Rapport d'analyse
Projet de structures de données II

Activité d'Apprentissage S-INFO-020

Membres du groupe:
LABEEUW Dorian (dorian.labeeuw@student.umons.ac.be)
JOERTZ Jonathan (jonathan.joertz@student.umons.ac.be)

Année Académique 2017-2018
Bachelier en Sciences Informatiques Bloc 3

Faculté des Sciences, Université de Mons

November 21, 2017

Abstract

Ce *rapport d'analyse* est rendu dans le cadre de l'AA S-INFO-020 "Projet de structures de données II", dispensé par le professeur *Bruyère Véronique* en année académique 2017-2018.

Contents

1 Énoncé du problème

Dans le cadre de ce projet, nous devons comprendre et implémenter une solution au problème de *map overlay* selon un article de recherche fourni. L'utilité de ce problème est de retrouver l'ensemble des points d'intersections d'un ensemble de segments donné. Afin de réussir une implémentation efficace (on entend par là que l'on recherche une solution meilleure que la solution naïve en $O(n^2)$), il nous est nécessaire d'utiliser 2 structures de données expliquées ci-après (voir Description de Q et Description de T) et 3 algorithmes dont nous fournissons aussi une explication plus loin dans ce rapport (voir Explication de *FindIntersections(S)*, Explication de *HandleEventPoint(p)* et Explication de *FindNewEvent(s_l, s_r, p)*). De plus, certains cas poseront problème, nous donnons ici une brève explication de la façon dont nous avons prévu de les gérer (voir Gestion des segments horizontaux et Gestion des points d'intersection à au moins 3 segments). Une des applications du problème de *map overlay* se retrouve dans les cartes informatisées ou les *GPS*.

2 Description des structures de données

2.1 Description de Q

2.1.1 Description

Q est une *event queue* stockée dans un arbre binaire de recherche balancé sans doublons dont l'ordre correspond à un Ordre sous-jacent défini plus loin. Les éléments stockés dans un noeud de Q sont un point d'évènement et tous les segments commençant en ce point.

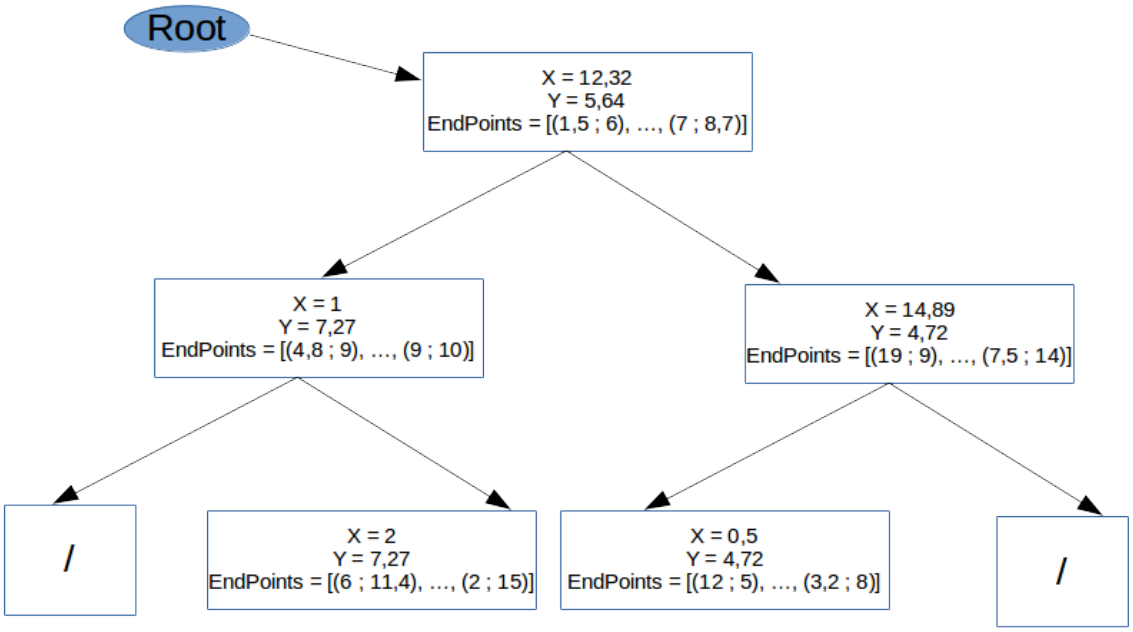
Les 3 opérations que Q doit supporter sont la suppression, la recherche du minimum et l'insertion sans doublons.

Q sera sans doublons car on ne veut traiter qu'une seule fois un point d'évènement.

La complexité des algorithmes d'insertion et de suppression (et par extension de recherche) peuvent et doivent être implémentés en O(log(m)) où m est le nombre de noeuds présents dans Q.

Le nombre de noeuds de Q est plus petit que $s + i$ où s est le nombre de segments et i le nombre d'intersections. Ceci est le pire cas, en effet, ce cas-là apparait si tous les *upper endpoints* (points de début des segments) sont au-dessus de la sweep line, si aucun *lower endpoint* n'est au-dessus d'elle et si chaque segment ne croise que ses voisins et pas les segments plus loin.

2.1.2 Exemple



2.1.3 Ordre sous-jacent

L'ordre est défini par le symbole $\prec$. Supposons que p et q sont deux points d'évènement. Alors on a $p \prec q \leftrightarrow (p_y > q_y) \vee (p_y = q_y \wedge p_x < q_x)$.

Le cas particulier des *segments horizontaux* se gère sans ajout particulier. En effet, le point gauche du segment sera considéré comme devant celui de droite.

2.2 Description de T

2.2.1 Description

2.2.2 Exemple