Київський національний університет імені Тараса Шевченка

Факультет комп’ютерних наук та кібернетики

Кафедра інтелектуальних інформацій систем

Алгоритми та складніcть

Лабораторна робота №2.1

Завдання №2.1 Реалізувати АА-дерево

Тип даних – Раціональні числа

Виконала студентка 2-го курсу

Групи ІПС - 22

Мандріченко Ксенія

2023

**Зміст**

[Теоретичні відомості 3](#_Toc102339827)

[Алгоритм 3](#_Toc102339828)

[Складність 5](#_Toc102339829)

[Мова програмування 5](#_Toc102339830)

[Модулі програми 6](#_Toc102339831)

[Інтерфейс користувача 7](#_Toc102339832)

[Приклади 7](#_Toc102339833)

[Висновок 8](#_Toc102339834)

[Література 8](#_Toc102339835)

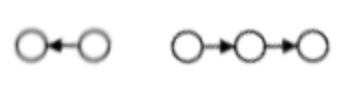
# **Теоретичні відомості**

# **АA-дерево** — структура даних, що є збалансованим двійковим деревом пошуку, яке є різновидом червоно-чорного дерева з додатковими обмеженнями. На відміну від червоно-чорних дерев, до однієї вершини можна приєднати вершину тільки того ж рівня, тільки одну і лише праворуч.

# АA-дерево названо за першими літерами імені та прізвища винахідника, Арне Андерссона, який вперше запропонував цю модифікацію червоно-чорного дерева у 1993 році.

**Властивості АА-дерева:**

* Рівень кожного листового вузла дорівнює одиниці.
* Рівень кожного лівого дочірнього елемента рівно на одиницю менше, ніж у його батька.
* Рівень кожного правого нащадка дорівнює або на одиницю менше, ніж у його батька.
* Рівень кожного правого онука значно менше, ніж у його прабатька.
* Кожен вузол рівня більше одиниці має двох дочірніх елементів.

В АА-дереві через суворі обмеження необхідно обробляти лише два види можливих розташувань вершин, щоб перевірити чи дотримується головне правило «один правий горизонтальний зв'язок». Тобто ми повинні перевірити, чи немає лівого горизонтального зв'язку, як на першому малюнку нижче і чи немає двох послідовних правих горизонтальних зв'язків, як на правому малюнку.

**Горизонтальне ребро** - ребро, що з'єднує вершини з однаковим рівнем.

У AA-дереві дозволені праві ребра, що не йдуть поспіль, і заборонені всі ліві горизонтальні ребра.

Для балансування АА-дерева потрібні такі дві операції:

**Skew()**- Усунення лівого горизонтального ребра. Робимо праве обертання, щоб замінити піддерево, що містить лівий горизонтальний зв'язок, на піддерево, що містить дозволений правий горизонтальний зв'язок.

**Split()** - Усунення двох послідовних правих горизонтальних ребер. Робимо ліве обертання і збільшуємо рівень, щоб замінити піддерево, що містить два або більше послідовних правих горизонтальних зв'язків, на вершину, що містить два піддерева з меншим рівнем.

# **Алгоритм**

Це програма, яка реалізує структуру даних AA- дерева для зберігання та пошуку раціональних чисел.

1. Програма визначає структуру під назвою Rational, яка представляє раціональне число. Вона має два цілих члени даних - num (чисельник) і denom (знаменник).

2.Програма визначає структуру під назвою Node, яка представляє вузол у AA- дереві. Кожен вузол містить поле даних Rational, поле рівня (яке використовується для балансування дерева) і вказівники на його лівий і правий дочірні вузли.

3.Програма визначає клас під назвою AATree, який представляє структуру даних AA- дерева. Клас має один приватний член даних, вказівник на кореневий вузол дерева.

4.Клас AATree визначає три функції: insert(), search() і inorder(). Функція insert() приймає параметр Rational і вставляє число в дерево. Функція search() приймає параметр Rational і шукає число в дереві, повертаючи true, якщо число знайдено, і false в іншому випадку. Функція inorder() використовується для тестування та друкує дерево в порядку зростання.

5.Функція insert() реалізована рекурсивно. Якщо поточний вузол дорівнює нулю, новий вузол створюється з даним Rational і повертається. Якщо даний Rational уже є в дереві, повертається існуючий вузол. В іншому випадку Rational рекурсивно вставляється в ліве або праве піддерево, залежно від його значення відносно поточного вузла.

6.Після того, як Rational вставлено, AA- дерево балансує себе за допомогою операцій нахилу та розділення. Якщо ліве піддерево вузла має той самий рівень, що й вузол, виконується обертання вправо. Якщо праве піддерево вузла має рівень на одиницю, що перевищує рівень вузла, а його праве піддерево має той самий рівень, що й вузол, виконується поворот ліворуч і рівень вузла збільшується.

7.Функція search() реалізована ітераційно. Починаючи з кореневого вузла, функція переміщається по дереву, порівнюючи Rational кожного вузла з даним Rational і рухаючись ліворуч або праворуч відповідно, доки не буде знайдено число або досягнуто нульового вузла.

8.Функція inorder() реалізована рекурсивно. Вона перетинає дерево в порядку, друкуючи кожне раціональне значення, коли воно зустрічається.

**Складність**

Часова складність операції вставки в AA-Tree, реалізоване у цій програмі, дорівнює O(log n), де n – кількість вузлів у дереві.

Операція пошуку має таку ж часову складність, як і операція вставки, тобто O(log n), оскільки вона слідує подібному шляху пошуку, що й операція вставки.

Обхід дерева за порядком, реалізований методом inorder, має часову складність O(n), де n — кількість вузлів у дереві.

Часова складність методів rotateRight і rotateLeft становить O(1), оскільки вони виконують операції з постійним часом.

# **Мова програмування**

С++

# 

# **Модулі програми**

**struct Rational** //структура під назвою Rational, яка представляє раціональне число, має два цілих члени даних - num (чисельник) і denom (знаменник).

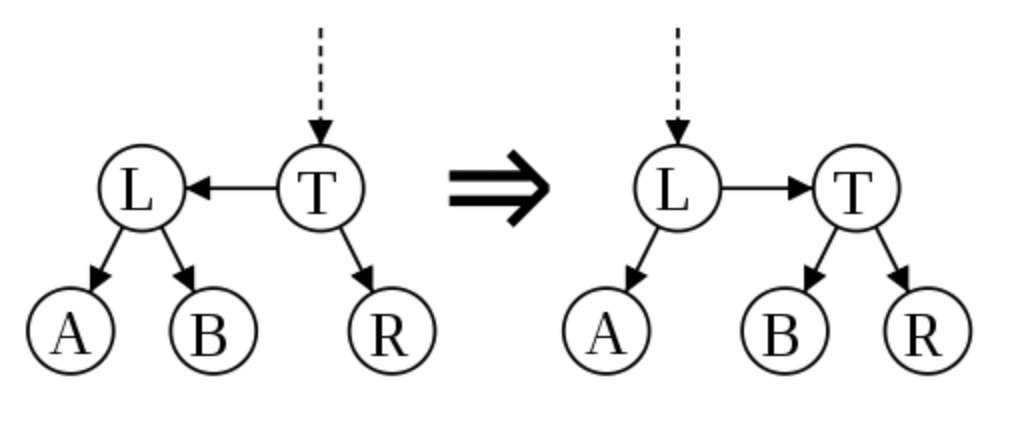
**struct Node** //структура під назвою Node, яка представляє вузол у AA- дереві.

**class AATree** //Клас AATree визначає три публічні функції-члени: insert(), search() і inorder().

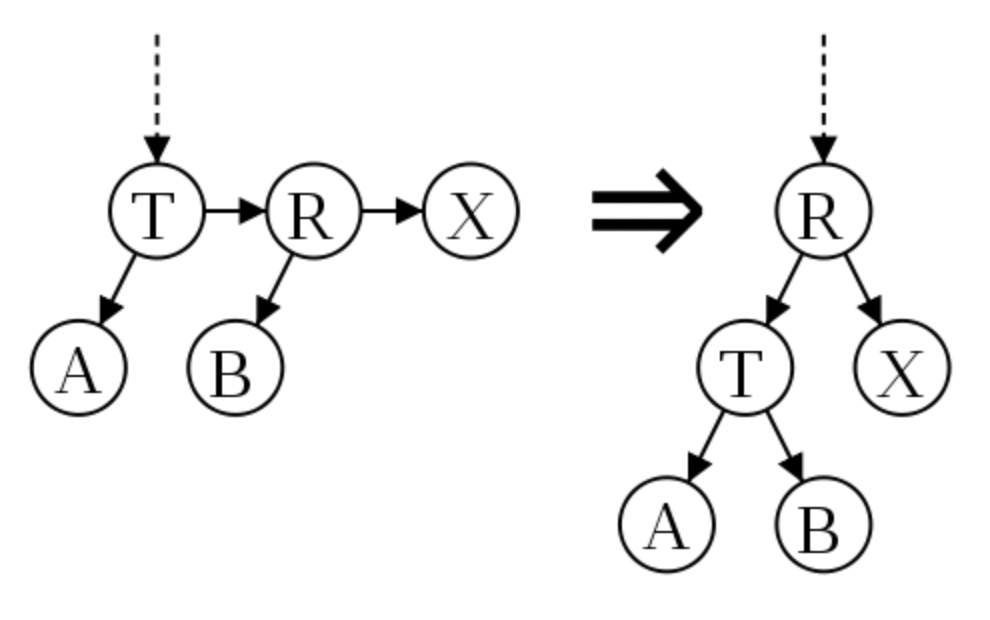
**void insert** //Функція insert() приймає параметр Rational і вставляє число в дерево.

**bool search** //Функція search() приймає параметр Rational і шукає число в дереві, повертаючи true, якщо число знайдено, і false в іншому випадку.

**void inorder** //Функція inorder() використовується для тестування та друкує дерево в порядку.

**rotateRight** //Операція «повернути вправо» використовується як частина операції нахилу та виконується, коли лівий дочірній вузол має той самий рівень, що й батьківський вузол. Ця операція повертає піддерево, що має корінь у заданому вузлі, праворуч, фактично роблячи лівого дочірнього елемента новим батьком піддерева. 

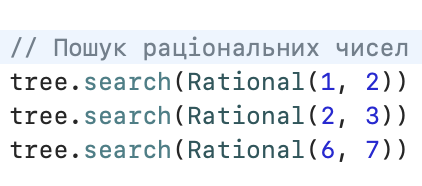
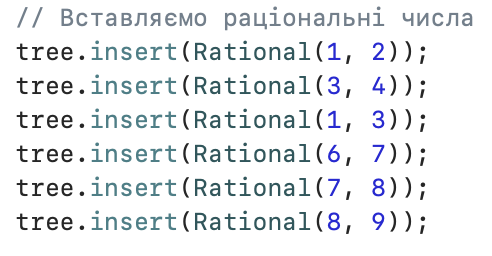
**rotateLeft** // функція, яка виконує поворот ліворуч заданого вузла в AA-дереві. Функція приймає вказівник на вузол, який потрібно обертати, і повертає вказівник на новий корінь піддерева після обертання.Обертання вліво використовується в AA-дереві, щоб підтримувати властивості нахилу та рівня дерева. При обертанні вліво вузол обертається разом зі своїм правим дочірнім елементом, так що правий дочірній елемент стає новим коренем піддерева, а вихідний вузол стає його лівим дочірнім елементом. Лівий дочірній елемент нового кореня стає правим дочірнім елементом початкового вузла.



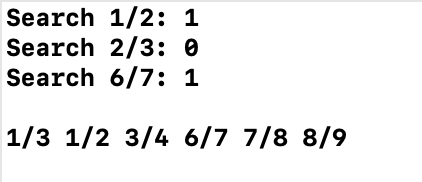
# **Інтерфейс користувача**

Вхідні дані задаються у програмі, а результат виводиться у консоль.

# **Приклад**



Вивід програми:



**1)** На основі отриманих даних можна зробити висновок, що дана реалізація AATree здатна правильно вставляти та шукати раціональні числа. Результати показують, що під час пошуку раціональних чисел 1/2 і 6/7 алгоритм правильно повертає 1, вказуючи, що ці числа присутні в дереві. Однак під час пошуку раціонального числа 2/3 алгоритм правильно повертає 0, що вказує на відсутність числа в дереві.

Крім того, вихідні дані функції inorder показують, що елементи друкуються в порядку зростання, що вказує на те, що дерево збалансовано та зберігає порядок елементів.

Таким чином, на основі отриманих даних ми можемо зробити висновок, що реалізація AATree є правильною та ефективною для вставки, пошуку та підтримки порядку раціональних чисел у дереві.

**Висновок**

Підсумовуючи, реалізація структури даних AA-дерева для раціональних чисел мовою C++ забезпечує ефективний спосіб зберігання та керування впорядкованими даними. Реалізація AA-дерева базується на концепції призначення рівня кожному вузлу та балансування дерева шляхом забезпечення того, що рівень лівого дочірнього вузла менший або дорівнює рівню вузла, а рівень правого вузла дитини менше рівня вузла.

Реалізація надає такі базові функції, як вставка раціонального числа в дерево, пошук раціонального числа в дереві та виконання обходу дерева. Ці функції працюють правильно та ефективно завдяки самобалансуванню AA- дерева, яке гарантує, що дерево завжди збалансоване.

Загалом, реалізація AA-дерева є корисною та ефективною структурою даних для керування впорядкованими даними, і її можна використовувати в різноманітних програмах, де потрібні впорядковані дані.

# **Література**

1. <https://neerc.ifmo/wiki/index.php?title=AA-дерево>
2. <https://www.geeksforgeeks.org/aa-trees-set-1-introduction/>
3. Лекція з предмету «Алгоритми та складність »
4. https://studfile.net/preview/3571362/