СБОРНИК **НАУЧНЫХ ТРУДОВ НГТУ. – 2016. – № 3(85).** –98–106

СОВРЕМЕННЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

УДК 004.434:004.94

АСИНХРОННАЯ ПИРАМИДАЛЬНАЯ СОРТИРОВКА*

Н.О. ИВАНОВ[1], Д.О. ТИНГАЙКИН[2]

[1] *630073, РФ, г. Новосибирск, пр. Карла Маркса, 20, Новосибирский государственный технический университет, студент АВТФ. E-mail: stenrulf@ gmail.com*
[2] *630073, РФ, г. Новосибирск, пр. Карла Маркса, 20, Новосибирский государственный технический университет, студент АВТФ. E-mail: anikantonsd@ yandex.ru*

Асинхронизм – неодновременное выполнение различных задач. Этот термин используется в информатике для описания процедур, которые могут выполняться независимо от основного потока. Поток – наименьшая единица обработки, исполнение которой может быть назначено ядром операционной системы. Многопоточность – свойство платформы создавать процессы, которые могут порождать дополнительные потоки, выполнение которых не зависит от времени. Для синхронизации потоков используются такие методы, как использование объектов синхронизации (mutex), общие ресурсы (семафоры), события (1bit event), критические секции, условные переменные, порты завершения и т. д.

Асинхронное программирование – термин, подразумевающий создание многопоточных приложений. Многоядерные процессоры располагаются на одном процессорном кристалле или корпусе. Именно многоядерные процессоры влияют на производительность многопоточных приложений, но для достижения результата необходима оптимизация. Оптимизация – процесс создания максимально лучших характеристик или соотношений и минимизации расходов. В программировании оптимизация – процесс написания максимально быстрого кода, т. е. минимализация тактов процессора.

Ключевые слова: сортировка, асинхронное программирование, пирамидальная сортировка, параллельные вычисления, многопоточность, оптимизация

DOI: 10.17212/2307-6879-2016-3-98-106

ВВЕДЕНИЕ

Задача сортировки – это классическая задача, которую должен уметь решать любой программист. Почти в любой программе происходит работа с большим количеством данных (например, с элементами базы данных),

* Статья получена 24 августа 2016 г.

отправка пакетов команд на робота или работа с документом. Для упорядочивания однотипной информации как раз и используется сортировка. В литературе [1–6] рассмотрено множество видов сортировки, таких как пузырьковая сортировка, шейкерная сортировка, сортировка вставками и др.

Упорядочение – это такое множество, элементы которого подчинены правилу предшествования или следования. Натуральный ряд чисел – пример множества, упорядоченного по величине.

Самым популярным методом сортировки является так называемая быстрая сортировка, ***qsort*** [3]. Но в этой статье мы рассмотрим другой метод – пирамидальную сортировку, ***heapsort*** [1].

Для сравнения эффективности алгоритмов рассчитывают сложность алгоритма: сложность по времени, использование дополнительной памяти [1, 7].

Одним из способов ускорения программы является использование асинхронных вычислений [8–9].

1. ОПИСАНИЕ ПИРАМИДАЛЬНОЙ СОРТИРОВКИ

Пирамидальная сортировка (или сортировка кучей) – классический алгоритм, с которым должен быть знаком каждый программист. Проверенная временем «пирамида» интересна тем, что независимо от входных данных у нее одна и та же сложность по времени, $\boldsymbol{O(nlog(n))}$. Стоит отметить, что сложность популярной сортировки *qsort* составляет в худшем случае $\boldsymbol{O(n^2)}$.

Сортировка пирамидой использует бинарное сортирующее дерево. Сортирующее дерево – это такое дерево, которое обладает следующими свойствами.

1. Каждый лист имеет глубину $\boldsymbol{d}$ либо $\boldsymbol{(d-1)}$, где $\boldsymbol{d}$ – максимальная глубина дерева.

2. Значение в любой родительской вершине не меньше (либо не больше, в зависимости от порядка сортировки) значения ее потомков.

3. Наиболее удобным представлением сортирующего дерева в виде массива является такой массив $\{\boldsymbol{a_0} \ldots \boldsymbol{a_n}\}$, в котором $\boldsymbol{a_0}$ – корень дерева, а потомки элемента $\boldsymbol{a_i}$ – элементы $\boldsymbol{a_{2i+1}}$ и $\boldsymbol{a_{2i+2}}$.

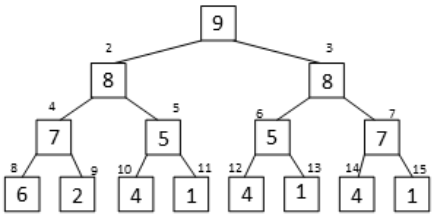

Рис. 1. Пример сортирующего дерева

Пирамидальная сортировка состоит из двух этапов.

1. Формирование из входного массива сортирующего дерева, такого что $a_i \geq a_{2i+1}$ и $a_i \leq a_{2i+2}$ при $1 \leq i \leq n/2$.

2. Выполняется замена максимального элемента на последний элемент кучи, после чего идет **восстановление свойства кучи**, и это действие выполняется ровно n раз, где n – размер массива, каждая итерация выполняется для $n-1$ элементов, т. е. максимальные элементы, поставленные в конец, не рассматриваются. Процесс повторяется до тех пор, пока в сортирующем дереве не останется один элемент, это будет означать, что массив отсортирован.

Нам не удалось найти примера реализации асинхронной пирамидальной сортировки. Поэтому мы хотим предложить вам свой вариант решения этой задачи.

2. ПОСТАНОВКА ЗАДАЧИ

- Найти способ эффективно использовать многопоточность в пирамидальной сортировке.
- Использовать преимущества многопроцессорных машин.

3. ОПИСАНИЕ АСИНХРОННОГО АЛГОРИТМА

Можно разбить массив, представленный в виде бинарного дерева, на два поддерева.

Первое поддерево – это все дочерние элементы вершины по индексу 2, второе поддерево включает в себя все дочерние элементы вершины по индексу 3.

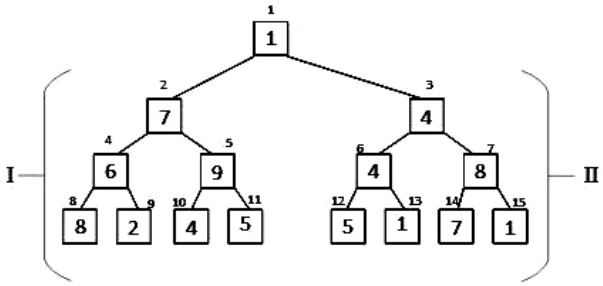

Рис. 2. Выделение поддеревьев

Теперь можно распараллелить поиск максимальных элементов. Запустим дополнительный поток для формирования свойств кучи в первом поддереве. Второе поддерево можно обрабатывать в основном потоке, по завершении этих операций останется лишь поставить максимальный элемент в верхушку пирамиды.

Ниже предоставлена схема алгоритма (рис. 3 и 4). Данный вид представления был выбран для того, чтобы алгоритм был независим от языка программирования. Алгоритм был основан на стандартной однопоточной пирамидальной сортировке [1].

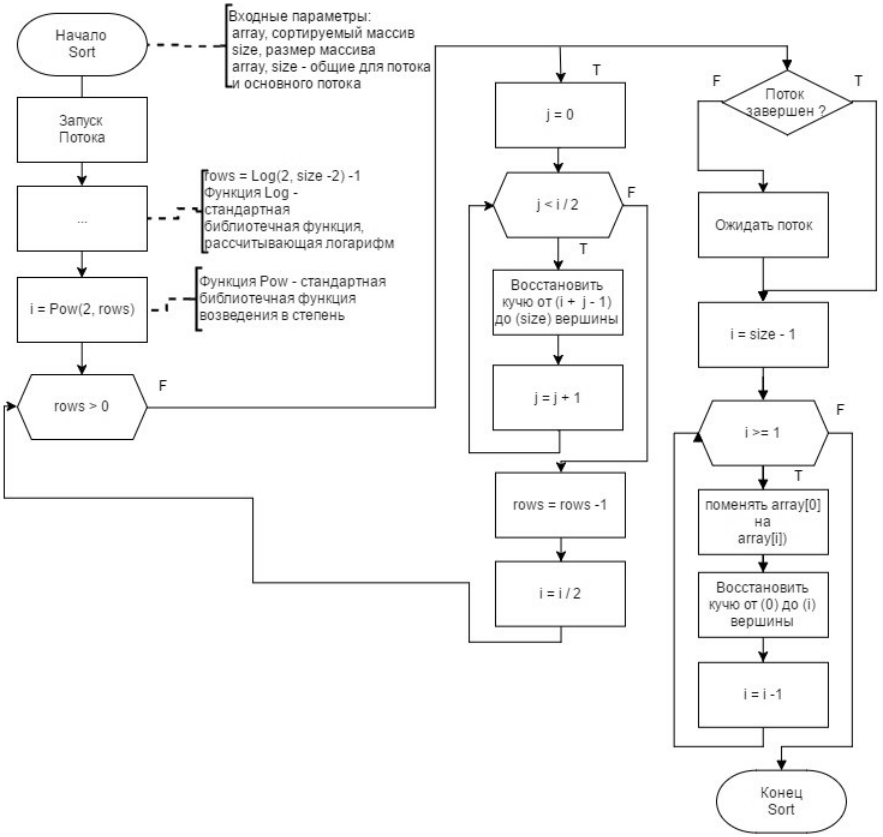

Рис. 3. Алгоритм сортировки

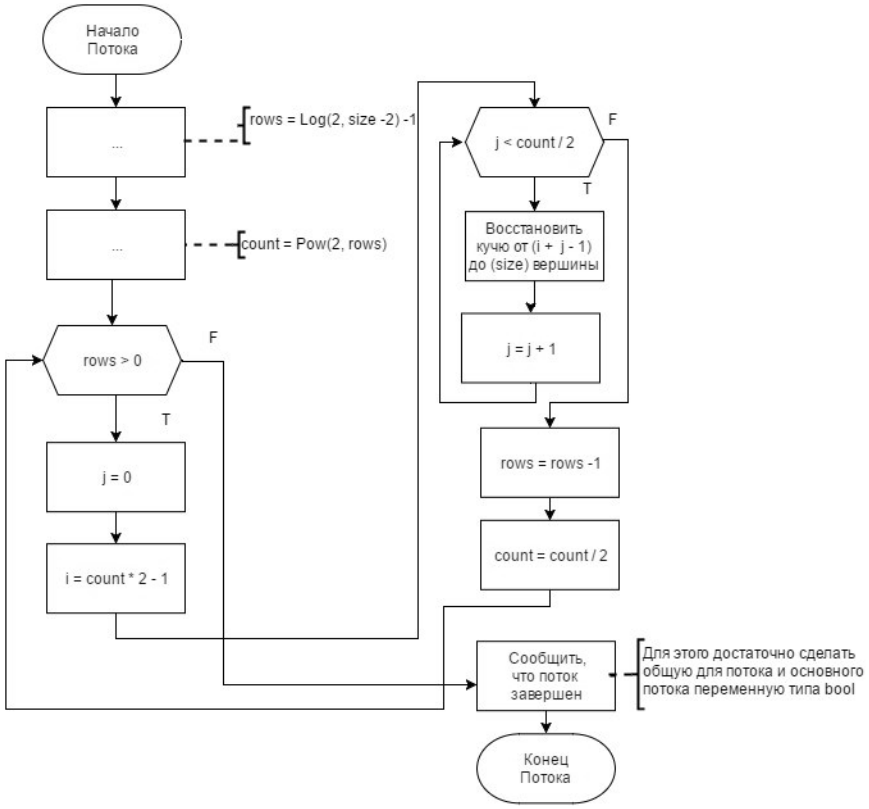

Рис. 4. Алгоритм для потока

4. ПРИМЕР ФОРМИРОВАНИЯ СОРТИРУЮЩЕГО ДЕРЕВА

Рассмотрим выполнение асинхронного алгоритма на примере массива: {1, 7, 4, 6, 9, 4, 8, 8, 2, 4, 5, 5, 1, 7, 1}.

Представим массив в виде двоичного дерева (рис. 5).

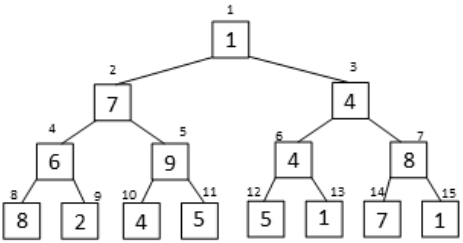

Рис. 5

Построим сортирующее дерево (левое и правые поддеревья сортируются разными потоками) (рис. 6).

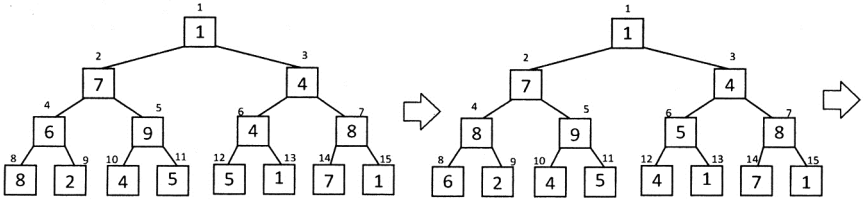

Рис. 6

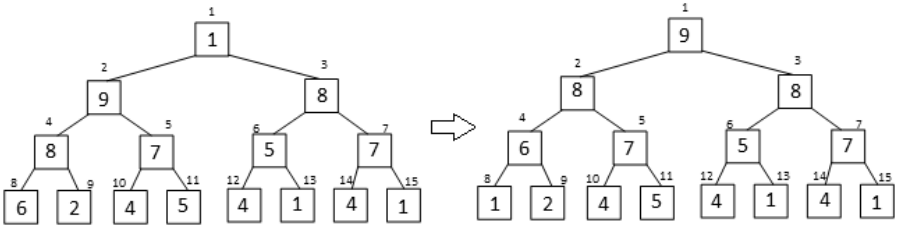

Рис. 7. Формирование сортирующего дерева

На рисунках показан способ эффективного использования многопоточности в сортировке.

БЛАГОДАРНОСТЬ

Авторы выражают искреннюю благодарность профессору кафедры автоматики А.А. Воеводе за помощь при выполнении работ, а также полезное обсуждение полученных результатов.

ЗАКЛЮЧЕНИЕ

В данной работе мы взглянули по-новому на классический метод сортировки *heapsort* и доказали, что существуют потокобезопасный способ распараллелить эту сортировку на этапе построения сортирующего дерева. Эту реализацию можно эффективно использовать на многопроцессорных машинах при обработке большого количества данных.

СПИСОК ЛИТЕРАТУРЫ

1. *Кормен Т., Лейзерсон Ч., Ривест Р.* Сортировка с помощью кучи // Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы. Построение и анализ: пер. с англ. – М.: МЦНМО, 2002. – Гл. 7. – С. 138–150.

2. *Левитин А.В*. Метод преобразования. Пирамиды и пирамидальная сортировка // Левитин А.В. Алгоритмы: введение в разработку и анализ: пер. с англ. – М.: Вильямс, 2006. – Гл. 6. – С. 275–284.

3. *Кнут Д.* Искусство программирования. Т. 3. Сортировка и поиск. – 2-е изд. – М.: Вильямс, 2007. – 824 с.

4. *Седжвик Р.* Фундаментальные алгоритмы на С. Ч. 1–4. Анализ. Структуры данных. Сортировка. Поиск. – СПб.: ДиаСофтЮП, 2003. – 672 с.

5. *Hetland M.L.* Python algorithms: mastering basic algorithms in the python language. – New York: Apress, 2010. – 336 p.

6. *Левитин А.В.* Метод грубой силы. Пузырьковая сортировка // Левитин А.В. Алгоритмы: введение в разработку и анализ. – М.: Вильямс, 2006. – Гл. 3. – С. 144–146.

7. *Sipser M.* Introduction to the theory of computation. – Boston: Thomson Course Technology, 2006. – 437 p.

8. *Kongmunvattana A.*, *Tzeng N.-F.* Logging and recovery in adaptive software distributed shared memory systems // Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems, SRDS'99, Lausanne, Switzerland, 1999. – Los Alamitos, CA, 1999. – doi: 10.1109/RELDIS.1999.805096.

9. Asynchronous programming – reference documentation. Version: 3.1.8 [Electronic resource] / G. Rocher, P. Ledbrook, M. Palmer, J. Brown, L. Daley, B. Beckwith, L. Hotari. – Available at: http://docs.grails.org/3.1.8/guide/async.html (accessed: 15.08.2016).

Иванов Никита Олегович, студент кафедры автоматики Новосибирского государственного технического университета по направлению «Информатика и вычислительная техника». Основное направление научных исследований – автоматизированное проектирование программных систем. E-mail: stenrulf@gmail.com

Тингайкин Денис Олегович, студент кафедры автоматики Новосибирского государственного технического университета по направлению «Информатика и вычислительная техника». Основное направление научных исследований – автоматизированное проектирование программных систем. E-mail: please.die.liet@gmail.com

Asynchronous HeapSort*

N.O. Ivanov[1], D.O. Tingajkin[2]

[1] *Novosibirsk State Technical University, 20 Karl Marks Avenue, Novosibirsk, 630073, Russian Federation, student of the automation department. E-mail: stenrulf@gmail.com*
[2] *Novosibirsk State Technical University, 20 Karl Marks Avenue, Novosibirsk, 630073, Russian Federation, student of the automation department. E-mail: please.die.liet@gmail.com*

Asynchrony – not simultaneous execution of various tasks. This term is used to describe a computer procedures that can be performed independently of the main stream. Flow – the smallest unit of processing, the execution of which can be assigned to the operating system kernel. Multithreading – platform property to create processes that can generate additional flows, the implementation of which does not depend on time. To synchronize threads using techniques such as the use of synchronization objects (the mutex), shared resources (semaphores), events (1bit event), critical sections, condition variables, completion ports, and so on...
Asynchronous programming – a term implying the creation of multi-threaded applications. Multi-core processor – a CPU with more than one processing core, which are located on a single processor crystals or housing. It is multi-core processors affect the performance of multithreaded applications, but to achieve the result required optimization. Optimization - the process of creating the best performance or maximum ratios, and minimize costs. In programming optimization – the process of writing the code as quickly as possible, that is, minimizing CPU cycles.

Keywords: sorting, asynchronous programming, HeapSort, parallel computing, multithreading optimization

DOI: 10.17212/2307-6879-2016-3-98-106

* *Received 24 August 2016.*

REFERENCES

1. Cormen T., Leiserson C., Rivest R. Sortirovka s pomoshch'yu kuchi [Heapsort]. Cormen T., Leiserson C., Rivest R. *Algoritmy. Postroenie i analiz [Introduction to algorithms]*. *Translated from English. Moscow,* MTsNMO Publ., 2002, ch. 7, pp. 138–150. (In Russian)

2. Levitin A.V. Metod preobrazovaniya. Piramidy i piramidal'naya sortirovka [Conversion method. Pyramids and peers-further sorting]. Levitin A.V. *Algoritmy: vvedenie v razrabotku i analiz* [Introduction to the design and analysis of algorithms]. *Translated from English.* Moscow, Williams Publ., 2006, ch. 6, pp. 275–284. (In Russian)

3. Knuth D.E. *The art of computer programming*. Vol. 3. Sorting and searching. Reading: Addison-Wesley, 1986 (Russ. ed.: Knut D. *Iskusstvo programmirovaniya*. T. 3. *Sortirovka i poisk*. 2nd ed. Moscow, Williams Publ., 2007. 824 p.).

4. Sedgewick R. *Algorithms in C*. Pt. 1–4. *Fundamentals. Data structures. Sorting. Searching.* 3rd ed. Boston, Addison-Wesley, 2002 (Russ. ed.: Sedzhvik R. *Fundamental'nye algoritmy na C*. Ch. 1–4. *Analiz. Struktury dannykh. Sortirovka. Poisk*. St. Petersburg, DiaSoftYuP Publ., 2003. 672 p.).

5. Hetland M.L. *Python algorithms: mastering basic algorithms in the python language*. New York, Apress, 2010. 336 p.

6. Levitin A.V. Metod gruboi sily. Puzyr'kovaya sortirovka [Brute force. Bubble sort]. Levitin A.V. *Algoritmy: vvedenie v razrabotku i analiz* [Introduction to the design & analysis of algorithms]. *Translated from English.* Moscow, Williams Publ., 2006, ch. 3, pp. 144–146. (In Russian)

7. Sipser M. *Introduction to the theory of computation*. Boston, Thomson Course Technology, 2006. 437 p.

8. Kongmunvattana A., Tzeng N.-F. Logging and recovery in adaptive software distributed shared memory systems. *Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems, SRDS'99*, Lausanne, Switzerland, 1999. doi: 10.1109/RELDIS.1999.805096

9. Rocher G., Ledbrook P., Palmer M., Brown J., Daley L., Beckwith B., Hotari L. *Asynchronous programming – reference documentation. Version: 3.1.8*. Available at: http://docs.grails.org/3.1.8/guide/async.html (accessed 15.08.2016)