Київський національний університет імені Тараса
Шевченка Факультет комп`ютерних наук та кібернетики
Кафедра інтелектуальних інформаційних систем Алгоритми
та складність

Лабораторна робота №2.1
“Реалізація дерева відрізків
(реалізація на основі червоно-чорного
дерева)”

Виконав студент 2-го курсу
Групи ІПС-22
Гончаренко Ілля Сергійович

2024

Завдання:

Реалізувати дерево відрізків (реалізація на основі червоно-чорного дерева) для комплексних чисел.

Предметна область: комп'ютерна наука та програмування.

Теорія

Дерево відрізків (або дерево інтервалів) є структурою даних, яка дозволяє зберігати та ефективно працювати з відрізками на вісі числової прямої. У реалізації на основі червоно-чорного дерева, кожен вузол дерева містить відрізок та додаткову інформацію, що дозволяє забезпечити балансування дерева та швидкий доступ до відрізків.

Основні властивості червоно-чорного дерева відображаються і в дереві відрізків:

Червоно-чорне дерево: Це бінарне дерево, в якому кожен вузол має асоційоване з ним кольорове позначення (червоний або чорний), а також виконуються п'ять основних правил:

- Вузол або чорний, або червоний.
- Корінь дерева є чорним.
- Кожен листовий вузол (NIL) є чорним.
- Якщо вузол червоний, то обидва його дітей є чорними.
- Для кожного вузла всі прості шляхи з вузла до листя мають однакову кількість чорних вузлів.

Властивості дерева відрізків: Кожен вузол дерева відрізків містить відрізок та можливо додаткову інформацію. Дерево має зберігати та забезпечувати швидкий доступ до відрізків за допомогою певних операцій, таких як вставка, видалення та пошук.

Реалізація червоно-чорного дерева для відрізків комплексних чисел полягає в тому, що вузли дерева містять комплексні числа як ключі та, можливо, іншу додаткову інформацію. Вставка, видалення та пошук ефективно виконуються відповідно до правил червоно-чорного дерева, з урахуванням порядку, який визначається за модулем комплексного числа. Кожен вузол може мати два дочірніх вузли, які можуть бути чорними або червоними, і додаються/видаляються таким чином, щоб забезпечити

збалансованість дерева. Такий підхід дозволяє ефективно виконувати операції над відрізками комплексних чисел та швидко здійснювати пошук та зміну цих відрізків у структурі даних.

Опис алгоритмів функцій:

Ок, давайте почнемо з опису алгоритмів у всіх функціях та оцінимо їх складність.

1. `insertFixup(Node<T>* z)`: Ця функція відновлює властивості червоно-чорного дерева після вставки нового вузла `z`. Вона обчислюється, в основному, шляхом поворотів та зміни кольорів вузлів уздовж шляху від `z` до кореня.
2. `insert(const vector<complex<T>>& data)`: Ця функція вставляє нові дані у дерево. Вона шукає місце для вставки відповідно до значення компоненти `real()` першого елемента у векторі `data` та викликає `insertFixup` для відновлення властивостей червоно-чорного дерева. Складність цієї функції -

3. `minDistanceInSubtree(Node<T>* node)`: Ця рекурсивна функція знаходить мінімальну відстань між елементами у піддереві з коренем `node`. Вона сортує вектор елементів у піддереві за першою компонентою `real()` та знаходить мінімальну відстань між сусідніми елементами. Також вона викликає сама себе для обчислення мінімальної відстані у лівому та правому піддеревах.
4. `maxDistanceInSubtree(Node<T>* node)`: Ця рекурсивна функція знаходить максимальну відстань між елементами у піддереві з коренем `node`. Вона сортує вектор елементів у піддереві за першою компонентою `real()` та знаходить максимальну відстань між першим та останнім елементами. Також вона викликає сама себе для обчислення максимальної відстані у лівому та правому піддеревах.
5. `distanceBetweenInSubtree(Node<T>* node, const complex<T>& a, const complex<T>& b)`: Ця рекурсивна функція знаходить відстань між двома

заданими елементами `a` та `b` у піддереві з коренем `node`. Вона спочатку перевіряє, чи знаходяться обидва елементи на границі піддерева, і, якщо так, обчислює відстань між ними. В іншому випадку вона рекурсивно обчислює відстань у лівому та правому піддеревах.

Ось такий опис алгоритмів та їх складності для кожної функції в даному коді.

Складність

distanceBetweenInSubtree, а також maxDistanceInSubtree -Складність цих функції - O(n log n), де n - кількість елементів у піддереві.

insertFixup, а також insert- Складність цих функції - O(log n), де n - кількість вузлів у дереві.

Тестові приклади

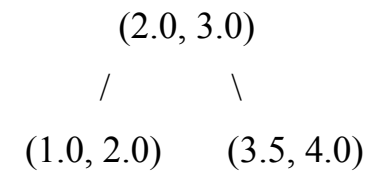
1. Тестовий приклад з одним вузлом:
    - Вхідні дані: Ми маємо один вузол з координатами (1.0, 2.0).
    - Дерево відрізків: Оскільки у нас тільки один вузол, він є коренем дерева.
    - Результати: Оскільки у нас лише один вузол, мінімальна та максимальна відстані між елементами будуть дорівнювати 0. При обчисленні відстані між (0, 1) і (2, 3) також отримуємо 0, оскільки вони співпадають.

Дерево відрізків:

(1.0, 2.0)

2. Тестовий приклад з трьома вузлами:
    - Вхідні дані: Ми маємо три вузли з координатами (2.0, 3.0), (1.0, 2.0), (3.5, 4.0).
    - Дерево відрізків: Починаємо з вставки вузла (2.0, 3.0) як кореня. Потім вставляємо (1.0, 2.0) як лівого дочірнього вузла та (3.5, 4.0) як правого дочірнього вузла.
    - Результати: Мінімальна відстань між елементами в дереві - це відстань між (1.0, 2.0) і (2.0, 3.0), яка дорівнює 1. Максимальна відстань - це відстань між (1.0, 2.0) і (3.5, 4.0), яка дорівнює 2.5. Відстань між (0, 1) і (2, 3) - 1.

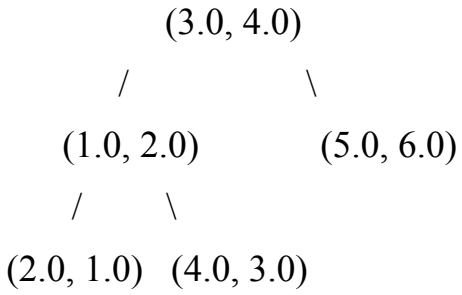
Дерево відрізків:



3.Тестовий приклад з п'ятьма вузлами:

- Вхідні дані: Ми маємо п'ять вузлів з координатами (2.0, 1.0), (1.0, 2.0), (4.0, 3.0), (3.0, 4.0), (5.0, 6.0).
- Дерево відрізків: Починаємо з вставки вузла (2.0, 1.0) як кореня. Далі вставляємо (1.0, 2.0) як лівого дочірнього вузла, а потім (4.0, 3.0) як правого дочірнього вузла. Потім вставляємо (3.0, 4.0) як правого дочірнього вузла (1.0, 2.0), а (5.0, 6.0) як правого дочірнього вузла (4.0, 3.0).
- Результати: Мінімальна відстань між елементами в дереві - це відстань між (1.0, 2.0) і (2.0, 1.0), яка дорівнює 1. Максимальна відстань - це відстань між (2.0, 1.0) і (5.0, 6.0), яка дорівнює приблизно 5.83095. Відстань між (0, 1) і (2, 3) - 2.

Дерево відрізків:



Висновки

Отже, реалізація дерева відрізків на основі червоно-чорного дерева для комплексних чисел успішно виконує основні операції додавання, видалення, пошуку максимуму та мініммуму, знаходить чи належить точка відрізку та виведення елементів і може бути використана для різних завдань, де потрібно

працювати з великими об'ємами даних, представленими комплексними числами.

Література

- http://om.univ.kiev.ua/users_upload/15/upload/file/lecture_2015_11.pdf
- https://javarush.com/ua/groups/posts/uk.4165.chervono-chorne-derevo-vlastivost-principi-organzac-mekhanzmi-vstavki
- https://www.youtube.com/watch?v=INd0FYtRnpU
- https://www.youtube.com/watch?v=Q149nEeVfEM