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»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

«ОБРАБОТКА РАЗРЕЖЕННЫХ МАТРИЦ»

по курсу «Типы и структуры данных»

Вариант 1

Студент: Писаренко Дмитрий Павлович

Группа: ИУ7-34Б

Студент _____ Писаренко Д.П.

подпись, дата *фамилия, и.о.*

Преподаватель _____ Барышникова М.Ю.

подпись, дата *фамилия, и.о.*

Цель работы

Цель работы - реализовать алгоритмы обработки разреженных матриц, сравнить эффективность использования этих алгоритмов (по времени выполнения и по требуемой памяти) со стандартными алгоритмами обработки матриц при различном процентном заполнении матриц ненулевыми значениями и при различных размерах матриц.

Условие задачи

Разработать программу умножения или сложения разреженных матриц. 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.

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

- вектор A содержит значения ненулевых элементов;
- вектор JA содержит номера столбцов для элементов вектора A;
- связный список IA, в элементе Nk которого находится номер компонент в A и JA, с которых начинается описание строки Nk матрицы A.

1. Смоделировать операцию сложения двух матриц, хранящихся в этой форме, с получением результата в той же форме.

2. Произвести операцию сложения, 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.

Техническое задание

Исходные данные

Выбор действия: целое число от 0 до 3.

0. Выход из программы

Не требует ввода от пользователя.

1. Сложение матриц, хранящихся в заданной форме:

Вводится количество строк, количество столбцов, затем способ заполнения матрицы (число от 1 до 2). При заполнении матрицы вручную необходимо указать количество ненулевых элементов, далее ввести их координаты (номера строк и столбцов) и значения; при автоматическом заполнении матрицы необходимо ввести процент заполнения матрицы ненулевыми элементами.

2. Сложение матриц, применяя стандартный алгоритм:

Вводится количество строк, количество столбцов, затем способ заполнения матрицы (число от 1 до 3). При заполнении вручную ненулевых элементов необходимо указать количество ненулевых элементов, далее ввести их координаты (номера строк и столбцов) и значения; при заполнении вручную всей матрицы необходимо ввести значения всех элементов; при автоматическом заполнении матрицы необходимо ввести процент заполнения матрицы ненулевыми элементами.

3. Сравнение времени выполнения этих двух алгоритмов

Не требует ввода от пользователя.

Структуры данных

```c
typedef struct matrix
{
    int **matrix;
    int rows;
    int cols;
} matrix_t;
```

```
int **matrix – матрица значений
int rows – кол-во строк
int cols – кол-во столбцов
```

Описание полей структуры matrix_t

```c
typedef struct sparse_matrix
{
    int *a;
    int *ja;
    int *ia;
    int nonzero_elems;
    int rows;
    int cols;
} sparse_matrix_t;
```

```
int *a – вектор, хранящий кол-во ненулевых элементов в строке
int *ja – вектор, хранящий номера столбцов ненулевых элементов
int *ia – вектор, хранящий значения ненулевых элементов
int nonzero_elems – количество ненулевых элементов
int rows – кол-во строк
int cols – кол-во столбцов
```

Описание полей структуры sparse_matrix_t

Способ обращения к программе

Работа с программой осуществляется с помощью консоли.

Сборка осуществляется с помощью команды **make build**

Запуск выполняется с помощью команды **./app.exe**

Дальнейшая работа производится с помощью меню:

```
dimasxt@dimasxt-VirtualBox:~/TaSD/lab_03$ ./app.exe
Меню:
1. Сложение матриц, хранящихся в заданной форме
2. Сложение матриц, применяя стандартный алгоритм
3. Сравнение времени выполнения этих двух алгоритмов
0. Выход
Введите номер команды:
```

Описание алгоритма

1. Вводится количество строк и столбцов для первой матрицы, пользователю предлагается выбрать способ заполнения матрицы (вручную или автоматически).

2. При выборе ввода вручную пользователю предлагается ввести количество ненулевых элементов, их номера строк и столбцов и значения, при выборе ввода автоматически пользователю предлагается ввести процент заполнения матрицы ненулевыми элементам.

3. Первые два пункта повторяются для второй матрицы. Разрешается заполнять матрицы разными способами (первую автоматически, вторую вручную или наоборот).

4. Производится сложение: для начала в вектор “a” результирующей матрицы добавляется количество элементов в векторах “a” первых двух матриц. Затем проверяется на ноль количество элементов отдельно каждой матрицы: если в какой-то из матриц нет элементов в этой строке, то в результирующую добавляются все значения векторов “ja” и “ia” из другой матрицы. Если же в обеих матрицах есть элементы в данной строке, производится еще проверка на совпадения, чтобы не добавлять один и тот же элемент два раза, после чего производится добавление значений векторов “ja” и “ia” в результирующую матрицу по возрастанию столбцов.

Тестирование

Позитивные тесты

#	Входные данные	Выходные данные	Результат
1	Ключ = 1 Валидные значения количества строк и столбцов у обеих матриц, заполняемых автоматически	Вектора a, ja и ia для первых двух матриц + вектора a, ja и ia для результирующей матрицы	Ожидание следующего ключа
2	Ключ = 1 Валидные значения количества строк и столбцов у обеих матриц, валидное количество ненулевых элементов, валидные номера строк, столбцов, значения ненулевых элементов	Вектора a, ja и ia для первых двух матриц + вектора a, ja и ia для результирующей матрицы	Ожидание следующего ключа
3	Ключ = 2 Валидные значения количества строк и столбцов у обеих матриц, заполняемых автоматически, матрица размером меньше, чем 10x10	Первые две матрицы и результирующая в обычном виде; Вектора a, ja и ia для первых двух матриц + вектора a, ja и ia для результирующей матрицы	Ожидание следующего ключа
4	Ключ = 3	Затраченные память и время для матриц разных размеров	Ожидание следующего ключа
5	Ключ = 0	Поток вывода пустой	Код возврата 0

Негативные тесты

#	Входные данные	Выходные данные	Результат
1	Номер команды = 6	Номер команды - цифра от 0 до 3	Код возврата 2
2	Отрицательное количество строк	Количество строк - натуральное число	Код возврата 4
3	Количество строк двух матриц не совпадает	В матрицах A и B разное количество строк/столбцов	Код возврата 22
4	Выбор способа заполнения матрицы = 9	Способ ввода - число от 1 до 2	Код возврата 9
5	Процент заполнения матрицы = 146	Процент заполнения - целое число от 1 до 100	Код возврата 11
6	Одинаковые координаты у введенных значений	Для одной и той же строки были введены одинаковые номера столбцов	Код возврата 31
7	Количество ненулевых элементов больше размера матрицы	Количество ненулевых элементов - число от 1 до d, где d – размер матрицы	Код возврата 14
8	В значении ненулевого элемента вводится 0	Ненулевой элемент должен быть не равен 0	Код возврата 16
9	Номер строки больше количества строк матрицы	Номер строки - число от 0 до n, n – последняя строка	Код возврата 18
10	Вместо номера строки вводятся буквы	Номер строки - целое число	Код возврата 17

Таблицы с результатами измерения времени и памяти

Время замерялось при 50 выполнениях функции.

Время в таблицах указано в мс.

Матрица	Размер	Процент заполнения ненулевыми элементами				
		10	20	30	40	50
Разреженная	100x100	1	2	3	3	5
Обычная	100x100	4	5	4	4	4
Разреженная	300x300	17	40	55	70	88
Обычная	300x300	54	56	55	54	57
Разреженная	500x500	57	105	188	208	267
Обычная	500x500	176	173	176	178	175
Разреженная	700x700	111	219	307	339	393
Обычная	700x700	359	371	337	353	372
Разреженная	900x900	187	321	401	453	510
Обычная	900x900	458	431	491	461	478

Матрица	Размер	Процент заполнения ненулевыми элементами				
		60	70	80	90	100
Разреженная	100x100	6	9	8	8	11
Обычная	100x100	6	5	4	5	4
Разреженная	300x300	95	108	110	120	134
Обычная	300x300	61	53	56	58	57
Разреженная	500x500	278	303	312	325	344
Обычная	500x500	181	169	175	188	181
Разреженная	700x700	425	462	504	508	534
Обычная	700x700	381	359	364	393	355
Разреженная	900x900	552	577	630	673	654
Обычная	900x900	467	422	501	477	478

Выделенная память в таблицах указана в байтах.

Матрица	Размер	Выделенная память				
		10	20	30	40	50
Разреженная	100x100	8400	16400	24400	32400	40400
Обычная	100x100	40000				
Разреженная	300x300	73200	145200	217200	289200	361200
Обычная	300x300	360000				
Разреженная	500x500	202000	402000	602000	802000	1002000
Обычная	500x500	1000000				
Разреженная	700x700	394800	786800	1178800	1570800	1962800
Обычная	700x700	1960000				
Разреженная	900x900	651600	1299600	1947600	2595600	3243600
Обычная	900x900	3240000				

Матрица	Размер	Выделенная память				
		60	70	80	90	100
Разреженная	100x100	48400	56400	64400	72400	80400
Обычная	100x100	40000				
Разреженная	300x300	433200	505200	577200	649200	721200
Обычная	300x300	360000				
Разреженная	500x500	1202000	1402000	1602000	1802000	2002000
Обычная	500x500	1000000				
Разреженная	700x700	2354800	2746800	3138800	3530800	3922800
Обычная	700x700	1960000				
Разреженная	900x900	3891600	4539600	5187600	5835600	6483600
Обычная	900x900	3240000				

По приведенным выше таблицам можно сделать следующие выводы:

- По времени выполнения: выбор способа хранения зависит от размера матрицы: при размерах от 100x100 до 500x500 разреженный вид выгоднее при количестве ненулевых элементов до 30% от общего. При размерах от 500x500 до 900x900 разреженный вид выгоднее при количестве ненулевых элементов до 40-50% от общего.

- По памяти: выбор способа хранения не зависит от размера матрицы. Хранение в разреженном виде выгоднее при количестве ненулевых элементов до 50% от общего.

Контрольные вопросы

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

Разреженная матрица – матрица с преимущественно нулевыми элементами.

Схемы хранения разреженных матриц: линейный связанный список, кольцевой связанный список, двунаправленный стек и очередь, диагональная схема хранения, связные схемы разреженного хранения.

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

Под хранение разреженной матрицы: память всех массивов, используемых для ее хранения (зависит от способа).

Под хранение обычной матрицы: кол-во строк * кол-во столбцов * размер элемента.

3. Каков принцип обработки разреженной матрицы?

Обрабатываются только ненулевые элементы.

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

Зависит от количества ненулевых элементов и размера матрицы. В моей реализации при количестве ненулевых элементов выше 30-40% следует применять стандартные алгоритмы обработки матриц.

Вывод

В ходе выполнения лабораторной работы была изучена обработка разреженных матриц. Произведено сравнение эффективности использования двух различных алгоритмов по времени и по памяти. Результаты сравнения эффективности были приведены в виде таблиц выше в отчете.

Про выбор способов хранения было выявлено:

- По времени выполнения: выбор способа хранения зависит от размера матрицы: при размерах от 100x100 до 500x500 разреженный вид выгоднее при количестве ненулевых элементов до 30% от общего. При размерах от 500x500 до 900x900 разреженный вид выгоднее при количестве ненулевых элементов до 40-50% от общего.

- По памяти: выбор способа хранения не зависит от размера матрицы. Хранение в разреженном виде выгоднее при количестве ненулевых элементов до 50% от общего.