УДК 681.3:007

ПОСТРОЕНИЕ ДВОИЧНОГО ДЕРЕВА НА ОСНОВЕ ПАРАЛЛЕЛЬНОЙ СОРТИРОВКИ

Ромм Я.Е., Чабанюк Д.А.

Таганрогский институт имени А.П. Чехова (филиал) ФГБОУ ВО РГЭУ (РИНХ), Таганрог, e-mail: romm@list.ru

В статье синтезированы параллельные алгоритмы построения двоичного дерева на основе параллельной и последовательной сортировок по матрицам сравнений. Даны три разновидности алгоритмов в конструктивной форме для множества из N элементов. Временная сложность построения двоичного дерева соответственно оценивается из соотношений $T(R) = O(1)$, $T(R) = O(\log_2 N)$, где число процессоров $R = (N^2 - N)/2$ и $T(1) = O(\log_2 N)$. Оценки получены на модели неветвящихся параллельных программ без учета операций обмена. Алгоритмы инвариантны относительно вида сортируемой по неубыванию последовательности, построение дерева выполняется со свойством единственности. Приводится пример работы параллельного алгоритма построения двоичного дерева с пошаговой интерпретацией процесса определения корней, ветвей и поддеревьев. Устанавливается взаимно однозначное соответствие двоичного дерева и отсортированной последовательности его элементов, а также возможность взаимного преобразования этих структур. Результаты ориентированы на создание эффективных методов динамической обработки баз данных.

Ключевые слова: структуры данных, двоичные деревья, алгоритмы параллельных и последовательных сортировок

CONSTRUCTING BINARY TREE BASED ON PARALLEL SORTING ALGORITHM

Romm Y.E., Chabanyuk D.A.

Taganrog Institute of Chekhov A.P. (branch) RGEU (RINH), Taganrog, e-mail: romm@list.ru

In the article synthesized parallel algorithms for constructing a binary tree based on parallel and serial sorting on comparison matrix. The below presents three types of algorithms in a constructive mode for a variety of N elements. The time complexity of constructing a binary tree respectively estimated out of relations $T(R) = O(1)$, $T(R) = O(\log_2 N)$, where the number of processors $R = (N^2 - N)/2$ and $T(1) = O(\log_2 N)$. An estimate was obtained on the straight-line model of parallel programs without taking into account operations of the exchange. Algorithms invariants on the form of a sequence sorted in ascending order, building tree is performed with the uniqueness property. This article provides an example of the parallel algorithm for constructing a binary tree with a turn-based interpretation of the definition of roots, branches and subtrees. Installed to-one correspondence of the binary tree and the sorted sequence of its elements, as well as the possibility of mutual transformation of these structures. The results are focused on the creation of effective methods of processing dynamic databases.

Keywords: data structures, binary trees, parallel and serial sorting algorithms

Древовидные структуры данных являются одними из наиболее распространенных в информатике и программировании, представляют собой иерархические структуры в виде набора связанных узлов. Такие структуры эффективны для представления динамических данных с целью быстрого поиска информации. Рост объемов обрабатываемой информации делает последовательное построение древовидных структур данных [2, 3] недостаточно эффективным, целесообразна разработка соответственных параллельных алгоритмов. В частности, актуальна задача синтеза и анализа параллельных алгоритмов построения двоичного дерева. Ниже параллельные алгоритмы рассматриваются на модели неветвящихся параллельных программ, временная сложность $T(R)$ алгоритма (кратко – время выполнения) измеряется количеством последовательных шагов без учета обмена, R – число процессоров [7].

Предлагаемое решение задачи реализуется на основе алгоритмов параллельной сортировки.

Параллельная модификация сортировки подсчетом

Для построения двоичного дерева ко всем элементам массива данных будет применяться максимально параллельная сортировка подсчетом по матрице сравнений [5]. Модификация известного алгоритма заключается в следующем. Для одномерного массива $a = (a_1, a_2, ..., a_n)$ сортируемых элементов составляется матрица сравнений, элемент α_{ij} которой определяется из соотношений

$$\alpha_{ij} = \operatorname{sign}\left(a_j - a_i\right) = \begin{cases} 1(+), & a_j > a_i; \\ 0, & a_j = a_i; \\ -1(-), & a_j < a_i; \quad i = \overline{1,n};\ j = \overline{1,n}. \end{cases}$$

Пусть, например, требуется отсортировать по неубыванию массив $a = (1, 0, –2, –4, 8, 2)$.

Матрица сравнений примет вид:

		a_1	a_2	a_3	a_4	a_5	a_6
		1	0	–2	–4	8	2
a_1	1	0	–	–	–	+	+
a_2	0	+	0	–	–	+	+
a_3	–2	+	+	0	–	+	+
a_4	–4	+	+	+	0	+	+
a_5	8	–	–	–	–	0	–
a_6	2	–	–	–	–	+	0

Входной элемент с номером j получает номер k в отсортированном массиве по правилу [5]: в j-м столбце матрицы подсчитывается количество нулей и плюсов сверху вниз до главной диагонали включительно и складывается с количеством только плюсов ниже диагонали в этом же столбце. Суммарное количество составит значение выходного номера k. Согласно данному правилу вставки получится

$$a_1 = 1 \to c_{(0+++)} \to c_4;$$

$$a_2 = 0 \to c_{(0++)} \to c_3;$$

$$a_3 = -2 \to c_{(0+)} \to c_2;$$

$$a_4 = -4 \to c_{(0)} \to c_1;$$

$$a_5 = 8 \to c_{(++++0+)} \to c_6;$$

$$a_6 = 2 \to c_{(++++\ 0)} \to c_5.$$

Отсортированный массив примет вид $c = (–4, –2, 0, 1, 2, 8)$. Программа модифицированной сортировки представлена в [6] и частично приводится ниже.

При параллельной обработке все сравнения могут выполняться одновременно и взаимно независимо, требуемое количество процессоров составит $\left(N^2 - N\right)/2$. Отсюда временная сложность сортировки оценивается из соотношения

$$T\left(\frac{N^2 - N}{2}\right) = \tau = O(1), \qquad (1)$$

где τ – время бинарного сравнения. Сортировка сохраняет порядок равных элементов, в явном виде задает взаимно однозначное соответствие входных и выходных индексов сортируемых элементов (адреса вставки: `c[k]: = a[j];` адреса входных элементов в порядке расположения в отсортированном массиве: `e[k]: = j;`):

```
for j:= 1 to n do
begin
k:= 0;
for i:= 1 to j do
if a[j] >= a[i] then k:= k+1;
for i := j+1 to n do
if a[j] > a[i] then k := k+1;
c[k] := a[j]; e[k] := j;
end;
```

В приводимом ниже (пример 1) соотношении последовательностей (4) `e[k]` совпадает по значению и смыслу с i_k. Использование входных индексов, записанных в `e[k]`, позволит рассматриваемые в дальнейшем преобразования выполнять без перемещения элементов.

Параллельное построение двоичного дерева

Двоичное дерево – это структура данных, для которой выполняются следующие условия: оба поддерева – левое и правое – являются двоичными деревьями. У всех узлов левого поддерева произвольного узла X значения элементов меньше, нежели значение элементов самого узла X. В то же время значения элементов всех узлов правого поддерева (того же узла X) больше, нежели значение элементов узла X [1].

Пусть задано некоторое множество из N элементов X_i с отношением порядка $\leq$, расположенных в виде одномерного массива. Массив сортируется с помощью максимально параллельной сортировки за время (1). В качестве корня двоичного дерева выбирается серединный элемент отсортированного массива C с округлением индекса середины до ближайшего целого, не меньшего самого числа: $j_{\text{ср}} = \left\lceil \frac{N}{2} \right\rceil$. Тогда все элементы отсортированного массива слева от $C_{j_{\text{ср}}}$ меньше либо равны (не превосходят в смысле данного отношения порядка) $C_{j_{\text{ср}}}$, они автоматически определяются принадлежащими левому поддереву (левому подмассиву). Аналогично элементы справа от $C_{j_{\text{ср}}}$ не меньше (с сохранением порядка равных элементов) $C_{j_{\text{ср}}}$ и автоматически определяются как принадлежащие правому поддереву (правому подмассиву).

Далее, левый подмассив рассматривается как новый массив для аналогичного определения его корня по номеру

$$j_{\text{ср.лев.1/2}} = \left\lceil \frac{1}{2}\left(\left\lceil \frac{N}{2} \right\rceil - 1 \right) \right\rceil,$$

или

$$j_{\text{ср.лев.1/2}} = \left\lceil \frac{j_{\text{ср}} - 1}{2} \right\rceil.$$

Такой корень, $C_{j_{\text{ср.лев.1/2}}}$, станет ближайшим слева потомком ранее найденного корня всего дерева $C_{j_{\text{ср}}}$, а также корнем левого поддерева. В силу сортировки левое поддерево с корнем $C_{j_{\text{ср.лев.1/2}}}$ сохраняет требуемые свойства: все элементы подмассива слева от $C_{j_{\text{ср.лев.1/2}}}$ не превосходят $C_{j_{\text{ср.лев.1/2}}}$ (в смысле данного отношения порядка), все элементы подмассива справа, аналогично, не меньше $C_{j_{\text{ср.лев.1/2}}}$.

Одновременно с левым правый подмассив рассматривается как новый массив для аналогичного определения его корня по номеру

$$j_{\text{ср.прав.1/2}} = \left\lceil \frac{N}{2} \right\rceil + \left\lceil \frac{1}{2}\left(\left\lceil \frac{N}{2} \right\rceil - 1 \right) \right\rceil,$$

или

$$j_{\text{ср. прав. 1/2}} = j_{\text{ср}} + \left\lceil \frac{j_{\text{ср}} - 1}{2} \right\rceil.$$

Такой корень, $C_{j_{\text{ср.прав.1/2}}}$, станет ближайшим справа потомком ранее найденного корня всего дерева $C_{j_{\text{ср}}}$, а также корнем правого поддерева. В силу сортировки правое поддерево с корнем $C_{j_{\text{ср.прав.1/2}}}$ сохраняет требуемые свойства: все элементы справа от $C_{j_{\text{ср.прав.1/2}}}$ не меньше $C_{j_{\text{ср.прав.1/2}}}$ (в смысле данного отношения порядка), все элементы слева не больше $C_{j_{\text{ср.прав.1/2}}}$. Далее, процесс итеративно повторяется по отношению к каждой паре смежных подмассивов слева и одновременно справа:

$$j_{\text{ср.лев.1/4,1}} = \left\lceil \frac{j_{\text{ср.лев.1/2}} - 1}{2} \right\rceil;$$

$$j_{\text{ср. лев. 1/4,2}} = j_{\text{ср. лев. 1/2}} + \left\lceil \frac{j_{\text{ср.лев.1/2}} - 1}{2} \right\rceil;$$

$$j_{\text{ср. прав. 1/4,1}} = j_{\text{ср. прав. 1/2}} - \left\lceil \frac{j_{\text{ср. прав. 1/2}} - j_{\text{ср}} - 1}{2} \right\rceil;$$

$$j_{\text{ср. прав. 1/4,2}} = j_{\text{ср. прав. 1/2}} + \left\lceil \frac{j_{\text{ср. прав. 1/2}} - j_{\text{ср}} - 1}{2} \right\rceil \text{ и т.д.}$$

Число шагов параллельного алгоритма построения двоичного дерева складывается из шага сортировки и последовательности параллельных шагов вычисления индексов корней поддеревьев, число которых равно целой части логарифма числа элементов входного множества:

$$T\left(\frac{N^2 - N}{2} \right) = \tau + \left[\log_2 N \right] \tau_0 = O\left(\log_2 N \right), \quad (2)$$

где τ – время бинарного сравнения; τ_0 – время вычисления индекса корня.

Пример 1. Двоичное дерево для массива из $N = 15$ элементов

$$X = (14, 9, 24, 7, 11, 20, 28, 3, 8, 10, 13, 17, 21, 25, 30) \quad (3)$$

содержит уровни корней с номерами от 0 до $\left[\log_2 N \right] - 1 = 2$.

Поэтапно выполняется параллельная сортировка подсчетом по матрице сравнений массива (3), отсортированный массив C примет вид

$$C = (3, 7, 8, 9, 10, 11, 13, 14, 17, 20, 21, 24, 25, 28, 30). \quad (4)$$

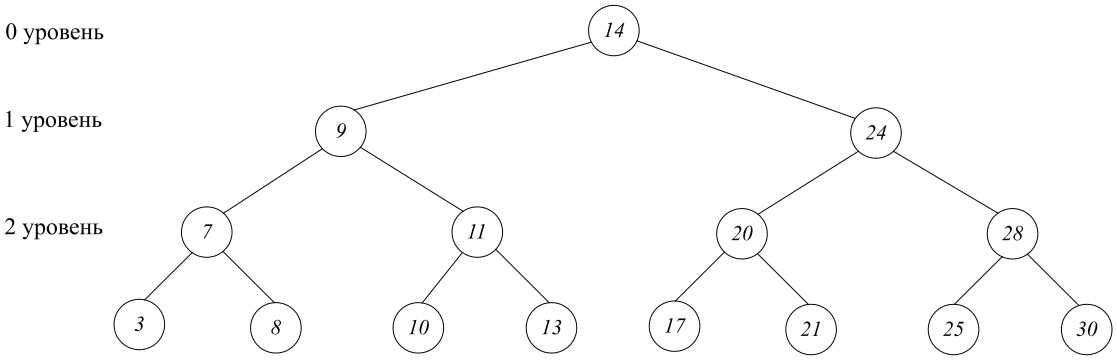
Корнем двоичного дерева является серединный элемент массива (4):

$$j_{\text{ср}} = \left\lceil \frac{15}{2} \right\rceil = 8, \quad C_8 = 14$$

(сформирован 0-й уровень дерева).

Левый подмассив рассматривается как новый массив для аналогичного определения его корня, $j_{\text{ср.лев.1/2}} = \left\lceil \frac{8-1}{2} \right\rceil = 4$, элемент $C_4 = 9$ является ближайшим слева потомком корня $C_{j_{\text{ср}}}$ и корнем левого поддерева. Одновременно правый подмассив рассматривается для аналогичного определения его корня, $j_{\text{ср.прав.1/2}} = 8 + \left\lceil \frac{8-1}{2} \right\rceil = 12$, элемент $C_{12} = 24$ – ближайший справа потомок корня $C_{j_{\text{ср}}}$ и корень правого поддерева (сформирован 1-й уровень дерева). Далее, $j_{\text{ср.лев.1/4,1}} = \left\lceil \frac{4-1}{2} \right\rceil = 2$, элемент $C_2 = 7$ является ближайшим слева потомком ранее найденного корня поддерева $C_{j_{\text{ср.лев.1/2}}}$, а также корнем левого от него поддерева. Одновременно смежный с левым правый подмассив рассматривается как новый массив для аналогичного определения его корня, $j_{\text{ср.лев.1/4,2}} = 4 + \left\lceil \frac{4-1}{2} \right\rceil = 6$, элемент $C_6 = 11$ является ближайшим справа

потомком корня поддерева $C_{j_{\text{ср.лев.1/2}}}$, а также корнем правого от него поддерева. Аналогично левый подмассив от корня $C_{j_{\text{ср.прав.1/2}}}$ правого поддерева рассматривается как новый массив для определения его корня, $j_{\text{ср.прав.1/4,1}} = 12 - \left\lceil \frac{12-8-1}{2} \right\rceil = 10,$ элемент $C_{10} = 20$ является ближайшим слева потомком корня поддерева $C_{j_{\text{ср.прав.1/2}}}$, а также корнем левого от него поддерева. Для смежного с рассмотренным правого подмассива: $j_{\text{ср.прав.1/4,2}} = 12 + \left\lceil \frac{12-8-1}{2} \right\rceil = 14,$ элемент $C_{14} = 28$ является ближайшим справа потомком корня поддерева $C_{j_{\text{ср.прав.1/2}}}$, а также корнем правого от него поддерева (сформирован 2-й уровень дерева). Слева и справа от каждого из четырех найденных корней текущего уровня осталось по одному потомку, которые составят нижний уровень дерева (рисунок).



Пример построения двоичного дерева на основе модифицированной сортировки подсчетом

Таким образом, имеет место.

Лемма 1. Двоичное дерево для массива из N элементов может быть построено параллельно на основе модифицированной параллельной сортировки подсчетом по матрице сравнений с логарифмической оценкой временной сложности (2).

Очевидная модификация заключается в том, что индексы всех серединных элементов всех уровней могут быть вычислены за один шаг: для r-го уровня дерева эти индексы образуются как элементы последовательности $k\left\lceil \frac{N}{2^{r+1}} \right\rceil$, $k = 1, 2, \ldots, r+1$, из которой исключаются индексы корней предшествующих уровней, $r = 0, 1, \ldots, \lceil \log_2 N \rceil - 1$.

Отсюда вытекает.

Теорема 1. Двоичное дерево для массива из N элементов может быть построено параллельно на основе рассматриваемой сортировки с единичной оценкой временной сложности (1).

Замечание 1. В силу устойчивости рассматриваемой сортировки, при условии округления до ближайшего целого не меньшего самого числа, в процессе вычисления индекса корня, двоичное дерево строится со свойством единственности.

Известен алгоритм последовательной сортировки слиянием по матрицам сравнений с явным заданием взаимно однозначного соответствия входных и выходных индексов сортируемых элементов [4], временная сложность которой $T(1) = O(N\log_2 N)$. Отсюда следует, что на одном процессоре с такой оценкой времени могут быть рассчитаны серединные элементы всех подмассивов.

Таким образом, справедливо следующее утверждение.

Теорема 2. Последовательное построение двоичного дерева для массива из N элементов может быть выполнено с временной сложностью $O(N\log_2 N)$.

Согласно изложенному можно утверждать, что двоичное дерево с единственностью строится на основе устойчивой сортировки, в явной форме задающей соответствие входных и выходных индексов сортируемых элементов. Вместе с тем верно и обратное: обход снизу вверх, слева направо элементов текущего поддерева, возобновляемый каждый раз при переходе от поддерева к предшествующему корню, влечет восстановление отсортированной последовательности. Так, например, для дерева на рисунке получится

3, 7 (корень), 8, 9 (корень), 10, 11 (корень), 13, 14 (корень), 17, 20 (корень), 21, 24 (корень), 25, 28 (корень), 30.

Очевидно, что процесс восстановления может быть максимально распараллелен.

Таким образом, между двоичным деревом и отсортированной последовательностью его элементов существует взаимно однозначное соответствие, которое в обе стороны реализуется конструктивным эффективно распараллеливаемым алгоритмом.

Заключение

В статье изложены разновидности алгоритмов построения двоичного дерева на основе устойчивой максимально-параллельной сортировки подсчетом. Временная сложность параллельного построения двоичного дерева в зависимости от разновидности представленных вариантов оценивается как $T\left(\left(N^2-N\right)/2\right)=O(1)$, $T\left(\left(N^2-N\right)/2\right)=O(\log_2 N)$, что улучшает известные оценки [8], в последовательном варианте – $T(1) = O(N\log_2 N)$. На этой основе устанавливается взаимно однозначное соответствие двоичного дерева и отсортированной последовательности его элементов. Предложенные алгоритмы могут использоваться для создания эффективных методов динамической обработки баз данных.

Список литературы

1. Акопов Р.Р. Двоичные деревья поиска // RSDN Magazine. – 2003. – № 5; url: http://rsdn.ru/article/alg/binstree.xml (дата обращения: 20.07.2015).

2. Ахо А., Хопкрофт Д., Ульман Д. Структуры данных и алгоритмы. – М.: Вильямс, 2003. – 384 с.

3. Вирт Н. Алгоритмы и структуры данных. – М.: ДМК Пресс, 2010. – 272 с.

4. Ромм Я.Е. Параллельная сортировка слиянием по матрицам сравнений. I // Кибернетика и системный анализ. – 1994. – № 5. – С. 3–23.

5. Ромм Я.Е. Параллельная сортировка слиянием по матрицам сравнений. II // Кибернетика и системный анализ. – 1995. – № 4. – С. 13–37.

6. Ромм Я.Е., Чабанюк Д.А. Параллельное построение декартова дерева с логарифмической оценкой временной сложности // Современные проблемы науки и образования. – 2015. – № 1; url: http://www.science-education.ru/121-18604 (дата обращения: 20.07.2015).

7. Солодовников В.И. Верхние оценки сложности решения систем линейных уравнений // В кн.: Теория сложности вычислений. I: Записки научных семинаров ЛОМИ АН СССР. – Л., 1982. – т. 118. – С. 159–187.

8. Kirkpatrick D.G., Przytycka T. Parallel Construction of binary trees with near optimal weighted path length // Algorithmica. – 1996. – Vol. 15. – P. 172–192.

References

1. Akopov R.R. Dvoichnye derevya poiska // RSDN Magazine. 2003. no. 5; url: http://rsdn.ru/article/alg/binstree.xml (data obrashcheniya: 20.07.2015).

2. Aho A., Hopkroft D., Ulman D. Struktury dannyh i algoritmy. M.: Vilyams, 2003. 384 p.

3. Virt N. Algoritmy i struktury dannyh. M.: DMK Press, 2010. 272 p.

4. Romm Ya.E. Parallelnaya sortirovka sliyaniem po matricam sravnenij. I // Kibernetika i sistemnyj analiz. 1994. no. 5. pp. 3–23.

5. Romm Ya.E. Parallelnaya sortirovka sliyaniem po matricam sravnenij. II // Kibernetika i sistemnyj analiz. 1995. no. 4. pp. 13–37.

6. Romm Ya.E., Chabanyuk D.A. Parallelnoe postroenie dekartova dereva s logarifmiche-skoj ocenkoj vremennoj slozhnosti // Sovremennye problemy nauki i obrazovaniya. 2015. no. 1; url: http://www.science-education.ru/121-18604 (data obrashcheniya: 20.07.2015).

7. Solodovnikov V.I. Verhnie ocenki slozhnosti resheniya sistem linejnyh uravnenij // V kn.: Teoriya slozhnosti vychislenij. I: Zapiski nauchnyh seminarov LOMI AN SSSR. L., 1982. t. 118. pp. 159–187.

8. Kirkpatrick D.G., Przytycka T. Parallel Construction of binary trees with near optimal weighted path length // Algorithmica. 1996. Vol. 15, pp. 172–192.

Рецензенты:

Веселов Г.Е., д.т.н., директор Института компьютерных технологий и информационной безопасности, Инженерно-технологическая академия, Южный федеральный университет, г. Таганрог;

Карелин В.П., д.т.н., профессор, заведующий кафедрой прикладной математики и информационных технологий, Таганрогский институт управления и экономики, г. Таганрог.