Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Системы автоматизированного проектирования»

**Отчет**

По лабораторной работе №2

По курсу «Логика и основы алгоритмизации в инженерных задачах»

На тему «Оценка времени выполнения программ»

Выполнил студент гр.20ВВ3

Мирясов Н. А.

Проверили:

Митрохин М. А.

Юрова О. В.

Пенза, 2021

**Цель:** оценить время исполнения работы программы, реализованной разными алгоритмами. Научиться вычислять порядок сложности программы.

**Задание 1:**

Переписал алгоритм, данный в задании, на языке Java.

public static void main(String[] args) {  
 int[] result;  
 int number;  
 int[][] mas1;  
 int[][] mas2;  
 int[] mas = new int[10000];  
 long time1;  
 long time2;  
  
 number = 100;  
 time1 = System.*currentTimeMillis*();  
 mas1 = *rand*(number);  
 mas2 = *rand*(number);  
 result = *result*(number, mas, mas1, mas2);  
 time2 = System.*currentTimeMillis*();  
 System.*out*.println("\n" + (time2 - time1) + " milliseconds");

static int[][] rand(int n) {  
 int[][] mas = new int[n][n];  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < n; j++) {  
 mas[i][j] = (int) (Math.*random*() \* 100);  
 }  
 }  
 return mas;  
}  
  
static int[] result(int number, int[] mas, int[][] mas1, int[][] mas2) {  
 for (int i = 0; i < number; i++) {  
 for (int j = 0; j < number; j++) {  
 mas[i] = mas1[i][j] \* mas2[i][j];  
 }  
 }  
 return mas;  
}

1. O(N2)
2. В number я храню размер матрицы, перед выполнением функции изменяю значение. Провожу перемножение матриц 7 раз, с разными значениями и получаю разное время выполнения метода.
3. number = 100;  
   time1 = System.*currentTimeMillis*();  
   mas1 = *rand*(number);  
   mas2 = *rand*(number);  
   result = *result*(number, mas, mas1, mas2);  
   time2 = System.*currentTimeMillis*();  
   System.*out*.println("\n" + (time2 - time1) + " milliseconds");

number = 200;

number = 400;

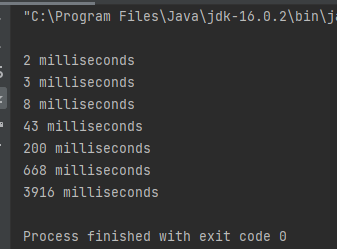
number = 1000;

number = 2000;

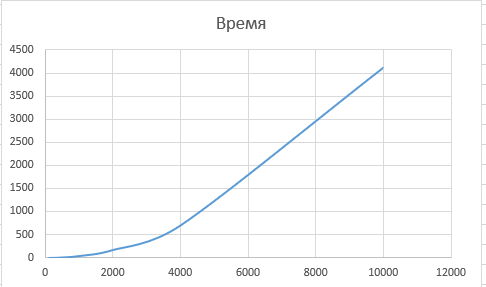
number = 4000;

number = 10000;

Получаю такой результат:



3. Построить график зависимости времени выполнения программы от размера матриц:

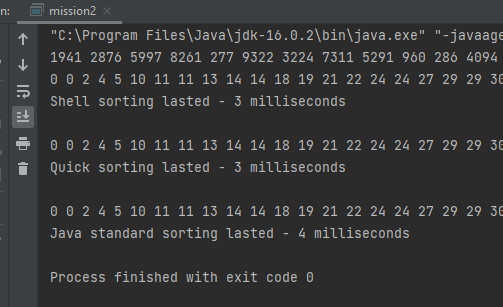


**Задание 2**

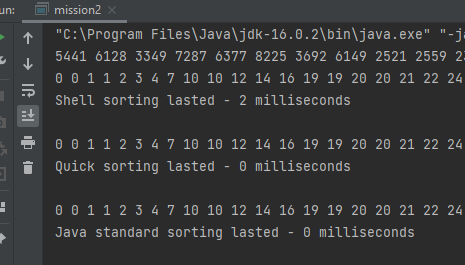
Переписал алгоритмы сортировки.

public static void shellSort(int[] array) {  
 int inner, outer;  
 int temp;  
  
 int h = 1;  
 while (h <= array.length / 3) {  
 h = h \* 3 + 1;  
 }  
  
 while (h > 0) {  
 for (outer = h; outer < array.length; outer++) {  
 temp = array[outer];  
 inner = outer;  
 while (inner > h - 1 && array[inner - h] >= temp) {  
 array[inner] = array[inner - h];  
 inner -= h;  
 }  
 array[inner] = temp;  
 }  
 h = (h - 1) / 3;  
 }  
}  
public static void quickSort(int[] source, int leftBorder, int rightBorder) {  
 int leftMarker = leftBorder;  
 int rightMarker = rightBorder;  
 int pivot = source[(leftMarker + rightMarker) / 2];  
 do {  
 while (source[leftMarker] < pivot) {  
 leftMarker++;  
 }  
 while (source[rightMarker] > pivot) {  
 rightMarker--;  
 }  
 if (leftMarker <= rightMarker) {  
 if (leftMarker < rightMarker) {  
 int tmp = source[leftMarker];  
 source[leftMarker] = source[rightMarker];  
 source[rightMarker] = tmp;  
 }  
 leftMarker++;  
 rightMarker--;  
 }  
 } while (leftMarker <= rightMarker);  
  
 if (leftMarker < rightBorder) {  
 *quickSort*(source, leftMarker, rightBorder);  
 }  
 if (leftBorder < rightMarker) {  
 *quickSort*(source, leftBorder, rightMarker);  
 }  
}

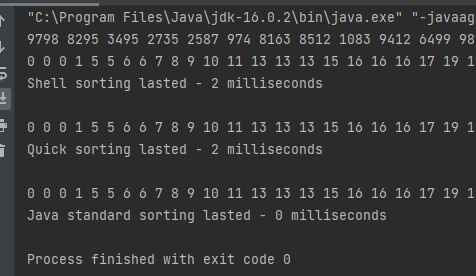
1. Оценил время работы каждого из реализованных алгоритмов на случайном наборе значений массива (10000 элементов).



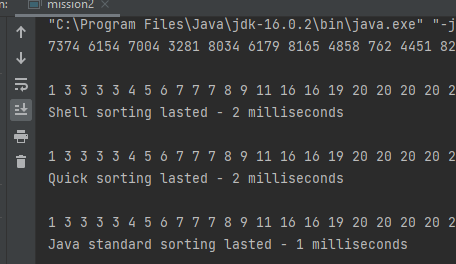
1. Оценил время работы каждого из реализованных алгоритмов на массиве, представляющем собой возрастающую последовательность чисел.



1. Оценил время работы каждого из реализованных алгоритмов на массиве, представляющем собой убывающую последовательность чисел.

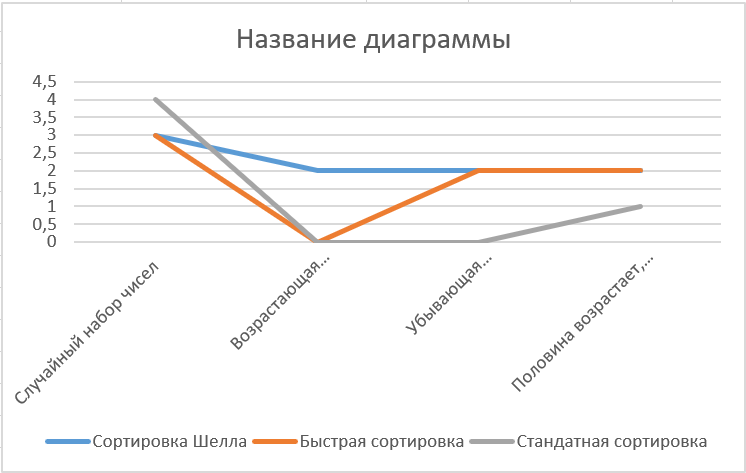


1. Оценил время работы каждого из реализованных алгоритмов на массиве, одна половина которого представляет собой возрастающую последовательность чисел, а вторая, – убывающую.



1. Оценил время работы стандартной функции qsort, реализующей алгоритм быстрой сортировки на выше указанных наборах данных.





Вывод: исходя их результатов выполнения алгоритмов и подсчета их времени, я пришел к выводу, что стандартная сортировка, встроенная в язык Java, быстрее и эффективнее, чем остальные предложенные.