Faire en latex

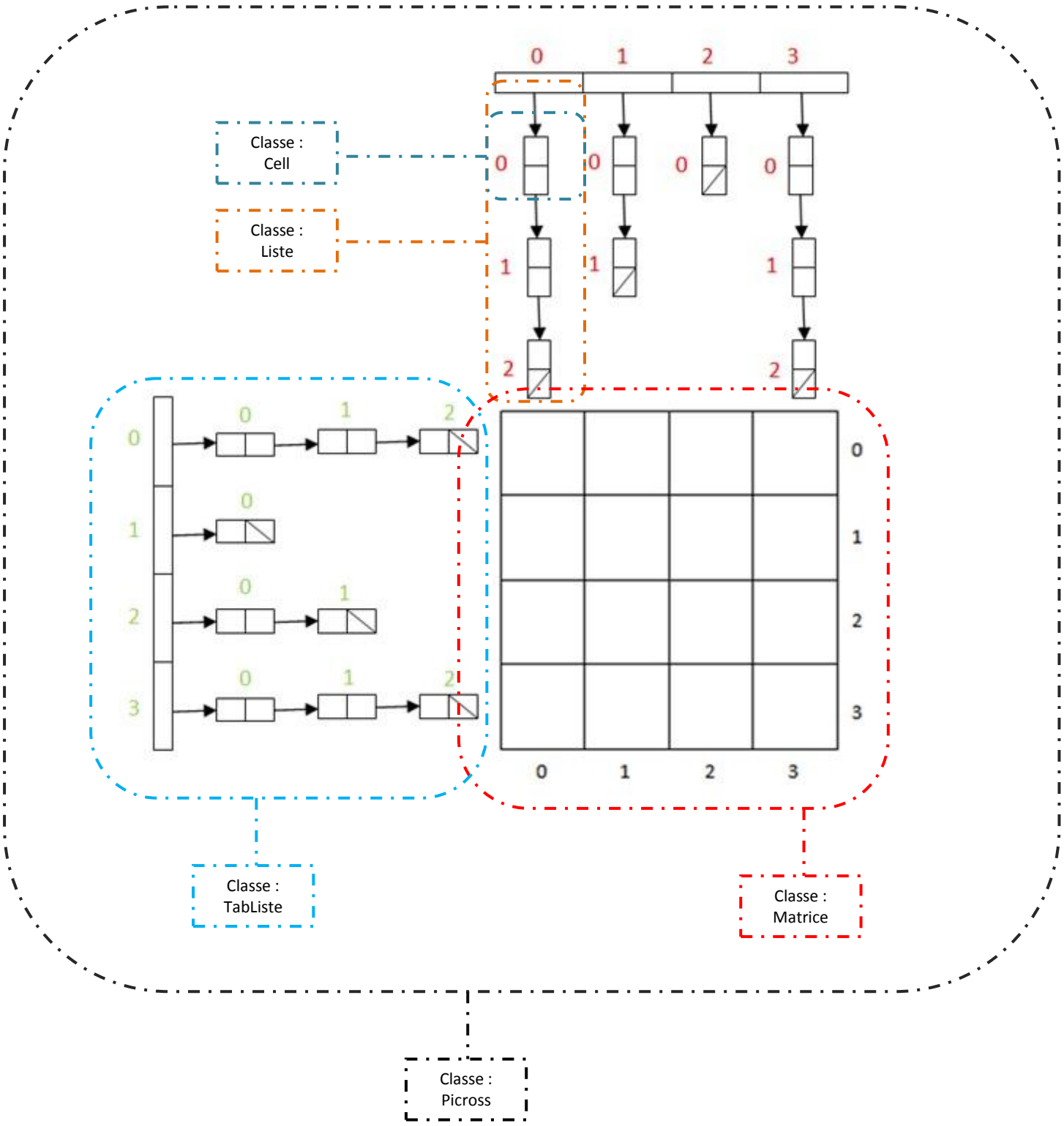

Schéma classe/structure projet Picross

Classe :Picross

Attribut :

Matrice Tab;
TabIndice Lignes ;
TabIndice Colonnes ;
Liste Ligmodif ;
Liste Colmodif ;

Constructeur:

Picross(Matrice Tab,TabIndice Lignes,TabIndice Colonnes) ;

Accesseurs(lecture/ecriture) :

Méthode :

Classe :TabListe

Attribut :

Liste[] Tab ;
Size_t taille ;

Constructeur:

TabListe(size_t taille) ;

Accesseurs(lecture/ecriture) :

Méthode :

Error() :void

Surcharge :

Operateur[] ;
Operator<< ;

Classe :Liste

Attribut :

Cell* tete;
Size_t longueur ;
Bool fini ;

Constructeur:

Liste ();//liste vide

Accesseurs(lecture/ecriture) :

Méthode :

Ajouter(Cell) :void
Operator() :cell*
IsNull() :bool
cutHd() :size_t
cutTl() :size_t
putFin(size_t) :void
add(cell*) :void
somCell : size_t
somElem :size_t
appartient :bool

Surcharge :

Operator<<
Operator=
Operator()

Classe :Matrice

Attribut :

int** tab;
Size_t nbl, nbc ;

Constructeur:

Matrice(size_t nbl,size_t nbc) ;

Accesseurs(lecture/ecriture) :

Surcharge :

Operator<< ;

Classe : Cell

Attribut :

Size_t val
Cell* suiv

Constructeur:

Indice ();//bool a false
Indice (size_t) ;//bool a false

Accesseurs(lecture/ecriture)
Surcharge :

Operator=

Les besoins du programme :

Pour les indices de chaque ligne/colonne:

-structure de taille dynamique (nombre variable d'indice par ligne)

-parcours en teta(n) (plusieurs méthodes recourant à un parcours total ou partiel de la structure)

-relation d'ordre (le parcours ne sont réaliser 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 chainés, car elle correspond parfaitement au spécification et parait finalement très proche de la réalité

Pour représenter la grille du picross :

-structure de taille fixe

-accès en temps constant a chaque case

-faciliter de copie de la structure (teta(n))

Choix: La matrice car très proche de la réalité

Pour l'ensemble des indices de lignes/colonnes :

-accéder en temps constant a chaque liste

-garder un indicage cohérent avec la matrice

Choix :Un tableau

Pour ligmodif /colmodif

-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 chainés car déjà implémentés 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 ;

Les classes

L'ensemble de nos classe dispose d'une surcharge de l'operateur << 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 resolution du picross.

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 choisit de la représenter par un size_t)

Suiv : qui est pointeur sur la cellule suivante de la liste

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)

Tête : Pointeur vers le premier élément de la liste.

Méthode remarquable :

Surcharge de l'opérateur() : Nous avons utiliser l'operateur() comme un operateur d'indexage d'une liste cela nous permet notamment de faciliter le parcours d'une liste.

La classe Tabliste :

Attribut s:

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.

La classe Matrice :

Attribut s:

Tab : la matrice d'entier (-1 : blanc 0 : indéterminé 1 : noir)

Nbl,nbc :Nombre de ligne, Nombre de colonnes

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.

Les algorithmes remarquables :

SLG :

Complexité pire des cas : teta n^2

Complexité meilleur des cas : teta n