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 ПО ДИСЦИПЛИНЕ:
ТИПЫ И СТРУКТУРЫ ДАННЫХ

ОБРАБОТКА РАЗРЕЖЕННЫХ МАТРИЦ

Студент **Попов Ю.А.**

Группа **ИУ7-32Б**

Название предприятия **НУК ИУ МГТУ им. Н. Э. Баумана**

Студент _____ **Попов Ю.А.**

Преподаватель _____ **Силантьева А.В.**

2024

Оглавление

Описание условия задачи

Смоделировать операцию сложения двух матриц, хранящихся в этой форме, с получением результата в той же форме. Произвести операцию сложения, применяя стандартный алгоритм работы с матрицами. Сравнить время выполнения операций и объем памяти при использовании этих 2-х алгоритмов при различном проценте заполнения матриц.

Описание технического задания

Файл с данными:

На первой строке файла хранятся 2 целых числа - количество строк и столбцов. В последующих строках лежат целые числа - элементы матрицы.

Входные данные:

Пользователь вводит целое число-команду от 0 до 7. Программа выполняет операции, в зависимости от выбранной опции. Чтобы корректно завершить программу, необходимо воспользоваться соответствующей командой. Необязательным параметром выступает аргумент для запуска приложения, который указывает программе на ручной ввод файла для заполнения матрицы

Выходные данные:

В зависимости от выбранного действия, пользователь получает разные выходные данные. Общими остаются уведомления об успешности или ошибки выполнения действия, и вывод информации на экран.

Действие программы:

Программа работает до тех пор, пока пользователь не введет число 0. Программное обеспечение позволяет ввести команды от 0 до 7, в случае если команда выходит за этот диапазон, или не подходит по типу данных, выводится соответствующее сообщение, и программа завершает работу. Разработанное программное обеспечение работает с матрицами и позволяет провести такие действия как: вывод и сложение обычных матриц, а также в режиме Compressed Sparse Column (CSC)

Обращение к программе:

Программа запускается через терминал, в командной строке, с помощью команды ./app.exe. При обращении с программой, пользователь может использовать параметр запуска, в котором можно указать флаг –manual. При указанном флаге программа даст возможность пользователю выбрать исходные файлы, иначе программа будет использовать файл по умолчанию. После запуска. пользователю будет доступно меню программы.

Аварийные ситуации:

1. ERR_FILE_INPUT – Ошибка, при чтении из файла.
2. ERR_RANGE_OPERATION – Ошибка при выборе операции.
3. ERR_INPUT_STRING – Ошибка при вводе строки.
4. ERR_STRING_OVERFLOW – Переполнение строки при вводе строки.
5. ERR_EMPTY_STRING - Пустая строка
6. ERR_FILENAME - Неправильное имя файла
7. ERR_RANGE_MATRIX_SIZE - Неверный размер матрицы, например символ вместо числа
8. ERR_INPUT_INTEGER_NUMBER - Ошибка при вводе целого числа

9. ERR_MATRIX_SIZE_NOT_EQ - Ошибка, при сложении, размер матриц не равен
10. ERR_MEMORY_ALLOCATION - Ошибка при выделении памяти

Во всех нештатных ситуациях, программа уведомляет о месте, где произошла ошибка.

Описание структуры данных

Структура для записи обычной матрицы

rows_count - количество строк

columns_count - количество столбцов

values - массив указателей на строки матриц.

```
typedef struct
{
    size_t rows_count;
    size_t columns_count;
    int **values;
} matrix_t;
```

листинг 1. Структура matrix_t

Структура для записи структуры в формате CSC.

values - массив значений матрицы

row_indices - массив индексов строк

col_ptr - массив указателей на столбцы

rows_count - количество строк в матрице

columns_count - количество столбцов в матрице

nz_count - количество ненулевых элементов

```
typedef struct
{
    int *values;
    int *row_indices;
    int *col_ptr;

    size_t rows_count;
```

```
    size_t columns_count;
    size_t nz_count;
} csc_t;
```

листинг 2. Структура hostel_t

Описание алгоритма

1. После запуска программы выводит приглашение к вводу и меню.
2. Программа принимает номер операции, и запускает соответствующие действия в соответствии с номер команды
3. Матрицы складываются двумя методами
4. Если пользователь вводит 0, то программа завершнается.

Описание основных функций

```
void print_matrix(const matrix_t matrix);
```

Функция реализует вывод обычной матрицы на экран

Входные параметры: матрица

Выходные параметры: вывод матрицы на экран

```
int fill_matrix_from_file(FILE *file, matrix_t *matrix);
```

Функция реализует запись матрицы из файла

Входные параметры: указатель на файловую переменную, указатель на матрицу

Выходные параметры: заполненная матрица

Возвращаемый результат: код возврата.

```
int random_fill_default_matrix(matrix_t *matrix);
```

Функция заполняет матрицу случайным образом.

Входные параметры: указатель на матрицу

Выходные параметры: заполненная матрица

Возвращаемый результат: код возврата.

```
int add_matrix_t(matrix_t matrix_1, matrix_t matrix_2, matrix_t *res);
```
Функция складывает матрицы

Входные параметры: две матрицы и указатель на результат.

Выходные параметры: результирующая матрица

Возвращаемый результат: код возврата.

```
csc_t convert_to_csc(matrix_t matrix);
```
Функция конвертирует матрицу в формат CSC.

Входные параметры: матрица

Возвращаемый результат: матрица в формате CSC.

```
void print_csc_matrix(const csc_t matrix);
```
Функция выводит матрицу в формате CSC

Входные параметры: матрица

```
int sum_csc_matrix(csc_t matrix_1, csc_t matrix_2, csc_t *result);
```
Сложение двух матриц в формате CSC.

Входные параметры: две матрицы и указатель на результирующую матрицу.

Выходные параметры: результирующая матрица равная сумме двух матриц.

Возвращаемый результат: код возврата.

Набор тестов

№	Описание теста	Пользовательский ввод	Вывод
1	Некорректный ввод операции	abc	Ошибка при выборе операции. Можно число от 0 до 7
2	Некорректный ввод операции. Выход за диапазон	10000	Ошибка при выборе операции. Можно число от 0 до 7
3	Некорректный ввод операции.	-100	Ошибка, введите

	Отрицательное число		число от 1 до 12
4	Корректный файл с матрицей		
5	В записи матрицы есть символы	2 2 a 1 2 3	Ошибка, при чтении из файла
6	Ошибка в записи размеров матрицы	ф 2 2 1 2 3	Ошибка, при чтении из файла
7	Количество элементов матрицы меньше ее размера	10 10 2 1 2 3	Ошибка, при чтении из файла
8	Сложение матриц	2 2 1 2 3 4 2 2 5 6 7 8	2 2 6 8 10 12
9	Вывод двух матриц	2 2 1 2 3 4 2 2 5 6 7 8	2 2 1 2 3 4 2 2 5 6 7 8
10	Вывод матриц в формате CSC	2 2 1 2 3 4 2 2 5 6 7 8	Первая матрица Values 1 3 2 4 Row Indices 0 1 0 1 Column Pointers 0 2 4 Вторая матрица Values 5 7 6 8 Row Indices 0 1 0 1 Column Pointers 0 2 4
10	Сложение матриц в формате CSC	2 2 1 2 3 4 2 2 5 6 7 8	Values 6 10 8 12 Row Indices 0 1 0 1 Column Pointers 0 2 4

Оценка эффективности

Временные замеры (нс)

Временные замеры осуществлялись с помощью библиотеки time.h. Каждый запуск, таблица заново считывалась из памяти. Каждый размер запускался пока общий RSE (Relative standard error) не будет меньше 1%. В таблице 1 представлено время работы алгоритма (в наносекундах) для обычной матрицы и матрицы в формате CSC

Размер	Процент заполнения	Обычная матрица	Матрица в формате CSC	Отношение скорости выполнения
10	0	412,54	73,21	5,64
	10	407,47	324,30	1,26
	20	414,16	455,57	0,91
	50	414,17	917,12	0,45
	100	416,50	695,39	0,60
60	0	11 757,58	255,13	46,08
	10	11 135,71	6 832,86	1,63
	20	10 685,50	13 025,14	0,82
	50	10 793,00	28 664,77	0,38
	100	10 725,00	23 533,75	0,46
110	0	42 510,97	468,79	90,68
	10	35 236,50	21 471,00	1,64
	20	35 036,00	42 767,77	0,82
	50	35 156,33	118 528,79	0,30
	100	37 113,42	118 445,14	0,31
310	0	276 963,36	1 214,92	227,97
	10	281 274,05	168 757,59	1,67

	20	285 252,17	328 259,12	0,87
	50	274 664,62	970 639,50	0,28
	100	272 357,74	917 409,17	0,30
460	0	626 584,83	1 753,48	357,34
	10	650 902,73	363 327,57	1,79
	20	635 586,90	725 524,84	0,88
	50	600 257,04	2 123 231,33	0,28
	100	601 738,31	2 076 172,50	0,29

Таблица 1. Сравнение времени работы исследуемых алгоритмов.

Объем занимаемой памяти (байты)

Объем, занимаемой памяти матрицами каждого типа представлен в таблице 2.

Размер	**Процент заполнения**	**Обычная матрица**	**Матрица в формате CSC**	**Отношение размера обычной к CSC**
10	0	424	156	2,72
	10	424	236	1,80
	20	424	316	1,34
	50	424	556	0,76
	70	424	716	0,59
	100	424	956	0,44
60	0	14424	356	40,52
	10	14424	3236	4,46
	20	14424	6116	2,36
	50	14424	14756	0,98

	70	14424	20516	0,70
	100	14424	29156	0,49
110	0	48424	556	87,09
	10	48424	10236	4,73
	20	48424	19916	2,43
	50	48424	48956	0,99
	70	48424	68316	0,71
	100	48424	97356	0,50
410	0	672424	1756	382,93
	10	672424	136236	4,94
	20	672424	270716	2,48
	50	672424	674156	1,00
	70	672424	943116	0,71
	100	672424	1346556	0,50

Таблица 2. Оценка занимаемой памяти, относительно размера.

Результаты были обработаны с помощью дополнительной программы на языке Python. Был построен 3-х мерный график зависимости скорости/памяти от размера матрицы и процента ее заполнения. Графики приведены на рисунке 1 и 2.

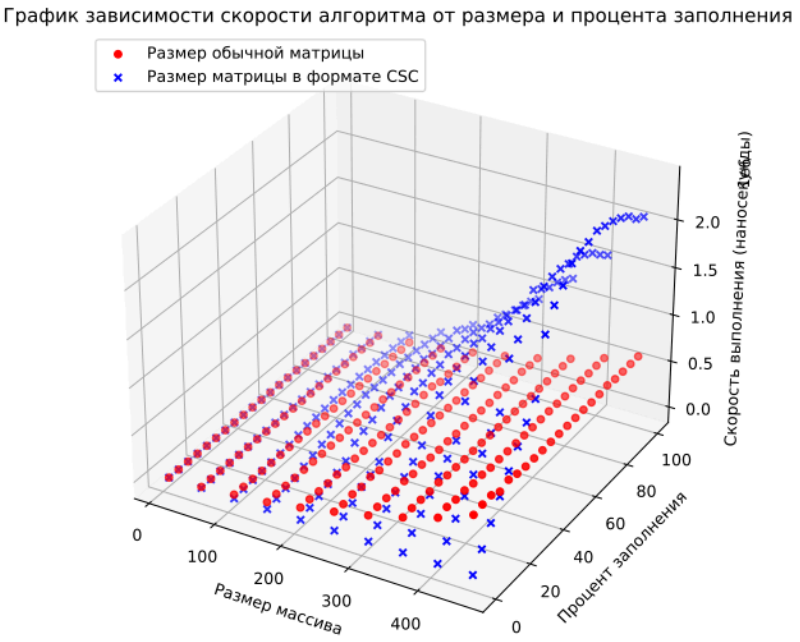

Рис. 1. График зависимости скорости работы алгоритма сложения матриц от размера и процента их заполнения.

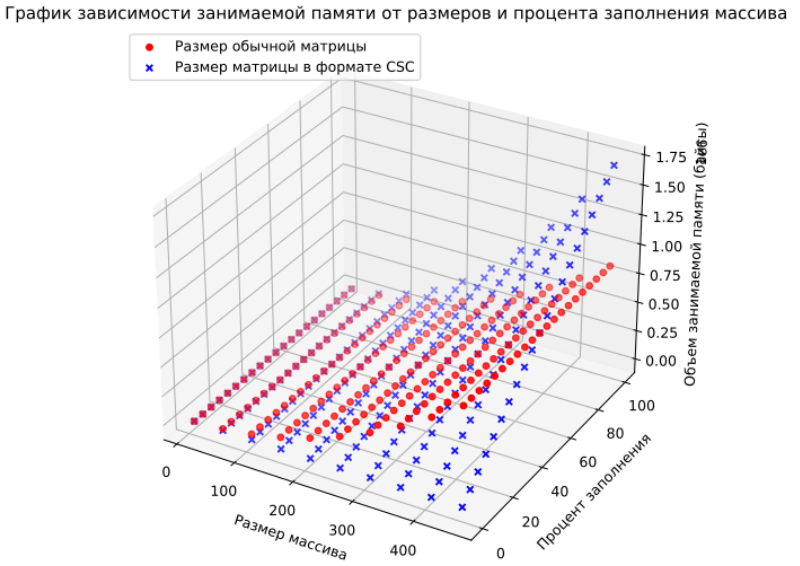

Рис. 2. График размера матриц от размера и процента их заполнения.

По графикам можно заметить, что метод CSC матриц выгоден по памяти при использовании матриц, заполненных меньше чем на 40%, и по скорости при заполнении меньше 20%. Я посчитал корреляцию между размером и относительной эффективностью алгоритма с помощью метода Пирсона. У меня получилось значение **0.117213**, это означает что существует слабая положительная линейная связь, то есть чем больше размер матрицы, тем эффективнее становится алгоритм.

Ответы на контрольные вопросы

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

Разреженная матрица - многомерный массив, содержащий большое количество нулевых элементов. Я знаю несколько способов хранения:

- Обычная матрица
- COO (Coordinate List) - Координатный способ, предполагает хранение трех массивов: массив значений и 2 массива соответствующих индексов строки и столбца.
- CSR (Compressed Sparse Row) - Представляет собой хранение, массива значений, индексов столбца и указателей на строку
- CSC (Compressed Sparse Column) - Аналогичен CSR, только вместо указателей на строку хранятся указатели на столбец.

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

В моей программе памяти под оба типа матриц выделяется динамически. Обычная матрица представляет собой массив указателей на

строку. Под обычную матрицу выделяется M * N * sizeof(int) байт памяти, где M - количество строк, а N - количество столбцов. Под CSC матрицу выделяется 2 * NZ * sizeof(int) + (N + 1) * sizeof(int), где NZ - количество ненулевых элементов, а N - количество столбцов.

3. Каков принцип обработки разреженной матрицы?

Разреженные матрицы содержат большое кол-во нулей, поэтому в сохраняются только ненулевые элементы. Это дает выигрыш по памяти и скорости по сравнению с алгоритмами обработки обычных матриц.

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

По результатам эксперемента было выяснено, что применять стандартный алгоритмы выгоднее, если количество значимых элементов более 20%, а также если размеры матриц очень маленькие.

ВЫВОД

При написании эффективных алгоритма для работы с матрицами, необходимо уделить внимание методам обработки разреженных матриц. Метод хранения Compressed Sparse Column работает быстрее при заполненности матрицы менее 20% и занимает меньше место до 40% заполнения. К тому же, чем больше размер матрицы, тем меньше процент заполненности необходим для того, чтобы стандартный способ сложения матриц проигрывал по скорости методу обработки разреженных матриц.