Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Кафедра інтелектуальних програмних систем

Алгоритми та складність

Лабораторна робота №2

“Реалізувати АА-дерево.”

Виконала студентка 2-го курсу

Групи ІПС-22

Челушкіна Валерія Олександрівна

Київ – 2025

**Завдання**

Реалізація АА-дерева для раціональних чисел.

**Теорія.**

АА-дерево – це форма збалансованого дерева, яке використовується для ефективного зберігання та отримання впорядкованих даних. Дерева AA названі на честь свого автора, Арне Андерссона.

АА-дерево є різновидом червоно-чорного дерева, форми бінарного дерева пошуку, яке підтримує ефективне додавання та видалення вузлів. На відміну від червоно-чорних дерев, червоні вузли в дереві AA можна додати лише як правий дочірній елемент. Іншими словами, жоден червоний вузол не може бути лівим дочірнім елементом.

Для балансування використовуються дві операції: SKEW (перекрут) та SPLIT (розбиття).

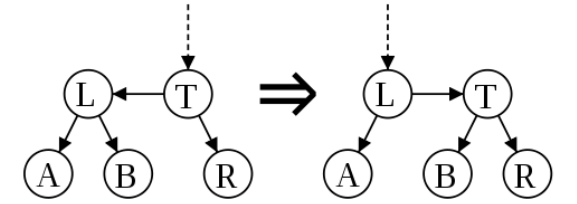
Для АА-дерев справедливі такі твердження:

* Рівень кожного листа дорівнює 1
* Рівень кожного лівого дочірнього елемента рівно на одиницю менший, ніж рівень його батька.
* Рівень кожної правої дитини дорівнює або на одиницю менше рівня її батька.
* Рівень кожного правого онука строго нижчий, ніж рівень його дідуся. Тобто не може бути 2 горизонтальних рівні підряд.
* Кожен вузол рівня вище одного має двох дочірніх вузлів.

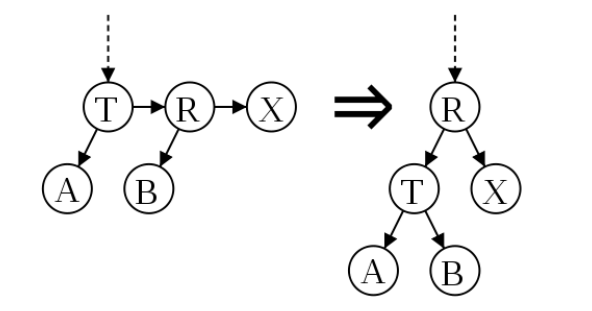
**Алгоритм.**

Розглянемо основні операції, потрібні для створення АА-дерева.

SKEW (перекрут) - усунення лівого зв’язку на одному рівні.



SPLIT (розбиття) - усунення двох правих зв’язків на одному рівні.



Додавання вузла:

Спочатку додаємо вершину як зазвичай, потім підіймаємось вгору, виконуючи для кожного вузла спочатку SKEW, а потім SPLIT.

Видалення вузла:

Після видалення листа при русі вгору спочатку за необхідності відбувається пониження рівня (рівень вузла має бути на 1 більшим мінімального рівня його синів), потім тричі викликається SKEW і двічі SPLIT:

T <= decrease\_level(T)

T <= skew(T)

right[T] <= skew(right[T])

right[right[T]] <= skew(right[right[T]])

T <= split(T)

right[T] <= split(right[T])

**Складність.**

Всі операції для створення АА-дерева виконуються за час O(lg n).

**Мова реалізації алгоритму.**

С++

**Модулі програми.**

* **Клас Rational.**

*Поля класу*: int num, den – чисельник та знаменник числа відповідно.

*Основні методи класу:*

Rational(int n = 0, int d = 1)– конструктор.

friend std::ostream &operator<<(std::ostream &os, const Rational &r) – перевиражений оператор виводу.

* **Клас AATree**

AANode \*skew(AANode \*node) – операція SKEW.

AANode \*split(AANode \*node) - операція SPLIT.

AANode \*insert(AANode \*node, Rational key) – вставка нового значення.

AANode \*decrease\_level(AANode \*node) – зниження рівня.

AANode \*remove(AANode \*node, Rational key) – видалення елемента.

int findLevel(AANode \*node, Rational key) – знаходження рівня елемента

void printTree(AANode \*node) – виведення дерева порівнево.

**Інтерфейс користувача.**

Введення даних відбувається із програми.

*Вхідні дані:*

вектор раціональних чисел

*Вихідні дані:*

дерево

**Тестові приклади**

**Приклад**

Подивимось приклад створення невеликого АА-дерева.

Почнемо з того що додаємо до пустого дерева вузол 10, рівень=1:

10 (L=1)

Додаємо вузол 20. Воно причепиться справа горизонтально до 10:

10 (L=1) -> 20 (L=1)

Додаємо 30:

10 (L=1) -> 20 (L=1) -> 30 (L=1)

Виникає порушення (2 послідовні праві дитини) тому перебалансовуємо дерево, використовуючи SKEW і SPLIT.

20 (L=2)

/ \

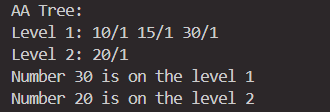
10 (L=1) 30 (L=1)

Додаємо 15:

20 (L=2)

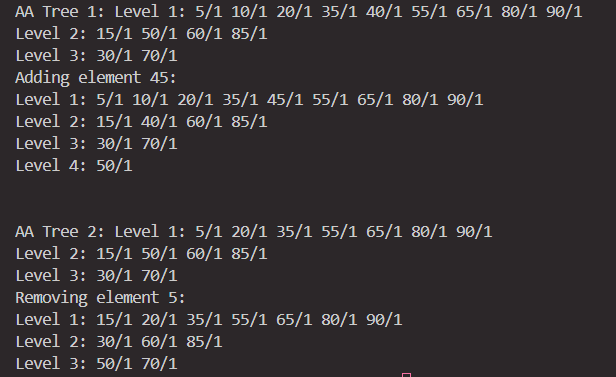
/ \

10 (L=1) -> 15 (L=1) 30 (L=1)

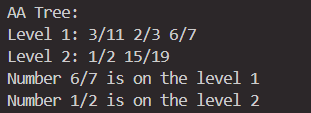


Ще декілька прикладів у середовищі С++:

*Приклад 1.*



*Приклад 2.*

**

**Висновки.**

У ході виконання лабораторної роботи було реалізовано АА-дерево як різновид збалансованого бінарного дерева пошуку. Досліджено його основні властивості та механізми балансування, зокрема операції SKEW (перекрут) і SPLIT (розбиття), які забезпечують збереження структурної рівноваги.

Практична реалізація показала, що АА-дерево ефективно підтримує впорядкованість даних і забезпечує швидке виконання операцій пошуку, вставки та видалення. Балансування відбувається за рахунок обмеження структури червоних вузлів, що запобігає надмірному зростанню висоти дерева.

Порівняння з іншими збалансованими деревами, зокрема червоно-чорними, підтвердило, що АА-дерева мають простішу реалізацію, хоча можуть бути трохи вищими. Це робить їх зручними для використання в ситуаціях, де важливе поєднання ефективності та простоти реалізації.

**Література.**

* Лекція з предмету «Алгоритми та складність #3»
* [**https://web.eecs.umich.edu/~sugih/courses/eecs281/f11/lectures/12-AAtrees+Treaps.pdf**](https://web.eecs.umich.edu/~sugih/courses/eecs281/f11/lectures/12-AAtrees+Treaps.pdf)