Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа № 15

По дисциплине «Основы алгоритмизации и программирования»

На тему «Анализ алгоритмов сортировок»

Выполнил:

Студент 1 курса 10 группы

Жамойдо Артём Игоревич

Преподаватель: асс. Андронова М.В.

Минск, 2024

В соответствии со своим вариантом написать программу для ***сортировок*** массивов указанными в таблице методами. Исходные массивы заполняются случайными числами.

Определить зависимость времени выполнения алгоритмов от количества элементов для каждого из алгоритмов. Выполнить моделирование для массивов размером 1000, 2000, 3000, 4000, 5000 (в зависимости от быстродействия компьютера размеры массивов можно увеличивать).

**Вариант 4**

Ввести массивы **А** и **В**. В массив **С** перенести те элементы массива **А**, которые больше минимального элемента массива **В**. Массив **С** отсортировать по убыванию, используя алгоритмы сортировок: пирамидальная сортировка, сортировка слиянием.

#include <iostream>

#include <vector>

#include <algorithm>

#include <ctime>

#include <random>

using namespace std;

// Функция для заполнения массива случайными числами

void fillArrayWithRandomNumbers(vector<int>& arr, int size, int min, int max)

{

random\_device rd;

mt19937 rng(rd());

uniform\_int\_distribution<int> uni(min, max);

arr.clear();

for (int i = 0; i < size; i++)

{

int num = uni(rng);

arr.push\_back(num);

}

}

// Функция для вывода массива

void printArray(const vector<int>& arr)

{

for (int num : arr)

{

cout << num << " ";

}

cout << endl;

}

// Функция для сортировки массива пирамидальной сортировкой

void heapSort(vector<int>& arr)

{

make\_heap(arr.begin(), arr.end());

sort\_heap(arr.begin(), arr.end());

}

// Функция для сортировки массива слиянием

void mergeSort(vector<int>& arr)

{

if (arr.size() <= 1)

{

return;

}

int middle = arr.size() / 2;

vector<int> left(arr.begin(), arr.begin() + middle);

vector<int> right(arr.begin() + middle, arr.end());

mergeSort(left);

mergeSort(right);

merge(left.begin(), left.end(), right.begin(), right.end(), arr.begin());

}

int main()

{

setlocale(0, "ru");

int size;

cout << "Введите размер массивов: ";

cin >> size;

// Заполнение массивов случайными числами

vector<int> arrA;

fillArrayWithRandomNumbers(arrA, size, 1, 100);

vector<int> arrB;

fillArrayWithRandomNumbers(arrB, size, 1, 100);

// Вывод исходных массивов

cout << "Массив A:" << endl;

printArray(arrA);

cout << "Массив B:" << endl;

printArray(arrB);

// Формирование массива C

int minB = \*min\_element(arrB.begin(), arrB.end());

vector<int> arrC;

for (int num : arrA)

{

if (num > minB)

{

arrC.push\_back(num);

}

}

// Сортировка массива C пирамидальной сортировкой

clock\_t startTime = clock();

heapSort(arrC);

clock\_t endTime = clock();

// Вывод отсортированного массива C

cout << "Массив C (пирамидальная сортировка):" << endl;

printArray(arrC);

// Вывод времени выполнения пирамидальной сортировки

double heapSortTime = double(endTime - startTime) / CLOCKS\_PER\_SEC;

cout << "Время выполнения пирамидальной сортировки: " << heapSortTime << " секунд" << endl;

// Сортировка массива C сортировкой слиянием

startTime = clock();

mergeSort(arrC);

endTime = clock();

// Вывод отсортированного массива C

cout << "Массив C (сортировка слиянием):" << endl;

printArray(arrC);

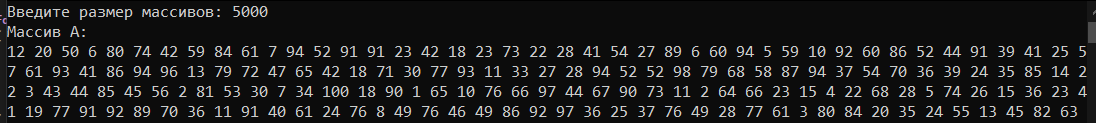
// Вывод времени выполнения сортировки слиянием

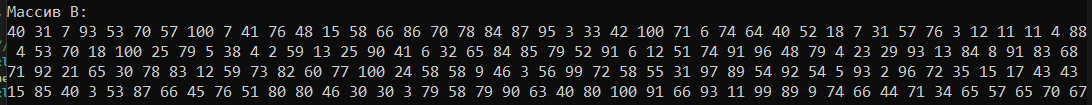
double mergeSortTime = double(endTime - startTime) / CLOCKS\_PER\_SEC;

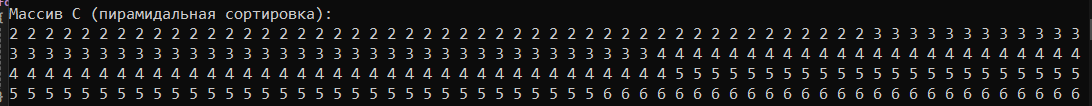
cout << "Время выполнения сортировки слиянием: " << mergeSortTime << " секунд" << endl;

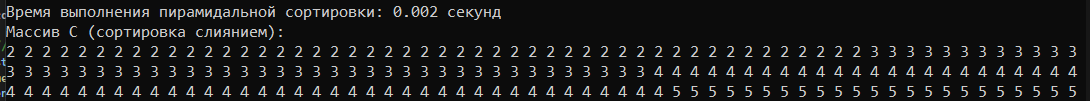
return 0;

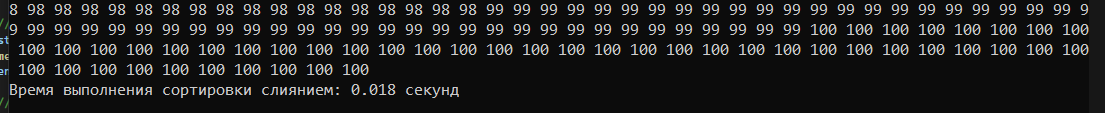
}



****

****

****

****

Доп. задачи

1. Разработать функцию сортировки **подсчетом** и добавить к списку исследуемых сортировок. Алгоритм: упорядоченный массив получается из исходного путем сравнения всех пар элементов массива. Для каждого из элементов подсчитывается количество элементов, меньших его. Например, пусть есть массив **B = <20, -5, 10, 8, 7>** .Для первого числа имеется 4 элемента, которые меньше первого. Для второго − 0 элементов и т. д. Таким образом, формируется массив счетчиков **S = < 4,   0, 3,** 2, 1>, элементы которого дают местоположения элементов в результирующем массиве B’ = <-5, 7, 8, 10, 20>.

#include <iostream>

#include <vector>

using namespace std;

// Функция сортировки подсчетом

void countingSort(vector<int>& arr)

{

int n = arr.size();

// Находим минимальное и максимальное значения в массиве

int minVal = \*min\_element(arr.begin(), arr.end());

int maxVal = \*max\_element(arr.begin(), arr.end());

// Определяем размер массива счетчиков

int countSize = maxVal - minVal + 1;

vector<int> count(countSize, 0);

// Подсчитываем количество элементов

for (int num : arr)

{

count[num - minVal]++;

}

int idx = 0;

// Формируем упорядоченный массив

for (int i = 0; i < countSize; i++)

{

while (count[i] > 0)

{

arr[idx] = i + minVal;

count[i]--;

idx++;

}

}

}

int main()

{

setlocale(0, "ru");

vector<int> arr = { 20, -5, 10, 8, 7 }; // Исходный массив

cout << "Исходный массив:" << endl;

for (int num : arr)

{

cout << num << " ";

}

cout << endl;

countingSort(arr);

cout << "Отсортированный массив (сортировка подсчетом):" << endl;

for (int num : arr)

{

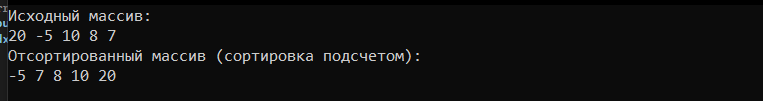
cout << num << " ";

}

cout << endl;

return 0;

}



2. Разработать функцию сортировки выбором и добавить к списку исследуемых сортировок. Алгоритм: осуществляется поиск элемента, имеющего наименьшее значение. После того, как этот элемент найден, он меняется местами с первым элементом. Затем, начиная со второго элемента массива, осуществляется поиск следующего наименьшего значения элемента. Найденный элемент меняется местами со вторым элементом. Этот процесс продолжается до тех пор, пока все числа не будут расположены в порядке возрастания.

#include <iostream>

#include <vector>

using namespace std;

// Функция сортировки выбором

void selectionSort(vector<int>& arr)

{

int n = arr.size();

for (int i = 0; i < n - 1; i++)

{

int minIdx = i;

// Находим индекс наименьшего элемента в оставшейся части массива

for (int j = i + 1; j < n; j++)

{

if (arr[j] < arr[minIdx])

{

minIdx = j;

}

}

// Меняем местами текущий элемент с наименьшим элементом

swap(arr[i], arr[minIdx]);

}

}

int main()

{

setlocale(0, "ru");

vector<int> arr = { 5, -2, 9, 3, 7 }; // Исходный массив

cout << "Исходный массив:" << endl;

for (int num : arr)

{

cout << num << " ";

}

cout << endl;

selectionSort(arr);

cout << "Отсортированный массив (сортировка выбором):" << endl;

for (int num : arr)

{

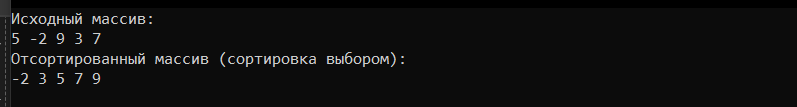
cout << num << " ";

}

cout << endl;

return 0;

}



3. Разработать функцию сортировки методом простой вставки и добавить к списку исследуемых сортировок. Алгоритм: массив в процессе сортировки делится на две части: упорядоченную и неупорядоченную. Вначале весь массив неупорядочен. На каждом шаге из неупорядоченной части извлекается первый элемент, который вставляется на нужное место упорядоченной части. При этом размер упорядоченной части увеличивается на единицу. В конце весь массив окажется упорядоченным.

#include <iostream>

#include <vector>

using namespace std;

// Функция сортировки вставками

void insertionSort(vector<int>& arr)

{

int n = arr.size();

for (int i = 1; i < n; i++)

{

int key = arr[i]; // Текущий элемент, который нужно вставить в упорядоченную часть

int j = i - 1;

// Перемещаем элементы большие, чем key, на одну позицию вперед

while (j >= 0 && arr[j] > key)

{

arr[j + 1] = arr[j];

j--;

}

// Вставляем key на правильную позицию в упорядоченной части

arr[j + 1] = key;

}

}

int main()

{

setlocale(0, "ru");

vector<int> arr = { 50, -20, -50, 20, -30, 30 }; // Исходный массив

cout << "Исходный массив:" << endl;

for (int num : arr)

{

cout << num << " ";

}

cout << endl;

insertionSort(arr);

cout << "Отсортированный массив (сортировка вставками):" << endl;

for (int num : arr)

{

cout << num << " ";

}

cout << endl;

return 0;

}

