Université de Mons
Faculté des sciences
Département d'Informatique

Stuctures de données II

Rapport de l'exercice préléminaire

Professeur :
Véronique Bruyère
Assistant :
Pierre Hauweele

Auteurs :
Theo Godin
Thomas Bernard

Année académique 2022-2023

Table des matières

1 Question 1

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. En effet, on voit qu'il s'agit d'une file à priorité sur les coordonnées x. On va toujours prendre le point possédant la coordonnée x minimum afin de le placer dans le noeud racine et ce pour chaque noeud racine de chaque sous arbre. Ce qui fait que chaque racine de chaque sous arbre est un x minimal du jeu de points restants pour le sous arbre étudié. Donc, chacun des fils a bien une coordonée x inférieure ou égale à celle de son père.
En revanche, il ne s'agit pas d'un tas. En effet, les couches ne sont pas remplies de droite à gauche. De plus, la condition de répartition établie sur la médiane des y impose qu'il y ait autant de noeuds dans les sous arbres gauche et droit de la racine.
En conclusion, aucune condition n'impose de remplir la dernière couche complètement de droite à gauche s'il n'y a pas assez de noeuds pour le faire. On ne peut donc pas avoir de tas.
La coordonnée x minimum se trouve donc dans à la racine de l'arbre et la coordonnée x maximum dans une des feuilles.

2 Question 2

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. On voit qu'il s'agit d'un arbre binaire de recherche. Tout d'abord, les conditions pour avoir un ABR sont que pour tout noeud n, son fils gauche f_g est tel que $n.data > f_g.data$ et son fils droit f_d est tel que $n.data < f_d.data$.
Ici la "data" qui est utilisée pour comparer chaque noeud avec ses fils avant une insertion est y_{mid} la médiane des coordonnées y des points présents dans l'ensemble actuellement considéré. On retrouvera donc dans le sous arbre gauche de la racine l'ensemble des points dont les coordonnées y sont strictement inférieures à la médiane y_{mid}. Par conséquent, la médiane des coordonnées y des points du sous arbre gauche qui servira de "data" pour le fils gauche de la racine sera également strictement inférieure à la médiane y_{mid}. Pour le sous arbre droit il s'agit du même raisonnement mais en partant du fait que les coordonnées y des points du sous arbre droit seront strictement supérieures à la médiane y_{mid}, i.e pour la médiane des coordonnées y de ces points.
La coordonnée y minimum se trouve donc dans la feuille la plus à gauche de l'ABR puisqu'elle est strictement inférieure à toutes les médianes elle-même inférieures à celle de la racine. Et la coordonnée maximum se trouvera dans la feuille la plus à droite de l'ABR comme elle est strictement suppérieure à toutes les médianes elles-même supérieures à celle de la racine.

3 Question 3

De quelle façon est équilibré un arbre de recherche à priorité ? Le fait que les noeuds soient répartis dans les sous-arbres droit et gauche de la racine à l'aide d'une médiane assure par la définition de la médiane qu'il y aura autant de noeuds possédant une coordonnée $y < y_{mid}$ que de noeuds possédant une coordonnée $y > y_{mid}$. Ce principe est respecté de manière récursive pour chaque noeud de chaque sous arbre. On peut donc voir l'équilibre de cette structure comme le fait qu'il y ait autant de noeuds à droite qu'à gauche de chaque racine de chaque sous arbre.

4 Question 4

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 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 permet de calculer efficacement la médiane. L'algorithme de construction de l'arbre de recherche à priorité peut être trouvé sur internet (Wikipédia). A l'aide du master theorem vu au cours de SDD1 on peut prouver que cet algorithme est en $O(n \log n)$. On peut voir ci-dessous l'algorithme de création d'un arbre de recherche à priorité avec le calcul de complexité ainsi que la source du code :

```
tree construct_tree(data) {
   if length(data) > 1 {

       node_point = find_point_with_minimum_priority(data) // Select
           the point with the lowest priority

       reduced_data = remove_point_from_data(data, node_point)
       node_key = calculate_median(reduced_data) // calculate median,
           excluding the selected point

       // Divide the points
       left_data = []
       right_data = []

       for (point in reduced_data) {
           if point.key <= node_key
               left_data.append(point)
           else
               right_data.append(point)
       }

       left_subtree = construct_tree(left_data)
       right_subtree = construct_tree(right_data)

```

```
23          return node // Node containing the node_key, node_point and the
                left and right subtrees
24
25      } else if length(data) == 1 {
26          return leaf node // Leaf node containing the single remaining
                data point
27      } else if length(data) == 0 {
28          return null // This node is empty
29      }
30  }
```

Source : `https://en.wikipedia.org/wiki/Priority_search_tree`

Calcul de complexité : Les lignes 2,4,6,10,11,23,25,26,27 et 28 sont en O(1).

1. **find_point_with_minimmum_priority(data) :** Il s'agit de trouver la coordonnée minimum en x on va donc itérer au maximum n fois en O(1) on a donc du O(n).
2. **calculate_median(reduced_data) :** Si l'on a trié les données selon les y au sein d'une liste au préalable on peut calculer la médiane de reduced_data en O(1) car elle se trouver à l'indice : $\frac{len(reduced\_data)}{2}$
3. **boucle for :** Toutes les instructions au sein de la boucle for sont en O(1) on effectue au maximum n itérations dans la boucle ce qui veut dire que celle-ci est en O(n).
4. **Appels récursifs des lignes 20 et 21 :** On va appliquer le Master theorem afin de calculer la complexité des appels récursifs. La formule est la suivante : $T(n) = aT(\frac{n}{b}) + O(n^d)$ où :
   (a) **n :** taille du problème = n
   (b) **a :** nombre de sous-problèmes dans lequel l'appel est divisé. Dans notre cas $a = 2$.
   (c) $\frac{n}{b}$ **:** La taille de chaque sous problème. Dans notre cas c'est $\frac{n-1}{2}$
   (d) $O(n^d)$ **:** Correspond au coût de gestion du reste de la fonction ici en O(n) donc $d = 1$

   Le master Theorem dit que si $d = \log_b(a)$ alors on a $T(n) = O(n^d \log_b(n))$. On a bien que $1 = \log_2(2)$ donc Notre appel récursif est effectué en $O(n \log_2 n)$.

En conclusion si l'on additionne tout, on obtient que l'algorithme est bien en $O(n \log_2 n)$

5 Question 5

Ici la question implique de modifier la manière dont la fenêtre de recherche est bornée sur la coordonnée x. Rappellons que les coordonnées x sont organisées sous forme d'une file à priorité. Ce qui implique que les fils d'un noeud ont une coordonnée x supérieure ou égale à celle de leur père. L'algorithme proposé est initialement rédigé pour des requêtes bornées par un x maximum. Il y a donc une condition qui contrôle pour chaque noeud de chaque sous-arbre en partant de la racine si la coordonnée x est inférieure ou égale à la borne supérieure de la fenêtre de recherche.

1. **Fenêtre bornée par un x minimum.** Dans ce cas, on peut utiliser la définition de file à priorité à notre avantage. En effet, on change la condition de contrôle de la coordonnée x. A présent au lieu de contrôler si la coordonée x est inférieure ou égale à la borne supérieure de la fenêtre de recherche, on contrôle si la coordonnée x est supérieure ou égale à la borne inférieure de la fenêtre de recherche. Si c'est le cas, alors on peut accepter l'entièreté du sous arbres car les fils de la racine de ce sous arbres auront une coordonnée supérieure ou égale à celle de leur père.
2. **Fenêtre bornée par un x minimum et un x maximum.** Dans ce cas on associe les deux conditions. Il faut que pour chaque noeud du sous arbre considéré, la coordonnée x soit supérieure ou égale à la borne inférieure et inférieure ou égale à la borne supérieure de la fenêtre de recherche. Dans le cas où l'on rencontre un noeud dont la coordonée x est supérieure à celle de la borne supérieure on considère par la définition d'une file à priorité que tous ses descendants ont une coordonnée x supérieure à celle de la borne supérieure. On peut donc arrêter la recherche dans ce sous arbre.
3. **Fenêtre non bornée sur un y minimum.** Dans le cas où la fenêtre n'est pas bornée sur un y minimum on peut établir que tous les sous abres à gauche du noeud v_{split} auront une coordonnée en y comprise dans l'intervalle car la coordonnée y minimale se trouve dans la feuille la plus à gauche de l'arbre.
4. **Fenêtre non bornée sur un y maximum.** Dans le cas où la fenêtre n'est pas bornée sur un y maximum on peut établir que tous les sous arbres à droite du noeud v_{split} auront une coordonnée en y comprise dans l'intervalle car la coordonnée y maximale se trouve dans la feuille la plus à droite de l'arbre.

6 Question 6

Les auteurs du Chapitre 10 font l'hypothèse que tous les points ont des coordonnées bien distinctes en x et en y. Expliquez pourquoi. Dans l'arbre de recherche à priorité, la comparaison de noeuds se fait par rapport à la coordonée x et à la médiane y_{mid}. Un problème se pose lorsque 2 points devant être insérés ont une coordonée x identique et une coordonnée y se trouvant du même côté de la médiane y_{mid} de la racine du sous arbre. En effet, imaginons deux points $(2, 3)$ et $(2, 4)$ et la racine du sous arbre dont la médiane est $y_{mid} = 2$ dans lequel ils doivent être insérés. Les deux noeuds doivent figurer dans le sous arbres droit. Dès lors on va se servir de la règle de la file de priorité sur x et on va sélectionner la coordonnée x la plus petite en priorité. Or, ici, x = 2 dans les deux cas. On ne peut donc pas déterminer quel noeud sera le fils droit et quel noeud deviendra un potentiel fils de ce fils droit.

7 Question 7

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. La technique présentée est celle des "composite-numbers". Au lieu de définir un point par sa coordonnée x et sa coordonnée y tq $p = (p_x, p_y)$ on va le définir à l'aide d'un composite-number pour chaque coordonnée comme suit : $p = ((p_x|p_y), (p_y|p_x))$. Dès lors, les coordonnées x et y deviennent des composite-number et si l'on reprend l'exemple donné à la question 6 on voit qu'à présent les points seront respectivement représentés comme suit : $((2|3), (3|2))$ et $((2|4), (4|2))$. On voit à présent que l'on peut trouver lequel de ces deux points a une coordonnée x plus petite en comparant les deux membres de guache du composite-number. On aura bien que 2=2 mais également que 3<4. Ceci est également valable dans le cas où 2 coordonnées y sont identiques.

8 Question 8

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 ? Un point est représenté par une coordonnée x et y comme suit : $p = (x, y)$, si on lui applique la technique des composite-numbers, il sera représenté comme suit : $p = ((p_x|p_y), (p_y|p_x))$. On va donc essayer d'adapter cette notation à un segment de droite. Tout d'abord, on peut établir qu'un segment de droite est défini par 2 points : $p = (x, y)$ et $q = (x', y')$ qui sont ses extrémités. On peut également établir que dans les cas des segments de droite le problème énoncé à la question 6, n'est pas d'application car même si 2 segments avaient une extrémité en commun, ils ne pourraient pas avoir l'autre extrémité en commun sans être identiques, on pourrait donc se servir d'une des deux extrémités afin de déterminer quel segment est le plus

proche du x parent ou de la médiane parent. Dès lors, nous avions pensé à écrire un segment comme suit : $segment = ((x|y), (x'|y'))$ mais il s'avère que cela rendra le calcul de la médiane plus compliqué du fait qu'il faudra accéder aux 2 termes du composite-number. En conclusion un segment sera représenté comme suit : $segment = ((x|x'), (y|y'))$ c'est à dire, avec les coordonnées x de chaque extrémité comme première composante et les coordonnées y de chaque extrémité comme deuxième composante.