

TP n°5

Modélisation

On veut colorer la carte géographique de la Ruritanie de sorte que deux départements adjacents n'aient pas la même couleur. On veut également utiliser seulement trois couleurs.

Pour modéliser ce problème, on utilise une formule de la logique propositionnelle p telle que p est satisfaisable si et seulement si la carte peut être coloriée avec trois couleurs. De plus, chaque affectation qui satisfait p correspond à une coloration possible. La caractérisation (utilisée dans le cours) est la suivante :

Exercice 1 *Encodage des variables*

On va tout d'abord résoudre la question de la numérotation des variables. En effet, *minisat* lit des variables numérotées (x_2, x_5, x_{42}), tandis qu'ici on a des variables avec des paires d'indices. De plus, en informatique, les variables sont souvent numérotées à partir de 0, mais dans le format DIMACS elles sont numérotées à partir de 1 car 0 annonce une fin de ligne. On va donc encoder les paires en nombres en utilisant la fonction suivante :

$$\text{code}(n, j) = 3n + j + 1$$

La paire (0,0) devient 1, la paire (1,2) devient 6, etc.

L'intéressant est que cette fonction est *inversible* : étant donné un entier k , on peut trouver une et une seule paire (n, j) telle que $\text{code}(n, j) = k$ (Pourquoi et comment ?)

1. Écrire une méthode `int code (int n, int j)` qui calcule la fonction décrite ci-dessus.
2. Écrire une méthode `int departement (int k)` qui calcule le département correspondant au code k .
3. Écrire une méthode `int couleur (int k)` qui calcule la couleur du département correspondant au code k .

Exercice 2 *Écriture des contraintes*

Il faut maintenant exprimer dans un fichier DIMACS les contraintes du problème de la 3-coloration de la Ruritanie.

1. Écrire une méthode `String auPlusUneCouleurDepartement(int n)` qui, pour chaque département n génère les lignes DIMACS, correspondant à la formule

$$U_n = \bigwedge_{0 \leq j \leq 2} (x_{n,j} \rightarrow (\bigwedge_{h \neq j} \neg x_{n,h}))$$

Il faudra d'abord mettre cette formule en forme normale de conjonction.

2. Écrire une méthode `String auPlusUneCouleurCarte(int tot)` qui, donne le nombre total de départements $\text{tot} = N$, génère les lignes DIMACS, correspondant à la formule

$$U = \bigwedge_{n < N} U_n$$

3. Écrire une méthode `String auMoinsUneCouleurDepartement(int n)` qui, pour chaque département n génère une ligne DIMACS, correspondant à la formule

$$E_n = \bigvee_{0 \leq j \leq 2} (x_{n,j})$$

4. Écrire une méthode `String auMoinsUneCouleurCarte(int tot)` qui, donne le nombre total de départements $\text{tot} = N$, génère les lignes DIMACS, correspondant à la formule

$$E = \bigwedge_{n < N} E_n$$

5. Ecrire une methode `String departementsAdjacents(int n1, int n2)` qui, pour chaque paire $(n1, n2)$ genere trois lignes DIMACS, correspondant a la formule

$$C_{n1,n2} = \bigwedge_{0 \leq j \leq 2} (\neg x_{n1,j} \vee \neg x_{n2,j})$$

Combien de variables a-t-on defini et combien de lignes DIMACS a-t-on genere au total ?

Exercice 3 *Lecture du fichier carte*

Les cartes sont decrites par un fichier de la forme suivante :

1. la premiere ligne contient un entier qui indique le nombre total de departements
2. les lignes suivantes contiennent chacune deux entiers qui representent une paire de departements adjacents.

Ecrire une methode `boolean[][] grapheAdjacents(String nom)` qui renvoie un tableau de booleens a double entree representant le graphe des departements adjacents de la carte decrite dans le fichier `nom`. Ce tableau sera de taille $n \times n$, ou n est le nombre de departements et la case (i, j) aura la valeur `true` si la ligne i j apparait dans le fichier et `false` sinon.

Exercice 4 *Génération du fichier DIMACS*

En utilisant les methodes precedentes, ecrire une methode `String carteToDIMACS(String nom)` qui, lisant en entree le fichier `nom` decrivant une carte, genere les lignes DIMACS correspondant a la formule $U \wedge E \wedge \bigwedge_{\{n1,n2\} \in A} C_{n1,n2}$, ou A est l'ensemble des paires de nœuds pas le fichier.

On commencera par ecrire les lignes correspondant a la formule, en comptant le nombre de variables et le nombre de clauses, puis on ajoutera l'en-tête du fichier DIMACS.

Exercice 5 *Résolution avec minisat*

Ecrire une methode `boolean est3Colorable(String nom)` qui

1. genere le fichier DIMACS correspondant a la carte `nom`
2. fait un appel a *minisat* (utiliser `Execution.exec`, cf. page web du cours)
3. lit le fichier de sortie de *minisat* et renvoie `true` si la carte est 3-colorable, `false` sinon.

Tester avec les cartes fournies. L'Ile-de-France et la Ruritanie sont-elles 3-colorables ?

Exercice 6 *Obtenir toutes les solutions*

On cherche dans cet exercice a obtenir toutes les manieres possibles de 3-colorer la Ruritanie.

Pour cela on utilisera de maniere repetee *minisat* en rajoutant, a chaque iteration, dans le fichier DIMACS donne en argument, une clause correspondant a la negation de l'affectation retournée par *minisat* lors de l'iteration precedente. Ainsi, si par exemple *minisat* retourne l'affectation 2 3 -1 0, on ajoutera la ligne -2 -3 1 0.

1. Ecrire une methode `int[][] lireAffectationMinisat(String s)` qui prend un argument un fichier `s` correspondant a une reponse de *minisat* et retourne sous forme de tableau d'entiers l'affectation proposee dans `s` (nul si c'est insatisfiable).
2. Ecrire une methode `int[][] negationAffectation(int[][] aff)` qui retourne la negation de l'affectation donnee en argument.

3. Ecrire une methode `ajouteLigneDIMACS(String s, int[] ligne)` qui ajoute au fichier DIMACS `s` la clause representee par `ligne` en actualisant le nombre de clauses.

Ecrire en n une methode `String toutesColorations(String nom)` qui retourne toutes les colorations possibles d'une carte representee par le fichier `nom`. Tester avec les cartes fournies.