АВЛ-деревья, выполнение операций над ними.

М. И. Сергеев, А. Г. Янишевская

Омский государственный технический университет

Аннотация: Рассматривается проблема невозможности дачи оценки времени выполнения операции поиска данных в информационной системе при использовании классических бинарных деревьев поиска. Предлагается способ решения данной проблемы путем использования АВЛ-деревьев. Обосновываются преимущества и недостатки использования АВЛ-деревьев. Также приводятся алгоритмы выполнения операций над АВЛ-деревьями для работы с данными.
Ключевые слова: АВЛ-дерево, бинарное дерево поиска, балансировка дерева, поиск данных в двоичном дереве по ключу.

Введение

Большинство современных информационных систем содержат в себе базы данных объем которых может превышать несколько гигабайт и к поиску данных в таких базах предъявляются особые требования, такие как максимальная скорость, прогнозируемость времени поиска и точность нахождения информации [1]. Простые алгоритмы перебора не способны обеспечить максимальную скорость и предоставить оценку времени выполнения операции поиска ввиду чего повсеместно заменяются на алгоритмы поиска с использованием двоичных деревьев. Доказательством этого служит сравнение скорости поиска в современных СУБД. Именно СУБД, использующие индексацию и двоичные деревья поиска показывают наибольшую производительность при поиске данных по таблице, содержащей большой объем данных [2].

Двоичные деревья поиска в стандартной реализации позволяют увеличить скорость поиска данных в информационных системах и предоставляют точные результаты, однако не способны предоставить оценку среднего времени выполнения операции поиска по базе данных фиксированного объема. В данной статье рассмотрен модифицированный вариант двоичных деревьев поиска – АВЛ-деревья, который позволяет давать

точную оценку скорости выполнения операции поиска данных, а так же уменьшает время нахождения искомой информации. Так же будут рассмотрены операции вставки и удаление узлов и поддеревьев из АВЛ-дерева и приведены алгоритмы выполнения данных операций.

Определение АВЛ-дерева

Для того, чтобы дать понятие АВЛ-дерева нам необходимо дать определение двоичного дерева поиска. Двоичное дерево поиска – иерархическая структура данных в которой каждый узел имеет не более чем два потомка, причем значение ключа любого узла левого поддерева произвольного узла X меньше, чем значение самого узла X, а значение любого узла правого поддерева узла X больше либо равно значению ключа узла X [3].

Важной характеристикой двоичного дерева поиска, непосредственно влияющей на скорость поиска данных является коэффициент сбалансированности. Коэффициентом сбалансированности называют некоторую константу k, на которую могут отличаться высоты левого и правого поддерева любого произвольного узла X.

Таким образом АВЛ-дерево – это двоичное дерево поиска, для которого определен коэффициент сбалансированности k = 1 [4].

Свойства АВЛ-дерева

Из определения АВЛ-дерева, следует, что разница высот левого и правого поддерева любого произвольного узла X лежит в диапозоне {-1, 0, 1}. Исходя из этого свойства было доказанно, что высота h АВЛ-дерева, содержащего n узлов может быть рассчитана по формуле (1) [5].

$$h \leq [1.45 * log_2(n+2)] \quad (1)$$

Таким образом можно сказать, что АВЛ-деревья являются наиболее эффективными в обработке ввиду того, что максимальное количество шагов,

для обнаружения искомого узла равно высоте дерева, которая является минимальной среди всех существующих двоичных деревьев поиска и максимально приближена к высоте идеально сбалансированного дерева. Время T нахождения произвольного узла X АВЛ-дереве, содержащем n элементов может быть рассчитана по формуле (2) [6].

$$T = O(log_2(n)) \qquad (2)$$

Время выполнения операций вставки и удаления узлов напрямую зависит от скорости поиска узла по дереву и следовательно является является минимальным по сравнению с другими видами двоичных деревьев поиска, однако в следствии выполнения данных операций коэффициент может произойти разбалансировка дерева, т.е. для некоторого узла X высота левого и правого узла станет отличаться на 2, что приведет к потере свойств АВЛ-дерева. Для того, чтобы не допустить этого над АВЛ-деревом определена операция балансировки. Время выполнения операции балансировки фиксировано и она осуществляется путем поворота узлов и изменению связей в дереве. Выделяют 4 типа поворотов (вращений) [7]:

1. Малое правое вращение;
2. Большое правое вращение;
3. Малое левое вращение;
4. Большое левое вращение.

Оба вида большого вращения представляют собой комбинации малых поворотов.

Балансировка АВЛ-дерева

Всего существует 2 типа разбалансированности дерева. Для исправления первого случая используется тип порота 1 и 3, а для второго 2 и 4 [8-10].

Рассмотрим первый случай. На рисунке 1 изображено сбалансированное поддерево, где x и y – узлы, а A, B, C – поддеревья. Добавления к поддереву A узла v приведет к нарушению баланса поддерева. Выполним балансировку при помощи правого малого поворота узла y (рисунок 2). Операция выполнения левого малого поворота осуществляется семерично малому правому.

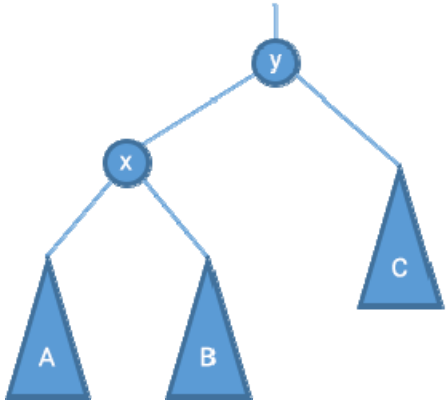

Рис 1. – Сбалансированное поддерево

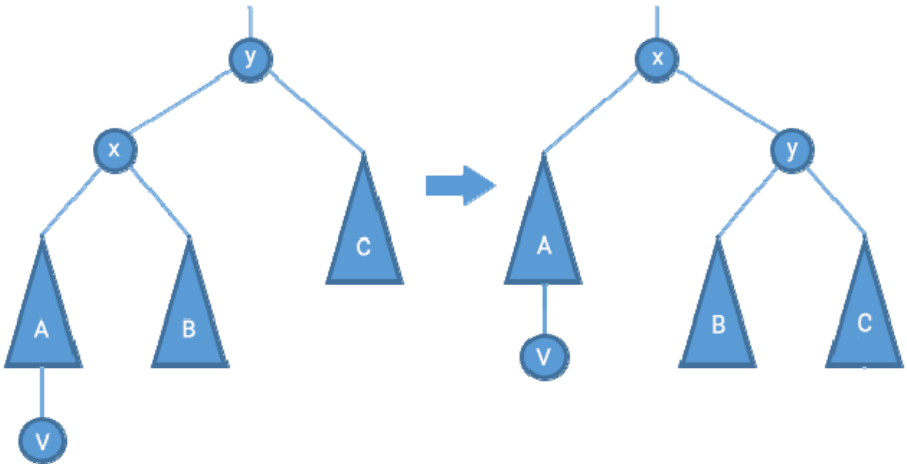

Рис 2. – Правый малый поворот узла y

Второй случай нарушения баланса дерева исправляется путем выполнения большого правого или большого левого поворота. Рассмотрим дерево изображенное на рисунке 3.

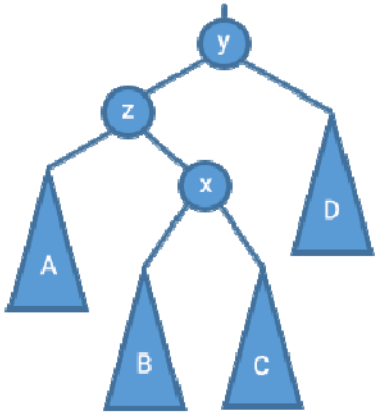

Рис. 3 – Сбалансированное дерево. Случай 2

При добавлении узлов в поддерево A или D баланс дерева не будет нарушен, однако при добавлении узла с поддерево B или C необходимо будет произвести балансировку. Для этого выполним большой правый поворот (рисунок 4). Большой левый поворот выполняется симметрично правому.

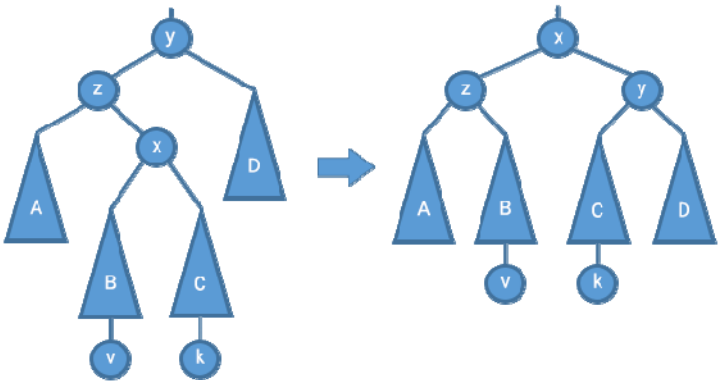

Рис. 4 – Большой правый поворот.

Вывод

Рассмотрев АВЛ-деревья и изучив их свойства мы можем сделать вывод, что их использование вместо простых двоичных деревьев поиска приведет к уменьшению времени операции поиска, вставки и удаления данных при одинаковом количестве элементов в дереве, однако это приведет к усложнению алгоритма взаимодействия с деревом ввиду того, что после выполнения каждой операции вставки и удаления придется выполнять

операцию проверки дерева на сбалансированность, и при обнаружении разбалансировки выполнять операции поворота узлов дерева.

Несмотря на это мы все равно получаем преимущества в виде повышения скорости выполнения операций, возможности более точно предсказывать время выполнения операций и снижения нагрузки на оборудование благодаря минимизации высоты дерева.

Литература

1. Радчук В.А. Закономерности развития рынка информационных услуг на современном этапе (обзор) // Инженерный вестник Дона, 2011, №3. URL: ivdon.ru/ru/magazine/archive/n3y2011/494.

2. Шаякбаров Н.Ф., Зорин Д.С. Анализ производительности систем управления базами данных при работе с большим объемом информации. // Инженерный вестник Дона, 2015, №2 ч.2. URL: ivdon.ru/ru/magazine/archive/n2p2y2015/2974.

3. Вирт Н., Алгоритмы и структуры данных. СПб.: Невский Диалект, 2008. 352 с. ISBN 978-5-7940-0065-8.

4. Адельсон-Вельский Г. М., Ландис Е. М. Один алгоритм организации информации // Доклады АН СССР, 1962. Т. 146, № 2. с. 263—266.

5. Lewis H.R., Denenberg L. Data Structures and Their Algorithms. Addison-Wesley. 1991. 509p.

6. Paul E. Dictionary of Algorithms and Data Structures. NIST // URL: xlinux.nist.gov/dads.

7. Кнут Д., Искусство программирования, том 3. Сортировка и поиск 2-е изд. М.:Издательский дом «Вильямс», 2007. 824 с. ISBN 0-201-89685-0.

8. Кормен, Т., Лейзерсон, Ч., Ривест, Р., Штайн, К. Алгоритмы: построение и анализ / Под ред. И. В. Красикова. 2-е изд. М.: Издательский дом «Вильямс», 2005. 1296 с. ISBN 5-8459-0857-4.

9. Tarjan R. E. Data structures and networks algorithms. SIAM. 1983. 125 p.

10. Bfaff P. Performance analysis of BSTs in system software SIGMETRICS Perform. Eval. vol. 32, 2004. pp. 410–411.

References

1. Radchuk V.A. Inženernyj vestnik Dona (Rus), 2011, №3. URL: ivdon.ru/ru/magazine/archive/n3y2011/494.

2. Shayakbarov N.F., Zorin D.S. Inženernyj vestnik Dona (Rus), 2015, №2. URL: ivdon.ru/ru/magazine/archive/n2p2y2015/2974.

3. Virt N., Algoritmy i struktury dannykh. [Algorithms and data structures] SPb.: Nevskiy Dialekt, 2008. 352 p. ISBN 978-5-7940-0065-8

4. Adel'son-Vel'skiy G. M., Landis E. M. Doklady AN SSSR, 1962. V. 146, № 2. pp. 263—266.

5. Lewis H.R., Denenberg L. Data structures and their algorithms. addison-Wesley. 1991. 509 p.

6. Paul E. Dictionary of algorithms and data structures. NIST. URL: xlinux.nist.gov/dads.

7. Knut D., Iskusstvo programmirovaniya, tom 3. Sortirovka i poisk 2e izd. [The art of programming] M.: Izdatel'skiy dom «Vil'yams», 2007. 824 p. ISBN 0-201-89685-0.

8. Kormen, T., Leyzerson, Ch., Rivest, R., Shtayn, K. Algoritmy: postroenie i analiz [Introduction to algorithms]. Pod red. I. V. Krasikova. 2e izd. M.: Izdatel'skiy dom «Vil'yams», 2005. 1296 p. ISBN 5-8459-0857-4.

9. Tarjan R. E. Data structures and networks algorithms. SIAM. 1983. 125 p.

10. Bfaff P. Performance analysis of BSTs in system software. SIGMETRICS Perform. Eval. vol. 32, 2004. pp. 410–411.