Московский технический университет связи и информатики

Кафедра «Информатика»

Лабораторная работа по дисциплине «Теория и методы программирования» № 2

Тема: «*РАЗРАБОТКА И ИССЛЕДОВАНИЕ АЛГОРИТМОВ СОРТИРОВКИ ДАННЫХ»*

Выполнил:

Блёсткин Михаил

Группа БИБ2203

Вариант 6

Москва 2023 г.

**Цель:**

Изучить основные алгоритмы сортировки данных и освоить их на практике. Проверить и исследовать работу алгоритмов на различных наборах данных.

**Постановка задачи:**

1. Разработать подпрограммы сортировок массива типа int short с помощью алгоритмов быстрой сортировки и блочной сортировки.
2. Отладить правильность работы сортировок на массивах с количеством элементов N = 50 сгенерированные датчиком случайных чисел в диапазоне [10060, 30706]. Кроме того, контролировать правильность сортировки путём подсчёта контрольной суммы и числа серий в массиве.
3. Для сравнения алгоритмов сортировки составить таблицу для N = 100, 200, 300, 400, 500

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Количество элементов | 50 | 100 | 200 | 300 | 400 | 500 |
| Алгоритм сортировки | Время выполнения алгоритма сортировки (в миллисекундах) | | | | | |
| Быстрая сортировка | 0.001 | 0.001 | 0.001 | 0.002 | 0.002 | 0.004 |
| Блочная сортировка | 0.001 | 0.02 | 0.004 | 0.031 | 0.069 | 0.167 |

**Алгоритм решения задачи:**

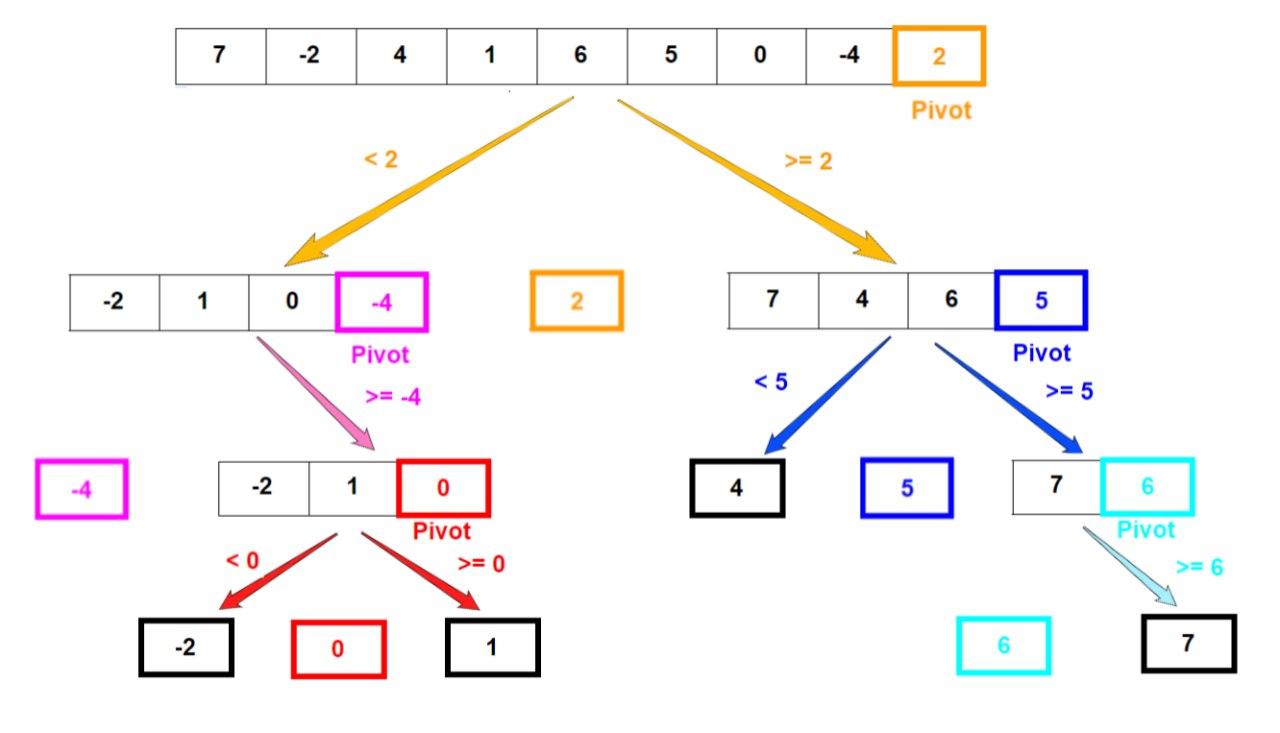
Быстрая сортировка:

Быстрый метод сортировки функционирует по принципу "разделяй и властвуй".

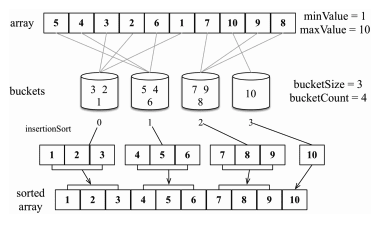
* Массив  типа  разбивается на два (возможно пустых) подмассива  и  , таких, что каждый элемент меньше или равен  который в свою очередь, не превышает любой элемент подмассива. Индекс вычисляется в ходе процедуры разбиения.
* Подмассивы    сортируются с помощью рекурсивного вызова процедуры быстрой сортировки.
* Поскольку подмассивы сортируются на месте, для их объединения не требуются никакие действия: весь массив оказывается отсортированным.

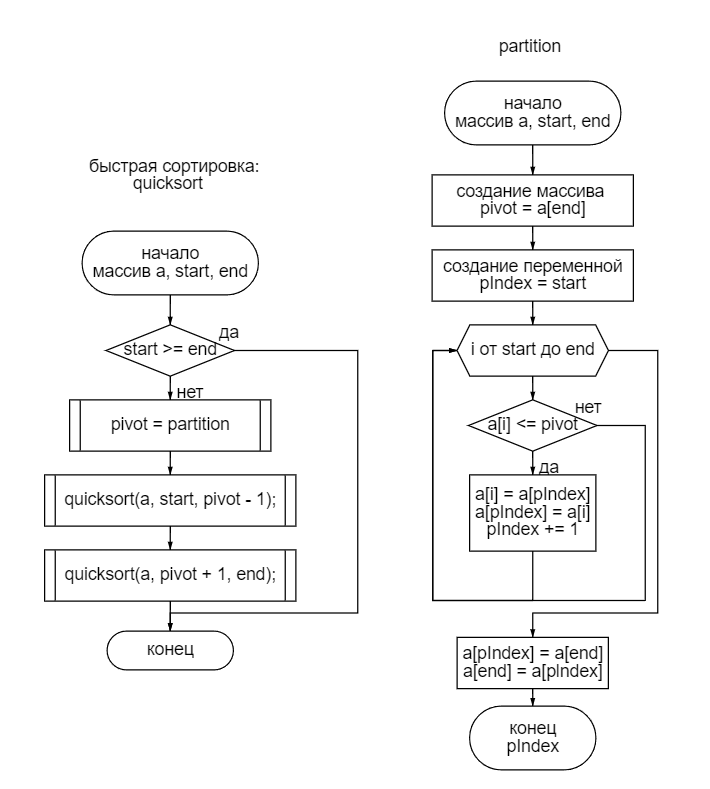
**Разбиение массива**

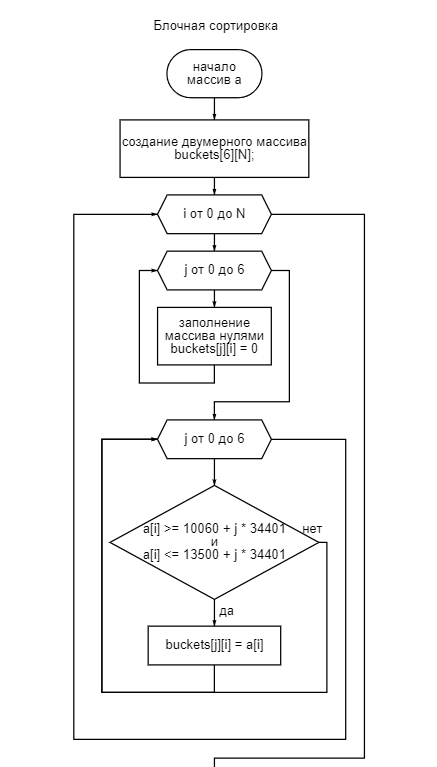
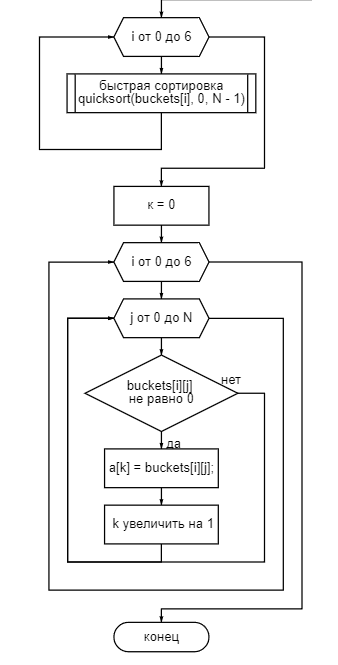
* Основной шаг алгоритма сортировки — процедура , которая переставляет элементы массива типа  нужным образом. Разбиение осуществляется с использованием следующей стратегии. Прежде всего, в качестве разделяющего элемента произвольно выбирается элемент . Далее начинается просмотр с левого конца массива, который продолжается до тех пор, пока не будет найден элемент, превосходящий по значению разделяющий элемент, затем выполняется просмотр, начиная с правого конца массива, который продолжается до тех пор, пока не отыскивается элемент, который по значению меньше разделяющего. Оба элемента, на которых просмотр был прерван, очевидно, находятся не на своих местах в разделенном массиве, и потому они меняются местами. Так продолжаем дальше, пока не убедимся в том, что слева от левого указателя не осталось ни одного элемента, который был бы больше по значению разделяющего, и ни одного элемента справа от правого указателя, которые были бы меньше по значению разделяющего элемента.
* Переменная    сохраняет значение разделяющего , a  и  представляет собой, соответственно, указатели левого и правого просмотра. Цикл разделения увеличивает значение  и уменьшает значение  на 1, причем условие, что ни один элемент слева от  не больше  и ни один элемент справа от  не меньше , не нарушается. Как только значения указателей пересекаются, процедура разбиения завершается.



Блочная сортировка:

1. Создание блока для каждого элемента в массиве.
2. Перебор списка сегментов и добавление элементов из массива. В конечном итоге мы получим элементов в каждом блоке.
3. Сортировка каждого непустого блока. Так как мы работаем с небольшим набором данных, в каждом сегменте не будет слишком много элементов. Этот этап реализован с помощью быстрой сортировки.
4. Перебор блоков по порядку. Когда содержимое каждого сегмента отсортировано, мы получаем список, в котором элементы расположены в соответствии с заданными критериями.

****

****

**Листинг программы**

//Задача 1:

#include <stdlib.h>

#include <stdio.h>

#include <time.h>

#include <iostream>

#include <algorithm>

using namespace std;

const int N = 50;

void output(short int a[], int sum, int ser)

{

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

printf("%hu ", a[i]);

printf("\nКонтрольная сумма: %d\nКоличество серий: %d\n\n", sum, ser);

}

int partition(short int a[], int start, int end)

{

short int pivot = a[end];

int pIndex = start;

for (int i = start; i < end; i++)

{

if (a[i] <= pivot)

{

swap(a[i], a[pIndex]);

pIndex++;

}

}

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

return pIndex;

}

void quicksort(short 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 bucketsort(short int a[])

{

short int buckets[6][N];

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

{

for (int j = 0; j < 6; j++) buckets[j][i] = 0;

if (a[i] >= 10060 && a[i] <= 13500) buckets[0][i] = a[i];

if (a[i] >= 13501 && a[i] <= 16941) buckets[1][i] = a[i];

if (a[i] >= 16942 && a[i] <= 20382) buckets[2][i] = a[i];

if (a[i] >= 20383 && a[i] <= 23823) buckets[3][i] = a[i];

if (a[i] >= 23824 && a[i] <= 27264) buckets[4][i] = a[i];

if (a[i] >= 27265 && a[i] <= 30706) buckets[5][i] = a[i];

}

for (int i = 0; i < 6; i++)

quicksort(buckets[i], 0, N - 1);

int k = 0;

for (int i = 0; i < 6; i++)

{

for (int j = 0; j < N; j++)

{

if (buckets[i][j] != 0)

{

a[k] = buckets[i][j];

k++;

}

}

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

time\_t t;

srand((unsigned)time(&t));

short int array1[N], array2[N], array1Copy[N], array2Copy[N];

int sum1 = 0, sum2 = 0, ser1 = 1, ser2 = 1;

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

{

int t = rand() % (30706 - 10060) + 10060;

array1[i] = (short int)t;

if (array1[i] < array1[i - 1]) ser1++;

sum1 += array1[i];

array1Copy[i] = (short int)t;

t = rand() % (30706 - 10060) + 10060;

array2[i] = (short int)t;

if (array2[i] < array2[i - 1]) ser2++;

sum2 += array2[i];

array2Copy[i] = (short int)t;

}

printf("Массивы до сортировки:\n\n");

output(array1, sum1, ser1);

output(array2, sum2, ser2);

sum1 = 0; sum2 = 0; ser1 = 1; ser2 = 1;

quicksort(array1, 0, N - 1);

quicksort(array2, 0, N - 1);

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

{

if (array1[i] < array1[i - 1]) ser1++;

sum1 += array1[i];

if (array2[i] < array2[i - 1]) ser2++;

sum2 += array2[i];

}

printf("Массивы после быстрой сортировки:\n\n");

output(array1, sum1, ser1);

output(array2, sum2, ser2);

sum1 = 0; sum2 = 0; ser1 = 1; ser2 = 1;

bucketsort(array1Copy);

bucketsort(array2Copy);

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

{

if (array1[i] < array1[i - 1]) ser1++;

sum1 += array1[i];

if (array2[i] < array2[i - 1]) ser2++;

sum2 += array2[i];

}

printf("Массивы после блочной сортировки:\n\n");

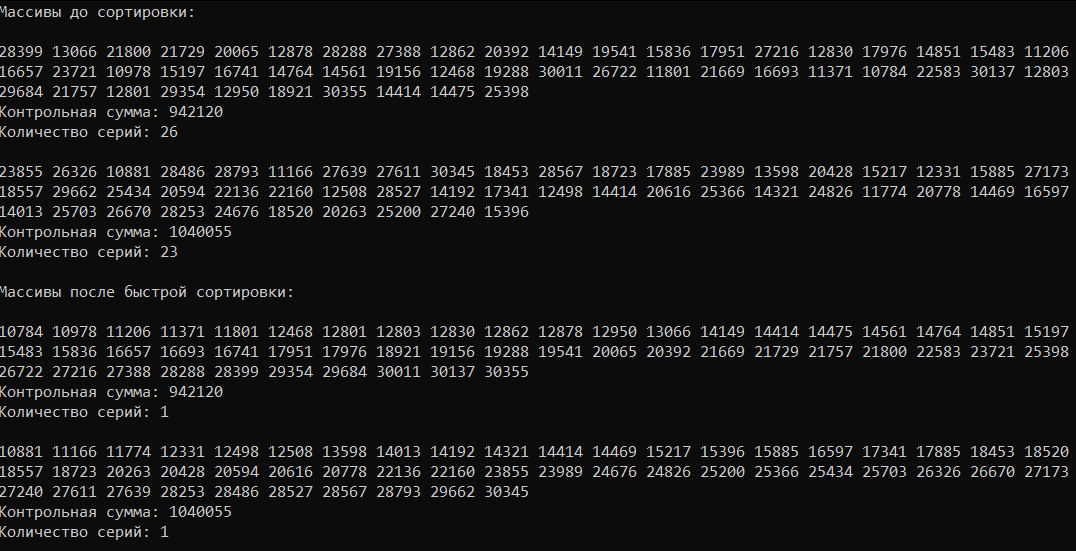
output(array1, sum1, ser1);

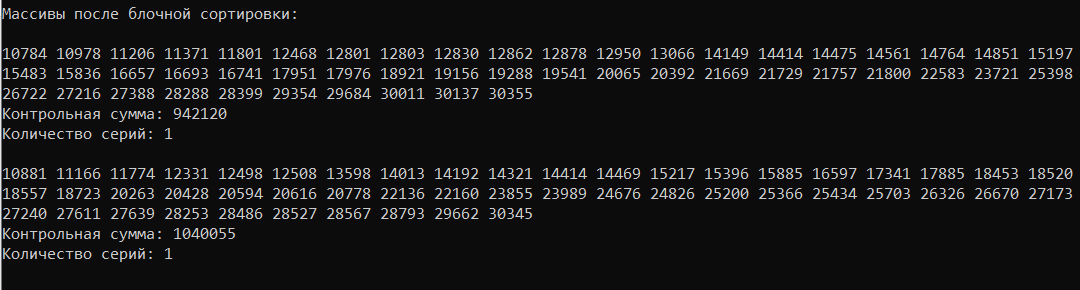
output(array2, sum2, ser2);

}

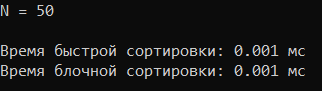
**Результат работы программы**

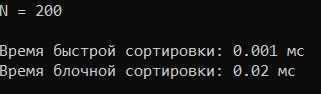
Задача 2

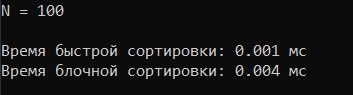


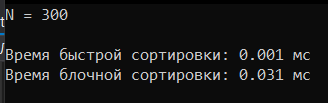


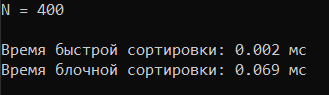
Задача 3

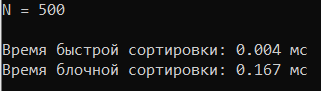












**Вывод**

Мной было реализована программа по сортировке двух массивов случайных чисел по алгоритму быстрой сортировки и блочной сортировки. В результате быстрая сортировка оказалась быстрее блочной.

Мной было изучены основные алгоритмы сортировки данных и освоены на практике. Проверена и исследована работа алгоритмов на различных наборах данных.