

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 colorée avec trois couleurs. De plus, chaque affectation qui satisfait p correspond à une coloration possible de la carte.

L'idée est d'utiliser des variables $x_{n,j}$, pour représenter le fait que le département n est coloré avec la couleur j .

Pour exprimer le fait que le département n peut avoir au plus une couleur on utilise la formule :

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

Pour exprimer le fait que chaque département peut avoir au plus une couleur on utilise donc la formule :

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

Pour exprimer le fait que le département n doit avoir au moins une couleur on utilise la formule :

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

Pour exprimer le fait que chaque département doit avoir au moins une couleur on utilise donc la formule :

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

Pour dire que deux départements $n1, n2$ ne peuvent pas avoir la même couleur :

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

Pour dire que deux départements adjacents ne peuvent pas avoir la même couleur, on doit utiliser l'ensemble A , qui définit l'adjacence :

$$C = \bigwedge_{\{n1,n2\} \in A} C_{n1,n2}$$

Une affectation qui satisfait la formule $U \wedge E \wedge C$ correspond à une coloration de la carte. Donc $U \wedge E \wedge C$ est satisfaisable si et seulement si la carte est 3-colorable.

Le but de ce TP est de modéliser ce problème et de le résoudre en utilisant *minisat*. A partir d'un fichier donné qui représente la carte de la Ruritanie, on générera un fichier DIMACS modélisant les contraintes que l'on donnera à *minisat* afin de connaître les éventuelles solutions.

On utilisera pour cela les classes d'entrée/sortie `Lecteur` et `Ecrivain` des méthodes pour manipuler des chaînes de caractères (`String.split`, `Integer.parseInt`,... consulter la documentation Java en ligne). On créera une nouvelle classe `Coloriage`

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, donné 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, donné le nombre total de départements $\text{tot} = N$, génère les lignes DIMACS, correspondant à la formule

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

5. Écrire une méthode `String departementsAdjacents(int n1, int n2)` qui, pour chaque paire $(n1, n2)$ génère trois lignes DIMACS, correspondant à 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 défini et combien de lignes DIMACS a-t-on généré au total ?

Exercice 3 Lecture du fichier carte

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

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

Ecrire une méthode `boolean[][] grapheAdjacents(String nom)` qui renvoie un tableau de booléens à double entrée représentant le graphe des départements adjacents de la carte décrite dans le fichier `nom`. Ce tableau sera de taille $n \times n$, où n est le nombre de départements et la case (i, j) aura la valeur `true` si la ligne i j apparaît dans le fichier et `false` sinon.

Exercice 4 Génération du fichier DIMACS

En utilisant les méthodes précédentes, écrire une méthode `String carteToDIMACS(String nom)` qui, lisant en entrée le fichier `nom` décrivant une carte, génère les lignes DIMACS correspondant à la formule $U \wedge E \wedge \bigwedge_{\{n1,n2\} \in A} C_{n1,n2}$, où A est l'ensemble des paires définies pas le fichier.

On commencera par écrire les lignes correspondant à 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 méthode `boolean est3Colorable(String nom)` qui

1. génère le fichier DIMACS correspondant à la carte `nom`
2. fait un appel à `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 à obtenir toutes les manières possibles de 3-colorer la Ruritanie.

Pour cela on utilisera de manière répétée `minisat` en rajoutant, à chaque itération, dans le fichier DIMACS donné en argument, une clause correspondant à la négation de l'affectation retournée par `minisat` lors de l'itération précédente. Ainsi, si par exemple `minisat` retourne l'affectation `2 3 -1 0`, on ajoutera la ligne `-2 -3 1 0`.

1. Ecrire une méthode `int[] lireAffectationMinisat(String s)` qui prend un argument un fichier `s` correspondant à une réponse de `minisat` et retourne sous forme de tableau d'entiers l'affectation proposée dans `s` (`null` si c'est insatisfiable).
2. Ecrire une méthode `int[] negationAffectation(int[] aff)` qui retourne la négation de l'affectation donnée en argument.

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

Ecrire enfin une méthode `String toutesColorations(String nom)` qui retourne toutes les colorations possibles d'une carte représentée par le fichier `nom`. Tester avec les cartes fournies.