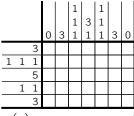


# Informatique I - Projet N° 02 † † †

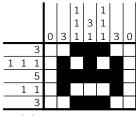
## Sujet 2 : Résolution de NonoGrams!

## I - Fonctionnement

Un nonogram, aussi connu sous le nom de *picross*, est un jeu de logique qui mène l'utilisateur à colorier une grille en noir et blanc. Pour chaque ligne et pour chaque colonne, une liste de nombres donne la taille des séries de cases noires successives. La figure 1 donne un exemple de puzzle avec sa solution.



(a) nonogram proposé



(b) nonogram résolu

Figure 1 – Exemple de nonogram

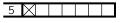
Dans le cas général, le seul moyen d'identifier la solution unique d'un nonogram est de tester toutes les combinaisons possibles. Heureusement, pour la majorité des puzzles prévus pour les humains, des règles de résolution permettent de compléter le dessin par itérations successives.

Par exemple, la  $1^{re}$  colonne de la figure 1 peut être résolue individuellement :

0

Une seule solution est possible, avec toutes les cases vides :

La 3<sup>e</sup> ligne donne alors :



Ce qui permet de remplir certaines cases noires (deux cases restent inconnues) :



La résolution du puzzle sera principalement basée sur la résolution d'une *ligne* (notion qui regroupe indistinctement les lignes et les colonnes du puzzle original). Une ligne est constituée d'une série d'instructions (les valeurs des indices) et d'une série de cases pouvant être dans un des trois états : blanc, noir ou inconnu.

### II - Travail à réaliser

Votre programme finale devra réaliser les différentes fonctionnalités suivantes :

→ Modélisation et sauvegarde de Nonograms :

Les nonograms seront représentés dans des fichiers suivant le format défini ci-dessous. Votre programme devra permettre de charger un nonogram depuis un nom de fichier fourni comme paramètre en ligne de commande <sup>1</sup>.

Pour la représentation d'un nonogram dans un fichier, la première ligne indique le nombre de lignes et de colonnes. Chaque ligne consécutive représente une des lignes puis une des colonnes avec le nombre d'indices suivi de leurs valeurs. L'exemple

<sup>1.</sup> http://www.adaic.org/resources/add\_content/standards/05rm/html/RM-A-15.html





nombre de lignes (5) et nombre de colonnes (7) 1 3 indications de la 1re ligne (1 élément) : 3 indications de la 2º ligne (3 éléments) : 1 1 1 indications de la 3º ligne (1 élément) : 5 3 1 1 1 1 5 indications de la 4e ligne (2 éléments) : 1 1 1 1 indications de la 5e ligne (1 élément) : 3 3 indications de la 1<sup>re</sup> colonne (0 élément) 0 indications de la 2e colonne (1 élément) : 3 suivant représente le puzzle de la figure 1. indications de la  $3^e$  colonne (3 éléments) :  $1\ 1\ 1$ indications de la  $4^e$  colonne (2 éléments) :  $3\ 1$ 2 3 indications de la 5e colonne (3 éléments) : 1 1 1 3 1 1 1 indications de la 6e colonne (1 élément) : 3 1 3 indications de la 7e colonne (0 élément) : 0

#### → Vérification de la cohérence lignes/colonnes :

Votre programme devra vérifier que le nombre total de cases noires sur l'ensemble des lignes est bien égale au total des cases noires sur l'ensemble des colonnes. Par exemple, le puzzle figure 2 possède trois cases noires au total selon les indices des lignes, et deux cases noires au total selon les indices des colonnes. Ce puzzle est donc impossible.



Figure 2 – Nonogram impossible (totaux différent)

De même, la somme des cases d'une ligne ou d'une colonne ne devra pas dépasser les dimensions du puzzle. Par exemple, le puzzle figure 3 est impossible, car la ligne devrait contenir au moins quatre cases (2+1+1), tandis que la dernière colonne devrait en contenir au moins deux.



Figure 3 – Nonogram impossible (débordement de lignes)

#### → Affichage de l'état d'un puzzle :

Votre programme devra permettre de visualiser l'état d'un nonogram sur la sortie console, en distinguant les cases noires ('@'), blanches ('.') et inconnues ('?').

Puzzle en cours de résolution :

1						1		1		
						1	3	1		
3				0	3	1	1	1	3	0
			3		?	@	@	@	?	
5	1	1	1		?		@	?	?	
			5		?	@	@	?	?	
7		1	1		?				?	
			3		?	@	?	@	?	
9										

#### Puzzle résolu :

						1		1		
2						1	3	1		
				0	3	1	1	1	3	0
4			3			@	@	@		
	1	1	1		@		@		@	
6			5		@	@	@	@	@	
		1	1		@				@	
8			3			@	@	@		

#### → Résolution d'un nonogram :

Votre programme devra permettre à l'utilisateur de choisir une ligne (ou une colonne) à résoudre, puis de saisir pour celle-ci de nouvelles valeurs de cases.

Votre programme devra également vérifier si un puzzle complété (donc sans case inconnue) est correct ou non.



## **Recommandations:**

Ce projet doit vous permettre de mettre en œuvre les nombreux et importants principes vus en cours. Ainsi vous devrez porter une attention toute particulière sur les points suivants :

- \* la modélisation du problème ; point incontournable pour la bonne réalisation des fonctionnalités demandées dans votre projet, vous devrez concevoir une structure de données la plus juste et abstraite possible.
- \* l'architecture modulaire; en effet vous devrez décomposer votre en travail en plusieurs modules, i.e. package. Chacun de ceux-ci devront être homogènes (cohérence entre Types et fonctionnalités présents dans celui-ci), complets et commentés.
- \* **tests des fonctionnalités** ; pour chaque module réalisé, vous testerez l'ensemble des fonctionnalités présentes en choisissant des jeux de tests adaptés.

## À produire à la fin du projet :

À l'issue de la dernière séance de projet, vous déposerez sur le moodle (à l'aide de la section de dépôt concernant votre sujet de projet) de l'école une unique archive portant le nom des deux élèves constituant le binôme, et contenant :

- \* l'ensemble de votre code source,
- \* un rapport au format Pdf contenant :
  - → les choix de conception effectués (et commentés) lors de la réalisation de votre structure de données;
  - → une représentation et explication de votre architecture modulaire (i.e. package)
  - → pour chaque package réalisé, les jeux de tests effectués.



