|  | **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)** |
| --- | --- |

**Отчет**

**по лабораторной работе №3**

**по теме**

**«Обработка разреженных матриц»**

**Вариант 6.**

Дисциплина: Типы и структуры данных

Студент ИУ7-31Б:

Косарев Алексей

Проверила:

Барышникова М.Ю.

Москва, 2021

1. **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 вектора-строки и матрицы, 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.

1. **Описание ТЗ**

* Описание исходных данных

Исходными данными являются матрица и вектор, записанные во входных файлах, названия которых подаются в параметры командной строки.

Также матрица и вектор могут быть введены вручную или случайным образом путем выбора соответствующего пункта меню после запуска программы.

* Описание результатов программы

Меню:

1. Ввод матрицы из файла

2. Ввод матрицы случайным образом

3. Ввод матрицы вручную

4. Ввод вектора из файла

5. Ввод вектора случайным образом

6. Ввод вектора вручную

7. Умножение матрицы на вектор-строку стандартным способом

8. Умножение матрицы на вектор-строку в разреженном формате

9. Вывод матрицы (в двух видах)

10. Вывод вектора (в двух видах)

11. Вывод результата в стандартном формате

12. Вывод результата в разреженном формате

13. Результаты анализа по времени и памяти

14. Удаление матрицы

15. Удаление вектора-строки

0. Выход из программы

1. **Описание задачи, реализуемой в программе**

Программа реализует ввод и вывод матриц и векторов-строк; перевод матрицы и вектора из стандартного формата в разреженный и обратно (для матрицы); умножение вектора-строки на матрицу и вывод результата в стандартном и разреженном виде.

1. **Способ обращения к программе**

Обращение к программе происходит через консоль, путём запуска файла с расширением .exe

(./main.exe) и ввода параметров командной строки (название входных текстовых файлов)

1. **Описание возможных аварийных ситуаций и ошибок пользователя**

#define OK 0 // Нет ошибок

#define INCORRECT\_ARGS 1 // Некорректные аргументы командной строки

#define FILE\_OPEN\_ERR 2 // Ошибка открытия файла

#define FILE\_ERR 3 // Некорректный файл

#define FILE\_CLOSE\_ERR 4 // Ошибка закрытия файла

#define EMPTY\_FILE 5 // Пустой файл

#define INCORRECT\_INPUT 6 // Некорректный ввод

#define IMPOSSIBLE\_TO\_MULTIPLY 7 // Невозможно умножить вектор на матрицу

1. **Описание внутренних структур данных**

#define MAX\_MATRIX\_SIZE 135 // Максимальный размер матрицы

#define MAX\_MATRIX\_ELEMS 18225 // Максимальное количество элементов матрицы

// Вектор в стандартном виде

typedef struct

{

int len;

int data[MAX\_MATRIX\_SIZE];

} vector\_t;

// Вектор в разреженном виде

typedef struct

{

int len;

int data[MAX\_MATRIX\_SIZE]; // Массив ненулевых элементов

int columns[MAX\_MATRIX\_SIZE]; // Номера столбцов ненулевых элементов

int pointer[MAX\_MATRIX\_SIZE + 1];

} vector\_s\_t;

// Матрица в стандартном виде

typedef struct

{

int rows;

int columns;

int data[MAX\_MATRIX\_SIZE][MAX\_MATRIX\_SIZE];

} matrix\_t;

// Матрица в разреженном виде

typedef struct

{

int rows;

int columns;

int data[MAX\_MATRIX\_ELEMS]; // Массив ненулевых элементов

int strings[MAX\_MATRIX\_ELEMS]; // Номера строк ненулевых элементов

int pointer[MAX\_MATRIX\_SIZE + 1]; // Количество ненулевых элементов от каждого столбца до нулевого

} matrix\_s\_t;

1. **Тесты**

Положительные тесты:

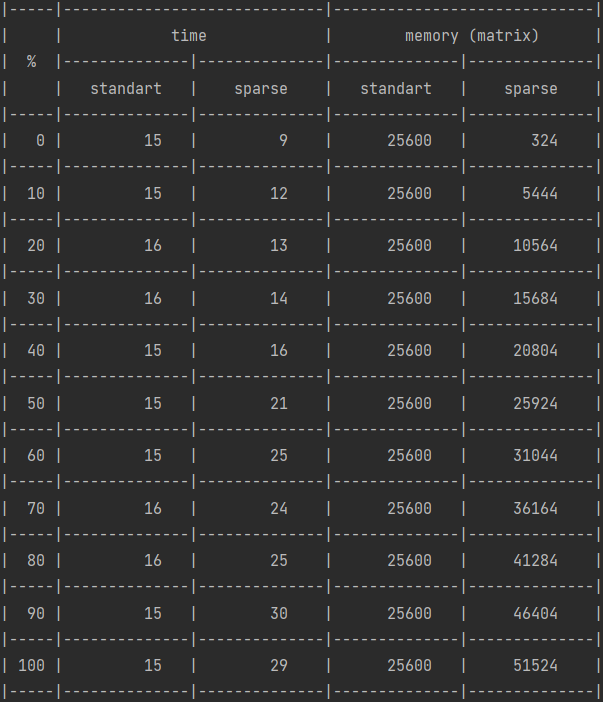
1. Во входном файле ненулевая матрица
2. Во входном файле нулевая матрица
3. Ввод матрицы случайным способом с 0% заполненности
4. Ввод матрицы случайным способом с 50% заполненности
5. Ввод матрицы случайным способом с 100% заполненности
6. Ввод матрицы вручную
7. Во входном файле ненулевой вектор
8. Во входном файле нулевой вектор
9. Ввод вектора случайным способом с 0% заполненности
10. Ввод вектора случайным способом с 50% заполненности
11. Ввод вектора случайным способом с 100% заполненности
12. Ввод вектора вручную
13. Умножение вектора-строки на матрицу с получением ненулевой матрицы
14. Умножение вектора-строки на матрицу с получением нулевой матрицы

Негативные тесты:

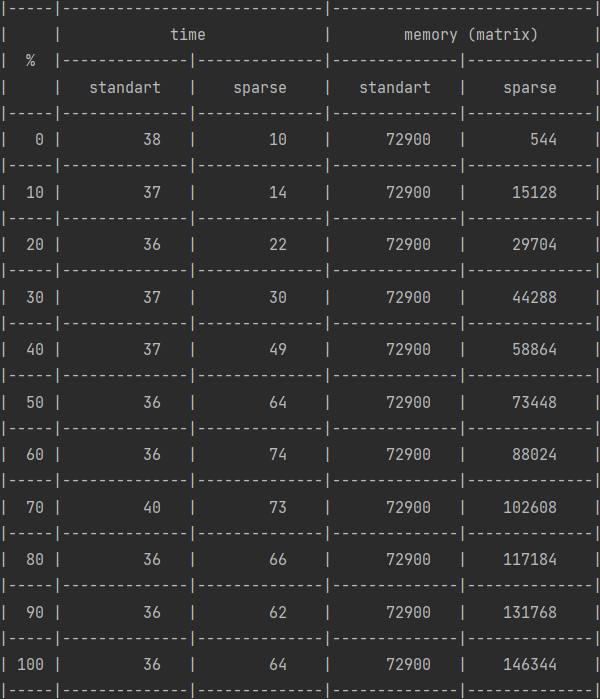
1. Некорректный ввод параметров командной строки
2. Пустой файл
3. Несуществующий файл
4. Во входном файле вместо размерности матрицы - буквы
5. Во входном файле отрицательная размерность матрицы
6. Во входном файле вместо элемента матрицы - буква
7. Во входном файле вместо размерности вектора - буква
8. Во входном файле отрицательная размерность вектора
9. Во входном файле вместо элемента вектора - буква
10. **Временная эффективность и затраты памяти**

% - процент заполненности матрицы

Для размерности матрицы 80х80 и вектора 1х80:



Для размерности матрицы 135х135 и вектора 1х135:



Из результатов анализа использования разных представлений матрицы и разных способов умножения вектора-строки на матрицу видно, что умножение в разреженном формате начинает уступать по времени стандартному умножению после ~40% заполненности. Таким образом умножение в разреженном формате дает временное преимущество при заполненности матрицы в среднем до 40%.

По памяти выгодней использовать разреженный формат записи матрицы, если заполненность матрицы меньше 50%, иначе выгодней записывать матрицу в стандартном виде.

1. **Вывод**

В процессе выполнения данной лабораторной работы я изучил различные варианты представления матриц (стандартный и разреженный).

Проведя анализ обработки матриц по времени и памяти, я понял, что при заполненности матрицы до ~40% выгоднее (по времени) использовать разреженный формат ее представления, а по памяти выгодней использовать разреженный формат записи только при заполненности до 50%.

1. **Ответы на контрольные вопросы**
2. Что такое разреженная матрица, какие схемы хранения таких матриц Вы знаете?

Разреженная матрица – это матрица с преимущественно нулевыми элементами.

Схемы хранения матриц:

* Связная схема хранения матриц, предложенная Кнутом
* Схема, предложенная Чангом и Густавсоном, называемая "разреженный строчный формат"

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

Для обычной матрицы память выделяется для хранения всех элементов. Количество памяти, выделяемой под разряженную матрицу зависит от количества ненулевых элементов.

1. Каков принцип обработки разреженной матрицы?

В основу принципов обработки разреженных матриц легли алгоритмы работы только с ненулевыми элементами.

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

Разреженная матрица эффективна, только когда в массиве много нулевых элементов. Если их незначительное количество, то целесообразнее использовать стандартный формат представления и обработки матриц.