МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Национальный исследовательский
Нижегородский государственный университет им. Н.И. Лобачевского»
(ННГУ)**
Институт информационных технологий, математики и механики

Направление подготовки: «Прикладная математика и информатика»

ОТЧЕТ

по лабораторной работе

Тема:

«Сортировка массивов разными способами»

Выполнил:

студент группы 3824Б1ПМ4
Торсеев Н.Е

_____
подпись

Преподаватель:
Куклин А.Е.

_____
подпись

Нижний Новгород
2024

Содержание:

Введение

Сортировка — одна из фундаментальных задач в программировании. Она подразумевает упорядочивание элементов в массиве или списке по определённому критерию, например, по возрастанию или убыванию.
Эффективные алгоритмы сортировки используются во многих приложениях, таких как базы данных, поисковые системы и системы обработки данных. Например, в интернет-магазинах сортировка товаров по цене, рейтингу или популярности делает выбор подходящего товара более удобным для пользователей.
В этом отчете рассматриваются три популярных алгоритма сортировки:

1. **Сортировка выбором (Selection Sort)**
2. **Сортировка вставками (Insertion Sort)**
3. **Сортировка пузырьком (Bubble Sort)**

Постановка задачи

Задача состояла в создании программы, которая сначало создает массив с генерацией в нем случайных чисел. А потом сортирует его тремя способами: сортировкой выбором, сортировкой вставками, сортировкой пузырьком. Далее программа должна вывести время за которое она отсортировала массив этими способами, и я должен выяснить какой метод сортировки является лучшим для определенного размера массива.

Описание алгоритмов

1. **Сортировка выбором (Selection Sort)**

Описание:

Сортировка выбором — это алгоритм сортировки, который находит наименьший элемент из неотсортированной части массива и меняет его местами с первым элементом этой части. Процесс повторяется для оставшихся элементов, пока весь массив не будет отсортирован.
Суть сортировки выбором в том, что алгоритм сравнивает каждый элемент с каждым и в случае необходимости производит обмен, приводя последовательность к необходимому упорядоченному виду.

Принцип работы:

1. Массив делится на две части: отсортированную и неотсортированную. Изначально весь список считается несортированным.
2. Начиная с первого элемента в неотсортированной части, определяется минимальный элемент из этой части массива и помещается в текущую позицию.
3. То же самое делается для остальных элементов в неотсортированной части, один за другим постепенно увеличивая отсортированную часть, пока не будет отсортирован весь массив.

2. **Сортировка вставками (Insertion Sort)**

Описание:

Сортировка вставками — это алгоритм сортировки, в котором элементы входной последовательности просматриваются по одному, и каждый новый поступивший элемент размещается в подходящее место среди ранее упорядоченных элементов.
При сортировке вставками массив постепенно перебирается слева направо. При этом каждый последующий элемент размещается так, чтобы он оказался между ближайшими элементами с минимальным и максимальным значением.

Принцип работы:

1. Массив делят на две части — отсортированную и неотсортированную.
2. Из неотсортированной части извлекают любой элемент.
3. Сравнивают его со значениями в отсортированном подмассиве справа налево, пока не определят подходящую позицию (то есть, в тот момент, когда найдут первое число, которое меньше, чем извлекаемый элемент).
4. Затем сдвигают все отсортированные элементы, которые находятся справа от этого числа вправо, чтобы образовалось место для элемента, и вставляют его туда, тем самым расширяя отсортированную часть массива.

3. **Сортировка пузырьком (Bubble Sort)**

Описание:

Сортировка пузырьком — это метод сортировки массивов и списков путём последовательного сравнения соседних элементов и их обмена, если предшествующий оказывается больше последующего (при сортировке по возрастанию).

Принцип работы:

1. Начинаем с первого элемента массива.
2. Сравниваем текущий элемент со следующим.
3. Если текущий элемент больше следующего, меняем их местами.
4. Переходим к следующему элементу и повторяем шаги 2-3.
5. После завершения прохода по массиву, повторяем процесс, пока не будет выполнен полный проход без изменений.

Описание программной реализации

В данной программе реализованы три алгоритма сортировки: сортировка выбором, сортировка вставками и сортировка пузырьком. Программа позволяет пользователю выбрать длину изначального массива, который требует сортировки. Ниже представлено подробное описание каждой части программы.

1. **Подключение библиотек**

• **stdio.h**: Библиотека для ввода и вывода данных.
• **stdlib.h**: Библиотека для работы с памятью и генерации случайных чисел.

• **time.h**: Библиотека для работы с временем, используется для измерения времени выполнения сортировок.
• **malloc.h**: Библиотека для использования функций динамического распределения памяти.

2. Алгоритмы сортировки

• **Сортировка выбором (search_sort)**:

(На каждой итерации находит наименьший элемент в неотсортированной части массива и перемещает его в начало отсортированной части)

• **Сортировка вставками (insertion_sort)**:

(Строит отсортированный массив, вставляя каждый элемент в правильное положение относительно уже отсортированных элементов)

• **Сортировка пузырьком (bubble_sort)**:

(Проходит по массиву и сравнивает соседние элементы, меняя их местами, если они находятся в неправильном порядке. Процесс повторяется до тех пор, пока не будет выполнен полный проход без изменений)

Каждый из алгоритмов реализован в отдельной функции, принимающей массив и его размер в качестве аргументов.

3. Генерация массива

С помощью библиотеки malloc.h выделяем нужное количество памяти массива. Для генерации случайных чисел используется функция rand().

4. Основная функция

В основной функции происходит:
• Запрос размера массива у пользователя.
• Выделение памяти для массива mas с помощью функции malloc. Здесь mas = (int*)malloc(size * sizeof(int)); выделяет память для массива целых чисел размером size.
Использование sizeof(int) позволяет определить, сколько байт нужно выделить для массива целых чисел.

5. Измерение времени выполнения

Для каждой сортировки используется функция clock() для измерения времени выполнения. Время выполнения каждой сортировки сохраняется в переменных `start_time`, `total_time`, `end_time` .

6. Вывод результатов

После завершения сортировок программа выводит время выполнения каждого алгоритма на экран.

7. Освобождение памяти

В конце программы освобождается память, выделенная для массива mas, с помощью функции free(), что предотвращает утечки памяти.

Результаты экспериментов

Я проводил эксперименты над массивами, содержащими 5000, 10000, 50000, 100000 элементов

5000 элементов:
а) Selection Sort – 0.037 с
б) Insertion Sort – 0.019 с
в) Bubble Sort – 0.054 с
Лучший результат показал Insertion Sort.

10000 элементов:
а) Selection Sort – 0.128 с
б) Insertion Sort – 0.061 с
в) Bubble Sort – 0.239 с
Лучший результат показал Insertion Sort.

50000 элементов:
а) Selection Sort – 2.869 с
б) Insertion Sort – 1.55 с
в) Bubble Sort – 6.97 с
Лучший результат показал Insertion Sort.

100000 элементов:
а) Selection Sort – 11.479 с
б) Insertion Sort – 6.31 с
в) Bubble Sort – 29.046 с
Лучший результат показал Insertion Sort.

Заключение

Я провел эксперименты с временем сортировки массивов и делаю вывод, что чем больше количество элементов в массиве, тем сильнее отличается время выполнения трех разных сортировок . В итоге я считаю, что для маленьких массивов можно выбрать любую из этих трех сортировок(так как время их выполнения отличается незначительно), но для массивов с большим количеством элементов стоит пользоваться сортировкой вставками(Insertion Sort) .

Литература

https://github.com/qcha/JBook/blob/master/algorithms/sorting/bubble.md
https://github.com/qcha/JBook/blob/master/algorithms/sorting/insertion.md

Приложение

https://github.com/nikitasup/labaaaaa.git