**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ЛАБОРАТОРНАЯ РАБОТА №3

По дисциплине: **«Технологии разработки программного обеспечения»**

на тему: ***Моделирование и алгоритмизация как средства проектирования программного обеспечения***

Выполнил: студент группы ИТП-21

Пронуза М.Ю.

Проверил:

Малиновский И.Л.

Гомель 2024

**Цель работы**: научиться проектировать, разрабатывать и отлаживать алгоритмы сортировки.

**Задание:**

1. Описать блок-схему алгоритма согласно варианта (таблица 3).

2. На основе описанной блок-схемы реализовать алгоритм средствами языка Java.

3. Протестировать разработанный алгоритм при помощи модульных тестов.

4. Создать консольное приложение для демонстрации работы алгоритма.

5. Составить отчет о проделанной работе.

**Ход выполнения и результаты работы:**

Текстовый вариант задания для варианта 17 представлен на рисунке 1.



Рисунок 1 – Задание варианта 17

На рисунке 2 представлено консольное приложение программы.

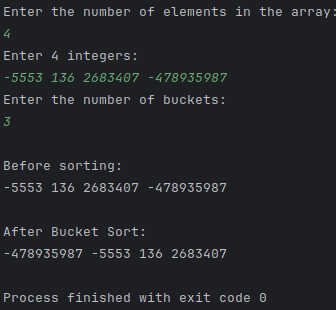


Рисунок 2 – консольное приложение программы

На рисунке 3 представлена блок-схема алгоритма сортировки.

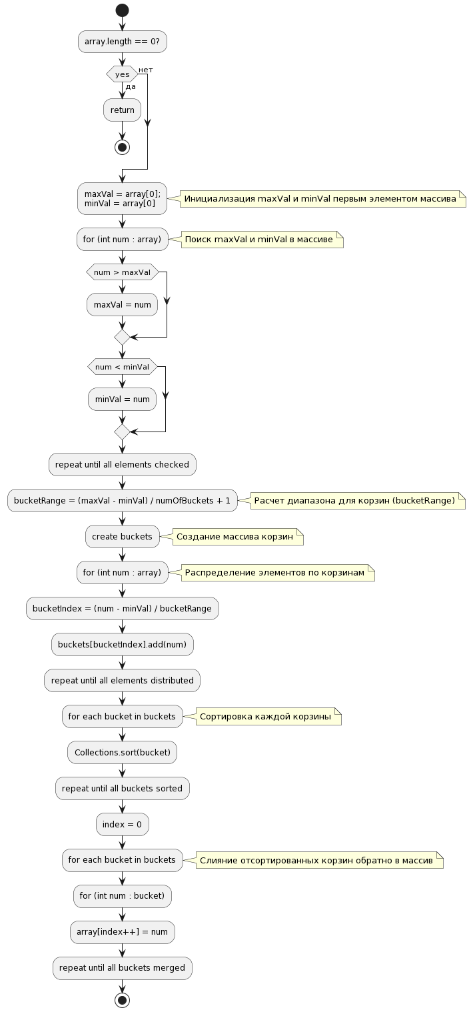


Рисунок 3 – блок-схема алгоритма сортировки

На рисунке 4 представлен результат выполнения модульных тестов.



Рисунок 4 – результат выполнения модульных тестов

Текст класса *BucketSort.java* представлен в приложении А.

Текст класса *BucketSortTest.java* представлен в приложении Б.

Текст класса *Main.java* представлен в приложении В.

**Вывод:** научились проектировать, разрабатывать и отлаживать алгоритмы сортировки.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Текст *BucketSort.java***

package lab3;

import java.util.ArrayList;

import java.util.Collections;

public class BucketSort {

public static void bucketSort(int[] array, int numOfBuckets) {

if (array.length == 0) return;

int maxVal = array[0];

int minVal = array[0];

for (int num : array) {

if (num > maxVal) maxVal = num;

if (num < minVal) minVal = num;

}

int bucketRange = (maxVal - minVal) / numOfBuckets + 1;

ArrayList<Integer>[] buckets = new ArrayList[numOfBuckets];

for (int i = 0; i < numOfBuckets; i++) {

buckets[i] = new ArrayList<>();

}

for (int num : array) {

int bucketIndex = (num - minVal) / bucketRange;

buckets[bucketIndex].add(num);

}

int index = 0;

for (ArrayList<Integer> bucket : buckets) {

Collections.sort(bucket);

for (int num : bucket) {

array[index++] = num;

}

}

}

public static void printArray(int[] array) {

for (int num : array) {

System.out.print(num + " ");

}

System.out.println();

}

}

**ПРИЛОЖЕНИЕ Б**

(обязательное)

**Текст *BucketSortTest.java***

package lab3;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertArrayEquals;

class BucketSortTest {

@Test

void testBucketSortWithPositiveNumbers() {

int[] array = {5, 3, 2, 8, 1, 4};

BucketSort.bucketSort(array, 3);

assertArrayEquals(new int[]{1, 2, 3, 4, 5, 8}, array);

}

@Test

void testBucketSortWithNegativeNumbers() {

int[] array = {-3, -1, -4, -2};

BucketSort.bucketSort(array, 2);

assertArrayEquals(new int[]{-4, -3, -2, -1}, array);

}

@Test

void testBucketSortWithMixedNumbers() {

int[] array = {-5, 4, -3, 2, 1};

BucketSort.bucketSort(array, 5);

assertArrayEquals(new int[]{-5, -3, 1, 2, 4}, array);

}

@Test

void testBucketSortWithSingleElement() {

int[] array = {42};

BucketSort.bucketSort(array, 1);

assertArrayEquals(new int[]{42}, array);

}

@Test

void testBucketSortWithEmptyArray() {

int[] array = {};

BucketSort.bucketSort(array, 3);

assertArrayEquals(new int[]{}, array);

}

@Test

void testBucketSortWithIdenticalElements() {

int[] array = {7, 7, 7, 7};

BucketSort.bucketSort(array, 2);

assertArrayEquals(new int[]{7, 7, 7, 7}, array);

}

@Test

void testBucketSortWithZerosAndNegatives() {

int[] array = {0, -1, -2, 0};

BucketSort.bucketSort(array, 2);

assertArrayEquals(new int[]{-2, -1, 0, 0}, array);

}

@Test

void testBucketSortWithLargeNumbers() {

int[] array = {1000, 2000, 1500, 500};

BucketSort.bucketSort(array, 4);

assertArrayEquals(new int[]{500, 1000, 1500, 2000}, array);

}

@Test

void testBucketSortWithLargeNumberOfBuckets() {

int[] array = {5, 3, 2, 8, 1, 4};

BucketSort.bucketSort(array, 6); // Используем число корзин равное числу элементов

assertArrayEquals(new int[]{1, 2, 3, 4, 5, 8}, array);

}

@Test

void testBucketSortWithMoreComplexArray() {

int[] array = {33, 22, 55, 44, 11, 66, 77};

BucketSort.bucketSort(array, 3);

assertArrayEquals(new int[]{11, 22, 33, 44, 55, 66, 77}, array);

}

@Test

void testBucketSortWithDecimals() {

int[] array = {9, 7, 5, 3, 2, 8, 1, 4, 6};

BucketSort.bucketSort(array, 3);

assertArrayEquals(new int[]{1, 2, 3, 4, 5, 6, 7, 8, 9}, array);

}

@Test

void testBucketSortWithBackwardsArray() {

int[] array = {9, 8, 7, 6, 5, 4, 3, 2, 1};

BucketSort.bucketSort(array, 4);

assertArrayEquals(new int[]{1, 2, 3, 4, 5, 6, 7, 8, 9}, array);

}

@Test

void testBucketSortWithAllZeros() {

int[] array = {0, 0, 0, 0};

BucketSort.bucketSort(array, 2);

assertArrayEquals(new int[]{0, 0, 0, 0}, array);

}

@Test

void testBucketSortWithVerySmallAndLargeNumbers() {

int[] array = {-1000, 2000, 0, -500, 1500, 500};

BucketSort.bucketSort(array, 4);

assertArrayEquals(new int[]{-1000, -500, 0, 500, 1500, 2000}, array);

}

@Test

void testBucketSortWithRandomNumbers() {

int[] array = {42, 24, 12, 56, 78, 67, 89, 23, 45};

BucketSort.bucketSort(array, 5);

assertArrayEquals(new int[]{12, 23, 24, 42, 45, 56, 67, 78, 89}, array);

}

@Test

void testBucketSortWithALongArray() {

int[] array = new int[100];

int[] expected = new int[100];

for (int i = 0; i < 100; i++) {

array[i] = 100 - i;

}

for (int i = 0; i < 100; i++) {

expected[i] = i + 1;

}

BucketSort.bucketSort(array, 10);

assertArrayEquals(expected, array);

}

}

**ПРИЛОЖЕНИЕ В**

(обязательное)

**Текст Main*.java***

package lab3;

import java.util.Scanner;

import java.util.InputMismatchException;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter the number of elements in the array:");

int n = scanner.nextInt();

int[] array = new int[n];

System.out.println("Enter " + n + " integers:");

for (int i = 0; i < n; i++) {

try {

array[i] = scanner.nextInt();

} catch (InputMismatchException e) {

System.out.println("Please enter only integers. Restart the program and try again.");

return;

}

}

int numOfBuckets;

while (true) {

System.out.println("Enter the number of buckets:");

try {

numOfBuckets = scanner.nextInt();

if (numOfBuckets <= 0) {

System.out.println("Number of buckets should be a positive integer. Please try again.");

} else {

break;

}

} catch (InputMismatchException e) {

System.out.println("Please enter a valid integer for the number of buckets.");

scanner.next(); // to consume the incorrect input

}

}

System.out.println("\nBefore sorting:");

BucketSort.printArray(array);

BucketSort.bucketSort(array, numOfBuckets);

System.out.println("\nAfter Bucket Sort:");

BucketSort.printArray(array);

}

}