ИЗВЕСТИЯ
ПЕНЗЕНСКОГО ГОСУДАРСТВЕННОГО
ПЕДАГОГИЧЕСКОГО УНИВЕРСИТЕТА
имени В.Г. БЕЛИНСКОГО
ФИЗИКО-МАТЕМАТИЧЕСКИЕ
И ТЕХНИЧЕСКИЕ НАУКИ
№30 2012

IZVESTIA
PENZENSKOGO GOSUDARSTVENNOGO
PEDAGOGICHESKOGO UNIVERSITETA
IMENI V.G. BELINSKOGO
PHYSICAL, MATHEMATICAL
AND TECHNICAL SCIENCES
№30 2012

УДК: 004.6

САМООРГАНИЗУЮЩИЕСЯ СТРУКТУРЫ ДАННЫХ

© А. М. ВОЛОДИН
Пензенский государственный университет,
кафедра прикладной математики и информатики
e-mail: alexgreen.penza@gmail.com

Володин А. М. — Самоорганизующиеся структуры данных // Известия ПГПУ им. В. Г. Белинского. 2012. № 30. С. 271–277. — Рассмотрены известные самоорганизующиеся структуры данных, построенные на основе линейных списков и бинарных деревьев поиска. Приведены оценки сложности онлайн-алгоритмов, используемых в этих структурах. Исследованы самосовершенствующиеся алгоритмы, которые улучшают свою производительность, приспосабливаясь к неизвестному распределению входных данных. Отмечены достоинства и недостатки алгоритмов. Описаны преобразования СД на основе синергетического подхода.

Ключевые слова: Самоорганизующаяся структура данных, онлайн-алгоритм, самосовершенствующийся алгоритм, оффлайн-алгоритм, оптимальный анализ, исключение, преобразование.

Volodin A. M. — Self-organizing data structures // Izv. Penz. gos. pedagog. univ. im.i V. G. Belinskogo. 2012. № 30. P. 271–277. — Well-known self-organizing data structures based on linear lists and binary search trees are considered. Online algorithms complexity used in these data structures is given. Self-improving algorithms that improve its performance by adapting to an unknown distribution of input data are investigated. The advantages and disadvantages of the algorithms are pointed out. The data structures transformations using synergetic approach are described.

Keywords: Self-organizing data structure, self-improving algorithm, online algorithm, offline algorithm, competitive analysis, exception, transfor-mation.

В процессе создания информационных систем (ИС) часто приходится сталкиваться с проблемой выбора оптимальной структуры данных (СД) из набора потенциально возможных. От выбора СД зависят сложность реализации и эффективность выполнения программ. Каждый вид СД имеет свои особенности, проявляющиеся в их узкой специализации — способности эффективной организации и обработке определенного количества данных с определенными отношениями. Это определяет пригодность любой СД для решения только определенного класса задач при определенных условиях, что часто не совпадает с реально решаемыми задачами, которые к тому же изменяются в процессе использования ИС.

Разработчики ИС обычно используют стандартные библиотеки компонентов. При этом сначала выбирается наиболее простая возможная СД, а в более сложных условиях используются специальные классы коллекций. Однако любой список шаблонов, даже самый исчерпывающий, не может охватить все ситуации, встречающиеся в процессе программирования. Возникают проблемы, решить которые нельзя заранее

известными способами. Преимущества, получаемые заменой класса реализации, часто ограничены, поэтому приходится делать более масштабные алгоритмические изменения.

Наиболее часто приходится решать задачи, связанные с улучшением производительности. Операции обработки данных должны эффективно выполняться вне зависимости от числа элементов, содержащихся в СД. Например, миллионный элемент данных должен добавляться в коллекцию так же быстро, как и первый. При этом в процессе использования ИС типы и интенсивности различных запросов могут существенно изменяться.

Операции обработки данных предполагают определенную последовательность просмотра элементов, поэтому для повышения эффективности использования СД необходимо минимизировать число обращений к данным. Количество сравнений при поиске будет минимальным, когда наиболее часто используемые элементы данных расположены в наиболее доступных участках памяти, например, в начале списка или на верхних уровнях дерева.

В ИС также решаются задачи, для которых входные данные заранее неизвестны. Это требует эффективной реализация запросов, последовательность которых неопределена. Для решения этой проблемы разрабатываются алгоритмы, работающие в реальном времени, или онлайн-алгоритмы (online algorithms) [1], и самоорганизующиеся СД (self-organizing data structures) [2], которые осуществляют самокоррекцию в процессе использования. Онлайн-алгоритмы основаны на переупорядочивании элементов в самоорганизующейся СД в процессе выполнения операций обработки данных (чаще всего поиска элементов).

Самоорганизующиеся СД имеют ряд преимуществ по сравнению со СД, в которых используются оффлайн-алгоритмы (offline algorithm) [1], основанные на известных представлениях о свойствах входных данных. Например, если запросы на поиск данных поступают согласно определенному закону распределения, элементы СД упорядочиваются в порядке уменьшения вероятностей доступа к ним. Несмотря на то, что самоорганизующиеся СД работают в условиях, когда распределение вероятностей неизвестно, их состояние меняется таким образом, чтобы динамически приспособиться к этому распределению. Тем самым, уменьшается время выполнения будущих операций.

Онлайн-алгоритмы требуют дополнительных затрат на перестройку (реорганизацию) СД, преимущественно во время процедуры доступа к данным. Однако сложность таких преобразований (как правило, перемещения элементов) асимптотически не увеличивает сложность выполняемой операции обработки данных. При равной оценке асимптотической сложности операций обработки данных самоорганизующиеся СД в некоторых случаях (нерегулярность запросов, запросы с определенными свойствами) могут быть более эффективными, чем СД с оффлайн-алгоритмами.

Самоорганизующиеся СД могут быть реализованы без использования дополнительной памяти. В таблице 1 приведены оценки онлайн-алгоритмов для последовательностей и бинарных деревьев [1, 2].

Представленные онлайн-алгоритмы используют различные стратегии переупорядочения элементов (перемещение в последовательностях и повороты в деревьях) для того, чтобы общая стоимость запросов была минимальной.

Для последовательностей эффективное выполнение операций доступа, добавления и удаления элементов получило название проблемы обновления списка [1, 2]. Остается открытым вопрос, является ли эта проблема NP-трудной задачей. Наиболее быстрый известный оффлайн алгоритм решает эту задачу за время $O(2^n \cdot n! \cdot m)$, где n — количество элементов в списке, m — количество запросов.

Неизвестны оффлайн-алгоритмы, выполняющие последовательность основных операций обработки данных в бинарных деревьях с минимальной стоимостью за полиномиальное время. Одной из проблем является NP-полная задача поиска минимально возможного количества поворотов, необходимых для преобразования бинарного дерева T_1 в дерево T_2 с n узлами.

Таблица 1

Структура данных	Онлайн алгоритм	Использование дополнительной памяти	Оценка алгоритма
Последовательности	Move-to-front	нет	$c = 2$
	Transpose	нет	не является c-оптимальным
	Frequency-count	да	не является c-оптимальным
	Timestamp	да	$c = 2$
	MRI	да	$c = 2$
	Bit	да	$c = 1{,}75$
Бинарные деревья поиска	Move to root by rotation	нет	$\Theta(n)$-оптимален; $\Omega(n/\log n)$ - статически-оптимален; $O(1)$ - распределенно-оптимален
	Single-rotation	нет	$\Theta(n)$-оптимален; $\Omega(n/\log n)$ - статически-оптимален; $\Omega(\sqrt{n}/\log n)$ - распределенно-оптимален;
	Splaying (splay tree algorithm)	нет	$O(\log n)$-оптимален, $O(1)$ - статически-оптимален
	Dynamic monotone tree algorithm	да	$\Omega(n/\log n)$ - распределенно-оптимален
	WPL tree algorithm	да	$O(1)$ - распределенно-оптимален
	D-tree algorithm	да	$O(1)$-распределенно-оптимален

Для оценки онлайн-алгоритмов используется конкурентный (c-оптимальный) анализ (competitive analysis) [1, 2], сравнивающий сложность онлайн-алгоритма со сложностью оптимального оффлайн-алгоритма.

c-оптимальными (c-competitive) алгоритмами называются алгоритмы, которые для любой последовательности запросов σ дают оценку:

$$C_A(\sigma) \quad \leq c \cdot C_{OPT}(\sigma) + \alpha,$$

где $C_A(\sigma)$ — сложность онлайн-алгоритма A, $C_{OPT}(\sigma)$ — сложность соответствующего оптимального оффлайн-алгоритма, α — некоторая константа.

Параметр c называется коэффициентом оптимальности (competitive ratio) онлайн-алгоритма.

Описанное условие не всегда может быть выполнено, и онлайн-алгоритмы в общем случае не будут c-оптимальными для произвольной константы c. Например, как видно из таблицы 1, метод транспозиции (transpose) и использование счетчика частоты обращений (frequency-count) не являются cоптимальными.

Для бинарных деревьев используются следующие оценки.

Пусть для алгоритма A запись $A(\sigma,\ T_0)$ обозначает общую стоимость операций доступа и поворотов, выполняемых алгоритмом A над исходным бинарным деревом T_0 в процессе обработки последовательности запросов σ. $OPT(\sigma,\ T_0)$ — минимальная совокупная стоимость этих операций.

Алгоритм A называется $f(n)$-оптимальным ($f(n)$-competitive), если выполняется условие:

$$A(\sigma, T_0) \quad \leq \quad f(n) \cdot OPT(\sigma, T_0) + O(n),$$

где n — число элементов в бинарном дереве T_0.

Обозначим через S статический алгоритм, в котором выполняются операции поиска и не производятся повороты.

Алгоритм A называется $f(n)$-статически-оптимальным ($f(n)$-static-competitive), если:

$$A(\sigma, T_0) \quad \leq \quad f(n) \cdot min_T S(\sigma, T) + O(n),$$

где T — бинарное дерево поиска, содержащее те же элементы, что и T_0, $S(\sigma,\ T) = \sum\limits_{i=1}^{n} f_\sigma^i d_T^i$, f_σ^i — число обращений к элементу i в последовательности запросов σ, d_T^i — длина пути к элементу i в дереве T.

Пусть доступ к i элементу осуществляется с вероятностью p_i. Тогда ожидаемая стоимость запросов для бинарного дерева T равна

$$1 + \sum_{i=1}^{n} p_i d_T^i.$$

Обозначим через $S(P,T)$ стоимость выполнения запросов в дереве T, вероятность которых определяется распределением P. $\overline{S}(P) = \min_T S(P,T)$ — ожидаемая стоимость выполнения запросов с распределением вероятностей P для оптимального дерева поиска. Для алгоритма A через $\overline{A}(P, T_0)$ обозначается асимптотически ожидаемая стоимость обработки запросов, подчиняющихся распределению P и выполняющихся над исходным бинарным деревом T_0.

Алгоритм A называется $f(n)$-распределенно-оптимальным ($f(n)$-distribution-competitive), если:

$$\overline{A}(P, T_0) \quad \leq \quad f(n) \cdot \overline{S}(P).$$

Оценки онлайн-алгоритмов, подобные приведенным в таблице 1 для бинарных деревьев поиска, могут быть получены и для других иерархических структур. Например, в [3] рассматривается организация многоуровневого иерархического индекса на основе B-дерева, приспосабливающегося к изменениям интенсивности использования элементов индекса, объема индекса и наличия свободных ресурсов в системе.

Самоорганизующиеся СД имеют несколько недостатков. Во-первых, несмотря на то, что общее время выполнения запросов может быть невысоким, стоимость отдельных операций может быть высока. Во-вторых, самоорганизующиеся СД требуют дополнительных затрат на перестройку, особенно во время операций доступа к данным.

В рассмотренных самоорганизующихся СД применяются простые правила реорганизации, которые не всегда приводят к требуемому результату (эффективному выполнению будущих запросов). Подобные правила позволяют точнее настроить СД на прогнозируемые запросы, но не приводят к ускорению алгоритмов их обработки.

В [4] исследуются самосовершенствующиеся алгоритмы (self-improving algorithms), повышающие свою производительность на основе параметрической адаптации к неизвестному распределению входных данных. В отличие от самоорганизующихся СД, самосовершенствующиеся алгоритмы работают в режиме оффлайн, поэтому для их оценки не применим конкурентный анализ. В них изучаются свойства не отдельных наборов входных данных, а их распределение. При этом ни сами входные данные, ни их распределение заранее неизвестны.

Предполагается, что необходимо обработать некоторый набор последовательностей входных данных I_1, I_2, I_3,...,I_N,каждая из которых имеет фиксированную длину n: $I_i = (x_1, x_2, \ldots, x_n)$, где элемент x_i принимает значение из некоторого произвольного неизвестного распределения P_i. Целью самосовершенствующегося алгоритма является разработка способа обработки этих входных данных таким образом, чтобы время обработки было минимально для распределения $P = \prod_{i=1}^{N} P_i$.

Для улучшения производительности алгоритм использует дополнительную (служебную) информацию. Он начинает функционировать с фазы обучения, в течение которой собирается информация о распределении входных данных. Так как первоначально о распределении P_i ничего неизвестно, то алгоритм может гарантировать допустимую оценку времени выполнения операций лишь для наихудшего случая.

Работа алгоритма эквивалента выполнению функции $f(I)$, принимающей последовательность входных данных I в качестве аргумента. Самосовершенствующиеся алгоритмы используют данные прошлых запусков для улучшения быстродействия текущей обработки данных. После того как информация о распределении P_i собрана, алгоритм использует ее для ускорения выполнения функции $f(I)$.

Очевидно, что нет необходимости анализировать все множество распределений P, а достаточно выделить только некоторые типичные из них.

После обучения наступает этап устойчивого функционирования алгоритма — стационарной режим. На этом этапе используется релевантная информация, собранная на этапе обучения.

В [4] приводятся примеры самосовершенствующихся алгоритмов для решения задач сортировки последовательности чисел и вычисления триангуляции Делоне для множества точек на плоскости. В стационарном режиме эти алгоритмы работают за линейное время $O(n)$.

Служебная информация, которую используют самосовершенствующиеся алгоритмы, представляется в виде дополнительных СД. Например, алгоритм самосовершенствующейся сортировки сначала вычисляет интервалы распределения чисел и строит для них оптимальные деревья поиска [5]. Затем в стационарном режиме эти деревья используются для того, чтобы распределить по карманам новые входные данные. Для карманов применяется сортировка вставкой и после их конкатенации образуется результирующая упорядоченная последовательность. Тем самым, алгоритм самосовершенствующейся сортировки похож на алгоритм карманной сортировки [5], но в отличие от нее не требует, чтобы входные элементы были равномерно распределены в интервале [0, 1).

Самосовершенствующимся алгоритмам также свойственны недостатки:

— они работают только с входными данными фиксированной длины;

— в них выделяются всего два режима работы и ничего не говорится об их многократном выполнении;

— собранная информация о распределении входных данных не приводит к принципиальному изменению алгоритма в стационарном режиме. Эта информация используется лишь для классификации новых входных данных.

В [6] предлагается синергетический подход к организации СД. В основе этого метода лежит представление об эволюции (постепенном необратимом изменении) СД и методов обработки данных. В процессе эволюции могут изменяться не только конструктивные элементы СД [7], но и используемые алгоритмы. В результате выполнения операций обработки данных должны формироваться СД, максимально ориентированные на требуемое использование.

Алгоритмы преобразования СД должны быть основаны не на простом переупорядочивании элементов, а включать 4 вида основных преобразований: тождественные, количественные, качественные и относительные. В процессе использования СД могут изменяться структурные отношения и законы композиции, т. е. СД преобразуется в СД другого типа (например, список в дерево, бинарное сбалансированное дерево в бинарное несбалансированное дерево, бинарное дерево в m-арное дерево). При этом реорганизация СД не должна быть слишком трудоемкой. Алгоритмы реорганизации СД целесообразно реализо-

вывать как онлайн-алгоритмы, чтобы избежать блокировки системы (невыполнения операций обработки данных).

В [7] на основе общей теории систем Ю. А. Урманцева (ОТС-У) определено поле структур данных (ПСД) как пространство всех возможных структур данных. При этом СД одного качества, имеющие близкую структуру и обладающие сходными свойствами, представляются в виде R-системы. Такой подход позволяет состояниям реальных СД и их компонентов сопоставить точки ПСД, а изменение или эволюцию (смену состояний) — рассматривать как последовательные переходы (движение) от точки к точке в ПСД.

В [8] определены типы допустимых преобразований СД, согласованные с изменениями и эволюцией систем в ОТС-У. Особое значение имеет включение исключений (отклонений, аномалий). Исключение — это изменение СД с нарушение законов композиции. Например, нарушение условия сбалансированности у деревьев поиска, добавление элемента, нарушающего порядок в упорядоченной последовательности, включение дубликатов в множество и т. д.

К отклонениям приводят операции обработки данных, которые не могут быть полностью выполнены за допустимые промежутки времени. Так, добавление нового узла в бинарное дерево поиска требует определенных вращений для поддержания условия сбалансированности. Если эта операция не может быть сразу выполнена (из-за высокой стоимости), то ее целесообразно отложить. В дальнейшем во время “бездействия системы”, когда не выполняются запросы на обработку данных либо их поступление маловероятно, СД может быть реорганизована полностью (будут восстановлены законы композиции).

Если число исключений в СД превышает некоторый порог (снижается эффективность обработки данных и невозможно восстановить структурные отношения и законы композиции за допустимое время), то целесообразно выполнить R-альное преобразование, изменяющее тип СД. В качестве новой R-системы выбирается та, для которой значения коэффициента структурного соответствия [8] текущей СД максимально (количество отклонений является минимальным).

Таким образом, использование исключений является фиксацией информации о требуемом использовании данных и позволяет снизить затраты на преобразование СД (реорганизация СД производится не при каждом обращении) с сохранением высокой эффективность обработки данных.

Вследствие нестационарности потока запросов на обработку данных эволюции СД свойственно периодическое чередование режимов функционирования. Так, в *LS*-режиме [9] возрастания интенсивности запросов в ПСД формируется сложный спектр структур-аттракторов (образуются СД, ориентированные на эффективное выполнение специализированных запросов), а в *HS*-режиме [9] размывания структур формируются универсальные СД, ориентированные на более широкий тип запросов.

СПИСОК ЛИТЕРАТУРЫ

1. Albers S. Online algorithms: a surveys. Mathematical Programming, Volume 97, Numbers 1–2. Springer, Berlin, 2003. P. 3–26.

2. Albers S., Westbrook J. Self-organizing data structures, in Online Algorithms (Schloss Dagstuhl, 1996), Lecture Notes in Comput. Sci. 1442, Springer-Verlag, Berlin, 1998. P. 13–51.

3. Дрождин В. В. Организация адаптивного индекса // Математика и информатика: Межвузовский сборник. Пенза: ПГПУ, 1996. С. 80–85.

4. Ailon N., Chazelle B., Clarkson K., Liu D., Mulzer W., Seshadri C. Self-improving algorithms. SIAM Journal on Computing. 2011. Vol. 40. Issue 2. P. 350–375.

5. Кормен Т. Х., Лейзерсон Ч. И., Ривест Р. Л., Штайн К. Алгоритмы: построение и анализ. М.: Издательский дом “Вильямс”, 2007. 1296 с.

6. Дрождин В. В., Володин А. М. Синергетический подход к организации структур данных // Программные продукты и системы. 2010. № 1. С. 29–34.

7. Дрождин В. В., Володин А. М. Поле структур данных // Известия ПГПУ им. В. Г. Белинского. Физико-математические и технические науки. Пенза: ПГПУ, 2010. № 18 (28). С. 123–129.

8. Дрождин В. В., Володин А. М. Преобразование структур данных в поле структур данных // Известия ПГПУ им. В. Г. Белинского. Физико-математические и технические науки. Пенза: ПГПУ, 2011. № 26. С. 380–385.

9. Князева Е. Н., Курдюмов С. П. Основания синергетики: Синергетическое мировидение. М.: Книжный дом “ЛИБРОКОМ”, 2010. 256 с.