Санкт-Петербургский Национальный Исследовательский Университет

Информационных Технологий, Механики и Оптики

Факультет инфокоммуникационных технологий

**Лабораторная работа №1**

Выполнили:

Мануковская Д. М.

Сакулин И. М.

Сафронов И. С.

Проверил:

Мусаев А. А.

Санкт-Петербург,

2024

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc179747370)

[Задание 1 4](#_Toc179747371)

[Блочная сортировка 9](#_Toc179747372)

[Пирамидальная сортировка 11](#_Toc179747373)

[Сортировка слиянием 12](#_Toc179747374)

[ЗАКЛЮЧЕНИЕ 14](#_Toc179747375)

[СПИСОК ЛИТЕРАТУРЫ 15](#_Toc179747376)

ВВЕДЕНИЕ

В данной лабораторной работе необходимо написать программы для 5ти различных видов сортировки: быстрой, расчески, блочной, пирамидальной, слиянием. Оценить их достоинства, недостатки и сложность.

Цель работы: знакомство алгоритмами сортировок.

Задание 1

Для тестирования алгоритмов были созданы вспомогательные функции (рисунок 1). Первая – «generate\_lst» принимает границы диапазона и длину, а возвращает список случайных чисел заданной длины. Генерация случайных значений происходит с помощью функции «randint» модуля «random». По умолчанию заданы маленькие параметры для тестов.

Вторая функция «test». На вход к ней подаётся объект функции тестируемой сортировки, а также, опционально, сортируемый список. Если он не задан, то генерируется автоматически. Список сортируется с помощью переданной функции и с помощью встроенной в python «sorted», затем результаты сравниваются. После в терминал выводятся: название алгоритма, корректность его выполнения, изначальный, отсортированный через «sorted» и через переданную функцию сортировки списки.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 1 – Генерация списков и тестирование

Во второй части задания необходимо протестировать время исполнения алгоритмов через модуль «timeit». Так как тестирующая функция принимает на вход строку с кодом, для удобства была написан отдельный код, он тестирует все переданные функции через случайные списки, заданного размера и выводит время в терминал (рисунок 2).

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 2 – Тесты с помощью timeit

Быстрая сортировка является одной из самых быстрых сортировок, применяемых на практике, она относится к типу «разделяй и властвуй», когда мы делим большую задачу на маленькие подзадачи, тем самым решая большую задачу. Видоизменённые версии такой сортировки можно обнаружить во многих языках программирования в качестве встроенной.

Основной принцип её работы такой:

1. Проверяется базовый случай, когда список пустой или содержит всего 1 элемент;
2. Поиск индекса опорного элемента;
3. Все элементы делятся на 3 части: меньшие, равные и большие опорному;
4. Элементы меньшие и большие опорного сортируются рекурсивно;
5. Возвращается список, составленный из отсортированных элементов меньше опорного, равных и больше опорного.

Выбор опорного элемента – важный шаг, от которого зависит эффективность алгоритма. Оптимальным может оказаться любой элемент с равной вероятностью. Можно взять медианный от нескольких выбранных элементов, или просто по какому-то индексу, или первый, или применить прочие специальные методы. Если, например, брать первый при отсортированном массиве или такой, что список делиться на пустой и из всех остальных, то сложность достигнет максимальной O(n2).

Сложность:

Средней и оптимальной сложностью является O(n\*logn), потому что алгоритм делит на 2 части список log(n) итераций и на каждом из уровней рекурсии итерируется по n элементам.

В самом несбалансированном варианте каждое разделение даёт два подмассива размерами 1 и n-1, то есть при каждом рекурсивном вызове больший массив будет на 1 короче, чем в предыдущий раз [1]. Сложность в таком случае O(n2).

Достоинства:

* Самая быстрая на практике
* Простой принцип действия
* Допускается естественное распараллеливание процессов

Недостатки:

* При неудачных вводных данных сильно падает по скорости
* Зависимость от выбора опорного элемента

Реализация представлена на рисунке 3.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 3 – Быстрая сортировка

Сортировка расчёской – алгоритм, основанный на сравнении двух элементов и смене их мест. Основная идея в том, чтобы первоначально брать достаточно большое расстояние между сравниваемыми элементами и затем сужать это расстояние вплоть до 1.

Алгоритм делает следующие шаги:

1. Берёт максимальный шаг – расстояние между сравниваемыми элементами – как размер списка
2. Сокращает шаг на фактор уменьшения, оптимальная величина которого примерно 1.25
3. Сравнивает все элементы, находящиеся на расстоянии шага, если больший стоит раньше меньшего, меняет их местами
4. Если изменений за итерацию не было, заканчивает сортировку, иначе переходит к шагу 2.

Сложность:

В худшем случае сортировка произойдёт за O(n^2) операций, в лучшем за O(n\*logn), в среднем же она O(n^2/2^p).

Преимущества:

* простой код;
* стабильная сортировка [2].

Недостатки:

* медленней, чем алгоритмы со средней сложностью O(n\*logn)

На рисунке 4 представлена реализация этого алгоритма.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 4 – Сортировка расчёской

Быстрая сортировка и сортировка расчёской были отправлены в timeit, со следующими характеристиками (рисунок 5). Параметр repeat отвечает за количество повторений: чем больше это значение, тем дольше исполняются тесты, но значения сильнее усредняются. Второй аргумент length отвечает за размер входных данных – количество элементов в сортируемом списке.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 5 – Запуск тестов

В результате работы тестов были получены замеры времени выполнения (рисунок 6). На основе этих данных можно сделать выводы о том, что на малых данных скорости разняться не очень сильно, а на вот на больших массивах, разница ощутима заметна. Это были ожидаемые результаты, ведь оценка сложности прогнозировала то же самое.

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

Рисунок 6 – Результаты тестов

Блочная сортировка

Оно же имеет названия: Вики-сортировка: Wiki-sort, Сортировка святого Грааля: Grailsort

Блочная сортировка — это алгоритм сортировки , который сортирует массив путем разделения его на блоки фиксированного размера, сортировки каждого блока по отдельности, а затем объединения отсортированных блоков обратно в единый отсортированный массив. Блочная сортировка - хороший выбор для сортировки больших наборов данных, которые не помещаются в памяти. Он может эффективно сортировать данные по блокам, которые умещаются в памяти, а затем объединять отсортированные блоки вместе для получения окончательного отсортированного массива.

Временная сложность: O (n \* logn).

Подход к блочной сортировке заключается в следующем:

* Разделите входной массив на блоки фиксированного размера.
* Сортируйте каждый блок, используя алгоритм сортировки на основе сравнения (например, быструю сортировку или сортировку слиянием).
* Объедините отсортированные блоки обратно в единый отсортированный массив, используя очередь с приоритетом или минимальную кучу.

Преимущества блочной сортировки:

* Блочная сортировка имеет относительно хорошую временную сложность в наихудшем случае - O (n \* logn).
* Блочную сортировку можно эффективно распараллелить, параллельно сортируя каждый блок по отдельности.
* Блочная сортировка позволяет обрабатывать большие наборы данных, которые не умещаются в памяти, путем сортировки данных по блокам, которые умещаются в памяти.

Недостатки блочной сортировки:

* Блочная сортировка сопряжена с относительно высокими накладными расходами из-за необходимости разделить входные данные на блоки, а затем объединить отсортированные блоки вместе.
* Выбор размера блока может повлиять на производительность сортировки блоков. Если размер блока слишком мал, для сортировки потребуется больше блоков, что увеличивает накладные расходы. Если размер блока слишком велик, отдельные блоки могут не поместиться в памяти, что снизит эффективность блочной сортировки.

Изображение выглядит как текст, программное обеспечение, число, Шрифт

Автоматически созданное описание

Рисунок 7 – Блочная сортировка

Был реализован другой вариант блочной сортировки.

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Рисунок 8 – 2-й вариант блочной сортировки

Пирамидальная сортировка

Также известна как сортировка кучей. Этот популярный алгоритм, как и сортировки вставками или выборкой, сегментирует список на две части: отсортированную и неотсортированную. Алгоритм преобразует второй сегмент списка в структуру данных «куча» (heap), чтобы можно было эффективно определить самый большой элемент.

Пирамидальная сортировка включает следующие шаги:

1. Изначально, исходный список преобразуется в кучу. Куча представляет собой бинарное дерево, где каждый узел имеет значение, меньшее или равное значению его дочерних узлов.
2. После построения кучи наибольший элемент (корень кучи) вынимается и помещается в конец списка. Затем происходит восстановление свойств кучи для оставшихся элементов. Этот шаг повторяется до тех пор, пока вся куча не будет отсортирована.

Рассмотрим достоинства и недостатки пирамидальной сортировки.

Достоинства:

* Имеет доказанную оценку худшего случая.
* Требует всего O(1) дополнительной памяти.

Недостатки:

* Сложен в реализации.
* Неустойчив -- для обеспечения устойчивости нужно расширять ключ.
* На почти отсортированных массивах работает столь же долго, как и на хаотических данных.
* На одном шаге выборку приходится делать хаотично по всей длине массива – поэтому алгоритм плохо сочетается с кэшированием и подкачкой памяти.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 9 - Пирамидальная сортировка

Сортировка слиянием

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.

Этапы выполнения сортировки:

1. Массив разбивается на 2 примерно одинаковых меньшего размера.
2. Рекурсией массив разбивается на еще более маленькие, пока новый не будет состоять из 1 или менее элементов.
3. 2 массива поэлементно сравниваются между собой, в результате создается новый, который по сути является отсортированным слиянием этих двух исходных.
4. Таким образом, возвращаясь по рекурсии и сравнивая части, получаем отсортированный массив.

Пример реализации этого вида сортировки представлен на рисунке 9.

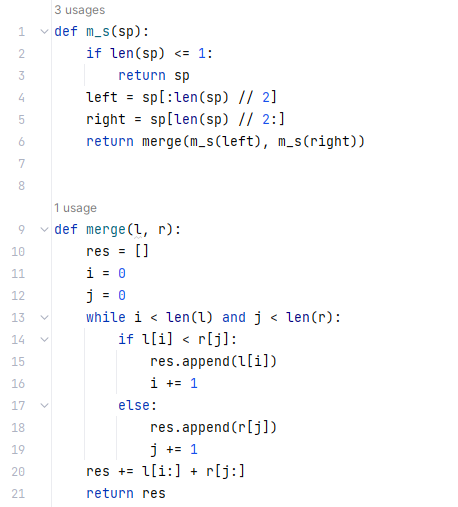


Рисунок 10 – Сортировка слиянием

Преимущества:

* Стабильное время выполнение. Сложность алгоритма O(nlogn).
* Быстрее для больших массивов в сравнении с сортировками вставки, пузырьком.
* Подходит для труднодоступных данных.
* Можно использовать для больших массивов.

Недостатки:

* Медленнее, чем быстрая сортировка.
* Требует пространство для временных значений равное исходному массиву.

ЗАКЛЮЧЕНИЕ

Цель работы достигнута. В ходе работы научились реализовать алгоритмы сортировок. Были созданы 6 программ для различных видов сортировок: быстрой, расчески, блочной, пирамидальной, слиянием. Проанализированы их недостатки, преимущества и сложность.

СПИСОК ЛИТЕРАТУРЫ

1. Википедия. Быстрая сортировка. [Электронный ресурс] – <https://clck.ru/3DupPa>
2. Википедия. Сортировка слиянием [Электронный ресурс] – <https://clck.ru/3DunZE>
3. Ряскова К.А. Исследование и классификация метода сортировки расческой [Электронный ресурс] / Ряскова К.А. – Режим доступа: <https://sci-article.ru/stat.php?i=1555662521>