INGI1122 - Projet 2017 : Rendu Glouton
Partie 1

Beznik Thomas – 18051400
Coppé Vianney – 49461400
Di Prinzio Florentin – 43711500

28 mars 2017

1 Théorie du problème

Il nous est donné une grille G sous forme de matrice, dont chaque tuile peut être ou non recouvrable.

A partir de cette grille, il nous est demandé de trouver une couverture dite localement optimale, c'est à dire une couverture dont chaque rectangle la composant ne doit pas avoir de voisin direct avec lequel il pourrait former un nouveau rectangle. Ici, la couverture est l'ensemble des rectangles en vert, délimités par les côtés rouges.

La grille est de la forme suivante :

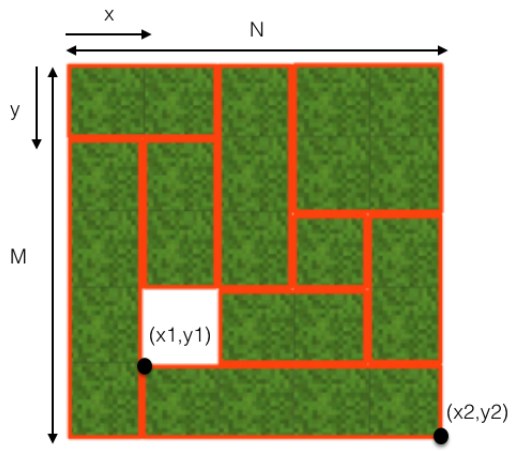

Figure 1: Grille à recouvrir

1.1 Décomposition du problème

Le problème peut être décomposé en plusieurs parties:

- Initialiser la couverture avec une liste de rectangles, dont la surface est équivalente à 1 tuile, recouvrant la totalité de la partie recouvrable.
- Parcourir la liste des rectangles de la Couverture, et pour chaque rectangle vérifier s'il peut ou non être fusionné avec les autres rectangles dans la liste, et ensuite les fusionner si possible.
- Si 2 rectangles peuvent en effet être fusionnés, il faut les retirer tous deux de la liste et rajouter le nouveau qui recouvre la surface des 2 précédents à la liste des Rectangle de la Couverture.

1.1.1 Prédicats

a,b : Rectangle

pDoesNotOverlap(a,b): True si le Rectangle a n'empiète pas sur le Rectangle b.
pCanMerge(a,b): True si la fusion du Rectangle a et b forme un nouveau Rectangle.
pInRectangle(r : Rectangle, x : int, y : int): True si la coordonnée (x,y) est dans le Rectangle r.
pCouvertureDescendante(C_0: Couverture, C: Couverture): True si la couverture C est une modification de la couverture C_0 via des merge de rectangles de celle-ci (elle "descend" de la couverture C_0).

1.1.2 Fonctions

canMerge(Rectangle a, Rectangle b) returns Boolean b
Retourne True si les rectangles a et b peuvent être fusionnés, sinon False.

merge(Rectangle a, Rectangle b) return Rectangle c
Retourne un Rectangle c qui recouvre totalement les Rectangle a et b, et contenu dans leurs limites.

improve()
Parcourt toutes les paires de Rectangle a, b et fait appel à merge(a,b) si canMerge(a,b) retourne True.

optimize()
Méthode principale de la classe Couverture qui en calcule la couverture localement optimale.

contains(int x, int y) return boolean b
Renvoie True si la tuile aux coordonnées (x,y) est recouverte par la couverture.

2 Spécification

Dans nos définitions, nous utilisons les constantes N et M, respectivement la taille en abscisses et en ordonnées de la grille.

2.1 Rectangle

Un rectangle est constitué de 2 paires (x, y) qui représentent deux coins opposés : le coin supérieur gauche et le coin inférieur droit (cfr. figure 1).
`Rectangle ::= rect(x1, y1, x2, y2)`

- $ok_{rectangle}$(`rect(x1, y1, x2, y2)`) $= x_1 < x_2 \wedge y_1 < y_2 \wedge 0 \leq x_1, x_2 < N \wedge 0 \leq y_1, y_2 < M$
- $abs_{rectangle}$(`rect(x1, y1, x2, y2)`) = rectangle avec comme coin supérieur gauche et coin inférieur droit les points (x_1, y_1) et (x_2, y_2). Les coordonnées des 2 coins restants peuvent en être déduit: (x_1, y_2) (inférieur gauche) et (x_2, y_1) (supérieur droit). La surface comprise entre ces 4 coordonnées représente le Rectangle.

2.2 Couverture

Une couverture est un ensemble de rectangles recouvrant partiellement une grille.
`Couverture ::= nil | couv(Rectangle r, Couverture c)`

- – $ok_{couverture}$(`nil`) $= true$
  – $ok_{couverture}$(`couv(Rectangle r, Couverture c)`) $= ok_{rectangle}$(`r`) $\wedge$ $ok_{couverture}$(`c`) $\wedge$ $no\_overlap$(`couv(Rectangle r, Couverture c)`)
- – $abs_{couverture}$(`nil`) $= \{\}$
  – $abs_{couverture}$(`couv(Rectangle r, Couverture c)`) $= \{abs_{rectangle}$(`r`)$\} \cup abs_{couverture}$(`c`)

Pour caractériser le fait qu'une couverture contient des rectangles qui ne se chevauchent pas, on définit le prédicat :
$no\_overlap(\{rect_1, rect_2, ..., rect_k\}) = \forall i, j : 1 \leq i < j \leq k : pDoesNotOverlap(rect_i, rect_j)$

2.3 Prédicats

a,b : Rectangle

pDoesNotOverlap(a,b): $(a.x1, a.x2 \leq b.x1 \vee a.x1, a.x2 \geq b.x2) \vee (a.y1, a.y2 \leq b.y1 \vee a.y1, a.y2 \geq b.y2)$

pCanMerge(a,b): $(a.y1 = b.y1 \wedge a.y2 = b.y2 \wedge (a.x1 = b.x2 \vee a.x2 = b.x1)) \vee (a.x1 = b.x1 \wedge a.x2 = b.x2 \wedge (a.y1 = b.y1 \vee a.y2 = b.y2))$

pInRectangle(r : Rectangle, x : int, y : int): r.x1 <= x <= r.x2 $\wedge$ r.y1 <= y <= $r.y2$

pCouvertureDescendante(C_0: Couverture, C: Couverture): $\forall$ Rectangle r $\in$ Couverture C_0 : $r \in C \vee$ ($\exists$ Rectangle r2 $\in$ C: pCanMerge(r,r2) $\wedge$ r3 = merge(r,r2) $\wedge$ r3 $\in$ C)

2.4 canMerge(Rectangle r1, Rectangle r2) returns boolean B

Pour que 2 rectangles soient *mergeable*, il faut qu'il y ait 1 arrête commune aux 2 rectangles.

@Pre: $ok_{rectangle}(r1) \wedge ok_{rectangle}(r2)$
@Post: B $\Leftrightarrow$ pCanMerge(r1, r2)
@Modifies: $\emptyset$

2.5 merge(Rectangle r1, Rectangle r2) returns Rectangle r3

Pour pouvoir utiliser la fonction merge sur les rectangles r1 et r2, il faut d'abord être sûr qu'ils puissent bien être mergés. Les sommets du rectangle résultant du merge doivent être les coins externes; donc le coin supérieur gauche $(x1, y1)$ (resp. droit, $(x2, y2)$) correspond au coin supérieur gauche (resp. droit) dont les coordonnées sont les plus petites (resp. grandes).

@Pre: canMerge(r1,r2)
@Post: r3.x1 = min(r1.x1,r2.x1) $\wedge$ r3.y1 = min(r1.y1,r2.y1) $\wedge$ r3.x2 = max(r1.x2,r2.x2) $\wedge$ r3.y2= max(r1.y2,r2.y2)
@Modifies: $\emptyset$

2.6 improve()

Improve va au moins merger deux rectangles si c'est possible et sinon elle va rien faire. Il faut aussi préciser dans la post que la couverture résultante est une transformation de l'originale et non pas une tout autre couverture (qu'elle "descend" de la première).

@Pre: Couverture C = C_0
@Post: pCouvertureDescendante(C_0,C)
@Modifies: Couverture C

2.7 optimize()

@Pre: Couverture C = C_0
Dans la liste de Rectangle de la Couverture C, il n'y a plus aucune paire de Rectangle non mergeable.
@Post: pCouvertureDescendante(C_0,C) $\wedge$ $\forall$ Rectangle r1, r2 $\in$ C: r1 $\neq$ r2 $\Rightarrow$ $\neg$ pCanMerge(r1,r2)
@Modifies: Couverture C

2.8 contains(int x, int y) returns boolean b

Les coordonnées (x,y) sont comprises dans la grille
@Pre: 0 <= $x < N \wedge 0$ <= $y < M$
Dans la liste de Rectangle de la Couverture C, il y a au moins un Rectangle r qui contient la coordonnée (x,y)
@Post: b $\Leftrightarrow$ $\exists$ Rectangle r $\in$ C: pInRectangle(x,y,r)
@Modifies: $\emptyset$