

Rapport de projet Picross

CONNES Victor, saisissez vos noms

April 14, 2015

Contents

1	Description de l'implémentation	3
1.1	Les besoins du programme :	3
1.1.1	Pour les indices de chaque ligne/colonne:	3
1.1.2	Pour représenter la grille du picross :	3
1.1.3	Pour l'ensemble des indices de lignes/colonnes :	3
1.1.4	Pour ligmodif / colmodif	3
1.2	Les classes	4
1.2.1	La classe Cell :	4
1.2.2	La classe Liste :	4
1.2.3	La classe Tabliste :	4
1.2.4	La classe Matrice :	5
1.3	La classe Picross :	5
1.4	Les algorithmes remarquables :	5
1.4.1	SLG :	5

1 Description de l'implémentation

1.1 Les besoins du programme :

1.1.1 Pour les indices de chaque ligne/colonne:

- Imperatif à satisfaire:
 - structure de taille dynamique (nombre variable d'indice par ligne)
 - parcours en $\theta(n)$ (plusieurs méthodes recourant à un parcours total ou partiel de la structure)
 - relation d'ordre (le parcours ne se réalise que dans l'ordre de l'indice le plus en haut vers celui le plus en bas respectivement gauche-droite pour les lignes)
- Choix : la liste simplement chaînée, car elle correspond parfaitement à la spécification et paraît finalement très proche de la réalité

1.1.2 Pour représenter la grille du picross :

- Imperatif à satisfaire:
 - structure de taille fixe
 - accès en temps constant à chaque case
 - faciliter la copie de la structure ($\theta(n)$)
- Choix: La matrice car très proche de la réalité

1.1.3 Pour l'ensemble des indices de lignes/colonnes :

- Imperatif à satisfaire:
 - accéder en temps constant à chaque liste
 - garder un indicage cohérent avec la matrice
- Choix : Un tableau

1.1.4 Pour ligmodif / colmodif

- Imperatif à satisfaire:
 - ajout d'un élément en temps constant
 - retrait d'un élément en temps constant
 - taille dynamique
- Choix : Le choix naturel serait l'ensemble mais on choisit la liste simplement chaînée car déjà implémentée pour les indices de chaque ligne/colonne. Son comportement est similaire dans le cas d'un ajout et d'un retrait en début de liste.

1.2 Les classes

L'ensemble de nos classes disposent d'une surcharge de l'opérateur[] afin de faciliter l'affichage dans le terminal. De plus, il existe d'autres classes dans notre programme qui permettent d'améliorer l'affichage en utilisant une interface graphique. Nous ne distinguerons ici seulement les classes permettant la résolution du picross.

1.2.1 La classe Cell :

Elle représente un indice logique

- Attributs :
 - Val : qui représente la valeur de cette indice logique (cette valeur étant strictement entière positive et a priori non borné nous avons choisis de la représenter par un size_t)
 - Suiv : qui est pointeur sur la cellule suivante de la liste

1.2.2 La classe Liste :

- Attributs :
 - Longueur : qui représente la longueur de la liste. Cette attribut et incrémenter ou décrémenter automatiquement dès qu'il y a variations de la taille de la liste.
 - Fini : Nous indique si la ligne/colonne a laquelle est rattaché la liste est entièrement rempli (booléen à true) ou non (booléen à false), cette attribut et notamment très utile pour vérifier la condition d'arrêt de notre résolution (cf. main)
 - Tte : Pointeur vers le premier élément de la liste.
- Méthode remarquable :
 - Surcharge de l'opérateur() : Nous avons utiliser l'opérateur() comme un opérateur d'indexage d'une liste cela nous permet notamment de faciliter le parcours d'une liste.

1.2.3 La classe Tabliste :

- Attributs :
 - Tab : le tableau de liste
 - Taille : la taille du tableau
- Méthode remarquable :
 - Surcharge de l'opérateur[] : Nous permet d'utiliser comme si elle était simplement un tableau.

1.2.4 La classe Matrice :

- Attributs:
 - Tab : la matrice d'entier (-1 : blanc 0 : indéterminé 1 : noir)
 - Nbl : Nombre de ligne
 - Nbc : Nombre de colonnes

1.3 La classe Picross :

La classe picross est la classe dont les instances représente les picross c'est dans cette classe que sont implémenté nos méthodes de résolution.

1.4 Les algorithmes remarquables :

1.4.1 SLG :

Spécification :

Complexité pire des cas : $\theta(n^2)$

Complexité meilleur des cas : $\theta(n)$