Алгоритмы упорядочивания. Задача упорядочивания, и её место в программных комплексах обработки данных (СУБД). Разнообразие алгоритмов упорядочивания

Алгоритм сортировки — это алгоритм для упорядочения элементов в списке. В случае, когда элемент списка имеет несколько полей, поле, служащее критерием порядка, называется ключом сортировки.

На практике в качестве ключа часто выступает число, а в остальных полях хранятся какие-либо данные, никак не влияющие на работу алгоритма.

Оценка алгоритма сортировки

Алгоритмы сортировки оцениваются по скорости выполнения и эффективности использования памяти:

Время — основной параметр, характеризующий быстродействие алгоритма. Называется также вычислительной сложностью. Для упорядочения важны худшее, среднее и лучшее поведение алгоритма в терминах мощности входного множества A. Если на вход алгоритму подаётся множество A, то обозначим . Для типичного алгоритма хорошее поведение — это и плохое поведение — это . Идеальное поведение для упорядочения — . Алгоритмы сортировки, использующие только абстрактную операцию сравнения ключей всегда нуждаются, по меньшей мере, в сравнениях. Тем не менее, существует алгоритм сортировки Хана (Yijie Han) с вычислительной сложностью , использующий тот факт, что пространство ключей ограничено (он чрезвычайно сложен, а за O - обозначением скрывается весьма большой коэффициент, что делает невозможным его применение в повседневной практике). Также существует понятие сортирующих сетей. Предполагая, что можно одновременно (например, при параллельном

вычислении) проводить несколько сравнений, можно отсортировать n чисел за операций. При этом число n должно быть заранее известно;

Память — ряд алгоритмов требует выделения дополнительной памяти под временное хранение данных. Как правило, эти алгоритмы требуют памяти. При оценке не учитывается место, которое занимает исходный массив и независящие от входной последовательности затраты, например, на хранение кода программы (так как всё это потребляет). Алгоритмы сортировки, не потребляющие дополнительной памяти, относят к сортировкам на месте.

Классификация алгоритмов сортировки

- Устойчивость (stability) — устойчивая сортировка не меняет взаимного расположения равных элементов.
- Естественность поведения — эффективность метода при обработке уже упорядоченных, или частично упорядоченных данных. Алгоритм ведёт себя естественно, если учитывает эту характеристику входной последовательности и работает лучше.

Использование операции сравнения. Алгоритмы, использующие для сортировки сравнение элементов между собой, называются основанными на сравнениях. Минимальная трудоемкость худшего случая для этих алгоритмов составляет , но они отличаются гибкостью применения. Для специальных случаев (типов данных) существуют более эффективные алгоритмы.

Ещё одним важным свойством алгоритма является его сфера применения. Здесь основных типов упорядочения два:

- Внутренняя сортировка оперирует с массивами, целиком помещающимися в оперативной памяти с произвольным доступом к любой ячейке. Данные обычно упорядочиваются на том же месте, без дополнительных затрат.

В современных архитектурах персональных компьютеров широко применяется подкачка и кэширование памяти. Алгоритм сортировки должен хорошо сочетаться с применяемыми алгоритмами кэширования и подкачки.

- Внешняя сортировка оперирует с запоминающими устройствами большого объёма, но с доступом не произвольным, а последовательным (упорядочение файлов), т. е. в данный момент мы 'видим' только один элемент, а затраты на перемотку по сравнению с памятью неоправданно велики. Это накладывает некоторые дополнительные ограничения на алгоритм и приводит к специальным методам упорядочения, обычно использующим дополнительное дисковое пространство. Кроме того, доступ к данным на носителе производится намного медленнее, чем операции с оперативной памятью.

Доступ к носителю осуществляется последовательным образом: в каждый момент времени можно считать или записать только элемент, следующий за текущим.

Объём данных не позволяет им разместиться в ОЗУ.

Также алгоритмы классифицируются по:

- потребности в дополнительной памяти или её отсутствии
- потребности в знаниях о структуре данных, выходящих за рамки операции сравнения, или отсутствии таковой

Список алгоритмов сортировки

Алгоритмы устойчивой сортировки

Сортировка пузырьком (англ. Bubble sort) — сложность алгоритма: ;

для каждой пары индексов производится обмен, если элементы расположены не по порядку.

Алгоритм считается учебным и практически не применяется вне учебной литературы, вместо него на практике применяется сортировка вставками.

Сортировка перемешиванием (Шейкерная, Cocktail sort, bidirectional bubble sort) — Сложность алгоритма:

Сортировка перемешиванием (англ. Cocktail sort) - разновидность пузырьковой сортировки. Отличается тем, что просмотры элементов выполняются один за другим в противоположных направлениях, при этом большие элементы стремятся к концу массива, а маленькие - к началу.

Лучший случай для этой сортировки - отсортированный массив , худший - отсортированный в обратном порядке .

Сортировка вставками (Insertion sort) — Сложность алгоритма: ; определяем где текущий элемент должен находиться в упорядоченном списке и вставляем его туда.

Сортировка вставками (англ. insertion sort) — простой алгоритм сортировки. Хотя этот метод сортировки намного менее эффективен, чем более сложные алгоритмы (такие как быстрая сортировка), у него есть ряд преимуществ:

- прост в реализации
- эффективен на небольших наборах данных, на наборах данных до десятков элементов может оказаться лучшим
- эффективен на наборах данных, которые уже частично отсортированы
- это устойчивый алгоритм сортировки (не меняет порядок элементов, которые уже отсортированы)
- может сортировать список по мере его получения
- не требует временной памяти, даже под стек

На каждом шаге алгоритма мы выбираем один из элементов входных данных и вставляем его на нужную позицию в уже отсортированном списке, до тех пор, пока набор входных данных не будет исчерпан. Метод выбора очередного элемента из исходного массива произволен; может использоваться практически любой алгоритм выбора.

Блочная сортировка (Корзинная сортировка, Bucket sort) — Сложность алгоритма: O(n); требуется O(k) дополнительной памяти и знание о природе сортируемых данных, выходящее за рамки функций "переставить" и "сравнить".

Блочная сортировка (Карманная сортировка, корзинная сортировка, англ. Bucket sort) — алгоритм сортировки, в котором сортируемые элементы делятся на конечное число отдельных блоков (корзин) так, что все элементы в одной корзине всегда больше (или меньше), чем в другой. Каждый блок затем сортируется отдельно, либо рекурсивно тем же методом, либо иным другим. Затем элементы помещаются обратно в массив. Этот тип сортировки может обладать линейным временем исполнения.

Данный алгоритм требует знаний о природе сортируемых данных, выходящих за рамки функций "сравнить" и "поменять местами", достаточных для сортировки слиянием, сортировки пирамидой, быстрой сортировки, сортировки Шелла, сортировки вставкой.

Преимущества: относится к классу быстрых алгоритмов с линейным временем исполнения O(N) (на удачных входных данных).

Недостатки: сильно деградирует при большом количестве мало отличных элементов, или же на неудачной функции получения номера корзины по содержимому элемента. В некоторых таких случаях для строк, возникающих в реализациях основанного на сортировке строк алгоритма сжатия BWT, оказывается, что быстрая сортировка строк в версии Седжвика значительно превосходит блочную сортировку скоростью.

Сортировка подсчётом (Counting sort) — Сложность алгоритма: ; требуется дополнительной памяти (рассмотрено 3 варианта)

Сортировка подсчётом — алгоритм сортировки, в котором используется диапазон чисел сортируемого массива (списка) для подсчёта совпадающих элементов. Алгоритм имеет преимущества, если нужно отсортировать много чисел из небольшого диапазона, например, миллион натуральных чисел меньших 1000.

Предположим, что входной массив состоит из n целых чисел в диапазоне от 0 до k − 1, где . Далее алгоритм будет обобщён для произвольного целочисленного диапазона. Существует несколько модификаций сортировки подсчётом, ниже рассмотрены три линейных и одна квадратичная.

Сортировка слиянием (Merge sort) — Сложность алгоритма: ; требуется O(n) дополнительной памяти; выстраиваем первую и вторую половину списка отдельно, а затем — сливаем упорядоченные списки

Сортировка слиянием (англ. merge sort) — алгоритм сортировки, который упорядочивает списки (или другие структуры данных, доступ к элементам которых можно получать только последовательно, например — потоки) в определённом порядке. Эта сортировка — хороший пример использования принципа «разделяй и властвуй». Сначала задача разбивается на несколько подзадач меньшего размера. Затем эти задачи решаются с помощью рекурсивного вызова или непосредственно, если их размер достаточно мал. Наконец, их решения комбинируются, и получается решение исходной задачи.

Для решения задачи сортировки эти три этапа выглядят так:

- Сортируемый массив разбивается на две половины примерно одинакового размера;

- Каждая из получившихся половин сортируется отдельно, например - тем же самым алгоритмом;
- Два упорядоченных массива половинного размера соединяются в один.

Рекурсивное разбиение задачи на меньшие происходит до тех пор, пока размер массива не достигнет единицы (любой массив длины 1 можно считать упорядоченным).

Нетривиальным этапом является соединение двух упорядоченных массивов в один. Основную идею слияния двух отсортированных массивов можно объяснить на следующем примере. Пусть мы имеем две стопки карт, лежащих рубашками вниз так, что в любой момент мы видим верхнюю карту в каждой из этих стопок. Пусть также, карты в каждой из этих стопок идут сверху вниз в неубывающем порядке. Как сделать из этих стопок одну? На каждом шаге мы берём меньшую из двух верхних карт и кладём её (рубашкой вверх) в результирующую стопку. Когда одна из оставшихся стопок становится пустой, мы добавляем все оставшиеся карты второй стопки к результирующей стопке.

Алгоритм был изобретён Джоном фон Нейманом в 1945 году.

In-place merge sort — Сложность алгоритма: , или в зависимости от применяемого алгоритма слияния.

Двоичное дерево сортировки (Binary tree sort) — Сложность алгоритма: ; требуется O(n) дополнительной памяти

Двоичное дерево поиска (binary search tree, BST) — это двоичное дерево, в котором данные, привязанные к каждому узлу, представляют собой пару (key, value) (ключ и значение), причём на ключах определена операция сравнения "меньше", и для всех узлов дерева выполнено свойство, называемое свойством дерева поиска:

у всех узлов левого поддерева произвольного узла n значение ключей меньше, нежели значения ключа узла n,

у всех узлов правого поддерева произвольного узла n значение ключей не меньше, нежели значения ключа узла n.

Двоичное дерево поиска является одной из возможных реализаций ассоциативного массива.

Цифровая сортировка (Сортировка по отделениям, Pigeonhole sort) — Сложность алгоритма: ; требуется дополнительной памяти

Цифровая сортировка (англ. pigeonhole sort) обладает линейной вычислительной сложностью (O(n)), что является лучшей возможной производительностью для алгоритма сортировки, так как в любом таком алгоритме каждый сортируемый элемент необходимо просмотреть хотя бы однажды. Однако, применение алгоритма цифровой сортировки целесообразно лишь тогда, когда сортируемые предметы имеют (или их можно отобразить в) диапазон возможных значений, который достаточно мал по сравнению с сортируемым списком. Эффективность алгоритма падает всякий раз, когда несколько различных элементов попадает в одну ячейку. Необходимость сортировки внутри ячеек лишает алгоритм смысла, так как каждый элемент придётся просматривать более одного раза. Так что, для простоты и с целью отличить «классическую» цифровую сортировку от её многочисленных вариантов, укажем, что подсчёт должен быть обратимым: если два элемента попадают в одну ячейку, то они должны иметь одинаковое значение. Несколько элементов с одним значением в одной ячейке не портят картину — их можно просто вставить в отсортированный список рядом, один за другим (это позволяет применять цифровую сортировку в качестве устойчивой).

Алгоритм цифровой сортировки действует следующим образом:

- Создаём массив изначально пустых «ячеек», по одной для каждой величины из диапазона ключей.
- Просматриваем изначальный массив, помещая каждый его элемент в свою ячейку.
- Проходим по массиву ячеек в нужном порядке и переносим элементы из непустых ячеек обратно в первоначальный массив.

Эффективность этого алгоритма неявно, но сильно зависит от плотности элементов в массиве ячеек. Если элементов этого массива намного больше, чем сортируемых предметов, то шаги 1 и 3 будут относительно медленными.

Сортировку подсчётом применяют редко, потому что её требования редко удовлетворяются, и часто бывает проще применить другие, более гибкие и почти такие же быстрые алгоритмы сортировки. В особенности, блочная сортировка является более практичным вариантом сортировки подсчётом. В некотором роде, быстрая сортировка представляет собой обобщённую сортировку подсчётом (всего с двумя ячейками в каждый момент времени).

Гномья сортировка (Gnome sort) — Сложность алгоритма:

Алгоритмы неустойчивой сортировки

Сортировка выбором (Selection sort) — Сложность алгоритма: ; поиск наименьшего или наибольшего элемента и помещения его в начало или конец упорядоченного списка

Сортировка Шелла (Shell sort) — Сложность алгоритма: ; попытка улучшить сортировку вставками

Сортировка расчёской (Comb sort) — Сложность алгоритма: .

Пирамидальная сортировка (Сортировка кучи, Heapsort) — Сложность

алгоритма: ; превращаем список в кучу, берём наибольший элемент и добавляем его в конец списка

Плавная сортировка (Smoothsort) — Сложность алгоритма:

Быстрая сортировка (Quicksort) — Сложность алгоритма: — среднее время, — худший случай; широко известен как быстрейший из известных для упорядочения больших случайных списков; с разбиением исходного набора данных на две половины так, что любой элемент первой половины упорядочен относительно любого элемента второй половины; затем алгоритм применяется рекурсивно к каждой половине

Introsort — Сложность алгоритма: , сочетание быстрой и пирамидальной сортировки. Пирамидальная сортировка применяется в случае, если глубина рекурсии превышает .

Patience sorting — Сложность алгоритма: — наихудший случай, требует дополнительно памяти, также находит самую длинную увеличивающуюся подпоследовательность

Обменная поразрядная сортировка — Сложность алгоритма: ; требуется дополнительной памяти

Непрактичные алгоритмы сортировки

Bogosort — в среднем. Произвольно перемешать массив, проверить порядок.

Сортировка перестановкой — — худшее время. Генерируются всевозможные перестановки исходного массива и для каждой осуществляется проверка верного порядка.

Глупая сортировка (Stupid sort) — ; рекурсивная версия требует дополнительно памяти

Bead Sort — or , требуется специализированное аппаратное обеспечение

Блинная сортировка (Pancake sorting) — , требуется специализированное аппаратное обеспечение

Алгоритмы, не основанные на сравнениях

- Блочная сортировка (Корзинная сортировка, Bucket sort)
- Лексикографическая или поразрядная сортировка (Radix sort)
- Сортировка подсчётом (Counting sort)
- Остальные алгоритмы сортировки
- Топологическая сортировка
- Внешняя сортировка