МОДЕЛЬ ОБРАБОТКИ СТРОК И СПИСКОВ ДАННЫХ ДЛЯ СИСТЕМ, УПРАВЛЯЕМЫХ ЗНАНИЯМИ

Ивашенко В. П.
Кафедра интеллектуальных информационных технологий, Факультет информационных технологий и управления, Белорусский государственный университет информатики и радиоэлектроники
Минск, Республика Беларусь
E-mail: ivashenko@bsuir.by

В статье рассматриваются команды и операции модели обработки строк и списков данных

Введение

При реализации систем, управляемых знаниями (СУЗ) [1], выделяются три уровня: уровень управления устройствами (ресурсами), уровень управления данными и уровень управления знаниями. Ранее были рассмотрены некоторые совместимые модели обработки информации [2] первых двух уровней, в том числе те, которые обеспечивают алгоритмическую универсальность [3]. В памяти с линейно организованным адресным пространством основными структурами данных (СД) с точки зрения реализации являются массивы, представляющие непрерывные структуры данных, и списки, которые могут храниться в памяти отдельными частями (множеством массивов). При работе с массивами, как и со списками, кроме основных, связанных с чтением и записью данных, требуется выполнять такие операции как переразмещение (реаллокация) и копирование данных. Если перечисленные операции для массивов в основном рассмотрены [4], то для списков некоторые их свойства по отношению к задаче управления данными остались невыявленными. Следует отметить, что одними из основных качеств интеллектуальных систем, к которым относятся СУЗ, являются обучаемость и открытость. При этом система должна обладать таким качеством как отзывчивость, т.е. системе следует быть или становиться доступной для ввода информации, в том числе для обучения, на протяжении непродолжительных временных интервалов, а не быть постоянно «самой себе на уме». С другой стороны, количество информации, которую нужно сообщить системе для того, чтобы её «обучить», не должно быть чрезмерным («учитель не должен решать задачу за ученика»), т.е. необходимо минимизировать количество сеансов ввода и объём «учебного материала». Эти требования можно сформулировать как требование минимизации при условиях, выраженных следующей системой.

$$\begin{cases} k*(1+t) \geq k+n \\ t \geq 0 \\ k > 0 \\ n > 0 \\ \alpha > 0 \\ \beta > 0 \end{cases} \quad (1)$$

Пусть команда обрабатывается не более чем за время $1+t$ $(t \geq 0)$. Пусть, чтобы обработать данные за время n $(n > 0)$, требуется k команд $(k > 0)$, тогда оценка времени их работы не менее времени чтения команд и времени обработки данных: $k*(1+t) \geq k+n$. Требуется минимизировать количество команд и максимально допустимое время их обработки.

$$\alpha * t + \beta * k \to \min \quad (2)$$

Решение этой задачи можно сформулировать как «принцип квадрата обучения»: квадрат количества команд (обучения) или время обработки каждой из них прямо пропорциональны времени, затраченному на решение задачи.

$$k = \sqrt{\frac{\alpha * n}{\beta}}; t = \sqrt{\frac{\beta * n}{\alpha}} \quad (3)$$

Исходя из этих условий и требования, отдельные операции, выполняемые системой, время их выполнения, по возможности должны соответствовать этому принципу.

Представление данных в виде списка может быть организовано для различных СД, в том числе логически доступных как строка или массив. Поэтому при работе с данными в виде списка можно выделить такие операции как доступ к элементу списка с указанным номером. Если элементами списка являются пары из элементов данных (ЭД), то это позволяет реализовать представление таких математических абстракций как функции и отношения на множестве ЭД соответствующего типа. Следует отметить, что далее рассматриваются списки, размер которых ограничен (разрядностью одной ячейки памяти): в случае недостатка свободной памяти операции не меняют исходные данные. Операции над списками неограниченного размера требуют отдельного рассмотрения.

Операции обработки строк и списков

Перечислим следующие операции над списками (строками): добавление элемента в начало списка, удаление элемента из начала списка, удаление всех элементов списка, конкатенация двух разных списков (строк), разделение списка (строки), доступ к элементу списка с указанным номером. Добавим к перечислению также

операции итерирования и операции над списками, хранящими для каждого ЭД его некоторый числовой индекс: операции добавления ЭД и его индекса, операции удаления ЭД и его индекса, поиска ЭД по его индексу. Кроме времени работы операции важной характеристикой её работы является её пропускная способность. Поэтому среди перечисленных операций нет операции копирования (создания) списка, но есть операция удаления элементов списка, так как она является критически важной с точки зрения пропускной способности: при удалении не создаётся никаких ЭД (кроме пустого списка). Для перечисленных операций за основу представления строк взята такая СД как «последовательностное дерево» (sequence tree) [5], которую можно рассматривать как модификацию такой СД как B-дерево. Время работы операций с этой структурой зависит от глубины дерева, которая в свою очередь зависит от количества хранимых элементов данных и количества ссылок в узле на другие узлы. Одной из самых затратных по времени операций является операция удаления элементов списка (строки), поэтому количество хранимых ссылок в узле определяется, исходя из соображений минимизации времени её работы. При выборе количества хранимых ссылок q ($q > 1$) близким к квадратному корню от (максимального) размера списка n, можно ограничить количество вызовов операции высвобождения памяти до $n/q \to \sqrt{n}$. Оценка времени и пропускной способности операций при этих условиях приведена в следующей таблице (см. табл. 1).

Таблица 1 – Характеристики операций

операция	временная сложность	оценка пропускной способности
push	$O(\log_q n * g(n))$	$O(\log_n q/g(n))$
pop	$O(h(r,q,n) + \log_q n * g(n))$	$O(1/(h(r,q,n) + \log_q n * g(n)))$
popall	$O(h(r,q,n) + g(n) * n/q)$	$O(1/(h(r,q,n) + g(n) * n/q))$
concat split	$O(h(r,q,n) + \log_q n * g(n) + f(n) * n/q)$	$O(1/(h(r,q,n) + \log_q n * g(n) + f(n) * n/q))$
valueat	$O(\log_2 n * f(n)))$	$O(\log_n 2/f(n))$
start increment decrement break	$O(h(r,q,n) + \log_q n * f(n))$	$O(1/(h(r,q,n) + \log_q n * f(n)))$
over	$O(f(n))$	$O(1/f(n))$
value	$O(f(n))$	$O(1/f(n))$
keep	$O(\log_q n * g(n))$	$O(\log_n q/g(n))$
memorize lose	$O(h(r,q,n) + (\log_2^2 n + n/q) * f(n) + \log_q n * g(n))$	$O(1/(h(r,q,n) + (\log_2^2 n + n/q) * f(n) + \log_q n * g(n)))$
remember	$O(h(r,q,n) + \log_2^2 n * f(n))$	$O(1/(h(r,q,n) + \log_2^2 n * f(n)))$

Здесь g – функция временных затрат на перераспределение (высвобождение, выделение) памяти для структуры (строки) с n ЭД, f – функция временных затрат на доступ к одной ячейке памяти, способной хранить структуру с n ЭД, h – функция временных затрат на обработку r итераторов над строкой длины n ($h(r,q,n) \sim r * f(r+n)$). Были проведены вычислительные эксперименты и сравнение результатов и характеристик работы (см. рис. 1) реализованных операций над представленными списками и «наивной» реализацией этих операций (https://bitbucket.org/version/openjsvvm/). Сравнение с аналогичными операциям над массивами JavaScript (IE 11) показало превосходство реализованных операций до четырёх раз.

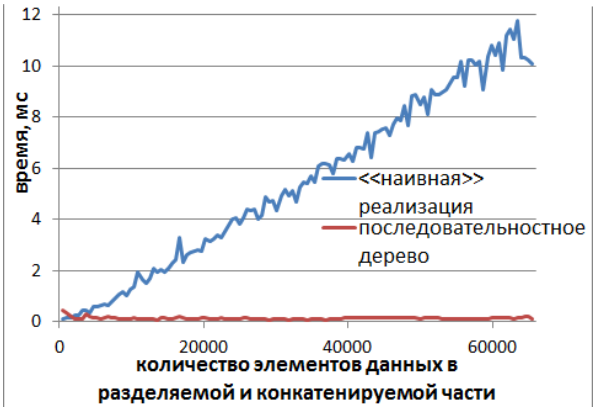

Рис. 1 – Сравнение времени выполнения операций разделения строки на две и их конкатенации

Заключение

Рассмотренные структуры, операции их спецификация и реализация, кроме непосредственного применения в задачах обработки данных, например при реализации пользовательского интерфейса, являются основой для реализации операций для обработки и представления знаний в соответствии с моделью унифицированного семантического представления знаний.

1. Голенков, В. В. Семантическая технология компонентного проектирования систем, управляемых знаниями / В. В. Голенков, Н. А. Гулякина // OSTIS. 2015. С. 57-–78.
2. Ивашенко, В. П. Модели обработки информации в интеллектуальных системах, основанных на семантических технологиях / В. П. Ивашенко, А. С. Бельчиков, А. П. Еремеев // Информационные технологии и системы 2016 (ИТС 2016) : материалы международной научной конференции (БГУИР, Минск, Беларусь, 26 октября 2016) / редкол. : Л. Ю. Шилин [и др.]. — Минск: БГУИР, 2016. — С. 106—107.
3. Ивашенко, В. П. Модели обработки информации и программные средства для универсальных моделей решения задач / В. П. Ивашенко, // Информационные технологии и системы 2017 (ИТС 2017) : материалы международной научной конференции (БГУИР, Минск, Беларусь, 25 октября 2017) / редкол. : Л. Ю. Шилин [и др.]. — Минск: БГУИР, 2017. — С. 106—107.
4. Ивашенко, В. П. Операции управления массивами данных в линейно адресуемой памяти / В. П. Ивашенко, С. В. Синцов // Доклады БГУИР. – 2016. – № 6 (100). – С. 86–93.
5. Bates, R. Sequence-Trees: Slicing and Concatenation of Sequences in Logarithmic Time / R. Bates // 14th Midwest Conference on Combinatorics, Cryptography, and Comp. Wichita: WSU, – 2002. – Vol.42. –P. 33—60.