Jeu Connexion

Ivan Dromigny--Chevreuil, Killian Fretaud

Groupe 501A

Licence 3 Informatique

2016 -2017

# Introduction

# 1. Les structures de données utilisées

Afin de parvenir à la résolution du problème, nous avons utilisé plusieurs structures de données :

* Classe-Union : Elle-même représenté par arbre sous forme de tableau.

# 2. Méthodes obligatoires

## 1. colorerCase

Fonction colorerCase (entier val) : booléen

Début

Si (valeur = 0 et (val = 1 ou val = 2 ou val = 3 )) alors

valeur 🡨 val ;

Si (val = 1) alors

couleur 🡨 rouge ;

Sinon

Change couleur bleu ;

FinSi

Retourner vrai ;

Sinon

Retourner faux ;

Fin

Description : Cette méthode vérifie si la case sélectionné est bien de couleur blanche (valeur = 0) et que le paramètre entré est bien un de ceux attendu. Si c’est le cas, la case est colorée en fonction de la valeur donné et vrai est retourné. Sinon retourne faux.

Raisons : On ne peut pas faire plus simple.

Complexité : O(1)

## 2. afficheComposante

Fonction afficheComposante(entier x, entier y)

Début

Si(TCase[x+y\*taille].valeur != 0) alors

entier rac 🡨 TComp[x+y\*taille].composante;

entier tmp[];

public void afficheComposante(int x, int y){

if (getVal(x, y) != 0) {

java.util.Timer t = new java.util.Timer();

int rac = getComp(x,y);

ArrayList<Integer> tmp = new ArrayList<Integer>();

// On ajoute toutes les Cellules de la composante dans un ArrayList temporaire

tmp.addAll(classe\_.getTousFils(rac%taille\_,rac/taille\_));

tmp.add(rac);

// Implémentation d'un timer afin de permettre un affichage en clignotement

class MonAction extends TimerTask {

int nbRep = 6;

public void run() {

if(nbRep > 0){

for (int i = 0; i < tmp.size(); ++i) {

tab\_[tmp.get(i)].colorerTemp();

}

--nbRep;

}

else

t.cancel();

}

}

// Appelle de la fonction pour le clignotement

t.scheduleAtFixedRate(new MonAction(),0, 1000);

}

}

Description : Si la case sélectionnée n’est pas blanche (valeur != 0) alors on récupère tous les éléments de la composante à laquelle la case appartient. Puis on fait clignoter en vert toutes les cases correspondantes. (ici 6 fois à intervalle de 1 sec)

C : nombre d’éléments dans la composante

Complexité : O(C)

3. existeCheminCases

Fonction existeCheminCases(entier x, entier y, entier z, entier t) : booléen

Si(TComp[x+y\*TComp.taille].composante = TComp[z+t\*Tcomp.taille].composante) alors

retourner vrai ;

Sinon

retourner faux ;

FinSi

Fin

Description : Vérifie si les deux cases font partie de la même composante, si oui retourne vrai, retourne faux sinon.

Complexité : O(1)

4. relierCasesMin

5. nombreEtoiles

Fonction nombreEtoiles(entier x, entier y) : entier

Début

entier compt 🡨 0 ;

entier rac 🡨 TComp[x+y\*taille].composante;

entier tmp[] ;

tmp.add(tout élément de rac);

Pour i de 0 à tmp.taille faire

Si(TCase[tmp[i]].estBase) alors

Compt 🡨 compt +1;

Finsi

FinPour

Retourner compt;

Fin

Description : Initialise un compteur à 0, puis récupère dans un tableau tous les éléments de la composante à laquelle la case sélectionnée appartient. Enfin incrémente le compteur de 1 pour chaque case étant une base.

C : nombre d’éléments dans la composante

Complexité : O(C)

Question : - si expliqué peut on : Tcomp.taille devient taille

- les raisons pour lesquelles l’algorithme proposée est correct

- les raisons pour lesquelles vous considérez cet algorithme aussi efficace que possible

- jeux de données

- doit-on faire décompte pour complexité