|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **ПРОГРАММНАЯ ИНЖЕНЕРИЯ (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04** Программная инженерия

**Отчет**

|  |  |
| --- | --- |
| **По лабораторной работе №** | 3 |

**Название:**

Обработка разреженных матриц

**Дисциплина:** Типы и структуры данных

|  |  |  |  |
| --- | --- | --- | --- |
| Студент | ИУ7-31Б |  | Н.Ю. Баринов |
|  | (Группа) |  | (И.О. Фамилия) |
|  |  |  |  |

Москва, 2021

**Описание условия задачи**

Разреженная (содержащая много нулей) матрица хранится в форме 3-х объектов:

- вектор A содержит значения ненулевых элементов;

- вектор IA содержит номера строк для элементов вектора A;

- связный список JA, в элементе Nk которого находится номер компонент

в A и IA, с которых начинается описание столбца Nk матрицы A.

1. Смоделировать операцию умножения матрицы и вектора-столбца,

хранящихся в этой форме, с получением результата в той же форме.

2. Произвести операцию умножения, применяя стандартный алгоритм работы с

матрицами.

3. Сравнить время выполнения операций и объем памяти при использовании

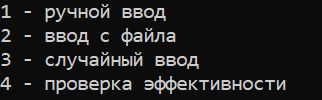
этих 2-х алгоритмов при различном проценте заполнения матриц.**Описание ТЗ**

1. **Описание исходных данных**

Исходными данными являются целочисленные матрицы.

Ввод данных происходит с помощью выбора 1, 2 пункта в меню.

Возможные варианты действий в меню можно увидеть ниже.



**- Значения элементов – целые числа.**

**- Размерность матрицы ограничена оперативной памятью компьютера.**

**- Максимальное значение элемента: 2 147 483 647**

**- Минимальное значение элемента: -2 147 483 647**

**- Элементы, при выборе в меню пункта 1(либо 2) вводятся следующим образом.**

**Пример ввода элемента:**

1 1 – корректный ввод (1 - строка 1 - столбец 0 - значение элемента)

0

Для прекращения ввода элементов вводится комбинация отрицательных чисел.

1. **Описание результатов программы**

Результатом программы будет сформированная база данных с возможными действиями:

1. Посчитать произведение матрицы на столбец (ручной ввод)
2. Посчитать произведение матрицы на столбец (ввод из файла)
3. Посчитать произведение случайной матрицы заданной плотности и столбца заданной плотности
4. Вывести таблицу времени работы обычного перемножения и перемножения разряженных матрицы и столбца

**Описание задачи, реализуемой в программе**

Программа реализует обработку матриц, используя различные способы их хранения в компьютере.

**Описание возможных аварийных ситуаций и ошибок пользователя**

Часть ошибочных ситуаций описана в заголовке Описание исходных данных

Возможный вывод программы в случае ошибки:

- MEMORY ERROR – произошла ошибка при выделении динамической памяти

- DIMENTION ERROR – произошла ошибка при попытке умножения матриц не соответствующих размерностей.

- INPUT ERROR – произошел некорректный ввод.

В случае возникновения ошибочной ситуации программа завершает работу с ненулевым кодом возврата.

**Описание внутренних структур данных**

Программа содержит в себе структуру для хранения матрицы в виде массива, для хранения матрицы в разреженном виде, а также структуру для реализации связного списка.

typedef struct

{

    int rows;

    int columns;

    int \*elements;

} simple\_matrix\_t;

**columns –** количество столбцов в матрице

**rows -** количество строк в матрице

**elements –** массив элементов матрицы

typedef struct

{

    int \*A;

    int \*IA;

    JA\_t \*JA\_head;

    int rows;

    int columns;

    int elements;

} matrix\_t;

**A –** ненулевые значения матрицы

**IA –** номера столбцов для ненулевых значений матрицы

**JA –** связный список, содержащий элементы, с которых начинается каждая строка матрицы. Последним элементом этого связного списка является количество ненулевых элементов матрицы.

**rows –** количество строк в матрице

**columns –** количество столбцов в матрице

**elements –** количество ненулевых элементов

Структура описания односвязного списка

typedef struct JA\_t JA\_t;

struct JA\_t

{

    int JA;

    JA\_t \*next;

};

**Тесты**

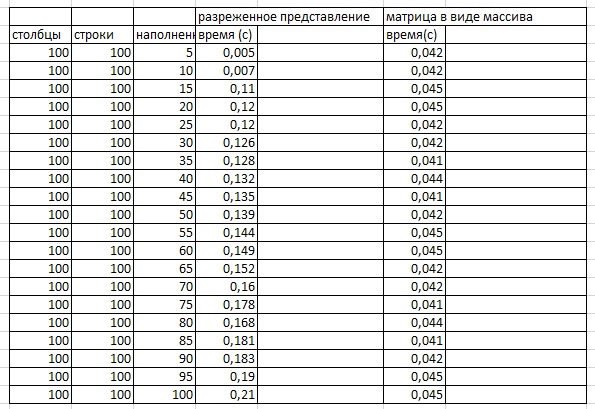
**Положительные тесты**

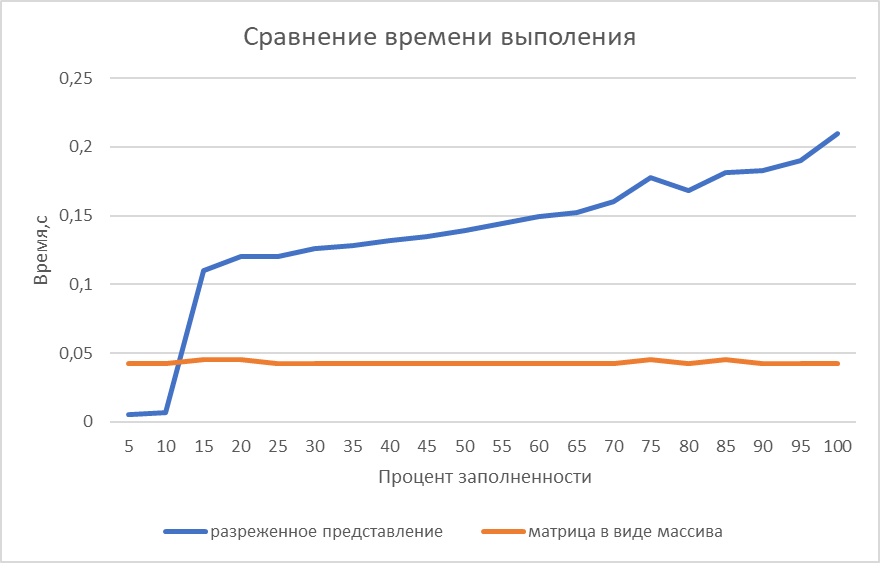
1. Сложение матриц размерности 1 на 1
2. Сложение матриц с отрицательными элементами
3. Сложение матриц, состоящих из нулей
4. Сложение матрицы, сгенерированной автоматически с матрицей, введенной пользователем.

**Негативные тесты**

1. Некорректный ввод количества строк в матрице
2. Некорректный ввод количества столбцов в матрице
3. Некорректный ввод элемента матрицы
4. Некорректный ввод заполненности матрицы
5. Попытка сложения матриц разной размерности

**Временная эффективность и затраты памяти**

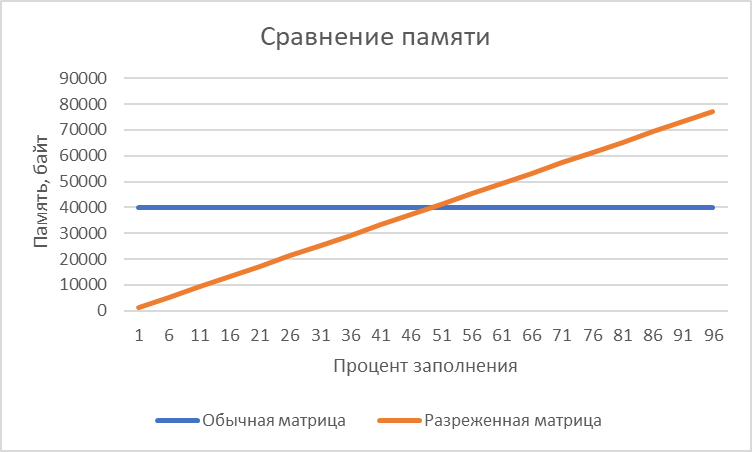
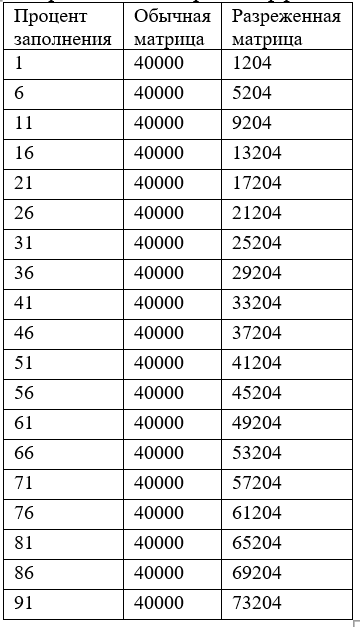




Размерность 100 на 100, как можно увидеть разреженная матрица сравнивается по эффективности при степени разреженности в примерно 12%.

**Затраты по памяти**

Разреженные матрицы эффективнее по памяти при плотности менее 50%.



**Вывод**

В процессе выполнения данной лабораторной работы я ознакомился со способами хранения матриц в компьютере.  
Периодически могут возникнуть такие ситуации, когда программисту необходимо будет выбрать наиболее эффективный способ хранения матрицы, при реализации какой-либо программы.

Оказалось, что хранение разреженных матриц лучше реализовывать в разреженном представлении если плотность матрицы не превышает 12%. Быстродействие для такой плотности увеличиться и количество потребляемой памяти снизится.

На отрезке от ~12% до ~50% программист сам должен выбрать способ хранения, так как время в данном случае у разреженных матриц хуже, но памяти затрачивается меньше. В случае с плотностью матрицы более 50% - лучше себя показала обычный способ хранения матрицы.

**Ответы на контрольные вопросы**

1. Что такое разреженная матрица, какие схемы хранения таких матриц Вы знаете?

Это матрица с большим количеством нулевых элементов. Можно хранить схемой кнута, кольцевой схемой и в разряженном строчном(столбовом) формате.

2. Каким образом и сколько памяти выделяется под хранение разреженной и обычной матрицы?

Для обычной матрицы выделяется память для всех элементов. Для разряженных матриц память выделяется только для данных о ненулевых элементах.

3. Каков принцип обработки разреженной матрицы?

Работа только с ненулевыми элементами.

4. В каком случае для матриц эффективнее применять стандартные алгоритмы обработки матриц? От чего это зависит?

Разряженная матрица эффективна, только когда в массиве много нулевых элементов. Если их незначительно количество, то целесообразнее использовать стандартные алгоритмы.