Содержание

Введение

Сортировка – это процесс упорядочивания элементов в массиве в соответствии с заданным критерием.

Задание:

1. описать три алгоритма сортировки: пузырьком (с флагом), выбором и вставками;
2. реализовать данные алгоритмы на языке C;
3. провести тестирование и анализ алгоритмов на различных типах данных:
    - лучший случай (best – уже отсортированный массив);
    - худший случай (worst – обратно отсортированный массив);
    - случайный массив (random – заполненный случайными числами).
4. оценить временную сложность (O) алгоритмов;
5. сравнить производительность реализаций алгоритмов по результатом замеров времени выполнения.

Цель работы

Цель работы заключается в сравнении трех нерекурсивных алгоритмов сортировки: пузырьком с флагом, выбором и вставками в различных сценариях, реализация их на языке C. В рамках работы требуется провести сравнительный анализ по временной сложности, и на основе полученных данных сформулировать выводы о производительности каждого алгоритма.

Описание алгоритмов

Сортировка пузырьком с флагом (Bubble Sort).

Алгоритм многократно проходит по массиву, сравнивая соседние элементы и меняя их местами, если они находятся в неправильном порядке.

```
void bubbleSort(int arr[], int n) {
    for (int i = 0; i < n - 1; i++) {
        int swapped = 0;
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                // Меняем элементы местами
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
                swapped = 1;
            }
        }
        // Если не было обменов, массив уже отсортирован
        if (!swapped) break;
    }
}
```

Листинг 1. Сортировка пузырьком с флагом

Лучший случай: O(N). Если массив отсортирован, то внутренний цикл выполняет N – 1 сравнений, но обмен не происходит. Алгоритм завершается за один проход внешнего цикла.

Худший случай: $O(N^2)$. На каждом из N – 1 проходов происходит N – i сравнений и обменов. По сумме арифметической прогрессии $\sum_{i=1}^{N-1} N - i = \frac{N(N-1)}{2} \rightarrow O(N^2)$.

Сортировка выбором (Selection Sort).

На каждом шаге алгоритм ищет минимальный элемент в неотсортированной части массива и помещает его в начало этой части.

```
void selectionSort(int arr[], int n) {
    for (int i = 0; i < n - 1; i++) {
        int minIndex = i;
        for (int j = i + 1; j < n; j++) {
            if (arr[j] < arr[minIndex]) {
                minIndex = j;
            }
        }
```

```
        // Меняем текущий элемент с минимальным
        int temp = arr[i];
        arr[i] = arr[minIndex];
        arr[minIndex] = temp;
    }
}
```

Листинг 2. Сортировка выбором

Лучший случай и худший случай: $O(N^2)$. Алгоритм всегда выполняет N – 1 итераций внешнего цикла. На каждой итерации поиск минимума требует N – i – 1 сравнений. Суммарно: $\sum_{i=1}^{N-1} N - i - 1 = \frac{N(N-1)}{2} \to O(N^2)$.

Сортировка вставками (Insert Sort).

Алгоритм строит отсортированную часть массива, постепенно вставляя каждый новый элемент в правильное место.

```
void insertionSort(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;
        // Перемещаем элементы больше key на одну позицию вперед
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
}
```

Листинг 3. Сортировка вставками

Лучший случай: O(N). Каждый элемент сравнивается только с предыдущим. Выполняется N – 1 сравнений.

Худший случай: $O(N^2)$. Каждый i-ый элемент требует i сдвигов. Суммарно $\sum_{i=1}^{N-1} N - i = \frac{N(N-1)}{2} \to O(N^2)$.

Средства реализации и архитектура

Язык программирования: C.

Среда разработки: CLion.

Компилятор: MinGW.

Для визуализации результатов: Python с библиотекой matplotlib.

Архитектура: CPU – Intel(R) Core(TM) i7-7600U CPU @ 2.80GHz (2.90 GHz) x64, RAM – 32 GB.

Для замера времени выполнения реализации алгоритмов использовались функции QueryPerformanceFrequency (возвращает частоту таймера (количество тиков в секунду)) и QueryPerformanceCounter (фиксирует текущее значение таймера до и после выполнения кода алгоритма) из WinAPI [4].

Результаты замеров

В таблицах 1-3 приведены результаты замеров времени выполнения агоритмов в лучшем, худшем и случайном случаях, а на рисунках 1-3 показаны графики зависимости временных характеристик от размера массива.

Размер	Пузырьком с флагом	Вставками	Выбором
10	0.00005	0.0001	0.00022
100	0.0003	0.00061	0.0129
500	0.001	0.0024	0.3371
1000	0.0044	0.0045	1.5337
2000	0.0053	0.006	4.6384
5000	0.0094	0.0156	28.3591
10000	0.023	0.0318	109.9488
25000	0.1136	0.1282	731.2663
50000	0.1016	0.1551	3047.963

Таблица 1. Время отработки алгоритмов при лучшем случае (мс)

Размер	Пузырьком с флагом	Вставками	Выбором
10	0.00034	0.00032	0.00034
100	0.031	0.0175	0.0152
500	0.828	0.4175	0.3414
1000	3.2472	1.8313	1.8194
2000	8.988	5.5146	6.27
5000	56.7457	36.0497	26.2004
10000	193.7575	130.5994	109.9503
25000	1459.7913	1004.0351	897.1359
50000	5002.2379	3341.4652	2578.742

Таблица 2. Время отработки алгоритмов при худшем случае (мс)

Размер	Пузырьком с флагом	Вставками	Выбором
10	0.00033	0.00018	0.00032
100	0.0261	0.0094	0.0151
500	1.0632	0.1987	0.5848
1000	2.4338	0.6985	1.2846
2000	10.0458	2.4994	4.8637
5000	50.4775	16.9069	26.849
10000	229.2933	64.5644	110.7014
25000	1991.3035	990.2702	931.6868
50000	6724.5013	1687.2455	2684.695

Таблица 3. Время отработки алгоритмов в случайном массиве (мс)

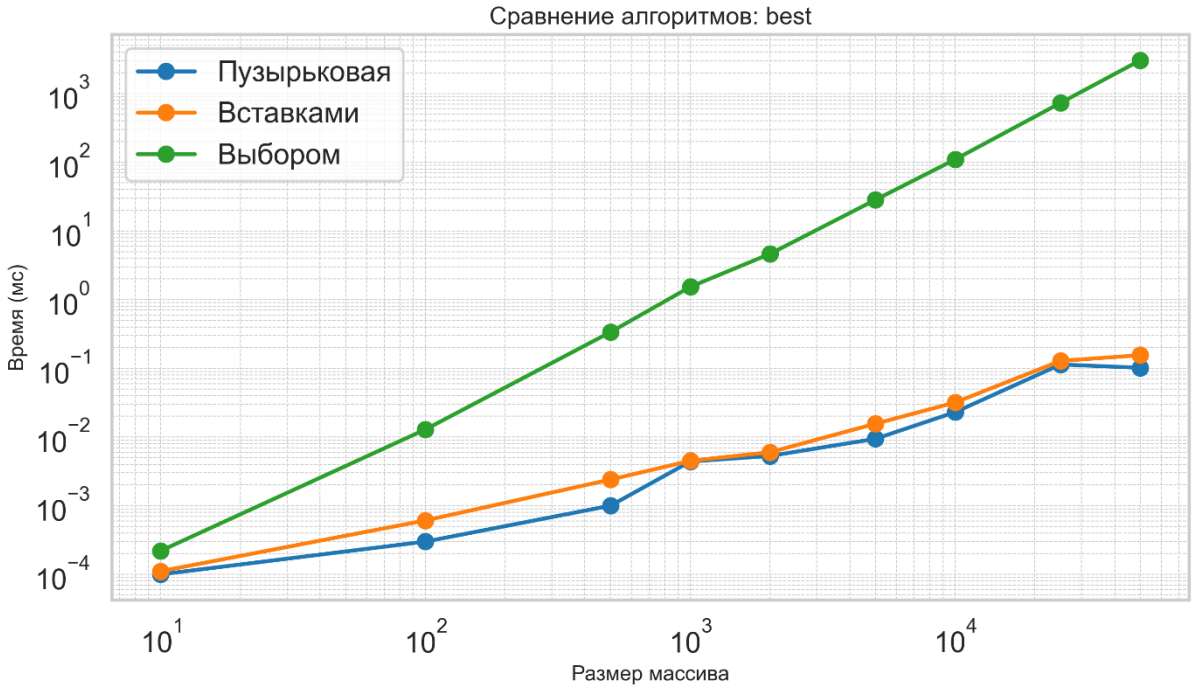

Рис.1. Графики зависимости времени отработки алгоритма от размера массива в лучшем случае

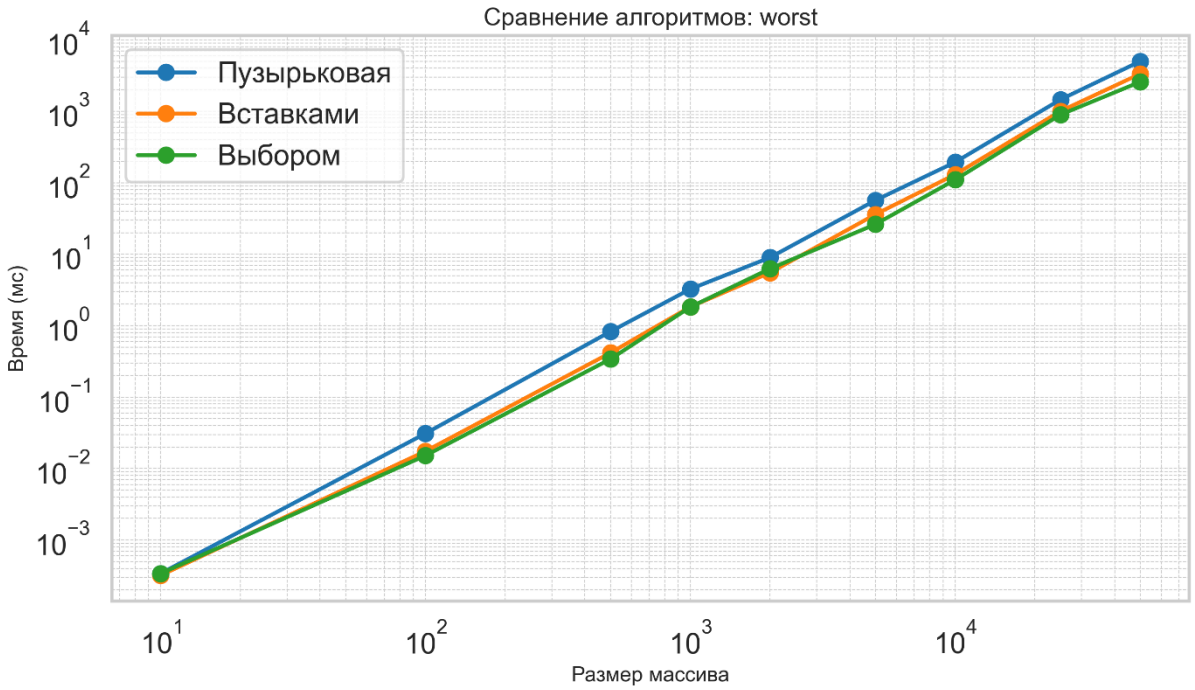

Рис.2. Графики зависимости времени отработки реализации алгоритма от размера массива в худшем случае

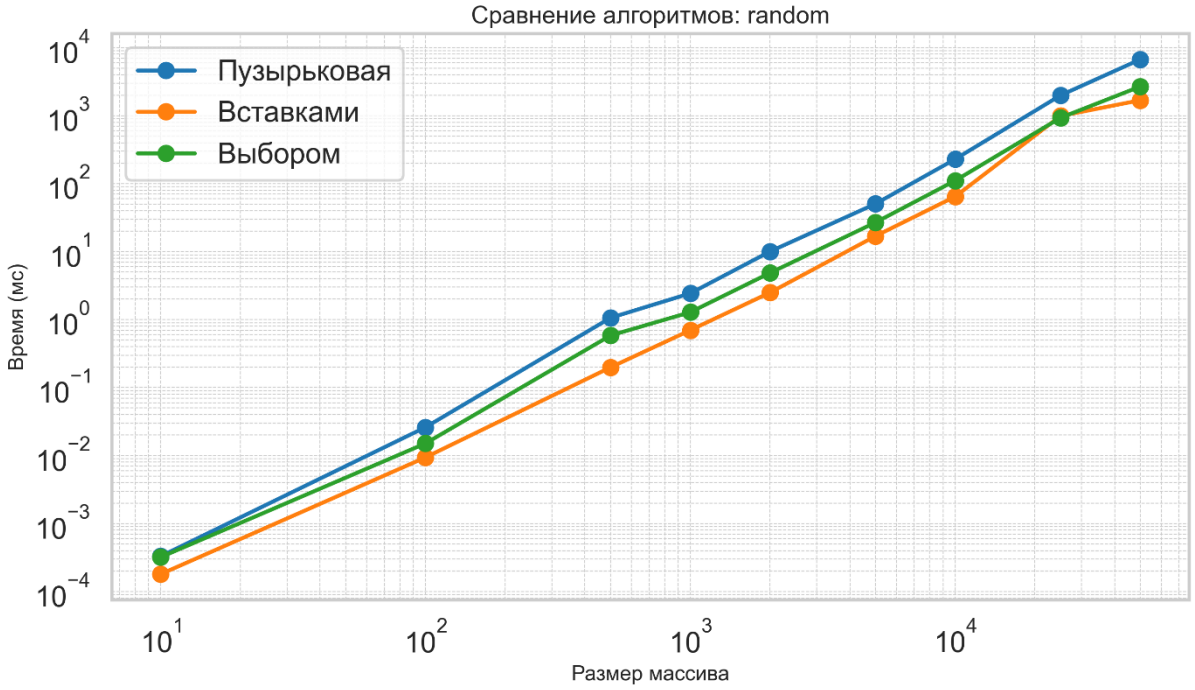

Рис.3. Графики зависимости времени отработки реализации алгоритма от размера случайно сгенерированного массива

Выводы из результатов

Анализируя экспериментальные данные по времени выполнения трех нерекурсивных алгоритмов сортировки (пузырьком с флагом, выбором и вставками) в различных сценариях, можно отметить некоторые закономерности.

Лучший случай (отсортированный по возрастанию массив) демонстрирует преимущество сортировок пузырьком с флагом и вставками над сортировкой выбором.

Заключение

В ходе лабораторной работы были изучены и реализованы три нерекурсивных алгоритма сортировки: пузырьковая, выбором и вставками. На основании теоретических изысканий и практических данных можно сделать следующие выводы:

1. Пузырьковая сортировка наиболее эффективна в случаях, когда массив уже упорядочен (или почти упорядочен). Однако в худшем случае она уступает по скорости остальным алгоритмам.
2. Сортировка выбором характеризуется постоянной сложностью не зависимо от порядка расположения элементов массива. На практике в худшем случае этот алгоритм показывает лучший результат за счет малого количества «дорогих» действий.
3. Сортировка вставками в лучшем и случайном массиве работает очень быстро.

Таким образом, выбор алгоритма сортировки определяется как требованием к трудоемкости, так и характером исходных данных.

Список литературы

1. Кнут Д. Искусство программирования. Том 3: Сортировка и поиск.
2. Седжвик Р. Алгоритмы на С.
3. Вирт Н. Алгоритмы и структуры данных.
4. QueryPerformanceCounter function (Windows) [Электронный ресурс]: документация Microsoft по Win32 API. [https://docs.microsoft.com/en-us/windows/win32/api/profileapi/nf-profileapi-queryperformancecounter]. Дата обращения: 11.04.2025.