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
Чеславский Константин Иванович

_____
подпись

Преподаватель:
Куклин А.Е.

_____
подпись

Нижний Новгород
2024

Содержание:

Введение

Сортировка является одной из основных задач в области алгоритмов и структур данных. Она подразумевает перестановку заданного массива или списка элементов в соответствии с оператором сравнения элементов. Он используется для определения их нового порядка в соответствующей структуре данных. Сортировка переупорядочивает все элементы либо по возрастанию, либо по убыванию. Оптимальные алгоритмы позволяют снизить время выполнения программ и обеспечить предсказуемое поведение системы.

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 сортирует заранее сгенерированный массив с помощью трех сортировок: сортировка выбором, сортировка вставками, сортировка пузырьком. Далее программа должна вывести время за которое она отсортировала массив с помощью разных сортировок, мы же в свою очередь должны выяснить какой метод сортировки является наилучшим для определенного размера массива.

Описание алгоритмов

1. Сортировка выбором (Selection Sort)

Описание:

Сортировка выбором — это алгоритм, который находит в исходном массиве максимальный или минимальный элементы, в зависимости от того, как необходимо сортировать массив: по возрастанию или по убыванию. Если массив должен быть отсортирован по возрастанию, то из исходного массива необходимо выбирать минимальные элементы. Если же массив необходимо отсортировать по убыванию, то выбирать следует максимальные элементы. Суть алгоритма сортировки выбором сводится к многократному поиску минимального (максимального) элементов в неотсортированной части массива.

Принцип работы:

1. На каждом шаге выбирается элемент с наименьшим значением из неотсортированной части массива.
2. Производится обмен элементов: выбранный элемент меняется местами с первым элементом неотсортированной части массива.
3. Сдвигается граница: граница между отсортированной и неотсортированной частями смещается, и процесс повторяется для оставшихся элементов.

4. Весь массив становится отсортированным после выполнения всех шагов.

2. Сортировка вставками (Insertion Sort)

Описание:

Сортировка вставками — алгоритм сортировки, в котором элементы входной последовательности просматриваются по одному, и каждый новый поступивший элемент размещается в подходящее место среди ранее упорядоченных элементов.

Принцип работы:

1. В начальный момент отсортированная последовательность пуста. На каждом шаге алгоритма выбирается один из элементов входных данных и помещается на нужную позицию в уже отсортированной последовательности.
2. В каждой позиции массива элемент проверяется на соответствие самому большому значению в отсортированном списке (которое оказывается рядом с ним, в предыдущей позиции массива). Если значение больше, элемент остаётся на месте и переходит к следующему. Если меньше, находится правильное положение в отсортированном списке, сдвигаются все большие значения вверх, освобождая пробел, и вставляется в это правильное положение.
3. Так происходит до тех пор, пока набор входных данных не будет исчерпан.

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 time1, time2 и time3.

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, 200000 элементов

5000 элементов:

Selection Sort – 0.02 с

Insertion Sort – 0.012 с

Bubble Sort – 0.04 с

Лучший результат в этом тесте показал Insertion sort.

10000 элементов:

Selection Sort – 0.082 с

Insertion Sort – 0.05 с

Bubble Sort – 0.197 с

Лучший результат в этом тесте показал Insertion sort.

50000 элементов:

Selection Sort – 2.055 с

Insertion Sort – 1.225 с

Bubble Sort – 5.487 с

Лучший результат в этом тесте показал Insertion sort.

200000 элементов:

Selection Sort – 32.651 с

Insertion Sort – 19.396 с

Bubble Sort – 90.922 с

Лучший результат в этом тесте показал Insertion sort.

Заключение

Из результатов экспериментов видно, что чем больше количество элементов в массиве, тем больше отличается время выполнения сортировки с помощью разных функций. В итоге я считаю, что для маленьких массивов можно выбрать любую из этих трех сортировок, но для больших массивов стоит пользоваться сортировкой вставками.

Литература

https://stackoverflow.com/questions/3557221/how-do-i-measure-time-in-c

Приложение

https://github.com/KAHATION/laba5/blob/master/laba5/1.cpp