Сравнение сбалансированных деревьев: AA-tree и Splay-tree

Студент группы Б9121-09.03.03 пикд

Панкратова Екатерина Денисовна

Руководитель практики

Доцент ИМКТ А.С. Кленин

Введение

Цель: описать реализацию данных алгоритмов и с помощью тестов сравнить их.

Задачи:

- Изучить теоретический материал
- Описать данные алгоритмы
- Реализовать данные алгоритмы
- Реализовать тесты к алгоритмам
- Описать результаты тестирования

История создания

AA-tree – это модификация красно-черного дерева, предложенная Арне Андерссоном в 1993 году. Это сбалансированное дерево, используемое для эффективного хранения и извлечения упорядоченных данных.

Splay-tree – двоичное дерево поиска, созданное Робертом Тарьяном и Даниелем Слейтор в 1983 году. Поддерживается свойство сбалансированности. Позволяет находить те данные, которые использовались недавно.

Неформальная постановка задачи

- Алгоритмы деревьев должны принимать на ввод числа и выдавать построенное дерево (обход: корень-левый потомок-правый потомок). Они должны соответствовать описанию данных деревьев и выполнять функции поиска, вставки и удаления элемента, функция вывода дерева. А также специфических функций для каждого дерева.
- Основной набор тестов должен определять время выполнения того или иного набора команд и проверять, какой алгоритм справляется лучше. Также следует написать тесты, проверяющие корректность работы алгоритма, его производительность.

Требования к окружению

- Требования для данной работы минимальные, подойдет любой пк или ноутбук. Любая операционная система и компилятор, установленный заранее или онлайн, поддерживающий C++.
- Тестирование проводилось на ноутбуке с операционной системой Windows 10. Использовались онлайн компилятор Replit и установленный компилятор Visual Studio Code с расширением C/C++ v1.13.9.

Архитектура системы

Система состоит изначально из 4 файлов:

- Main.cpp
- AA-tree.h
- Splay-tree.h
- Tests.txt

В результате работы файла Main.cpp создается еще один файл с результатами тестирования – Results.txt.

Функциональные требования

Система должна:

- Хранить информацию в виде чисел
- Иметь возможность удаления, поиска, вставки элемента
- Отвечать требованиям конкретного алгоритма
- Выполнять тестирование алгоритмов
- Выдавать отчет о тестировании

Средства реализации

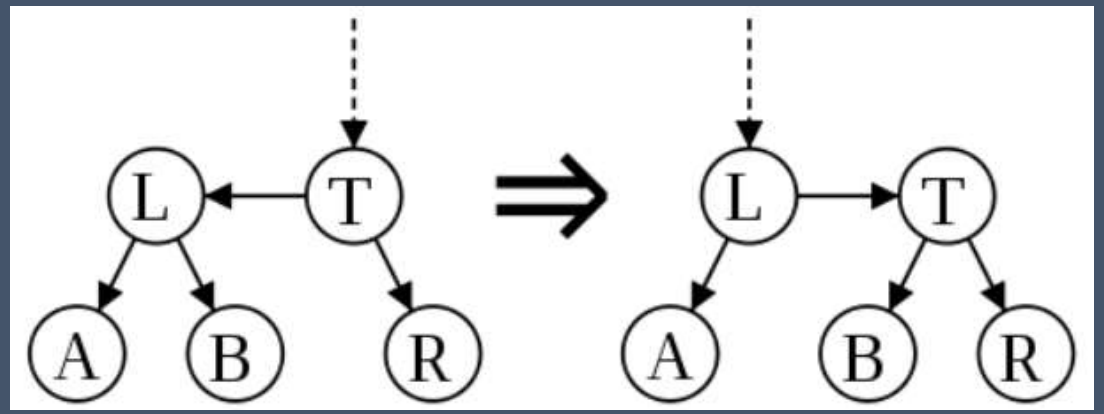
- В качестве языка программирования был выбран C++, так как он быстрее Python, подходит для целей работы и при этом более удобный и современный, чем C.
- В качестве среды разработки были выбраны Replit, так как писать код в браузере удобно, и Visual Studio Code, как один из самых популярных средств, так как в браузере имеется задержка при тестировании, влияющая на результаты тестов.

AA-tree. Свойства

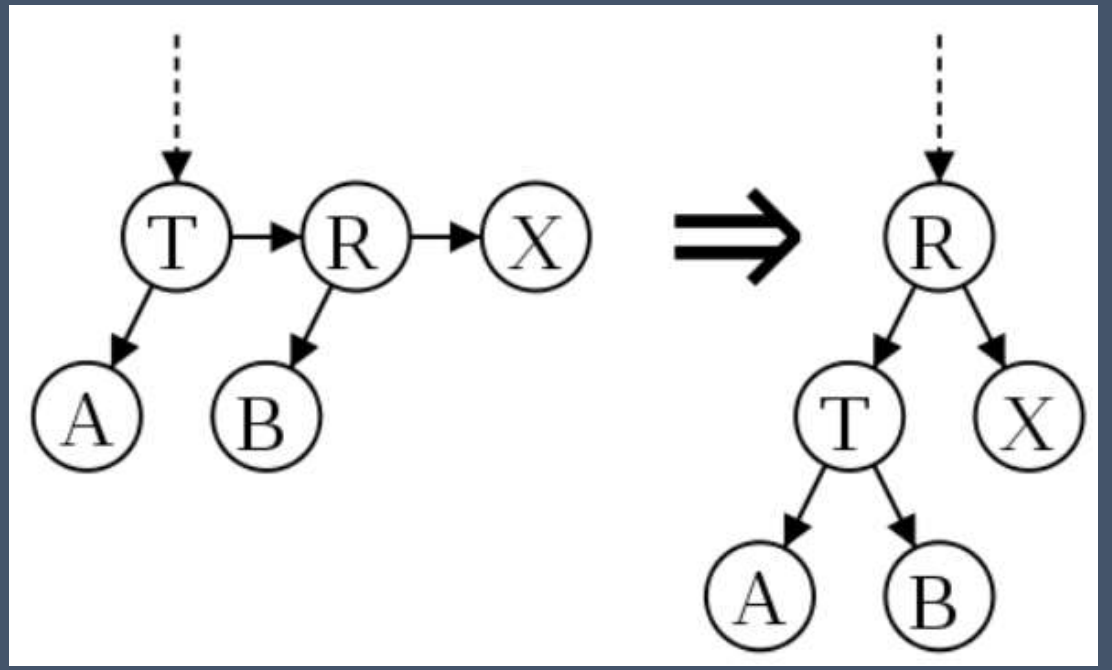
- Уровень каждого листа равен 1.
- Уровень каждого левого ребенка ровно на один меньше, чем у его родителя.
- Уровень каждого правого ребенка равен или на один меньше, чем у его родителя.
- Уровень каждого правого внука строго меньше, чем у его прародителя.
- Каждая вершина с уровнем больше 1 имеет двоих детей.

AA-tree. Балансировка

Skew()



Split()



AA-tree. Операции

- Вставка нового элемента происходит как в обычном дереве поиска, только на пути вверх необходимо делать ребалансировку, используя skew() и split().
- Удаление происходит также, как и в обычных деревьях, но чтобы сохранять баланс дерева необходимо делать skew(), split() и decreaseLevel() для каждой вершины.

AA-tree. Эффективность

- Все операции происходят за O (log n), потому что в сбалансированном двоичном дереве поиска почти все операции реализуются за O(n).
- Скорость работы AA-дерева эквивалентна скорости работы красно-черного дерева, но так как в реализации вместо цвета обычно хранят «уровень» вершины, дополнительные расходы по памяти достигают байта.

AA-tree. Преимущества

- Дерево сохраняет балансировку при следовании правилу «одна правая связь на одном уровне», что делает это дерево одним из самых быстрых, но в то же время простых в реализации.
- Для балансировки АА-дерева нужно всего две операции.

Splay-tree. Алгоритм

- Это дерево принадлежит классу «саморегулирующихся деревьев», которые поддерживают необходимый баланс ветвления дерева.
- «Расширяющие операции» (splay operation), частью которых являются вращения, выполняются при каждом обращении к дереву.

Splay-tree. Операция Splay

- Основная операция дерева. Заключается в перемещении вершины в корень при помощи последовательного выполнения трёх операций: Zig, Zig-Zig и Zig-Zag.

Splay-tree. Другие операции

- Search (поиск элемента) Поиск выполняется как в обычном двоичном дереве поиска. При нахождении элемента запускаем Splay() для него.
- Insert (добавление элемента) Запускаем Split() от добавляемого элемента и подвешиваем получившиеся деревья за элемент к добавлению.
- Delete (удаление элемента) Находим элемент в дереве, делаем Splay() для него, делаем текущим деревом Merge() его детей.

Splay-tree. Эффективность

Алгоритм	Среднее	Худший случай
Пробел	O (n)	O (n)
Поиск	амортизированный O (log n)	амортизированный O (log n)
Вставить	амортизированный O (log n)	амортизированный O (log n)
Удалить	амортизированный O (log n)	амортизированный O (log n)

Splay-tree. Преимущества и недостатки

- Хорошая производительность, так как оно самооптимизируется и часто используемые узлы будут приближаться к корню.
- Производительность в среднем такая же эффективная, как и у других деревьев.
- В Splay-деревьях не требуется хранить какие-либо дополнительные данные.
- Высота расширяемых деревьев может быть линейной, а значит фактическая стоимость одной операции может быть высокой.
- Когда шаблон доступа является случайным, появляются дополнительные накладные расходы на расширение по сравнению с менее динамичными альтернативами.

Реализация и тестирование

Заключение