G 18 : GATTA Nicolas et SCHELLEKENS Arnaud

Total : Satisfaisant ($19/32$)

Remarques

Question 1 : Bien

Si on ne considère que les coordonnées en x, voyez-vous que celles-ci sont organisées selon une file à priorité ? Où est la coordonnée minimum (maximum) ? Cette file à priorité est-elle un tas ? Justifiez.

— Pour un tas, il faut également que tous les niveaux de l'arbre soient plein à l'exception du dernier qui doit être rempli de gauche à droite. Ce qui n'est pas le cas pour les PST.

Question 2 : À améliorer

Si on ne considère que les coordonnées en y, voyez-vous que celles-ci sont organisées selon un arbre binaire de recherche ? Où est la coordonnée minimum (maximum) ? Justifiez.

— Je ne comprends pas ce que vous voulez dire par « en réorganisant les données ». On pose la question dans le cas d'un PST, on ne peut pas changer sa structure.
— L'idée est que si on regarde en fait les y_{mid}, c'est un ABR.

Question 3 : Très Bien

De quelle façon est équilibré un arbre de recherche à priorité ?

— OK.

Question 4 : À améliorer

La construction d'un arbre de recherche à priorité peut se faire en $O(n \log_2(n))$ si n est le nombre de points de $\mathbb{R}^2$ contenus dans l'arbre. Expliquez comment on peut y parvenir et comment le prouver. Il faut sans doute utiliser une autre structure de données qui permette de calculer efficacement la médiane.

— La liste est triée ?
— Vous ne décrivez pas la complexité de l'algorithme de construction. Qu'en-est-t'il de la construction des sous-ensembles de points pour les constructions récursives des sous-arbres ? La sélection du nœud de plus haute priorité ?
— Contentez-vous de pré-trier les données selon y et implémentez la construction comme décrite dans la définition du PST dans la référence. Le calcul de la médiane se fait en temps constant dans ce cas. Il faut juste veiller à ne pas casser l'ordre des y lors du partitionnement de l'ensemble des points autour de la médiane.

Question 5 : Satisfaisant

La structure d'arbre de recherche à priorité permet d'obtenir efficacement l'ensemble des points de T qui sont contenus dans une fenêtre donnée de la forme $]-\infty : x'] \times [y : y']$. Expliquez comment adapter l'algorithme proposé pour traiter une fenêtre de la forme $[x : +\infty[\times [y : y']$, $[x : x'] \times]-\infty : y']$ ou $[x : x'] \times [y : y']$.

— Si on doit inverser le traitement entre x et y, alors il faut également utiliser un arbre créé avec cette même inversion. Il faut donc construire un arbre pour chaque direction non-bornée. Je ne vois pas comment on peut combiner la recherche dans deux arbres à la fois pour répondre au troisième point.
— Contentez-vous d'adapter les algos `ReportInSubtree` et `QueryPrioSearchTree` en manipulant les bornes et en simplifiant là où il faut les appels sur les sous-arbres droit/gauche.

Question 6 : Très Bien

Les auteurs du chapitre 10 font l'hypothèse que tous les points ont des coordonnées bien distinctes en x et en y. Expliquez pourquoi.

— OK.

Question 7 : Bien

Une technique est présentée à la page 111. Celle-ci permet de simuler des coordonnées distinctes pour un ensemble de points quelconques et une requête. Expliquez comment.

— Et pour les requêtes ?

Question 8 : À améliorer

Dans le chapitre, la technique présentée est adaptée à des points. Quelles différences importantes aurait-on avec des segments de droite ? Comment peut-on adapter la technique ?

— Pour extraire le minimum (maximum ?), si le segment est horizontal, on n'a besoin que d'une extrémité.
— Les explications ne suffisent pas pour comprendre comment vous comptez adapter les algorithmes.
— Je passe quelques détails, mais voici une idée de méthode. Construisez deux arbres, un pour les segments horizontaux, un autre pour les verticaux (pour lequel on échange x et y dans les algos). Les points à insérer sont les points de gauche (du bas pour les segments verticaux) des segments. Que la fenêtre de requête soit non-bornée à gauche ou pas, on effectue une requête non-bornée vers la gauche (ou non bornée vers le bas pour les verticaux), sur l'arbre correspondant (celui pour les horizontaux/verticaux). Pour chaque point reporté par la requête de windowing, on vérifie si le segment « intersecte » bien la fenêtre de requête, si oui, il est rapporté.