УДК 004.67

БИТОНИЧЕСКАЯ СОРТИРОВКА КАК АЛЬТЕРНАТИВА СТАНДАРТНЫМ АЛГОРИТМАМ СОРТИРОВКИ

Мусонов В.М., Самуйлов С.В.

Пензенский государственный университет, Пенза, Россия

Аннотация. В статье делается попытка анализа алгоритмов сортировки стандартной библиотеки языка C++, известного алгоритма Шелла и алгоритма Bitonic, являющегося параллельным алгоритмом сортировки данных.

На текущий момент алгоритмы сортировки находят применение во многих областях человеческой деятельности. Во многих системах, созданных для обработки данных, применяются различные алгоритмы сортировки. Причем зачастую используются не самые оптимальные, но достаточно универсальные алгоритмы. Во многих программах применяются алгоритмы сортировок, написанные производителем операционной системы или языка, хранимые в стандартной библиотеке языка программирования. Подобные алгоритмы сортировки, встроенные, в стандартную библиотеку того или иного языка, являются универсальными, но не всегда оптимальными.

В статье будут рассмотрены три алгоритма сортировки: алгоритм сортировки стандартной библиотеки языка C++, алгоритм сортировки Шелла и алгоритм Bitonic. Реализация алгоритмов Шелла и Bitonic будет написана на языке C++ с использованием технологии Qt в среде macOS.

Точная реализация алгоритма из стандартной библиотеки языка C++ известна только разработчикам. Поэтому мы будем исследовать его как черный ящик, на вход которого подается неотсортированный массив, а на выходе – отсортированный. Останется только замерить время сортировки.

Сортировка Шелла. Является расширением и улучшением сортировки с помощью прямого включения [1]. Основная идея этого алгоритма заключается

в том, чтобы вначале устранить массовый беспорядок в массиве, сравнивая далеко стоящие друг от друга элементы. Далее интервал между сравниваемыми элементами постепенно уменьшается и, в конце концов, становится равным единице [2].

Формально суть метода заключается в следующем. Весь исходный массив S разбивается на группы с шагом h_1. В первую группу войдут элементы S_1, S_{h1+1}, S_{2h1+1}, ..., во вторую S_2, S_{h1+2}, S_{2h1+2}, ... и т.д. Затем каждая из h_1 групп сортируется методом прямого включения.

Далее описанный выше процесс повторяется для шага h_2 ($h_2<h_1$), h_3 ($h_3<h_2$) и т.д. Процесс заканчивается после того, как будет выполнена сортировка с шагом $h_t=1$.

Такой метод гарантированно дает упорядоченную последовательность элементов, так как на последнем шаге весь массив сортируется обычной сортировкой с помощью прямого включения.

Сложность алгоритма сортировки Шелла во многом зависит от выбора последовательности шагов h_1, h_2, …, h_t, и колеблется от $O(n^2)$ в худшем случае до $O(n^{7/6})$ в лучшем.

Сортировка Bitonic Sort. Это классический параллельный алгоритм сортировки. Он пригоден для реализации в аппаратном и параллельном массиве процессоров.

В основе данного алгоритма лежит понятие битонической последовательности [3]. Последовательность называется битонической, если она сначала увеличивается, а затем уменьшается. Другими словами, последовательность x_0, x_1, …, x_n является битонической, если существует индекс i ($0 <= i <= n$), такой, что

$$x_0 <= x_1 \ldots .. <= x_i \text{ и } x_i > = x_{i+1} \ldots ..> = x_n$$

Битоническая последовательность формируется следующим образом. Сначала формируется битоническая последовательность из четырех элементов, используя две последовательности из двух элементов. Рассмотрим

последовательность x_0, x_1, x_2, x_3. Сортируем x_0 и x_1 по возрастанию, x_2 и x_3 по убыванию. Объединяя эти две пары, формируем битоническую последовательность из четырех элементов.

Затем берутся две битонические последовательности из четырех элементов, первая сортируется в порядке возрастания, вторая – в порядке убывания и так далее, каждый раз удваивая длину последовательностей, пока не получим единую битоническую последовательность, первая половина которой отсортирована по возрастанию, вторая – по убыванию.

На втором этапе в полученной единой битонической последовательности сравниваем первый элемент первой половины с первым элементом второй половины, затем второй элемент первой половины со вторым элементом второй и т.д., выполняя перестановку пар элементов каждый раз, когда первый элемент больше второго. Т.е. сравнение пар происходит с шагом n/2.

Повторяем этот процесс с шагом n/4, n/8, … 2. В результате получает отсортированную последовательность элементов.

Анализ сложности алгоритма Bitonic. Чтобы сформировать отсортированную последовательность длины n из двух отсортированных последовательностей длины n/2 требуется log(n) сравнений. Следовательно, количество сравнений T(n) определяется по формуле:

$$T(n) = \log(n) + T(n/2)$$

Решение этого рекуррентного уравнения приводит к следующей формуле

$$T(n) = \log(n) + \log(n)\text{-}1 + \log(n)\text{ -}2 + \ldots + 1 = \log(n) \cdot (\log(n) + 1) / 2$$

В итоге сложность рассматриваемой сортировки – $O(n \log^2(n))$.

Рассмотренные выше алгоритмы сортировки были реализованы на языке C++, выполнено их сравнение (рисунок 1) и анализ.

Представленный на рисунке график является сводным графиком по всем перечисленным в данной работе алгоритмам. По оси OX представлено количество сортируемых данных, по оси OY – время, затраченное на выполнение сортировки.

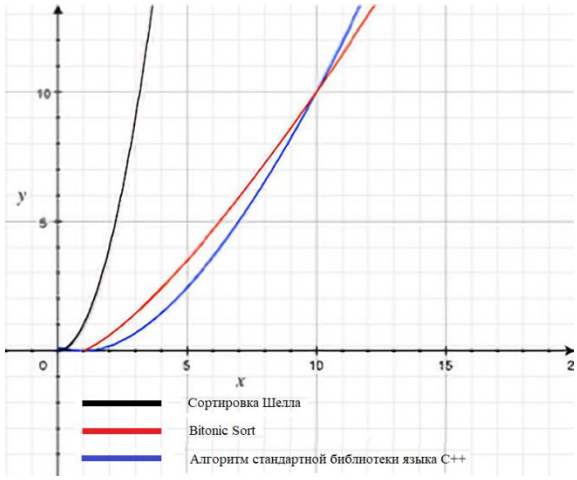

Рисунок 1 – Сравнение алгоритмов сортировки

Из графика видно, что наихудшую скорость работы демонстрирует алгоритм сортировки Шелла.

При детальном рассмотрении можно видеть, что при небольшом количестве входных данных Bitonic Sort работает хуже стандартной сортировки, но при росте количества сортируемых данных стандартный алгоритм языка C++ начинает ему проигрывать.

Следовательно, алгоритм битонической сортировки является эффективным по времени при сортировке больших объемов данных.

Список цитируемой литературы:

1. Самуйлов С.В., Фрунзе П.М. Методика анализа алгоритмов обработки данных. В сборнике: Новые информационные технологии и системы. Материалы XV Международной научно-технической конференции, посвященной 75-летию Пензенского государственного университета . 2018. С. 221-222.

2. Самуйлов С.В. Методика сравнительного анализа алгоритмов на примере алгоритмов последовательного поиска. Научно-методический электронный журнал Концепт. 2014. № 9. С. 46-50.

3. https://www.geeksforgeeks.org/bitonic-sort/(эл ресурс)