Санкт-Петербургский

Государственный Университет

Факультет прикладной математики – процессов управления

Задание по эмпирическому анализу

Алгоритм поразрядной MSD-сортировки.

Студент

Воробьёв Владислав Валерьевич

Группа 20.Б12-пу

г. Санкт-Петербург

4 декабря 2022 г.

Содержание

Краткая история

Поразрядная сортировка, возникшая к 1857 году, когда Герман Холлерит работал над вычислительными машинами. Это алгоритм, который использовался как способ сортировки перфокарт.

Первый компьютерный алгоритм с эффективным использованием памяти для этого метода сортировки был разработан в 1954 году в Массачусетском технологическом институте Гарольдом Х.Сюардом. Компьютеризированные сортировки по основанию ранее считались непрактичными из-за предполагаемой необходимости переменного распределения сегментов неизвестного размера. Новшество Сьюарда заключалось в использовании линейного сканирования для предварительного определения требуемых размеров сегментов и смещений, что позволило выполнить одно статическое выделение вспомогательной памяти.

Поразрядная сортировка может быть реализована таким образом, чтобы она начиналась либо по старшей цифре (MSD), либо по младшей цифре (LSD). Например, 1234 можно начать с 1 (MSD) или 4 (LSD).

При LSD сортировке обычно использует следующий порядок сортировки: короткие ключи идут перед более длинными ключами, а затем ключи той же длины сортируются лексикографически. Это совпадает с нормальным порядком целочисленных представлений, таких как последовательность [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11].

MSD сортировка больше всего подходит для сортировки строк или целочисленных представлений фиксированной длины. Последовательность вроде [b, c, e, d, f, g, ba] будет отсортирована как [b, ba, c, d, e, f, g].

Для сортировки по очередному разряду может применяться корзинная сортировка.

Помимо порядка обхода, сортировки MSD и LSD отличаются обработкой входных данных переменной длины. Сортировки LSD могут группироваться по длине, сортировать каждую группу по основанию, а затем объединять группы в порядке их размера. Сортировки MSD должны эффективно «расширять» все более короткие ключи до размера наибольшего ключа и соответственно сортировать их, что может быть сложнее, чем группировка, требуемая LSD.

Однако сортировки MSD более поддаются подразделению и рекурсии. Каждая корзина, созданная на шаге MSD, сама по себе может быть отсортирована по системе счисления с использованием следующего старшего разряда без ссылки на какие-либо другие корзины, созданные на

предыдущем шаге. Как только будет достигнута последняя цифра, объединение сегментов — это все, что требуется для завершения сортировки.

Область применения

Поразрядная сортировка применятся к данным, которые можно отсортировать лексикографически, например, к словам и целым числам. Также используется для стабильной сортировки строк. Это хороший вариант, когда алгоритм работает на параллельных машинах, что ускоряет сортировку.

Математический анализ алгоритма

Пусть у всех элементов одинаковое число разрядов. Если не одинаковое количество, то положим на более старших разрядах элементы с самым маленьким значением, например, для числовых значений это 0. Исходный массив делится на k частей(корзин). В первую корзину попадают элементы, у которых старший разряд с номером d=0 со значением 0. Во вторую корзину попадают элементы, у которых старший разряд с номером d=0 имеет значение 1. Элементы, которые попали в разные корзины, рекурсивно разделяются по следующему разряду с номером d=1, подход продолжается, пока не будут перебраны все разряды и размер корзины больше 1. Остановимся когда $d > m, left \geq right$, где m – максимальное число разрядов, $left$ и $right$ – левая и правая границы массива. Для подсчета количества элементов с различными значениями в сортируемом разряде, необходимы счетчики, которые фиксируются в массиве (count). Счетчики нужны для вычисления размеров корзин и определения границ разделения массива. Сортируемые объекты переносятся в массив **c**, в которым размещены корзины. Как корзины сформировались, содержимое из массива **c** переносится в исходный массив и выполняется рекурсивное разделение новых частей по следующему разряду.

Сложность Алгоритма

Если количество разрядов – k, а значения разрядов меньше b. То при сортировке массива из одинаковых элементов на каждом шаге элементы будут находится в неубывающей по размеру корзине, то время работы оценивается как $O(nk)$, притом что это время нельзя улучшить. Лучшем случаем для сортировки будет массив, при котором на каждом шаге каждая корзина будет делиться на b частей. Если размер корзины равен 1, сортировка перестанет рекурсивно запускаться в этой корзине. Асимптотика будет: $\Omega(n\ \log_b n)$. Функция трудоемкости данного алгоритма зависит не только от размера входных данных, но и от их значений, поэтому данный

алгоритм относится к классу количественно-параметрических по трудоемкости алгоритмов.

В итоге временная сложность в наилучшем случае равна ***O(N)***, а временная сложность в наихудшем случае равна ***O(N*M)*** , где M = средняя длина строк.

Характеристика входных данных

Входные данные в данной реализации представляют из себя массив чисел. Числа могут представляться разной длины или повторяющейся длины, также возможны числа с повторяющимися цифрами в разрядах.

Способ генерации входных данных

Генерация данных происходит в функции «`gen_random_generator`»

Программная реализация алгоритма

Весь программный код и генератор, может быть найден по ссылке:

https://github.com/VorobyovVV/radix_msd_sort

Вычислительный эксперимент и Анализ полученных результатов

График №1 отображает линейную зависимость (возможны погрешности вычисления), где количество элементов m=10 и зависимость от n:

График №1:

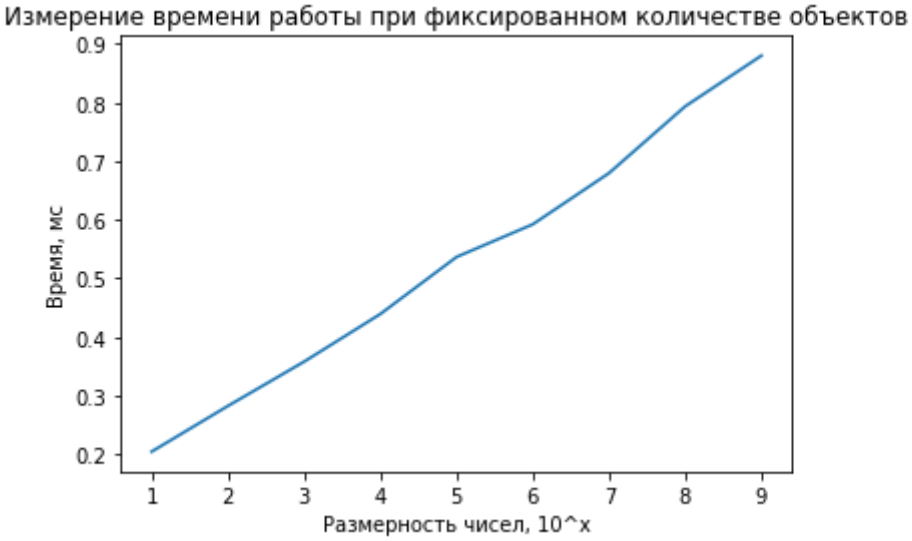
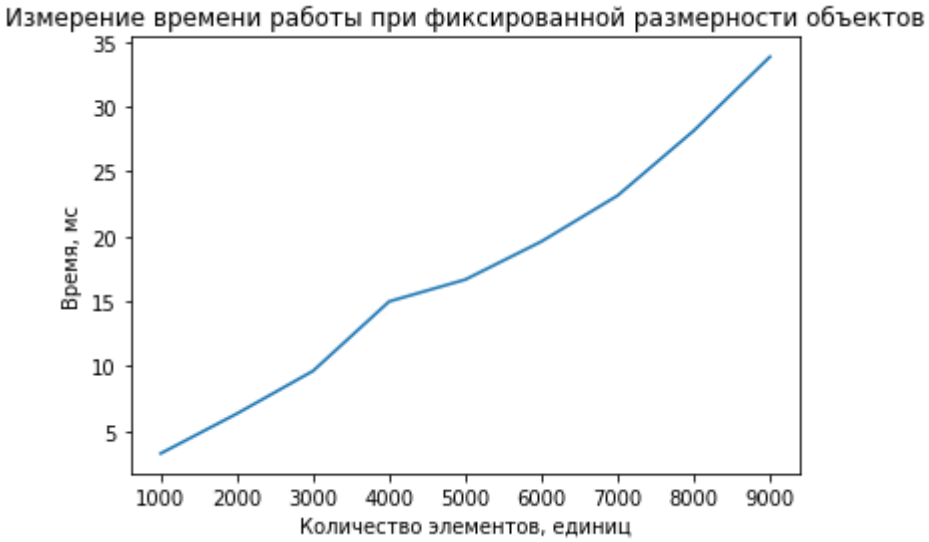


График №2 отображает линейную зависимость (возможны погрешности вычисления), где количество элементов n=1000 и зависимость от m:

График №2:



Характеристики использованной вычислительной среды и оборудования

Исследование проводилось в веб-интерактивной вычислительной среде **Jupyter Notebook.**

-Процессор: Intel(R) Core(TM) i5-3210M CPU @2.50GHz 2.50 Ghz

-Оперативная память: 8.00 ГБ

-Тип системы: 64-разрядная операционная система, процессор x64

Список использованной литературы

1.Т.Кормен, Ч.Лейзерсон, Р.Ривест, К.Штайн. Алгоритмы: построение и анализ.

2. Роберт Седжвик. Фундаментальные алгоритмы на C++.

3. Кнут Д.Э. – Искусство программирования.

4. Никлаус Вирт. Алгоритмы и структуры данных.