| 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 Программное обеспечение ЭВМ и информационные технологии

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3 «ОБРАБОТКА РАЗРЕЖЕННЫХ МАТРИЦ»**

Студент Щербина Михаил Александрович

Группа ИУ7 – 35Б

Приняла ???

[**Описание условия задачи**](#_v3r7gltbjvag) **3**

[Указания к выполнению работы](#_50dozklogsex) 3

[При тестировании программы необходимо:](#_6rn4o3sml75l) 4

[Описание технического задания](#_9g9dqs8974iy) 4

[Входные данные:](#_2u7tn8nlsxmb) 5

[Выходные данные:](#_y9qkcb6l7fkk) 5

[Функции программы:](#_azziwhd0kqdn) 5

[Обращение к программе:](#_risoxfhzxd5p) 6

[Аварийные ситуации:](#_u00bt7841yvu) 6

[Описание структуры данных](#_2juabhbb8hxe) 6

[Описание алгоритма](#_lus9ezsmkgy8) 8

[Набор тестов](#_sa0z9mmb5fn3) 8

[Вывод](#_yc70p3sqn0cx) 12

# Описание условия задачи

Разработать программу умножения. Предусмотреть возможность ввода данных, как с клавиатуры, так и использования заранее подготовленных данных. Матрицы хранятся и выводятся в форме трех объектов. Для небольших матриц можно дополнительно вывести матрицу в виде матрицы. Величина матриц - любая (допустим, 1000\*1000). Сравнить эффективность (по памяти и по времени выполнения) стандартных алгоритмов обработки матриц с алгоритмами обработки разреженных матриц при различной степени разреженности матриц и различной размерности матриц.

# Описание технического задания

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.

## Указания к выполнению работы

Все логически завершенные фрагменты алгоритма (ввод, вывод, обработка и т.п.) необходимо оформить как подпрограммы. При разработке интерфейса программы следует предусмотреть:

указание формата и диапазона вводимых данных,

указание операции, производимой программой,

наличие пояснений при выводе результата,

указание формата выводимых данных

возможность заполнения разреженных матриц вручную (даже при большой размерности, например, 1000\*1000) и автоматически с разным процентом разреженности.

## При тестировании программы необходимо:

o проверить правильность ввода o проконтролировать правильность вывода данных (т.е. их соответствие требуемому формату);

o проверить правильность выполнения операций; o обеспечить вывод сообщений при отсутствии входных данных («пустой ввод»); o обеспечить вывод сообщений при нулевых результате или вывод нулевого результата при ненулевом входе;

o обеспечить возможность ввода данных и вывода результата как при малых матрицах, так и при больших (например, 1000 \* 1000).

o сравнить время выполнения стандартного алгоритма обработки матриц и алгоритма обработки разреженных матриц при различной заполненности матриц (от 1 элемента до того количества нулей (в %), при котором становится неэффективно использование алгоритма сокращенного умножения).

o сравнить объем требуемой памяти для реализации стандартного алгоритма обработки матриц и алгоритма обработки разреженных матриц при различном проценте заполнения матриц и при различном их размере.

Следует также протестировать программу при полной загрузке системы, то есть при полном заполнении матриц. Программа должна адекватно реагировать на неверный ввод, пустой ввод и выход за границы матрицы или вектора.

## Описание технического задания

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.

## Входные данные:

Файлы с данными:

Обычная (dense) матрица

В header написаны размеры. Далее перечислены элементы

Разреженная (sparse) матрица

В header записаны размеры, кол-во ненулевых. Далее матрица записана в координатной форме.

Целое число, представляющее собой номер команды: целое число в диапазоне от 0 до n-команд.

## Выходные данные:

Результат -В координатном виде для обеих типов матриц

Выводятся ненулевые элементы, каждый с новой строки вида

“A[i] [j] = num ”

Где i - номер строки, j - номер столбца, num -значение

Количественная характеристика сравнения умножения матриц разного вида.

## Функции программы:

1. Ввод матрицы
2. Ввод вектора
3. Умножение матрицы метода на вектор
4. Профилирование
5. Вывод полезной информации
6. Выход

## Обращение к программе:

Запускается программа через терминал. Так же можно собрать программу используя cmake и запустить ее. ./app.exe

## Аварийные ситуации:

1. Некорректный ввод номера команды.

На входе: число, большее чем 12 или меньшее, чем 0.

На выходе: игнор ввода

2. Некорректный ввод номера команды.

На входе: пустой ввод.

На выходе: Игнор ввода

3. Файл пуст.

На входе: пустой файл.

На выходе: Игнор комманды, исходный буфер не изменяется

4. Выполнение вывода матрицы/вектора до заполнения матрицы/вектора.

На входе: номер команды для умножения

На выходе: сообщение «Input matrix and vector first!»

6. Неверный ввод при ручном заполнении матрицы/вектора.

На входе: строка, содержащая некорректные значения (дробь/буква).

На выходе: Сообщение об ошибке, игнор ввода

7. Неверный ввод из файла.

На входе: неверный / пустой файл, матрица с размерами / кол-вом ненулевых <= 0, кол-во элементов не соответствует размерам

На выходе: сообщение об ошибке, игнор ввода

8. Умножение матриц и вектора не допустимых размеров.

На входе: матрица и вектор.

На выходе: «Error: bad dimensions for vector and matrix!.»

## 

## 

## Описание структуры данных

struct cons\_t // Связный список

{

int value;

struct cons\_t \*next;

};

Value - значение ноды

Next - указатель на следующую ноду списка

typedef struct {

int rows;

int columns;

int \*\*data;

} matrix\_t;

Rows - кол-во строк

Columns -кол-во столбцов

Data - указатели на строки матрицы

typedef struct

{

int rows;

int columns;

int n;

vector\_t A; // A

vector\_t IA; // rows with non zero elements

cons\_t \*JA;

} sparse\_t;

Rows - кол-во строк

Columns - кол-во столбцов

N - кол-во ненулевых элементов

A - вектор с ненулевыми значениями

IA - вектор с индексами строк

JA - связный список (см задание)

union any\_matrix\_u

{

sparse\_t sparse;

matrix\_t dense;

};

Объединение двух типов для полиморфизма.

typedef struct

{

union any\_matrix\_u kind;

int type;

} any\_matrix\_t;

Тип включающий в себя либо разреженную либо плотную матрицу.

Type - тип (перечисление).

Kind - полиморфическое объединение.

## Описание алгоритма

1. Выводится меню данной программы.
2. Пользователь вводит номер команды из предложенного меню.
   1. Если производится умножение, типы преобразуются перед операцией.
3. Пока пользователь не введет 123 (выход из программы), ему будет предложено вводить номера команд и выполнять действия по выбору.

## Набор тестов

|  | Название теста | Пользовательский ввод | Результат |
| --- | --- | --- | --- |
| 1 | Некорректный ввод команды | 1435 | Игнор ввода вывод меню заново |
| 2 | Пустой ввод | - | Игнор ввода (сообщение об ошибке) |
| 3 | Файл пустой | Пустой файл | Игнор записи, игнор считанной матрицы в target, вывод сообщения об ошибке |
| 4 | Некорректный ввод из файла (размеры <= 0 или кол-во элементов не совпадает) | Некорректный файл | Игнор записи, игнор считанной матрицы в target, вывод сообщения об ошибке |
| 5 | Некорректный файл (не существует) | Несуществующий path | Игнор записи, игнор считанной матрицы в target, вывод сообщения об ошибке |
| 6 | Некорректный файл (не читается / не пишется) | Path к файлу | Игнор записи, игнор считанной матрицы в target, вывод сообщения об ошибке |
| 7 | При чтении с stdin неверный ввод | Плохой stdin | Игнор записи, игнор считанной матрицы в target, вывод сообщения об ошибке |
| 8 | Умножение, матрицы не введены | Комманда, матрицы | Вывод сообщения об ошибке «Input matrix and vector first!» |
| 9 | Умножение, матрицы введены, размеры некорректны | Комманда, матрицы | «Error: bad dimensions for vector and matrix!.» |
| 10 | Вывод нулевой матрицы | Нулевая матрица | Вывод \*empty matrix\* |
| 11 | Умножение матрицы на вектор (верное) | Матрица и вектор, команда | Умножается, выводится |
| 12 | Ввод матрицы (верный) | Матрица вводится, команда | Матрица сохраняется |
| 13 | Вывод информации | Команда | Информация выводится |

| 14 | Ввод неверных координат для sparse матрицы | Неверные координаты | Ввод прекращается, выдается ошибка |
| --- | --- | --- | --- |
| 15 | Ввод рандомной матрицы матрицы, sparsity < 0 | Неверный параметр | Ввод прекращается, ошибка |
| 16 | Ввод разреженной матрицы с 0 ненулевыми | 0 ненулевых | Матрица принимается |

**Matrix vector product time in ns**

**Matrix is 1000x1000, vector is 1x1000**

% dense sparse

1 29611 183

6 18293 5555

11 28642 9482

16 29544 13344

21 28547 17252

26 28582 19675

31 28749 23359

36 18046 27454

41 18246 31052

46 28680 34870

51 28711 39003

56 28519 44509

61 28607 47686

66 28694 52425

71 28806 55410

76 28753 61843

81 28951 63501

86 18145 67517

91 29078 75524

96 28979 77405

**Matrix size in bytes**

**Matrix is 10000x10000**

full dense sparse

1 400090016 16120188

6 400090016 56084412

11 400090016 96006804

16 400090016 135992428

21 400090016 175986316

26 400090016 215900852

31 400090016 255835948

36 400090016 295856836

41 400090016 335805884

46 400090016 375732348

51 400090016 415692108

56 400090016 455674852

61 400090016 495610076

66 400090016 535572796

71 400090016 575563412

76 400090016 615438468

81 400090016 655540012

86 400090016 695452484

91 400090016 735482052

96 400090016 775408132

Ответы на контрольные вопросы

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

Разреженная матрица — это матрица заполненная большим кол-вом нулей

Схемы хранения матрицы: связанная схема хранения (с помощью линейных связанных списков), кольцевая связанная схема хранения, двунаправленные стеки и очереди, диагональная схема хранения, строчной формат, столбцовый формат.

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

Под обычную матрицу (N – количество строк, M – количество столбцов) выделяется N \* M \* sizeof(тип) ячеек памяти.

Для разреженной матрицы количество ячеек памяти зависит от способа. В случае разреженного формата требуется количество ячеек с ненулевыми элементами M. Нужно определить M \* sizeof(тип) + M \* sizeof(int) значений памяти. Также нужна память под список с индексом строк (кол-во элементов < M).

3. Каков принцип обработки разреженной матрицы?

При обработке разреженной матрицы мы работаем только с ненулевыми элементами. Тогда количество операций будет пропорционально количеству ненулевых элементов.

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

Эффективнее применять стандартные алгоритмы выгоднее при достаточно малых размерах и очень большом количестве ненулевых элементов.

## Вывод

Выгодно использовать разреженные матрицы при большом кол-ве нулей (или при ~40% ненулевых элементов) в матрице, особенно при умножении на вектор-столбец, так как тратится меньший объем памяти (примерно в 3-5 раза меньше стандартного представления). Время выполнения может быть меньше при больших матрицах и большой разреженности.

Но уже при ~30% ненулевых элементов лучше использовать обычный алгоритм - время становится таким же или меньше. Если важна память, то так же при 50% ненулевых элементом выгоднее использовать стандартный алгоритм, так как затраты по памяти значительно меньше у обычного представления матрицы.

Хранить матрицы в разреженном виде выгодно только если они содержат большое количество нулей, во всех остальных случаях такой вид хранения проигрывает по памяти (из-за большого кол-ва метаданных).

Умножение “маленьких” разреженных матриц очень неэффективно.

## Использованные Функции

### Связный список

cons\_t \*cons\_new(int value); конструктор

void cons\_add(cons\_t \*self, int value); добавление элемента

void cons\_delete(cons\_t \*self); деструктор

int cons\_get(cons\_t \*self, int idx); геттер

int cons\_next(cons\_t \*self); след. элемент

size\_t cons\_size(cons\_t \*self); размер списка в памяти

### Матрица

matrix\_t matrix\_new(int n, int m); конструктор

void matrix\_delete(matrix\_t \*self); деструктор

matrix\_t matrix\_vector\_product(matrix\_t \*self, matrix\_t \*vector); задание

void matrix\_print(matrix\_t \*self); вывод

size\_t matrix\_size(matrix\_t \*self); размер в памяти

Разреженная матрица

void sparse\_delete(sparse\_t \*self); деструктор

void sparse\_print(sparse\_t \*self); кол-во элементов

sparse\_t sparse\_vector\_product(sparse\_t \*self, sparse\_t \*vector); задание

size\_t sparse\_size(sparse\_t \*self); размер в памяти

Динамический массив

vector\_t vector\_new(int capacity); конструктор

void vector\_delete(vector\_t \*self); деструктор

vector\_t vector\_realloc(vector\_t \*self); перевыделение

vector\_t vector\_from\_arr(int n, int \*arr); конструктор

void vector\_add(vector\_t \*self, int el); добавление элемента

int vector\_get(vector\_t \*self, int idx); геттер

size\_t vector\_size(vector\_t \*self); размер в памяти

Any Matrix

void any\_matrix\_as(any\_matrix\_t \*self, int type); преобразование типа

void any\_matrix\_delete(any\_matrix\_t \*self); удаление