L'algorithme devra permettre une visualisation *du coloriage obtenu en sortie, avec des cases de couleur noire, blanche ou indéterminée. Vous vérifierez que votre programme résout correctement l'instance 0.txt, qui correspond à l'exemple de l'introduction.*

1.4 Tests

(Q10) *Appliquez votre programme sur les instances 1.txt à 10.txt*[2]*. Vous indiquerez dans le rapport les temps de résolution dans un tableau. Vous fournirez dans le rapport la grille obtenue pour l'instance 9.txt.*

(Q11) *Appliquez votre programme sur l'instance 11.txt. Que remarquez-vous ? Expliquez.*

2 Méthode complète de résolution

La méthode proposée dans la section précédente est *incomplète*, c'est-à-dire que pour certaines grilles elle ne me permet pas de conclure à l'existence ou non d'un coloriage répondant aux contraintes. Pour pallier cet inconvénient (et donc obtenir une méthode *complète* de résolution), l'objet de cette section est d'intégrer la méthode précédente dans une procédure d'énumération des coloriages potentiels de la grille.

Cette approche est fondée sur un algorithme récursif, dont le pseudo-code est donné en annexe (Algorithme 2). Chaque branchement dans l'arbre des appels récursifs (arbre d'énumération) correspond au choix de colorier en blanc ou en noir une case de la grille, en parcourant les cases selon une numérotation de la gauche vers la droite et de haut en bas, de 0 à $NM - 1$. Pour illustration, la numérotation des cases utilisée pour une grille 4×4 est indiquée ci-dessous :

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Afin de limiter la taille de l'arbre d'énumération exploré, à chaque décision de colorier une case en blanc ou en noir, on propage depuis cette case pour tenter de colorier des cases supplémentaires en se servant de l'algorithme de la section 1.3 (ou éventuellement détecter une impossibilité induite par ce choix de coloration).

(Q12) *Montrez que cet algorithme est de complexité exponentielle en N et M.*

2.1 Implantation et tests

(Q13) *Implantez l'algorithme de résolution complète. Vous vérifierez que votre programme résout correctement l'instance 11.txt.*

(Q14) *Résolvez les instances 1.txt à 16.txt avec un* timeout *de 2 minutes. Donnez les temps de calcul dans un tableau. Pour les instances 12.txt à 16.txt, appliquez également la méthode de la*

2. Les instances, renumérotées, sont tirées du site `webpbn.com`.

section 1. Commentez. Vous fournirez dans le rapport la grille obtenue avec chacune des deux méthodes pour l'instance 15.txt.

3 Travail demandé

Le travail se fait obligatoirement en binôme du même groupe de TD. Il compte pour 25% de la note finale de l'UE. Le choix du langage de programmation est libre. Seuls les aspects purement algorithmiques seront abordés avec vos enseignants.

3.1 Rapport

Chaque binôme doit rédiger sur traitement de texte un rapport répondant aux différentes questions de l'énoncé (en prenant soin d'expliquer et/ou de commenter).

Vous créerez un répertoire ayant pour nom `nomBinome1_nomBinome2`, où `nomBinome1` et `nomBinome2` correspondent à vos noms de famille. Ce répertoire contiendra votre rapport (au format pdf) ainsi que les fichiers sources *commentés* de vos programmes. Compressez ce répertoire sous forme d'un fichier `zip`. Déposez ce fichier sur le site moodle de l'UE au plus tard le **mardi 12 mars 2024** à 23h59.

3.2 Soutenance

La soutenance aura lieu lors de la dernière séance de TP. Un planning des passages vous sera communiqué par mail.

NB : il pourra être demandé d'exécuter le code sur des instances autres que celle du projet. Votre programme devra donc pouvoir résoudre des instances fournies sur des fichiers `.txt` (au format spécifié dans le projet).

4 Annexe : pseudo-codes

Algorithme 1 Algorithme de résolution partielle

fonction COLORATION(A) /* A une grille entièrement non coloriée */
  $A' \leftarrow A$ /* pour ne pas modifier la grille A en entrée */
  LignesAVoir $\leftarrow \{0, \ldots, N-1\}$; ColonnesAVoir $\leftarrow \{0, \ldots, M-1\}$
  tant que LignesAVoir $\neq \emptyset$ ou ColonnesAVoir $\neq \emptyset$ **faire**
    pour i dans LignesAVoir **faire**
      (ok,A') $\leftarrow$ COLORELIG(A',i) /* Colorie par récurrence un max de cases de la ligne i de A' */
      /* ok=Faux si détection d'impossibilité, ok=Vrai sinon */
      si non(ok) **alors retourner** (Faux,V) /* V matrice vide */
      Nouveaux $\leftarrow \{j : j$ est la colonne d'une nouvelle case coloriée dans la ligne i de $A'\}$
      ColonnesAVoir $\leftarrow$ ColonnesAVoir $\cup$ Nouveaux ; LignesAVoir $\leftarrow$ LignesAVoir $- \{i\}$
    pour j dans ColonnesAVoir **faire**
      (ok,A') $\leftarrow$ COLORECOL(A',j) /* Colorie par récurrence un max de cases de la col j de A' */
      /* ok=Faux si détection d'impossibilité, ok=Vrai sinon */
      si non(ok) **alors retourner** (Faux,V) /* V matrice vide */
      Nouveaux $\leftarrow \{i : i$ est la ligne d'une nouvelle case coloriée dans la colonne j de $A'\}$
      LignesAVoir $\leftarrow$ LignesAVoir $\cup$ Nouveaux ; ColonnesAVoir $\leftarrow$ ColonnesAVoir $- \{j\}$
  si toutes les cases de A' sont coloriées **alors retourner** (Vrai,A')
  retourner (NeSaitPas,A') /* cas où l'on ne peut pas conclure */

Algorithme 2 Algorithme de résolution complète

fonction ENUMERATION(A) /* A une grille entièrement non coloriée */
  (ok,A') $\leftarrow$ COLORATION(A)
  si ok=Faux **alors retourner** (Faux,V) /* V matrice vide */
  retourner [ENUM_REC(A',0,blanc) **ou** ENUM_REC(A',0,noir)]

Algorithme 3 Algorithme récursif d'énumération

fonction ENUM_REC(A,k,c) /* A une grille partiellement coloriée, k un indice de case, c une couleur */
  si $k = NM$ **alors retourner** Vrai /* car toutes les cases de A sont coloriées */
  $i \leftarrow \lfloor \frac{k}{m} \rfloor$; $j \leftarrow k \ mod \ m$ /* ligne i et colonne j de la case k */
  (ok,A') $\leftarrow$ COLORIERETPROPAGER(A,i,j,c)
  /* COLORIERETPROPAGER idem COLORATION sauf que A est partiellement coloriée en entrée, et $\hookleftarrow$
   on commence par colorier la case (i, j) de A' avec la couleur c, et on remplace la ligne 3 du code par : $\hookleftarrow$
   3: LignesAVoir $\leftarrow \{i\}$; ColonnesAVoir $\leftarrow \{j\}$ */
  si ok=Faux **alors retourner** (Faux,V) /* V matrice vide */
  si ok=Vrai **alors retourner** (Vrai,A')
  $k' \leftarrow$ indice de la prochaine case indéterminée à partir de $k+1$
  retourner [ENUM_REC(A',k',blanc) **ou** ENUM_REC(A',k',noir)]