Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное 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
ПО ДИСЦИПЛИНЕ «ТИПЫ И СТРУКТУРЫ ДАННЫХ»

Обработка разреженных матриц

Вариант № 5

Студент: Бондарева **В. А.**

Группа: ИУ**7-34Б**

Преподаватель: **Никульшина Т. А.**

2025 г.

Условие задачи

Разработать программу умножения двух разреженных матриц (хранящихся в формате 3-х объектов) в форматах CSR и CSC. Полученный результат должен быть получен в CSR формате. Разработать программу для перемножения двух матриц, хранящихся в обычной форме. Сравнить время выполнения операций и объем памяти при использовании этих двух алгоритмов при различном проценте заполнения матриц. Сравнить эффективность двух подходов.

Описание ТЗ

Исходные данные

Исходными данными программы являются опция меню, представленная выборкой опций в консоли, ручной ввод матрицы, т.е. ввод размера матрицы, количества ее значимых элементов, ввод элементов матрицы. Также для выполнения пунктов меню, связанных с чтением данных из файла, исходными данными являются файлы.

На исходные данные наложены следующие ограничения:

1. Ограничение по формату вводимых данных.
2. Ограничение на ввод размеров матриц. Размеры матриц – положительные целые числа.
3. Ограничение на разрешение и содержание файлов для чтения и записи.

Результат

Результатом выполнения программы являются матрицы в обычном формате (плотные матрицы), матрицы в форматах CSR и CSC, а также таблица сравнения алгоритмов перемножения двух матриц.

Способ обращения к программе

Пользователь обращается к программе при помощи исполняемого файла app.exe.

Возможные аварийные ситуации и ошибки пользователя

Все возможные аварийные ситуации и ошибки пользователя описаны типом status_t, включающим в себя коды возврата программы:

1. Ошибка ввода/вывода (некорректный и/или не валидный ввод).
2. Неверно указано количество чего-либо.
3. Ошибка при работе с памятью.
4. Ошибка при попытке перемножения двух матриц.

5. Неизвестные ошибки (не классифицируемые).

Описание внутренних структур данных

```
typedef struct {
    int *A;
    int *JA;
    int *IA;
    size_t rows;
    size_t cols;
    size_t non_zero;
} CSR_matrix_t;

typedef struct {
    int *B;
    int *IB;
    int *JB;
    size_t rows;
    size_t cols;
    size_t non_zero;
} CSC_matrix_t;

typedef struct {
    int **data;
    size_t rows;
    size_t cols;
} dense_matrix_t;

typedef struct {
    int value;
    size_t row;
    size_t col;
} matrix_element_t;
```

Листинг 1. Внутренние структуры данных.

Структура CSR_matrix_t хранит матрицу в CSR формате. Она имеет следующие поля:

- A: массив значений матрицы, то есть массив ее ненулевых элементов
- JA: массив индексов столбцов ненулевых элементов
- IA: массив «индексации» строк
- rows: количество строк матрицы
- cols: количество столбцов матрицы
- non-zero: количество ненулевых элементов матрицы

Структура CSC_matrix_t хранит матрицу в CSC формате. Она имеет следующие поля:

- B: массив значений матрицы, то есть массив ее ненулевых элементов
- IB: массив индексов строк ненулевых элементов
- JB: массив «индексации»
- rows: количество строк матрицы
- cols: количество столбцов матрицы

- non-zero: количество ненулевых элементов матрицы

Структура dense_matrix_t хранит матрицу в обычном формате (т.е. как плотную). Она имеет следующие поля:

- data: двумерный массив значений матрицы, хранящий в себе $rows*cols$ элементов

- rows: количество строк матрицы

- cols: количество столбцов матрицы

Структура matrix_element_t хранит информацию об одном конкретном элементе матрицы, такую как:

- value: значение данного элемента матрицы

- row: индекс строки, на которой находится данный элемент

- col: индекс столбца, на котором находится данный элемент

Алгоритм

```
// освобождение матриц
status_t free_all_matr(void);
status_t free_dense_matrix(dense_matrix_t *dense_matrix);
status_t free_csr_matr(void);
status_t free_csc_matr(void);

// выделение память под матрицы
status_t allocate_dense_matrix(dense_matrix_t *dense_matr, size_t n, size_t
m);
status_t allocate_csr_matrix(size_t non_zero_quantity, size_t
rows_quantity);
status_t allocate_csc_matrix(size_t non_zero_quantity, size_t
cols_quantity);

// умножение матриц
status_t multiply_csr_and_csc(void);
status_t multiply_dense_matrices(void);

// чтение матриц из файла
status_t read_csc_from_file(CSC_matrix_t *csc_matrix);
status_t read_csr_from_file(CSR_matrix_t *csr_matrix);
status_t read_dense_from_file(dense_matrix_t *dense_matr);

// ввод данных
status_t input_cur_menu_opt(menu_option_t *cur_menu_opt);
status_t input_any_matrix(void);
status_t input_matrix_dimensions(size_t *rows, size_t *cols);

// сравнение эффективности и случайное заполнение
status_t compare_matrix_multiplication(void);
status_t fill_random_csr(CSR_matrix_t *mat, size_t num_random);
status_t fill_random_csc(CSC_matrix_t *mat, size_t num_random);
status_t fill_random_dense(dense_matrix_t *mat, size_t num_random);
```

Листинг 2. Сигнатуры основных функций.

Помимо основного алгоритма программы (main.c) основополагающими алгоритмами для реализации заданного функционала являются:

- алгоритмы ввода данных (ввод с консоли, чтение из файла)
- алгоритм перемножения плотных матриц
- алгоритм перемножения CSR и CSC матриц
- алгоритм сравнения эффективности двух способов перемножения

Рассмотрим каждый алгоритм.

```c
#include "data.h"
#include "matrix.h"
#include "input.h"
#include "output.h"
#include "process.h"
#include <stdbool.h>

int main(void)
{
    status_t exit_code = SUCCESS_CODE;  // основной код возврата из
программы
    status_t menu_opt_processing_status = SUCCESS_CODE;  // код возврата для
обработки ошибок внутри цикла
    menu_option_t cur_menu_opt;  // выбранная опция меню
    char buf;

    print_menu();
    exit_code = input_cur_menu_opt(&cur_menu_opt);
    while (cur_menu_opt != 0 && exit_code == SUCCESS_CODE)
    {
        menu_opt_processing_status = proccess_menu_option(cur_menu_opt);
        print_result(menu_opt_processing_status);
        print_menu();
        while ((buf = getchar()) != '\n' && buf != EOF);
        exit_code = input_cur_menu_opt(&cur_menu_opt);
    }

    print_final_common_result(exit_code);

    free_all_matr();

    return exit_code;
}
```

Листинг 3. Основной алгоритм программы.

Основной алгоритм:

1. Печатается пользовательское меню, считывается выбранная пользователем опция.
2. Пока опция корректна и не выбран выход, выполняется цикл обработки.
3. В зависимости от опции выполняется соответствующее действие.
4. Печатается результат выполнения действия, меню снова отображается.
5. После завершения цикла выводится итоговый результат работы программы.
6. Освобождаются все выделенные ресурсы и возвращается код завершения.

Листинг 4. Основной алгоритм.

1. Производится проверка входных данных.
2. Старый результат умножения освобождается.
3. Выделяется память под новый результат исходя из максимально возможной заполненности.
4. Для каждого ненулевого элемента по его индексам ищется «пара», производится перемножение соответствующих значений. Если результат не равен нулю, то его значение и позиция добавляются к набору результирующих ненулевых элементов.
5. Число ненулевых элементов сохраняется в структуре результата.
6. В случае ошибок освобождаются выделенные ресурсы, возвращается код ошибки.

Листинг 5. Алгоритм перемножения CSR и CSC матриц.

1. Производится проверка входных данных.
2. Освобождается память результата.
3. Память под результирующую матрицу перевыделяется.
4. Последовательно вычисляется каждый элемент результата через сумму произведений соответствующих элементов.
5. В случае ошибок освобождаются выделенные ресурсы, возвращается код ошибки.

Листинг 6. Алгоритм перемножения двух плотных матриц.

1. Считываются размеры матриц.
2. Освобождаются все ранее выделенные матрицы.
3. Выделяется память под две плотные матрицы требуемого размера.
4. Для каждого значения заполненности:
    a. Вычисляется количество ненулевых элементов.
    b. Выделяется память под разреженные CSR и CSC матрицы.
    c. По 100 раз матрицы случайно заполняются, перемножаются в плотном и разреженном формате, измеряется и усредняется время выполнения.
    d. Рассчитывается память, занимаемая плотными и разрежёнными матрицами.

e. Результаты для текущей заполненности выводятся в таблицу, память под текущие разрежённые матрицы очищается.
5. Освобождаются все выделенные матрицы.
6. Возвращается код статуса выполнения.

Листинг 7. Алгоритм функции измерения эффективности.

Тесты

Позитивные тесты

Номер теста	Описание	Ожидаемый результат
№1	Ввод двух матриц 3x3 стандартного вида вручную, перемножение. Исходные данные: 1 0 2 5 0 0 0 3 0 x 6 0 8 4 0 0 0 7 0	Успешное перемножение двух матриц. Результирующая матрица: 5 14 0 18 0 24 20 0 0
№2	Ввод матрицы CSR 3x3, ввод матрицы CSC 3x3, перемножение. 1 0 2 5 0 0 0 3 0 x 6 0 8 4 0 0 0 7 0	Успешное перемножение двух матриц. Результирующая CSR матрица: A: 5, 14, 18, 24, 20 JA: 0, 1, 0, 2, 0 IA: 0, 2, 4, 5
№3	Случайный ввод двух матриц 3x3 стандартного вида, перемножение. Исходные данные: 0 0 86 0 86 0 67 0 0 x 78 0 50 16 0 80 0 31 0	Успешное перемножение двух матриц. Результирующая матрица: 0 2666 0 0 5762 0 0 3856 0
№4	Случайный ввод двух матриц 3x3 в форматах CSR и CSC, перемножение. CSR матрица: Кол-во значимых элементов: 4 A: 50, 100, 81, 63 JA: 0, 1, 0 0 IA: 0, 2, 3, 4 CSC матрица: Кол-во значимых элементов: 4 B: 98, 69, 67, 94 IB: 2, 1, 1, 2 JB: 0, 1, 2, 4	Успешное перемножение двух матриц. Результирующая CSR матрица: Кол-во значимых элементов: 2 A: 6900, 6700 JA: 1, 2 IA: 0, 2, 2, 2
№5	Чтение из файла обычной матрицы. Содержание файла dense_matrix.txt: 2 3 1 2 3 4 5 6	Успешный ввод матрицы: 1 2 3 4 5 6
№6	Чтение из файла CSR матрицы. Содержание файла csr_matrix.txt: 3 3 5 1 2 4 2 6 0 1 1 1 2 0 2 3 5	Успешный ввод матрицы: Кол-во значимых элементов: 2 Кол-во строк: 3 Кол-во столбцов: 3 A: 1, 2, 4, 2, 6, JA: 0, 1, 1, 1, 2 IA: 0, 2, 3, 5
№7	Чтение из файла CSC матрицы. Содержание файла csc_matrix.txt: 4 4 6 5 1 3 4 7 8 0 2 3 1 2 0 0 2 3 5 6	Успешный ввод матрицы: Кол-во значимых элементов: 6 Кол-во строк: 4 Кол-во столбцов: 4

		B: 5, 1, 3, 4, 7, 8 IB: 0, 2, 3, 1, 2, 0 IA: 0, 2, 3, 5, 6

Таблица 1. Позитивные тесты.

Негативные тесты

Номер теста	Описание	Ожидаемый результат
№1	Ввод некорректных размеров матрицы (нечисловые значения).	Возврат ERR_RANGE. «Произошла ошибка обработки массива или обработки количества чего-либо :(»
№2	Ввод некорректных размеров матрицы (отрицательные значения).	Возврат ERR_RANGE. «Произошла ошибка обработки массива или обработки количества чего-либо :(»
№3	Ввод некорректных размеров матрицы (нули).	Возврат ERR_RANGE. «Произошла ошибка обработки массива или обработки количества чего-либо :(»
№4	Ввод некорректных значений матрицы (нечисловые значения).	Возврат ERR_IO. «Произошла ошибка ввода/вывода :(»
№5	Ввод некорректных значений матрицы (нецелые значения).	Возврат ERR_IO. «Произошла ошибка ввода/вывода :(»
№6	Ввод некорректной опции меню (выход из диапазона).	Возврат ERR_RANGE. «Произошла ошибка обработки массива или обработки количества чего-либо :(»
№7	Ввод некорректной опции меню (нечисловое значение).	Возврат ERR_IO. «Произошла ошибка ввода/вывода :(»
№8	Ввод имени несуществующего файла.	Возврат ERR_FILE. «»
№9	Попытка чтения файла с некорректным содержанием.	Возврат ERR_IO. «Произошла ошибка ввода/вывода :(»
№10	Попытка перемножения матриц с несоответствующими размерами.	Возврат ERR_RANGE. «Произошла ошибка обработки массива или обработки количества чего-либо :(»

Таблица 2. Негативные тесты.

Оценка эффективности

Сравнение времени перемножения двух матриц разных форматов.

Заполнение	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Рязреж.	46651	116161	212269	322534	487527	580162	717950	833078	1016704	1146677
Плотные	421046	380250	372836	370900	410402	372300	374027	368168	367287	368440

Таблица 3. Сравнение времени перемножения двух матриц 100 на 100 для разных форматов.

Заполнение	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Рязреж.	315940	887568	1688208	2393199	3157516	4178916	5216877	5095407	3883977	4712622
Плотные	2412494	2423426	2400435	2395187	2401313	2396206	2427662	1921512	1224223	1216127

Таблица 4. Сравнение времени перемножения двух матриц 150 на 150 для разных форматов.

Заполнение	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Рязреж.	812213	2315448	4102192	5941541	7344043	5572009	7157643	9242176	11294519	13797143
Плотные	5587066	5582653	5608879	5584017	4925655	2805559	2786561	2796359	2763004	2805933

Таблица 5. Сравнение времени перемножения двух матриц 200 на 200 для разных форматов

Заполнение	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Рязреж.	1670916	5086925	8629626	7213259	8952625	12810993	16968768	21963074	26198323	31035754
Плотные	11680851	11768372	11752343	6659833	5756468	5792168	5793478	5824344	5851836	5860151

Таблица 6. Сравнение времени перемножения двух матриц 250 на 250 для разных форматов.

Заполнение	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Рязреж.	3204291	9055581	8980418	12213859	18223737	25835642	33300128	41662240	50234055	59588269
Плотные	23939574	24165371	13725726	12028460	12162286	12061108	11919985	11939279	11974060	11965751

Таблица 7. Сравнение времени перемножения двух матриц 300 на 300 для разных форматов.

Сравнение объемов использованной памяти двух матриц разных форматов.

Заполнение	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Рязреж.	16904	32904	48904	64904	80904	96904	112904	128904	144904	160904
Плотные	81648	81648	81648	81648	81648	81648	81648	81648	81648	81648

Таблица 8. Сравнение объемов использованной памяти при разных формах хранения матриц 100 на 100.

Заполнение	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Рязреж.	37304	73304	109304	145304	181304	217304	253304	289304	325304	361304
Плотные	182448	182448	182448	182448	182448	182448	182448	182448	182448	182448

Таблица 9. Сравнение объемов использованной памяти при разных формах хранения матриц 150 на 150.

Заполнение	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Рязреж.	65704	129704	193704	257704	321704	385704	449704	513704	577704	641704
Плотные	323248	323248	323248	323248	323248	323248	323248	323248	323248	323248

Таблица 10. Сравнение объемов использованной памяти при разных формах хранения матриц 200 на 200.

Заполнение	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Рязреж.	102104	202104	302104	402104	502104	602104	702104	802104	902104	1002104
Плотные	504048	504048	504048	504048	504048	504048	504048	504048	504048	504048

Таблица 11. Сравнение объемов использованной памяти при разных формах хранения матриц 250 на 250.

Заполнение	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Рязреж.	146504	290504	434504	578504	722504	866504	1010504	1154504	1298504	1442504
Плотные	724848	724848	724848	724848	724848	724848	724848	724848	724848	724848

Таблица 12. Сравнение объемов использованной памяти при разных формах хранения матриц 300 на 300.

Достоинство использования для хранения и произведения операций над матрицами в форматах CSR и CSC лучше всего проявляется для разреженных матриц. Чем плотнее матрицы, над которыми производится операция перемножения, тем более эффективен для них «стандартный» алгоритм умножения «строка на столбец». Исходя из таблиц 3-7 в случаях, когда разреженные матрицы содержат до 5% значимых элементов от общей своей массы, указанные форматы хранения CSR и CSC имеют ряд преимуществ, таких как:

1. Экономия памяти. Всего для хранения разреженной матрицы понадобится $nonzero*sizeof(elem)+nonzero*sizeof(size\_t)+((rows+1)*sizeof(size\_t$ байт. Чем меньше процент заполненности матрицы, тем существеннее выигрыш по памяти для CSR/CSC форматов по сравнению с обычным форматом хранения.
2. Ускорение умножения. При перемножении CSR/CSC матриц исключаются операции умножения на нулевые элементы, что уменьшает общее количество операций и позволяет сократить общее время умножения.

Однако CSR/CSC форматы хранения имеют некоторые недостатки. Так, реализация операций удаления или добавления элементов в подобную матрицу трудоемко, ресурсоемко, так как все три поля структуры взаимосвязаны: операция модификации требует изменений во всех связанных массивах индексов. Также, если матрица достаточно плотная, то преимущества CSR/CSC форматов обращаются в недостатки: чтобы хранить абсолютно плотную матрицу в подобном представлении понадобится в три раза больше затрат по сравнению с обычным способом хранения.

В результате эффективность использования CSR/CSC матриц обусловлена конкретной задачей и ее условиями. Такой подход позволяет экономить память и значительно сокращать время перемножения разреженных матриц, однако требует скрупулезной реализации, адаптации и оптимизации алгоритмов.

Выводы

Время выполнения операции перемножения двух матриц CSR/CSC определяется размером и разреженностью матриц. Разрежённые матрицы, представленные CSR/CSC форматами, существенно экономят память по сравнению с матрицами в стандартном представлении, особенно при большом количестве нулевых элементов.

Однако с увеличением размера или заполненности разрежённой матрицы преимущества использования CSR/CSC форматов становятся менее выраженными, иногда перерастая в недостатки.

Контрольные вопросы

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

   Разреженная матрица — это матрица, преимущественно состоящая из нулевых элементов. Для её эффективного хранения используются различные схемы. Связная схема Кнута хранит ненулевые элементы вместе с их индексами и системой указателей на соседние элементы в строке и столбце. Однако наиболее распространённым и эффективным подходом является разреженный строчный формат (Чанг, Густавсон), который требует меньше памяти и оптимизирован для операций умножения, сложения и решения систем линейных уравнений.

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

   Для хранения разреженной матрицы в специальных оптимизированных форматах (например, CSR и CSC) выделяется значительно меньше памяти по сравнению со стандартным представлением матрицы, так как специальные форматы предполагают хранение информации только о ненулевых элементах матрицы. Таким образом для хранения одной CSR матрицы понадобится $nonzero*sizeof(elem)+nonzero*sizeof(size\_t)+((rows+1)*sizeof(size\_t$ байт, в то время как для хранения обычной матрицы понадобится $rows*cols*sizeof(elem)$ байт.

3. *Каков принцип обработки разреженной матрицы?*

   Обработка разреженных матриц имеет принципиальные отличия от работы с плотными матрицами. Ключевая идея заключается в исключении операций над нулевыми элементами, что позволяет значительно сократить объем вычислений и потребление памяти.

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

   Классические матричные алгоритмы демонстрируют максимальную эффективность при работе с плотными матрицами. Целесообразность их применения зависит от уровня заполненности матрицы и ее размера: с ростом количества ненулевых элементов специальные форматы хранения теряют свои преимущества. Для матриц с высокой плотностью данных или незначительной разреженностью стандартные методы обработки часто оказываются оптимальным выбором. Также стандартные методы обработки являются популярным решением в случаях, когда матрицы содержат сравнительно немного элементов, т.к. временные затраты на

перемножение подобных матриц являются несущественными, а в реализации перемножения с помощью специальным типов хранения нет необходимости.