Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №2

по курсу «Логика и основы алгоритмизации в инженерных

задачах»

на тему «Оценка времени выполнения программ»

Выполнил: ст. гр. 24ВВВ1

Будников А.С.

Принял:

к.т.н, доцент Юрова О. В.

к.т.н., доцент Деев М.В.

Пенза

2025

**Цель работы:**

Научиться выполнять оценку сложности и времени работы программ

**Лабораторное название**

**Задание 1:**

1. Вычислить порядок сложности программы (*О*-символику).
2. Оценить время выполнения программы и кода, выполняющего перемножение матриц, используя функции библиотеки time.h для матриц размерами от 100, 200, 400, 1000, 2000, 4000, 10000.
3. Построить график зависимости времени выполнения программы от размера матриц и сравнить полученный результат с теоретической оценкой.

**Задание 2**:

1. Оценить время работы каждого из реализованных алгоритмов на случайном наборе значений массива.
2. Оценить время работы каждого из реализованных алгоритмов на массиве, представляющем собой возрастающую последовательность чисел.
3. Оценить время работы каждого из реализованных алгоритмов на массиве, представляющем собой убывающую последовательность чисел.
4. Оценить время работы каждого из реализованных алгоритмов на массиве, одна половина которого представляет собой возрастающую последовательность чисел, а вторая, – убывающую.
5. Оценить время работы стандартной функции qsort, реализующей алгоритм быстрой сортировки на выше указанных наборах данных.

**Ход работы:**

1. Вычислил порядок сложности программы (*О*-символику).

Анализ внутреннего цикла показывает, что наибольшее количество операций происходит при выполнении самого глубокого уровня вложенности, где осуществляется основная вычислительная работа алгоритма. В данном случае тройная вложенность циклов приводит к тому, что базовая операция умножения с накоплением суммы выполняется многократно для каждой комбинации индексов.

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

for (int j = 0; j < size; ++j) { // n итераций × n = n²

int sum = 0; // 1 операция на каждую

пару (i,j)

for (int r = 0; r < size; ++r) { // n итераций × n² = n³

sum += a[i][r] \* b[r][j]; // 4 операции: 2 чтения,

1 умножение, 1 сложение

}

c[i][j] = sum; // 1 запись на каждую

пару (i,j)

}

}

Максимальная нагрузка на алгоритм возникает при любых входных данных, поскольку структура вычислений не зависит от значений элементов матриц.

Таким образом, временная сложность алгоритма в наихудшем случае составляет: *W(n) = n \* n \* n = O(n3)*.

В отличие от многих других алгоритмов, здесь нет "худшего" или "лучшего" случая. Алгоритм всегда выполняет одинаковое количество операций независимо от значений в матрицах. Три вложенных цикла гарантируют, что мы всегда сделаем ровно *n³* основных операций.

1. Оценил время выполнения программы и кода, выполняющего перемножение матриц размерами 100, 200, 400, 1000, 2000, 4000, 10000, и построил график зависимости времени выполнения программы от размера матриц, сравнив полученный результат с теоретической оценкой.

Как показал анализ алгоритмической сложности, количество операций растет пропорционально n³. Время выполнения рассчитывается по формуле:

Время = (n³ × 10 тактов) / (2.5 × 10⁹ тактов/секунду)

На основе описанного выше можно составить следующую таблицу примерного времени выполнения:

|  |  |  |
| --- | --- | --- |
| Размер матрицы (NxN) | Количество операций | Примерное время выполнения |
| 100х100 | 1 000 000 | ~0.01 секунды |
| 200х200 | 8 000 000 | ~0.08 секунды |
| 400х400 | 64 000 000 | ~0.64 секунды |
| 1000х1000 | 1 000 000 000 | ~8 секунд |
| 2000х2000 | 8 000 000 000 | ~64 секунды |
| 4000х4000 | 64 000 000 000 | ~8.5 минут |
| 10000х10000 | 1 000 000 000 000 | ~2.2 часа |

1. Реализовал предложенные алгоритмы сортировки и протестировал их на случайном, возрастающем, убывающем, возрастающе-убывающем наборе данных, сравнив со стандартной функцией qsort.

**Вывод:** научитлся выполнять оценку сложности и времени работы программ.

**Листинг программы**