Київський національний університет імені Тараса Шевченка

Факультет комп’ютерних наук та кібернетики

Кафедра інтелектуальних програмних систем

Алгоритми та складність

Завдання №2

“ Реалізація декартового дерева (treap)”

Варіант №4

Виконала студентка 2-го курсу

Групи ІПС-22

Клевчук Марія Вячеславівна

Київ - 2025

Завдання

Реалізувати декартове дерево (treap) для типу даних - комплексні числа.

Теорія

Декартове дерево — це двійкове дерево отримане з послідовності чисел. Кожен вузол містить значення ключа та пріоритету. Структура є деревом пошуку за ключами та купою пріоритетів.

Властивість купи - предок будь-якого не кореневого вузла містить менше значення ніж той вузол.

У декартовому дереві можуть повторюватися ключі, але вони мають бути однозначно розташовані або справа, або зліва. Повторів пріоритетів варто уникати.

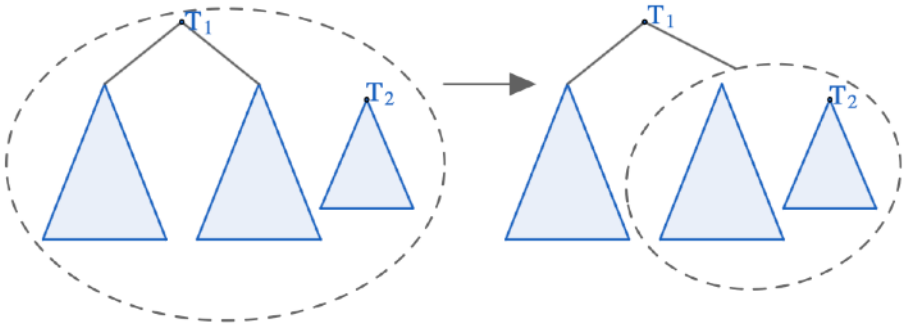
Висота декартового дерева з великою ймовірністю $\leq 4 \log_2 n$.

Для декартового дерева можна визначити дві базові операції - Merge (злиття) та Split (розбиття за ключем). На основі цих операцій відбувається вставка та видалення елементів у декартове дерево.

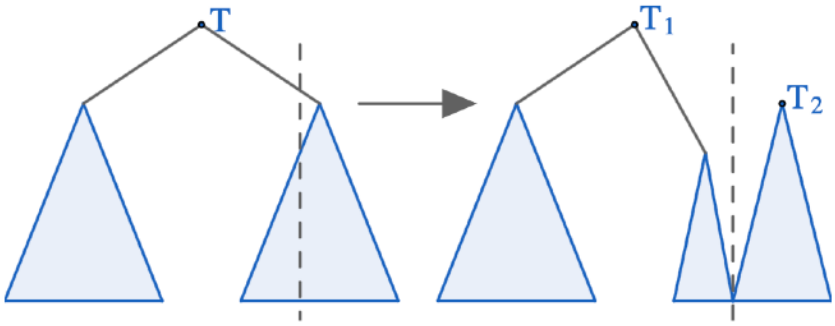
Алгоритм

Розглянемо алгоритми виконання основних операцій для декартових дерев.

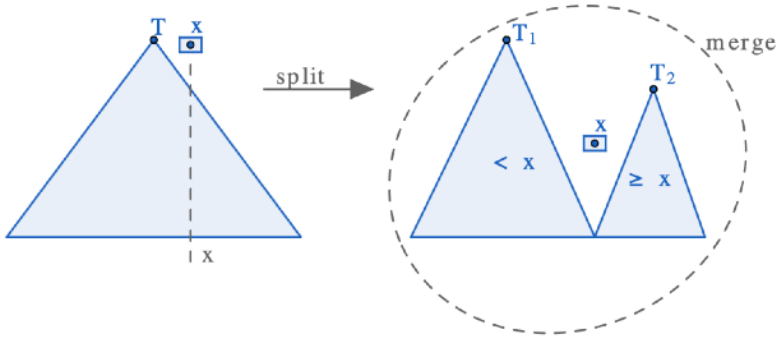
Merge (злиття). Для виконання злиття двох дерев має виконуватися умова, що всі ключі одного дерева не перевищують ключів іншого дерева. В якості нового кореня береться корінь дерева з більшим пріоритетом. Одне з піддерев зрозуміле, друге отримується рекурсивно.



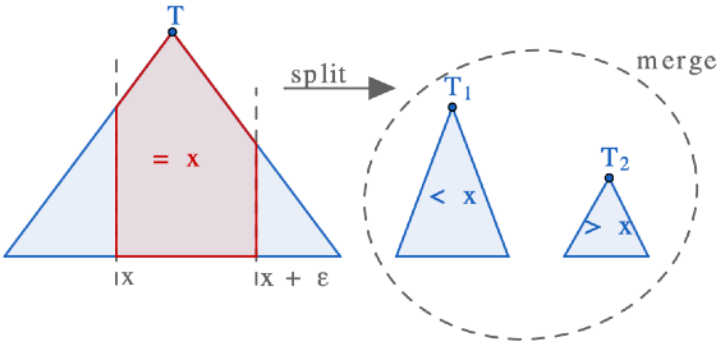
Split (розбиття за ключем). Результат: всі ключі одного дерева не перевищують ключів іншого. Звіряємо корінь з ключем. В залежності від результату бачимо, до якого з нових дерев належатиме корінь з одним із піддерев. Друге дерево рекурсивно виділяється з іншого піддерева.



Вставка елемента. Спускаємось по дереву за значенням ключа нового елемента, і знаходимо елемент, у якого значення пріоритету менше за пріоритет нового елемента. Викладаємо Split починаючи зі знайденого вузла. Отримані ліве та праве піддерево стають нащадками нового елемента.



Видалення елемента. Спускаємось по дереву і знаходимо елемент, який потрібно видалити. Викликаємо Merge для нащадків видаленого елемента.



Складність алгоритму

Час виконання операцій Merge та Split складає O(lg n). Такий же час виконання мають операції, що складаються з скінченної кількості викликів Merge та Split.

Мова реалізації алгоритму

C++

Модулі програми

• Клас Complex - для реалізації комплексних чисел.

Поля: int real — дійсна частина, int imag — уявна частина.

Методи:

- double modulus() const — обчислює модуль числа,

- bool operator<(const Complex& other) const — порівнює два комплексні числа за модулем, якщо модулі рівні — за дійсною частиною,

- bool operator==(const Complex& other) const — перевіряє рівність двох чисел.

• Клас Node - для реалізації декартового дерева.

Поля: Node* root — корінь дерева.

Методи:

- int getUniquePriority() — генерує унікальний випадковий пріоритет,
- void split(Node* t, Complex key, Node*& left, Node*& right) — розбиває дерево на дві частини за key,
- Node* merge(Node* left, Node* right) — об'єднує два піддерева,
- void insert(Complex key) — вставляє новий елемент у дерево,
- Node* erase(Node* root, Complex key) — рекурсивно видаляє вузол з дерева,
- int findLevel(Node* node, Complex key, int level) — знаходить рівень вузла в дереві,
- void printLevels() — друкує дерево по рівнях.

Інтерфейс користувача

Користувач задає кількість комплексних чисел, які потрібно внести у декартове дерево. У форматі real imag вводяться комплексні числа. У консоль виводиться декартове дерево.

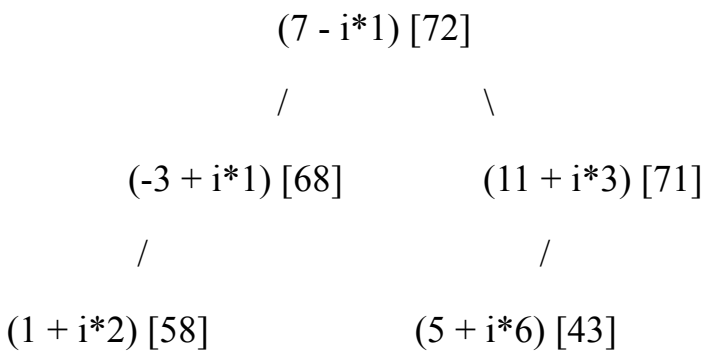
Через консоль користувач може ввести елемент, який необхідно видалити, додати в декартове дерево, або рівень якого необхідно знайти.

Тестові приклади

Приклад 1. Створення декартового дерева.

Створюємо декартове дерево з 5 елементів. Перший елемент голова - (з найбільшим пріоритетом). Далі поділяємо інші елементи на менші за голову

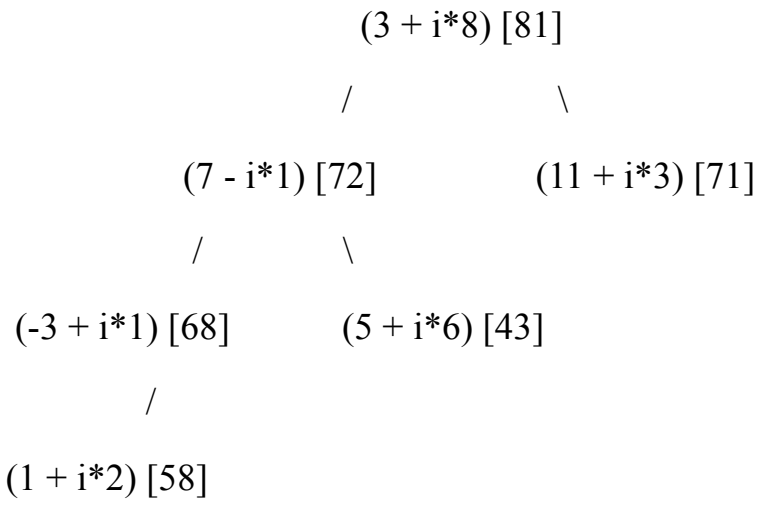
(утворюються ліве піддерево) та більші за голову (утворюють праве піддерево). Для кожного піддерева рекурсивно виконуємо ту ж процедуру.




```
Enter number of complex numbers: 5
Enter 5 complex numbers (real imag):
1 2
-3 1
5 6
11 3
7 -1
Treap levels:
Level 1: (7 - i*1) [72]
Level 2: (-3 + i*1) [68] (11 + i*3) [71]
Level 3: (1 + i*2) [58] (5 + i*6) [43]
```

Приклад 2. Вставка елемента.

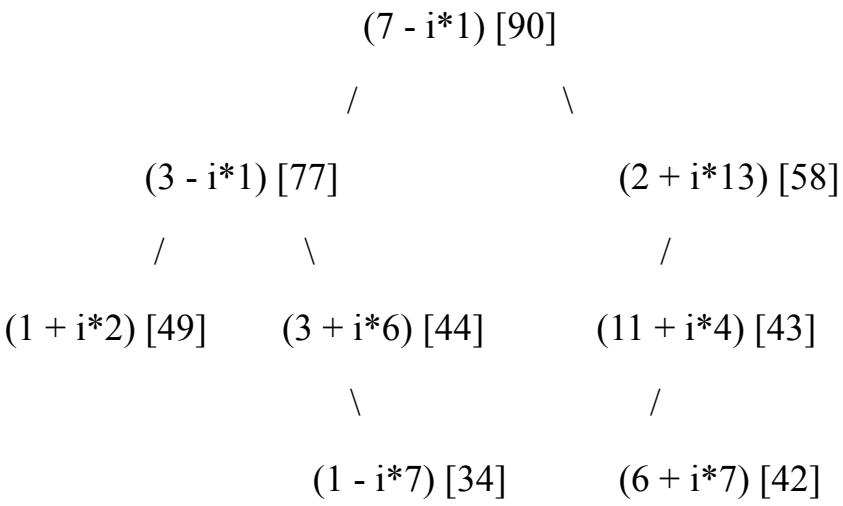
Нехай ми хочемо в попереднє дерево вставити елемент (3 + i*8) з пріоритетом 81. Так як пріоритет більший за будь-який існуючий елемент, то новий елемент стане головою.



```
Enter complex number to insert (real imag): 3 8
Treap levels:
Level 1: (3 + i*8) [81]
Level 2: (7 - i*1) [72] (11 + i*3) [71]
Level 3: (-3 + i*1) [68] (5 + i*6) [43]
Level 4: (1 + i*2) [58]
```

Приклад 3. Видалення елемента.

Утворимо нове декартове дерево.




```
Enter complex number to delete (real imag): 1 2
Treap levels:
Level 1: (7 - i*1) [90]
Level 2: (3 - i*1) [77] (2 + i*13) [58]
Level 3: (3 + i*6) [44] (11 + i*4) [43]
Level 4: (1 - i*7) [34] (6 + i*7) [42]
```

Видаляємо елемент 1 + i*2

```
Enter complex number to delete (real imag): 1 2
Treap levels:
Level 1: (7 - i*1) [90]
Level 2: (3 - i*1) [77] (2 + i*13) [58]
Level 3: (3 + i*6) [44] (11 + i*4) [43]
Level 4: (1 - i*7) [34] (6 + i*7) [42]
```

Приклад 4. Пошук рівня елемента.

```
Treap levels:
Level 1: (7 - i*1) [90]
Level 2: (3 - i*1) [77] (2 + i*13) [58]
Level 3: (1 + i*2) [49] (3 + i*6) [44] (11 + i*4) [43]
Level 4: (1 - i*7) [34] (6 + i*7) [42]
Enter complex number to find (real imag): 6 7
Element found at level 4
```

Висновки

У ході виконання лабораторної роботи було реалізовано декартове дерево (treap) — структуру даних, що поєднує властивості бінарного пошукового дерева та купи. Було досліджено алгоритми вставки, видалення та пошуку елементів у дереві, а також проведено тестування їхньої коректності.

Також було розглянуто механізм Split (розбиття) та Merge (об'єднання) дерев, які дозволяють гнучко модифікувати структуру для виконання складніших операцій.

Використані літературні джерела

- Алгоритми та складність. Лекція 3
- https://uk.wikipedia.org/wiki/Декартове_дерево
- https://en.oi-wiki.org/ds/cartesian-tree/
- https://cp-algorithms.com/data_structures/treap.html
- https://medium.com/@kush17041998/treap-cartesian-tree-86b3c43f8779