Алгоритмы и структуры данных в системах искусственного интеллекта		
Кафедра ЭВМ и С Лабораторная работа №1 «Сортировка массивов»	ФИО студента	Рюмин Д.Л.
	Группа	ЭВМ 1.2
	Дата выполнения	13.04.2023
	Дата отчета	
	Оценка (баллы)	
	Подпись преподавателя с расшифровкой	

Задание:

1) Реализовать по 1 варианту алгоритмов сортировки для сложности O(n^2), O (n*log2n) и O(n) : вставкой, слиянием и подсчетом. Сравнить с пузырьковой сортировкой.

Для каждого алгоритма выполнить запуски и измерить время работы на массивах следующей длины:

100, 200, 300 … 900 (с шагом 100)

1 000, 2 000, 3 000 … 9 000 (с шагом 1 000)

10 000, 20 000, 30 000 … 90 000 (с шагом 10 000)

100 000, 200 000, 300 000 … 900 000 (с шагом 100 000)

Занести измеренные времёна работы алгоритмов в таблицу (размеры массивов по строкам, алгоритмы - по столбцам).

2) Рассмотреть работу разных алгоритмов по представленному скомпилированному примеру на разном наборе входных данных.

3) Индивидуальные варианты: реализовать еще по 2 алгоритма сортировки по вариантам:

1. Сортировка выбором (selection sort) и быстрая (quicksort - обычная).

Для них также заполнить таблицу как в пункте 1.

Выполнение:

	Время (с.)				
	Insertion sort	**Merge sort**	**Count sort**	**Selection sort**	**Quick Sort**
100	0,0000079	0,0000754	0,0000025	0,000021	0,0000101
200	0,000026	0,0001251	0,0000034	0,0000762	0,0000212
300	0,000056	0,0002125	0,0000051	0,0001421	0,0000351
400	0,0001	0,0003012	0,0000072	0,0002215	0,0000416
500	0,00015	0,0003512	0,0000108	0,0003621	0,0000562
600	0,00023	0,0004621	0,0000112	0,0005274	0,0000809
700	0,00029	0,00054	0,0000122	0,0007053	0,000081
800	0,0004	0,0006221	0,0000139	0,0009053	0,0000942
900	0,00054	0,0007012	0,0000171	0,0010622	0,0001018
1 000	0,00068	0,0007512	0,0000207	0,0014072	0,0001542
2 000	0,0026	0,00161	0,0000406	0,0053126	0,0003128
3 000	0,006	0,002122	0,0000685	0,0117322	0,000419
4 000	0,01	0,003067	0,0000812	0,0216894	0,0006454
5 000	0,016	0,004012	0,0001016	0,0328421	0,0007427
6 000	0,022	0,004753	0,0001161	0,0481267	0,0009063
7 000	0,031	0,006023	0,0002012	0,0655126	0,0010352
8 000	0,039	0,006212	0,0002332	0,0854749	0,0012829
9 000	0,051	0,007251	0,000185	0,1083173	0,0014521
10 000	0,062	0,00812	0,0002035	0,134683	0,0020316
20 000	0,25	0,016237	0,0004152	0,521795	0,0036416
30 000	0,58	0,027582	0,0006064	1,227236	0,0054412
40 000	1,02	0,03215	0,0008512	2,208926	0,007361
50 000	1,8	0,041256	0,0008945	3,407126	0,0093619
60 000	2,6	0,050341	0,0010215	4,852858	0,0109457
70 000	3,32	0,05996	0,0010642	6,519452	0,0121617
80 000	4,15	0,066125	0,0012125	8,528392	0,0131623
90 000	4,96	0,0772152	0,0014015	10,872163	0,0151261
100 000	6,4	0,087437	0,0014954	12,521612	0,0173723
200 000	25,12	0,096216	0,00421613	42,62162	0,0241261
300 000	50,12	0,1012521	0,00561621	189,236231	0,0321623
400 000	140,17	0,11252162			0,0421612
500 000	350,62				0,0613216

Рисунок 1. Диаграмма зависимости времени от размера массива (сортировка вставкой).

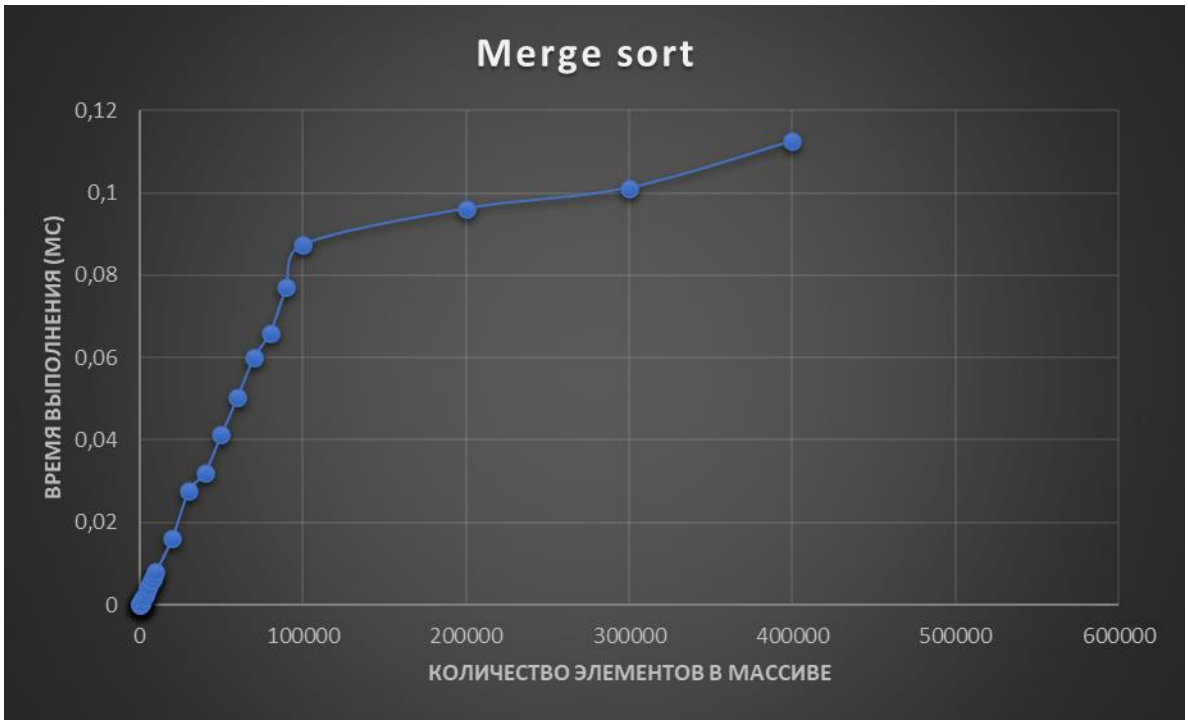

Рисунок 2. Диаграмма зависимости времени от размера массива (сортировка слиянием).

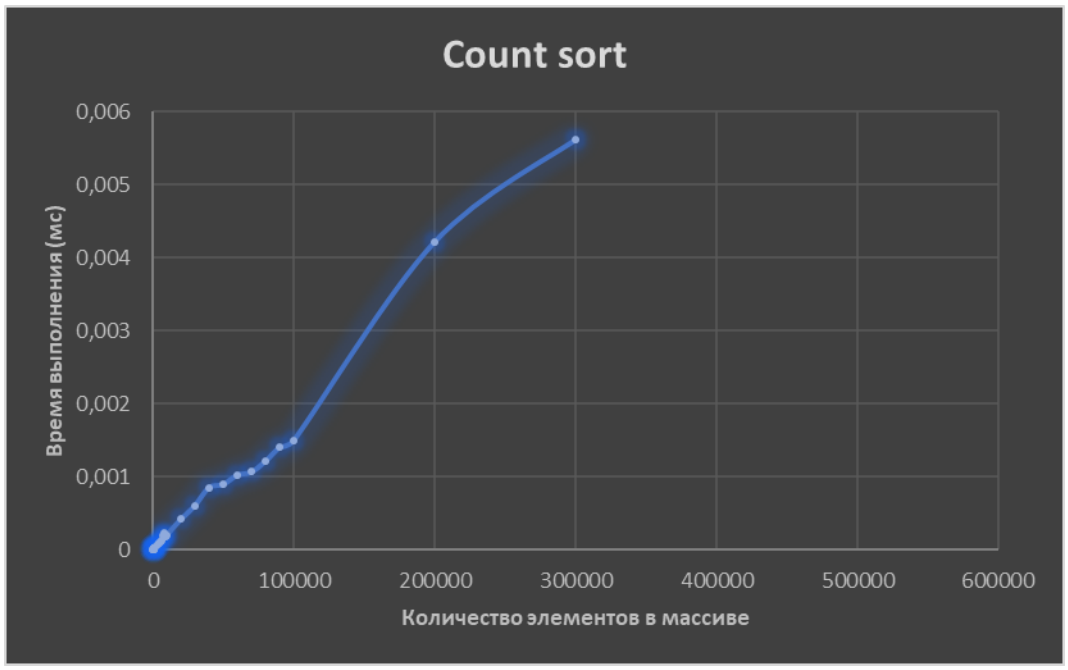

Рисунок 3. Диаграмма зависимости времени от размера массива (сортировка подсчетом).

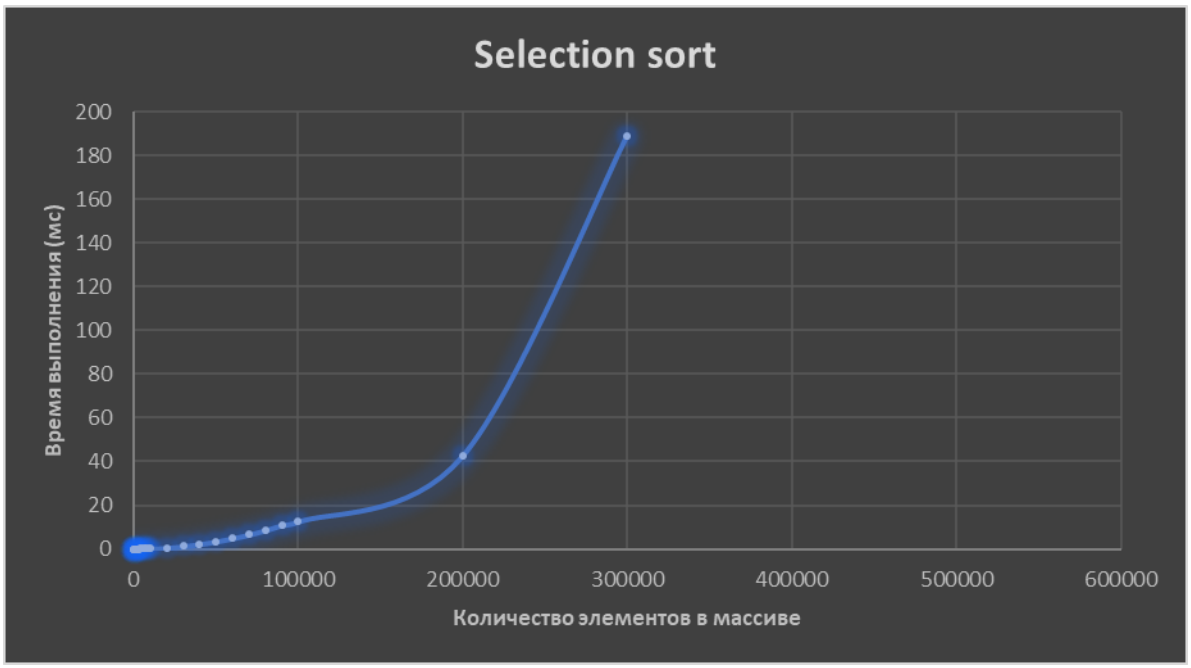

Рисунок 4. Диаграмма зависимости времени от размера массива (сортировка выбором).

Рисунок 5. Диаграмма зависимости времени от размера массива (быстрая сортировка).

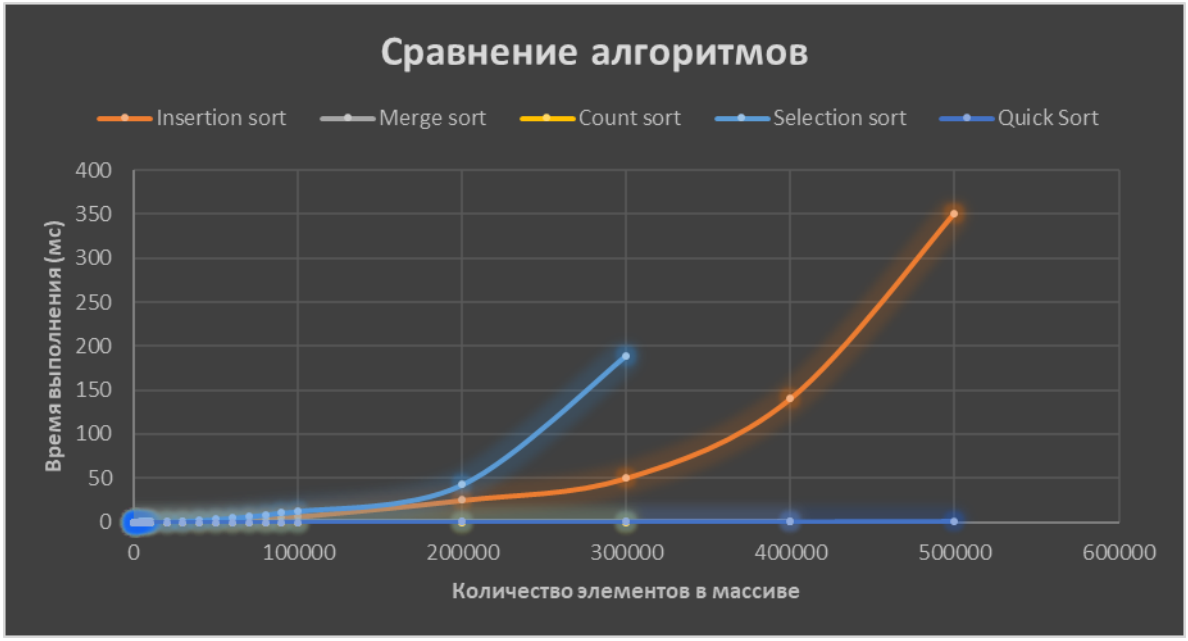

Рисунок 6. Диаграмма сравнения сортировок.

Код программы:

Sorts.cpp

```
#include <iostream>
#include <chrono>
#include <random>

using namespace std;
using namespace std::chrono;

const int n = 100000;

// ------------------------------ Сортировка вставкой ------------------------------ \\


void insertionSort(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
}


// ------------------------------ Сортировка слиянием ------------------------------ \\


void merge(int arr[], int left, int middle, int right) {
    int i, j, k;
    int n1 = middle - left + 1;
    int n2 = right - middle;

    int* L = new int[n1];
    int* R = new int[n2];

    for (i = 0; i < n1; i++)
        L[i] = arr[left + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[middle + 1 + j];

    i = 0;
    j = 0;
    k = left;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        }
        else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    while (j < n2) {
        arr[k] = R[j];
        j++;
```

```
        k++;
    }

    delete[] L;
    delete[] R;
}

void mergeSort(int arr[], int l, int r) {
    if (l < r) {
        int m = l + (r - l) / 2;

        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);

        merge(arr, l, m, r);
    }
}


// ------------------------------ Сортировка подсчетом ------------------------------ \\


void countingSort(int arr[], int n) {
    const int k = 200000;
    int count[k] = { 0 };

    for (int i = 0; i < n; i++) {
        count[arr[i]]++;
    }

    int index = 0;
    for (int i = 0; i < k;

        i++) {
        for (int j = 0; j < count[i]; j++) {
            arr[index] = i;
            index++;
        }
    }
}


// ------------------------------ Сортировка выбором ------------------------------ \\


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;
            }
        }
        std::swap(arr[i], arr[minIndex]);
    }
}


// ---------------------------------- Быстрая сортировка ---------------------------------- \\


int partition(int* arr, int start, int end)
{
    int pivot = arr[end];

    int pIndex = start;

    for (int i = start; i < end; i++) {
        if (arr[i] <= pivot) {
```

```
            swap(arr[i], arr[pIndex]);
            pIndex++;
        }
    }

    swap(arr[pIndex], arr[end]);

    return pIndex;
}

void quicksort(int a[], int start, int end)
{
    if (start >= end) {
        return;
    }

    int pivot = partition(a, start, end);

    quicksort(a, start, pivot - 1);

    quicksort(a, pivot + 1, end);
}

void printArr(int* arr) {
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
}




// ------------------------------ Main ------------------------------ \\




int main() {

    setlocale(LC_ALL, "Russian");

    // -------------------------- Сортировка вставкой -------------------------- \\

    /*

    int arr[n];
    random_device rd;
    mt19937 gen(rd());
    uniform_int_distribution<> dis(1, 300000);

    for (int i = 0; i < n; i++) {
        arr[i] = dis(gen);
    }

    auto start = high_resolution_clock::now();

    insertionSort(arr, n);

    auto stop = high_resolution_clock::now();
    auto duration = duration_cast<nanoseconds>(stop - start);

    cout << "Отсортированный массив: ";

    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
```

```
cout << "\nВремени потрачено: " << duration.count() << " ns" << endl;

return 0;


*/

// ------------------------- Сортировка слиянием ------------------------- \\

/*

int arr[n];
random_device rd;
mt19937 gen(rd());
uniform_int_distribution<> dis(1, 300000);

for (int i = 0; i < n; i++) {
    arr[i] = dis(gen);
}

auto start = high_resolution_clock::now();

mergeSort(arr, 0, n - 1);

auto stop = high_resolution_clock::now();
auto duration = duration_cast<milliseconds>(stop - start);

//cout << "Отсортированный массив: ";

//for (int i = 0; i < n; i++) {
//    cout << arr[i] << " ";
//}

cout << "\nВремени потрачено: " << duration.count() << " ns" << endl;

return 0;


// ------------------------- Сортировка подсчетом ------------------------- \\

*/

/*



int arr[n];

for (int i = 0; i < n; i++) {
    arr[i] = rand() % 200000;
}

auto start = steady_clock::now();

countingSort(arr, n);

auto end = steady_clock::now();

auto duration = duration_cast<nanoseconds>(end - start);


//cout << "Отсортированный массив: ";

//for (int i = 0; i < n; i++) {
//    cout << arr[i] << " ";
//}
```

```
cout << "Времени потрачено: " << duration.count() << " ns." << endl;

return 0;


// ------------------------- Сортировка выбором ------------------------- \\


*/

/*



int arr[n];

// заполнение массива случайными числами
for (int i = 0; i < n; i++) {
    arr[i] = rand() % n;
}

auto start = steady_clock::now();

selectionSort(arr, n);

auto end = steady_clock::now();

auto duration = duration_cast<nanoseconds>(end - start);

//cout << "Отсортированный массив: ";

//for (int i = 0; i < n; i++) {
//    cout << arr[i] << " ";
//}

cout << "Времени потрачено: " << duration.count() << " ns." << endl;

return 0;


// ------------------------- Быстрая сортировка ------------------------- \\

*/

/*


int arr[n];

for (int i = 0; i < n; i++) {
    arr[i] = rand() % n;
}

int N = sizeof(arr) / sizeof(arr[0]);

auto start = std::chrono::steady_clock::now();

quicksort(arr, 0, N-1);

auto end = std::chrono::steady_clock::now();

nanoseconds duration = duration_cast<nanoseconds>(end - start);

//for (int i = 0; i < n; i++) {
//    cout << arr[i] << " ";
//}

cout << "\nВремени потрачено: " << duration.count() << " ns." << endl;
```

```
    return 0;

    */

}
```