УДК 004.074
DOI: 10.17586/0021-3454-2024-67-4-352-358

БАЛАНСИРУЕМАЯ СТРУКТУРА ДАННЫХ С ПРИОРИТЕТАМИ ЭЛЕМЕНТОВ В ЗАДАЧЕ МОДЕЛИРОВАНИЯ ДИСКРЕТНЫХ ИСТОЧНИКОВ ИНФОРМАЦИИ

А. В. Аксенов

Санкт-Петербургский государственный университет аэрокосмического приборостроения, Санкт-Петербург, Россия,
aksenov@guap.ru

Аннотация. Рассматриваются особенности разработки балансируемой структуры данных, ориентированной на ускоренный доступ к элементам, имеющим высокий приоритет. Подобные структуры могут использоваться в задачах моделирования дискретных источников информации. Предложено самобалансирующееся двоичное дерево поиска, оптимизированное для эффективного хранения и поиска данных на основе приоритетов, коррелирующих с вероятностью порождения символов. Решение позволяет преодолеть ограничения существующих структур данных с учетом требований к памяти и производительности в контексте специфических задач обработки информации.

Ключевые слова: *структура данных, ускоренный доступ, двоичное дерево поиска, самобалансирующаяся структура, хранение данных, эффективный поиск данных*

Ссылка для цитирования: *Аксенов А. В.* Балансируемая структура данных с приоритетами элементов в задаче моделирования дискретных источников информации // Изв. вузов. Приборостроение. 2024. Т. 67, № 4. С. 352—358. DOI: 10.17586/0021-3454-2024-67-4-352-358.

BALANCEABLE DATA STRUCTURE WITH ELEMENT PRIORITIES IN THE PROBLEM OF DISCRETE INFORMATION SOURCE MODELING

A. V. Aksenov

St. Petersburg State University of Aerospace Instrumentation, St. Petersburg, Russia
aksenov@guap.ru

Abstract. The features of developing a balanceable data structure focused on accelerated access to elements with high priority are considered. Similar structures can be used in problems of modeling discrete information sources. A self-balancing binary search tree is proposed, optimized for efficient storage and retrieval of data based on priorities that correlate with the probability of symbol generation. The solution overcomes the limitations of existing data structures, taking into account memory and performance requirements in the context of specific information processing tasks.

Keywords: data structure, accelerated access, binary search tree, self-balancing structure, efficient data storage, efficient data retrieval

For citation: Aksenov A. V. Balanceable data structure with element priorities in the problem of discrete information source modeling. *Journal of Instrument Engineering.* 2024. Vol. 67, N 4. P. 352—358 (in Russian). DOI: 10.17586/0021-3454-2024-67-4-352-358.

Введение. В ряде задач, связанных с программной обработкой дискретных данных, возникает потребность в структуре, обеспечивающей хранение элементов некоторого заданного множества вкупе с некоторой информацией о них. При этом зачастую важную роль играет эффективность выполнения с этой структурой таких операций, как добавление и поиск.

Очевидно: если объем данных достаточно велик, а мощность множества возможных значений невелика, наиболее частой операцией, совершаемой со структурой данных, будет поиск некоторого элемента по его ключу. При этом, если предположить, что востребован-

ность элементов различается, то необходимо ускорить доступ именно к наиболее востребованным элементам. В частности, необходимость решения подобной задачи возникает при контекстном моделировании дискретных источников информации. При этом состояние источника определяется контекстом порожденных им символов выходного алфавита. Для каждого состояния задается вероятностное распределение возможных выходных символов, порождаемых при нахождении в данном состоянии. При этом задачей контекстного моделирования (поскольку набор состояний моделируемого источника заранее неизвестен) является получение априорной оценки вероятностного интервала для символа, который был порожден. Для символов с высокой вероятностью порождения целесообразно ускорить нахождение оценки интервала.

Специфика задачи накладывает ряд ограничений на физическую реализацию структуры данных. Так, объем памяти, занимаемый моделью, чрезвычайно велик. Он зависит от порядка моделирования, т.е. длины контекста, учитываемого при задании состояний модели, и ограничен сверху величиной

$$N_{\text{sym}}^{\text{ord}}\left(V_{\text{stat}}+V_{\text{data}}\right), \tag{1}$$

где N_{sym} — число символов входного алфавита; ord — порядок моделирования; V_{stat} — размер статистической информации в одиночном элементе; V_{data} — размер полезной информации в одиночном элементе.

Порядок моделирования может быть и неограниченным, тогда число состояний источника (а соответственно и размер модели) бесконечно.

Поскольку набор состояний источника заранее не известен, невозможно предсказать и размер модели, которая будет использована, даже когда осуществляется моделирование контекста конечной длины. Кроме того, многие состояния источника будут посещены столь малое число раз, что накопленная статистика будет крайне невелика. Также вероятны случаи, когда число различных порождаемых символов для состояния чрезвычайно мало. Отсюда следует практическое требование, чтобы структура данных, реализуемая для каждого состояния, занимала возможно меньший объем памяти, желательно пропорциональный количеству статистической информации, который она хранит.

Из этих соображений целесообразно отказаться от реализации структуры данных на основе хэш-таблиц, хотя значения константного времени добавления и поиска элемента следует признать приемлемыми.

Второе ограничение связано с тем, что, как было отмечено выше, задачей контекстного моделирования является оценивание вероятностного интервала порожденного символа, а для ее нахождения необходимо вычислить функцию от приоритетов других элементов структуры.

В настоящей работе предлагается структура, представляющая собой самобалансирующееся двоичное дерево поиска, хранящее данные в узлах, и вводится понятие балансировки дерева по весу.

Обзор самобалансирующихся древовидных структур данных. Если не принимать определенных мер, то эффективность выполнения операций с двоичным деревом поиска может значительно снизиться с увеличением числа его узлов. Во избежание подобных явлений используется балансировка, а деревья называются самобалансирующимися. Это означает, что при выполнении операций, таких как поиск, добавление и удаление элемента, помимо собственно полезного действия производятся преобразования структуры дерева, сохраняющие свойство поиска (ключи узлов левого поддерева меньше ключа корня этого поддерева, ключи узлов правого поддерева больше ключа корня этого поддерева), но приводящие дерево к сбалансированному виду. Для различных деревьев данные операции различны, как различны и критерии сбалансированности.

Чаще всего используется балансировка по высоте, под высотой при этом понимается расстояние от корня до листа. Механизмы, обеспечивающие сбалансированность по высоте, реализованы в АВЛ- и красно-черных деревьях.

АВЛ-деревья являются первыми широко известными самобалансирующимися древовидными структурами данных [1]. АВЛ-деревья поддерживают строгий баланс, основывающийся на высоте поддеревьев. Инвариант для узла АВЛ-дерева:

$$\left|H_{\text{left}} - H_{\text{right}}\right| \le 1, \tag{2}$$

где H_{left}, H_{right} — высота соответственно левого и правого поддеревьев узла.

В каждом узле дерева хранится дополнительная информация — его фактор баланса, который является целым числом в диапазоне [–1; 1] и представляет собой разность высоты левого и правого поддеревьев. В случае нарушения инварианта в каком-то узле выполняются вращения.

К достоинствам АВЛ-деревьев относится гарантированная $O(\log n)$ сложность операций в худшем случае, к недостаткам — довольно большое количество вращений, необходимость которых проистекает из жесткости условия сбалансированности [2].

В красно-черных деревьях узлы содержат бит, определяющий их „цвет“, который, в свою очередь, накладывает ряд ограничений на структуру дерева. При нарушении этих ограничений осуществляются вращения и „перекрашивания“ узлов [3].

К достоинствам красно-черных деревьев относится гарантированная $O(\log n)$ сложность операций в худшем случае, а также малое количество вращений при их выполнении [2].

Splay-деревья (в русскоязычной литературе также называемые расширяющимися или косыми) не являются балансируемыми по высоте, в отличие от красно-черных и АВЛ-деревьев [4].

Идея, лежащая в основе построения splay-деревьев, была взята из эвристики перехода на передний план (move-to-front) в самоорганизующихся списках: каждый раз узел дерева, к которому выполняется обращение, становится корнем. Для этого используется специальная операция splay.

Эта структура данных интересна тем, что разработана для приложений, в которых важен быстрый доступ к часто используемым элементам, поскольку они будут часто подвергаться операции splay и вследствие этого с высокой вероятностью располагаться ближе к корню.

К достоинствам splay-деревьев можно отнести отсутствие накладных расходов по памяти (в узлах дерева не хранится дополнительных полей) и хорошую производительность, когда доступ к некоторым узлам выполняется значительно чаще, чем к остальным, что является важным преимуществом в задаче контекстного моделирования. К недостаткам относится амортизированная $O(\log n)$ сложность операций (отдельные операции имеют сложность $\Omega(n)$), а также большое количество выполняемых вращений при каждом обращении к структуре данных [2].

В исследованиях splay-деревья показывают хорошую производительность при доступе к данным, имеющим последовательный или кластеризованный характер [5].

Scapegoat-деревья, в отличие от красно-черных и АВЛ-деревьев, опираются на весовой баланс поддеревьев, причем под весом поддерева понимается число узлов в нем [6].

Инвариант scapegoat-дерева имеет вид:

$$S_{\text{left}} \le \alpha S_{\text{total}}, \quad S_{\text{right}} \le \alpha S_{\text{total}}, \tag{3}$$

где α — фактор баланса в диапазоне [0,5; 1], позволяющий настроить строгость балансировки дерева: $\alpha = 0{,}5$ соответствует полному бинарному дереву, $\alpha = 1$ допускает вырождение в линейный список; S_{left} — число узлов поддерева с корнем в левом потомке текущего узла;

S_{right} — число узлов поддерева с корнем в правом потомке текущего узла; S_{total} — число узлов поддерева с корнем в текущем узле.

Интересно, что при нарушении инварианта выполняются не вращения, а полное перестроение дерева.

Достоинством scapegoat-деревьев является поддержка отложенных операций удаления узлов и балансировки, что снижает накладные расходы при работе с деревом.

К недостаткам относится амортизированная $O(\log n)$ сложность операций — отдельные операции имеют сложность $\Omega(n)$ [2].

Классические декартовы деревья не являются сбалансированными в строгом смысле слова, поскольку их структура зависит не от данных в элементах и не от порядка поступления этих данных на вход, а от случайно генерируемого и назначаемого каждому элементу приоритета, для которого дерево выполняет требование двоичной кучи (приоритет родителя должен быть больше приоритета потомков). Приоритет узла в течение жизни структуры данных не изменяется, вращения не используются, вместо них применяются операции разрезания и склеивания [7].

Преимуществом такой структуры данных можно назвать высокую вероятность $O(\log n)$ сложности операций с ней.

Представленные структуры данных широко используются, однако не вполне отвечают требованиям поставленной задачи. Декартовы, красно-черные, АВЛ- и scapegoat-деревья со случайными весами предназначены для поддержания близкой к оптимальной структуры, выполняя балансировку по высоте, но при этом игнорируя востребованность элементов.

Splay-деревья спроектированы так, чтобы учитывать востребованность элементов и ускорять доступ к наиболее востребованным, но обладают только амортизированной логарифмической сложностью, что накладывает ограничения на их применимость в ряде задач.

Описание предлагаемой структуры данных. Как уже упоминалось, предлагаемая структура данных предназначена для накопления статистики порожденных символов дискретного источника информации с целью создания его модели. Структура данных оперирует приоритетами узлов, связанных с вероятностью порождения соответствующих символов, и стремится минимизировать время доступа к узлам с высоким приоритетом.

В дальнейшем будем полагать, что приоритет символа прямо пропорционален оценке вероятности его порождения. Поскольку оценка априорна, следует говорить не о приоритете символа, а об оценке приоритета.

Для выполнения поставленного условия каждый элемент i рассматриваемой структуры должен хранить полезную информацию, а также быть аннотированным некоторым значением X_i, по которому можно оценить его приоритет: $P_i = f(X_i)$.

Аннотирование элемента i количеством его появлений в этом контексте (т.е. когда производился поиск элемента i в структуре) обеспечивает простоту вычисления и модификации данного элемента.

В принципе, на область значений оценки приоритета не накладывается каких-либо ограничений, поэтому для простоты можно положить, что $P_i = X_i$.

Особым случаем является ситуация, когда $P_i = 0$, т.е. символ ни разу не был встречен в текущем состоянии до этого момента. Данная проблема в контекстном моделировании известна как „проблема нулевой вероятности“ [8]. В описываемой структуре в такой ситуации будут выполняться добавление элемента и инициализация оценки его приоритета значением 1.

Для простоты назовем оценку приоритета узла весом. Это позволит ввести понятие меры веса поддерева и балансировки по весу. Исходя из принятых ранее допущений вес узла, число обращений к узлу и оценка приоритета узла — суть одно и то же число, и именно им аннотирован узел.

Распространенной практикой построения структур данных является кэширование мер поддеревьев, в частности, при реализации структуры данных *rope* (в русскоязычной литературе — „строп“, „веревка“ или „корд“), применяющейся для хранения и обработки операций со строками [9]. Показано, что кэшировать можно результат вычисления любой операции, если она и тип данных, к которому она применяется, образуют моноид — полугруппу с нейтральным элементом [10]. В настоящей работе моноид составляет тройка ⟨+, вес элемента, нулевой вес⟩.

Балансировка предлагаемой структуры данных по весу производится с использованием кэширования суммарного веса поддерева в его корне. Если точнее, то кэшируется суммарный вес левого поддерева, т.е. поддерева с корнем в виде левого потомка текущего узла, суммарный вес правого поддерева вычисляется как

$$P_{\text{right}} = P_{\text{total}} - P_{\text{left}} - \text{P}_{\text{root}}, \tag{4}$$

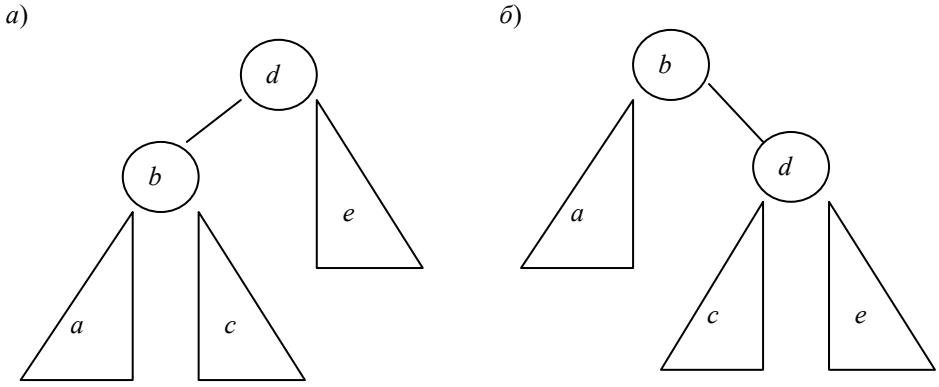
где P_{total} — вес поддерева с корнем в текущем узле; P_{left} — вес поддерева с корнем в левом потомке текущего узла; P_{right} — вес поддерева с корнем в правом потомке текущего узла; P_{root} — вес текущего узла.

Общий вес поддерева P_{total} запоминается в процессе спуска по дереву уровнем выше.

Также в процессе спуска инкрементируются хранящиеся в узлах меры P_{left}, если поиск продолжается в левом поддереве.

Для балансировки дерева предлагается использовать вращения, схожие по механизму с теми, что используются в структурах данных, балансируемых по высоте, однако использующие другие инварианты.

На рисунке приводится схема правого вращения (*а* — структура дерева до вращения; *б* — после вращения). При этом кругами обозначены узлы дерева, а треугольниками — левые или правые поддеревья (соответственно направлению острого угла треугольника). Каждый элемент дерева подписан буквой латинского алфавита, которая отражает лексикографический порядок, установленный в двоичном дереве поиска.



Инвариант для правого вращения имеет вид:

$$\left|P_a + P_b + P_c - P_e\right| \le \left|P_c + P_d + P_e - P_a\right|, \tag{5}$$

где P_i — вес узла или поддерева i. Смысл этого инварианта заключается в том, что разность суммарных весов левого и правого поддеревьев до потенциального правого вращения должна быть меньше, чем после преобразования. Если инвариант нарушается, это провоцирует правое вращение, которое восстанавливает инвариант. Инвариант левого вращения и его схема зеркально симметричны инварианту правого вращения (5) и его схеме.

При вращении переназначаются кэшированные в узлах меры суммарных весов.

Идея балансировки заключается в том, чтобы максимально равномерно распределить между правым и левым поддеревом каждого узла операции поиска элементов.

Интересным свойством предлагаемой структуры данных является то, что кэшированные веса являются не только средством для балансировки дерева по востребованности узлов, но и

данными для расчета вероятностного интервала символа, используемого для кодирования дискретного источника информации.

Предлагаемая структура данных имеет ряд особенностей, которые необходимо учитывать в различных сценариях ее поведения:

— в задачах общего типа она уступает остальным структурам данных в объеме накладных расходов по памяти, поскольку каждый узел хранит дополнительно два поля: собственный вес и вес своего левого поддерева;

— представляют существенную трудность и пока не реализованы операции склейки деревьев, разделения дерева на два, а также удаления узлов и множественные операции с узлами. Тем не менее, в задаче моделирования дискретных источников информации эти операции не применяются, значит, их реализация является второстепенной задачей;

— не учитывается возможная локальность поведения источника информации (ситуация, когда вероятностное распределение порождаемых символов лишь несущественно меняется на коротком отрезке времени, после чего происходит его значительное изменение). В отличие от splay-дерева, предлагаемая структура данных с трудом приспосабливается к изменениям характеристик источника. Впрочем, в контекстном моделировании применяются механизмы, нивелирующие эту проблему.

Заключение. Традиционно применяемые балансируемые древовидные структуры данных не вполне подходят под специфику задачи моделирования дискретного источника информации.

В предложенной структуре учитывается, что различные ее элементы неравнозначны по востребованности и содержат механизмы минимизации времени доступа к наиболее востребованным элементам.

СПИСОК ЛИТЕРАТУРЫ

1. *Адельсон-Вельский Г. М., Ландис Е. М.* Один алгоритм организации информации // Докл. АН СССР. 1962. Т. 146, № 2. С. 263—266.
2. *Кормен Т. Х., Лейзерсон Ч. Е., Ривест Р. Л., Штайн К.* Алгоритмы: построение и анализ. М.: Вильямс, 2013. 1328 с.
3. *Guibas L. J., Sedgewick R.* A dichromatic framework for balanced trees // Proc. of the 19th Annual Symp. on Foundations of Computer Science (SFCS 1978). Ann Arbor, MI, USA. 1978. P. 8—21.
4. *Sleator D., Tarjan R. E.* Self-Adjusting Binary Search Trees // J. of the Association for Computing Machinery. 1985. Vol. 32, N 3. P. 652—686.
5. *Pfaff B.* Performance Analysis of BSTs in System Software // ACM SIGMETRICS Performance Evaluation Review. 2004. Vol. 32, is. 1. P. 410—411.
6. *Galperin I., Rivest R.* Scapegoat trees // Proc. of the Fourth Annual ACM-SIAM Symp. on Discrete algorithms (SODA '93). Society for Industrial and Applied Mathematics, USA, 1993. P. 165—174.
7. *Seidel R., Aragon C. R.* Randomized search trees // Algorithmica. 1996. Vol. 16. P. 464—497.
8. *Cleary J., Teahan W.* Experiments on the zero frequency problem // Proc. of the DCC '95 Data Compression Conf. 1995. P. 480.
9. *Boehm H., Atkinson R., Plass M.* Ropes: an Alternative to Strings // Software—Practice and Experience. 1995. Vol. 25, N 12. P. 1315—1330.
10. *Hinze R., Paterson R.* Finger Trees: A Simple General-purpose Data Structure // J. of Functional Programming. 2006. Vol. 16, N 2. P. 197—217.

Сведения об авторе

Алексей Владимирович Аксенов — Санкт-Петербургский государственный университет аэрокосмического приборостроения, кафедра вычислительных систем и сетей; старший преподаватель; E-mail: aksenov@guap.ru

Поступила в редакцию 04.12.23; одобрена после рецензирования 14.12.23; принята к публикации 08.02.24.

REFERENCES

1. Adel'son-Vel'skii G.M., Landis E.M. *Doklady Akademii Nauk SSSR*, 1962, no. 2(146), pp. 263–266. (in Russ.)
2. Cormen Th.H., Leiserson Ch.E., Rivest R.L., Stein C. *Introduction to Algorithms*, MIT Press, 2009.
3. Guibas L.J., Sedgewick R. *Proceedings of the 19th Annual Symposium on Foundations of Computer Science (SFCS 1978)*, Ann Arbor, MI, USA, 1978, pp. 8–21.
4. Sleator D., Tarjan R.E. *Journal of the Association for Computing Machinery*, 1985, no. 3(32), pp. 652–686.
5. Pfaff B. *ACM SIGMETRICS Performance Evaluation Review*, 2004, no. 1(32), pp. 410–411.
6. Galperin I., Rivest R. *Proc. of the Fourth Annual ACM-SIAM Symp. on Discrete algorithms (SODA '93)*, Society for Industrial and Applied Mathematics, USA, 1993, pp. 165–174.
7. Seidel R., Aragon C.R. *Algorithmica*, 1996, vol. 16, pp. 464–497.
8. Cleary J., Teahan W. *Proceedings of the DCC '95 Data Compression Conference*, 1995, p. 480.
9. Boehm H., Atkinson R., Plass M. *Software—Practice and Experience*, 1995, no. 12(25), pp. 1315–1330.
10. Hinze R., Paterson R. *Journal of Functional Programming*, 2006, no. 2(16), pp. 197–217.

Data on author

Aleksey V. Aksenov — St. Petersburg State University of Aerospace Instrumentation, Department of Computing Systems and Networks; Senior Lecturer; E-mail: aksenov@guap.ru

Received 04.12.23; approved after reviewing 14.12.23; accepted for publication 08.02.24.