**Лабораторная работа №1 (Отчет)**

**Марченко Максим 9 группа**

**9 ВАРИАНТ**

Составить программу, которая формирует матрицу из n\*n случайных чисел. Определить произведение всех чисел в матрице. Значение n меняется в пределах от 5 до 10 тысяч.

**1.Название и цель работы.**

ИЗУЧЕНИЕ МЕТОДОВ ОЦЕНКИ АЛГОРИТМОВ

***Цель работы:*** Изучение методов оценки алгоритмов и программ и определение временной и емкостной сложности типовых алгоритмов и программ.

**2. Формулы асимптотической и верхней оценки сложности заданного алгоритма.**

***Верхняя оценка алгоритма:*** O(), так как в алгоритме присутствует двойной цикл.

***Асимптотическая оценка алгоритма для лучшего случая:*** Лучший случай будет достигаться тогда, когда первый же элемент матрицы m[i][j]=0, тогда:

Θ(n) = 1(инициализация i) + 1(сравнение n с 0) + 1(инициализация j) + 1(сравнение n с 0) + 3(двойная индексация) + 1(сравнение) + 1 (выход из циклов) = 9

***Асимптотическая оценка алгоритма в худшем случае:*** Худший случай будет тогда, когда матрица вообще не имеет элементов равных нулю, тогда:

Θ(n) = 1(инициализация i) +\* = 1(инициализация i) + 3n(присваивание, проверка условия, приращение) + n\* = 1(инициализация i) + 3n(присваивание, проверка условия, приращение) + n\*(1(инициализация j) + 3n(присваивание, проверка условия, приращение) + n\*(1(проверка условия if) + 3(двойная индексация) + 1 (проверка равенства) + 1(умножение) + 1(присваивание) + 3(двойная индексация)) + 1(выход из циклов)) = 13 + 5n +1

**3. Исходный код программы экспериментальной оценки временной сложности заданного алгоритма для массива большой размерности.**

*long start = System.nanoTime();*

*for (int i = 0; n > i; i++) {*

*for (int j = 0; n > j; j++) {*

*if(m[i][j]==0) break mainloop;*

*result \*= m[i][j];*

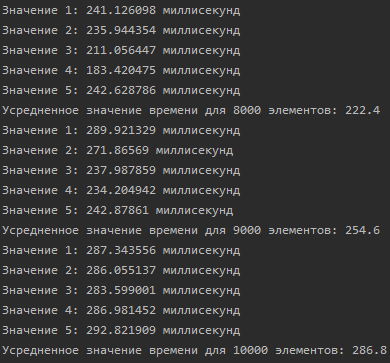
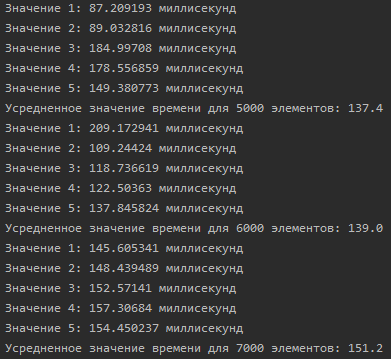
*}*

*}*

*long end = System.nanoTime() - start;*

**4. Значения временной сложности алгоритма, полученные экспериментальным способом, а также количество повторений алгоритма и объем исходных данных, при котором были получены эти значения.**

Высчитывал время выполнения для 5000, 6000, … 10000 элементов. Считал для каждого количества элементов 5 раз и брал усредненное значение.



**5. График зависимости времени выполнения алгоритма от объема исходных данных.**

График должен был получится параболической зависимости, но компьютер слишком мощный для этой задачи, график стремится к прямой. Изгибы можно объяснить скачками выделяемой памяти для данной задачи.

