УДК 004.424

С.С. СИНЕЛЬНИКОВ

Донецкий государственный институт искусственного интеллекта, Украина

ПОИСК В УПОРЯДОЧЕННОМ СПИСКЕ И СОРТИРОВКА СПИСКА С НАИМЕНЬШИМ КОЛИЧЕСТВОМ СРАВНЕНИЙ МЕТОДОМ ИТЕРАЦИЙ

Применен численный метод поиска данных – метод итераций для задач поиска и сортировки в динамических структурах данных – односвязном и двухсвязном списках. Улучшены методы сортировки вставками и естественного двухпутевого слияния для списков, что позволило уменьшить сложность процесса сортировки до порядка NlogN без использования лишних операций продвижения по списку. Проведен теоретический и практический анализ эффективности применения данного метода, который подтвердил для метода итераций уменьшение количества сравнений и увеличение скорости поиска по сравнению с другими методами. Выявлены слабые и сильные стороны итерационного метода поиска. Показаны условия, при которых следует применять данный метод.

поиск данных, численные методы, односвязный список, линейный поиск, итерационный метод

Введение

Хранение и поиск информации – одна из ключевых задач при разработке баз данных информационных систем, эффективность работы которых характеризуется скоростью доступа к определенным данным. Соответственно методы поиска, которые решают задачу доступа, и их реализация непосредственно влияют на эффективность информационных систем.

Для организации хранения данных выше описанные информационные системы используют, как правило, массивы или динамические структуры данных (односвязные и двусвязные списки, деревья) [1 – 4]. Достоинства массивов в том, что в них возможно достаточно легко и эффективно организовать поиск данных, но есть и недостатки – это операции добавления и удаления данных, которые для массива не эффективны. Эти недостатки были устранены в динамических структурах данных, но в отличие от массивов скорость поиска в них низкая (односвязные и двусвязные списки), так на данный момент применяется для нахождения элемента линейный поиск (причем независимо от того упорядочены данные в списке или нет).

В бинарных деревьях устранен этот недостаток за счет дополнительной памяти (указателей), использование которой в некоторых задачах не всегда приемлемо.

В данной работе предлагается метод, который в отличие от линейного метода использует меньшее количество сравнений для поиска и сортировки в односвязных и двусвязных списках, которые все чаще применяются в задачах хранения данных.

1. Разработка итерационного метода поиска для отсортированного односвязного списка

Задача поиска в массиве заключается в поиске индекса элемента, равного *key*, в массиве *f* с размерностью N. Для списков задача заключается в поиске адреса элемента со значением *key*.

В работе [5] показана связь численных методов поиска нуля функции [6 – 9] с задачей поиска индекса элемента массива по его ключу [1 – 4], было доказано, что метод итераций дает решение задачи поиска индекса, но при этом требует, чтобы был вычислен перед его применением коэффициент C:

$$C=\frac{1}{\max\{|f[I+1]-f[I]|,\ I=\overline{0,N-2}\}}. \qquad (1)$$

Коэффициент (1) используется для вычисления индекса искомого элемента в массиве в соответст-

вии с рекуррентной формулой (2) (поиск выполняется с начала массива):

$$I_{n+1} = I_n + (key - f[I_n]) \cdot C \,. \qquad (2)$$

В выражении (2) округление нового значения индекса производится в большую сторону.

Рассмотрим отсортированный односвязный список. Пронумеруем элементы списка, начиная с нуля, также как и массив, тогда возможно применение к списку метода итераций. В соответствии с (2) метод итераций дает полезную информацию – на сколько нужно продвинуться вперед по списку до искомого элемента без проверок на совпадение с ключом, которые выполняются в линейном поиске. Таким образом, производится однонаправленное движение по списку с меньшим количеством проверок на совпадение относительно линейного поиска с гарантированной сходимостью.

Итак, получаем алгоритм поиска элемента в отсортированном односвязном списке (рис. 1).

На рис. 1 **first** – указатель на первый элемент списка, **last** – указатель на последний элемент списка, key – искомый элемент. Все элементы списка представляют собой структуру, состоящую из следующих полей: **data** – данные; **next** – указатель на следующий элемент списка.

C – коэффициент, который вычисляется до применения алгоритма в соответствии с (1). Подсчет коэффициента C – недостаток данного алгоритма, но его можно частично устранить. Для этого возможно просчитывать максимальное расстояние между элементами списка во время добавления нового элемента в список. Данный подход распределит затраты по времени для подсчета коэффициента C. В случае, когда имеется двусвязный список, возможно движение как с начала списка, так и с его последнего элемента. Это может быть полезно при использовании параллельных вычислений, которые достаточно активно внедряются в современные компьютеры.

Заметим, что при поиске элемента появляются дополнительные вычисления, в которые входит медленно выполняемая операция – умножение. Это несколько замедляет работу алгоритма. При вычислении **low = low + (key - p->data)*c** округление производится в большую сторону.

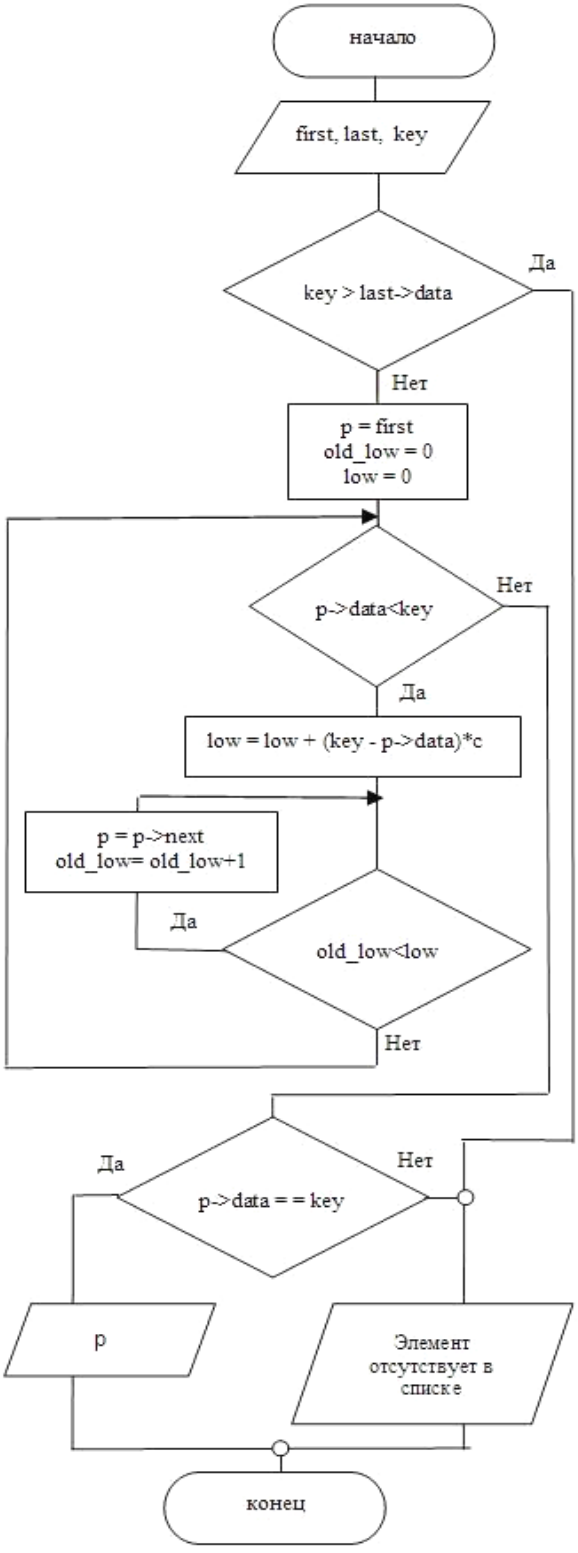

Рис. 1. Алгоритм поиска элемента в отсортированном односвязном списке

Теоретическая сложность алгоритма итерации в общем случае порядка $logN$, что значительно лучше, чем у линейного алгоритма, теоретическая сложность которого порядка N.

Наихудший вариант для данного метода наблюдается в том случае, когда элементы списка удовлетворяют условию (3):

$$\begin{cases} f[i] \le f[i+1]; \\ f[i+1] - f[i] \le f[i+2] - f[i+1]. \end{cases} \quad (3)$$

Условие (3) означает быстрый рост значений элементов списка относительно предыдущих элементов. Если выполняется условие (3), то константа C будет иметь малое значение, что приводит к медленному движению к искомому элементу. В этом случае метод итераций практически вырождается в линейный поиск с дополнительными вычислениями, что является его недостатком.

Заметим, что для двусвязного списка возможно применение выражения (4) для получения нового индекса:

$$I_{n+1} = I_n + (key - f[I_n]) \cdot C \cdot K, \quad (4),$$

где K – некоторая константа большая единицы, которая определяется опытным путем и задает дополнительную скорость движения к элементу.

Очевидно, что с применением (4) появляется риск перехода за искомый элемент. В случае, если это происходит, возможно движение в обратном направлении с меньшим значением K.

Таким образом, для двусвязных списков возможно применение различных модификаций итерационного метода поиска, что отражает его гибкость.

2. Анализ результатов применения итерационного метода поиска для односвязного отсортированного списка

Применим метод итерации для поиска данных и сравним его с линейным методом. В табл. 1 приведена статистика количества выполненных сравнений в зависимости от количества элементов в списке для данных с равномерным распределением. В ней для линейного метода общее количество сравнений всегда вычисляется по формуле (5):

$$\frac{N \cdot (N+1)}{2}. \quad (5)$$

Таблица 1

Количество сравнений, произведенных в отсортированном списке линейным и итерационным методами для данных с равномерным распределением

Количество элементов	Количество сравнений	
	Метод итераций	Линейный метод
100	800	5.050
1.000	11.500	500.500
10.000	15.000	50.005.000
20.000	317.000	200.010.000
30.000	500.000	450.015.000

Количество сравнений, произведенных итерационным методом, имеет порядок $logN$ для данных с равномерным распределением.

Количество сравнений, произведенных в отсортированном списке линейным и итерационным методами поиска для данных, удовлетворяющих условию (3), практически не отличаются. Этот результат говорит о нестабильности метода итераций, теоретическая сложность которого колеблется от $logN$ до N, в зависимости от характера данных. Причем, чем меньше константа C, тем ближе теоретическая сложность метода к порядку N.

Таким образом, метод итераций не приемлет резких скачков между рядом стоящими элементами, в случае возникновении которых скорость поиска резко падает.

3. Применение итерационного метода поиска для сортировки односвязного списка

Для сортировки односвязного списка, как правило, используют сортировку вставкой элемента на нужную позицию. На данный момент поиск места для вставки элемента определялся линейным методом, что приводило к теоретической сложности ал-

горитма сортировки к порядку N^2. Также для сортировки списков применяется метод двухпутевого слияния, теоретическая сложность которого порядка $NlogN$ (метод использует дополнительную память). К сожалению, применение его модификации метода естественного двухпутевого слияния нецелесообразно, так как бинарный поиск для списков не применим.

Исходя из результатов полученных в разделах 1 и 2, можно утверждать, что применение метода итерации для поиска места вставки элемента даст лучший результат по скорости. В таком случае получаем теоретическую сложность порядка $NlogN$ для метода сортировки итерационными вставками.

Если рассматривать итерационный метод поиска как основу для поиска элементов, выбираемых для слияния, то применение сортировки методом естественного двухпутевого слияния станет возможным. Такое усовершенствование позволит снизить теоретическую сложность процесса сортировки, которая, тем не менее, будет на уровне $NlogN$.

Заметим, что все рассуждения по процессам сортировки можно отнести и к двусвязным спискам. Таким образом, были улучшены методы сортировки вставками и естественного двухпутевого слияния для списков.

Заключение

В данной статье итерационный алгоритм поиска применен для односвязных и двусвязных списков. Теоретический и практический анализ показали эффективность и целесообразность применения данного метода. Выявлены слабые и сильные стороны итерационного метода поиска. Были показаны условия, при которых следует применять данный метод.

Показана возможность применения итерационного метода поиска к задаче сортировки списков. Улучшены методы сортировки вставками и естественного двухпутевого слияния для списков, что позволило уменьшить сложность процесса сортировки до $NlogN$.

Применение итерационного метода позволит в ряде случаев увеличить скорость поиска данных в любом программном комплексе, использующем поиск данных (например, сервере баз данных, файловых системах, базах данных интеллектуальных систем), что говорит о высокой практической значимости полученных результатов.

Литература

1. Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. Алгоритмы: построение и анализ. 2-е изд. – М.: Вильямс, 2005. – 1296 с.

2. Кнут Д. Искусство программирования. – 3-е изд. – М.: Вильямс, 2005. – 720 с.

3. Седжвик Р. Фундаментальные алгоритмы на C++: Ч. 1-4: Анализ, структуры данных, сортировка, поиск. – К.: Diasoft, 2001. – 687 с.

4. Страуструп Б. Язык программирования C++. – М.: Бином, 2005. – 1098 с.

5. Синельников С.С. Применение численных методов к задаче поиска данных в отсортированном массиве // Радіоелектронні і комп`ютерні системи. – 2007. – №1. – С. 68-73.

6. Березин И.С., Жидков Н.П. Методы вычислений. – Т.1. – М.: Физматгиз, 1962. – 632 с.

7. Вержбицкий В.М. Основы численных методов. – М.: Высш. шк., 2002. – 840 с.

8. Волков Е.А. Численные методы. – М.: Наука, 1982. – 254 с.

9. Калиткин Н.Н. Численные методы. – М.: Наука, 1978. – 512 с.

Поступила в редакцию 15.10.2007

Рецензент: канд. техн. наук С.А. Поливцев, Донецкий государственный институт проблем искусственного интеллекта, Донецк.