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»

Студент: Лысцев Никита Дмитриевич

Группа: ИУ7-33Б

Студент _____ Лысцев Н.Д

подпись, дата *фамилия,и.о.*

Преподаватель _____ Рыбкин Ю. А

подпись, дата *фамилия,и.о.*

Оценка _____

2022 г

Оглавление

1. 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

2. Описание условия 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 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.

3. Описание технического задания

3.1. Входные данные

Целые числа: количества строк и столбцов матрицы, количество строк вектора столбца, количество ненулевых элементов в матрице и в векторе-столбце.

При вводе матрицы и вектора столбца вручную также используются целые числа: индекс строки и столбца матрицы и индекс строки вектора-столбца.

3.2. Выходные данные

Результат умножения матрицы на вектор-столбец, вывод краткой информации об оценке эффективности использования алгоритмов стандартной обработки матриц и алгоритмов для обработки разреженных матриц.

3.3. Описание задачи, реализуемой программой

Задача программы – реализовать умножение матрицы на вектор столбец, используя стандартные алгоритмы обработки матриц и используя алгоритмы для обработки разреженных матриц.

3.4. Способ обращение к программе

Ввод и вывод всех данных осуществляется через консоль.

Также существует возможность заполнить матрицу случайными ненулевыми элементами в случайных местах по введённым пользователем размерности матрицы и количеству ненулевых элементов.

3.5. Описание возможных аварийных ситуаций и ошибок пользователя

Аварийные ситуации:

- Пустое поле ввода – ничего не произойдет, программа будет ждать ввода пользователя;

Ошибки пользователя:

- Некорректный ввод: превышение максимально допустимых размерностей матрицы, ввод некорректного индекса столбца или строки при ручном вводе, не валидные данные при вводе значения элемента.

4. Описание внутренних структур данных

Информация о матрице и векторе-столбце в стандартном их представлении хранится в структуре типа «std_matrix_t».

```
typedef struct
{
    int **matrix;
    int count_row;
    int count_col;
} std_matrix_t;
```

Листинг 1. Структура типа «std_matrix_t»

Поля структуры «std_matrix_t»:

1. matrix – целочисленная матрица;
2. count_row – целочисленная переменная, содержащая в себе количество строк матрицы. Максимально возможно количество строк равно 20000.

3. count_col -- целочисленная переменная, содержащая в себе количество столбцов матрицы. Максимально возможно количество столбцов для матрицы равно 20000, а для вектора-столбца равно 1.

Информация о матрице и векторе-столбце в разреженном их представлении хранится в структуре типа «std_matrix_t».

```
typedef struct
{
    int count_row;
    int count_col;
    int count_no_zero;

    int *A;
    int *JA;
    list_t IA;
} sparse_matrix_t;
```

Листинг 2. Структура типа «sparse_matrix_t»

Поля структуры «sparse_matrix_t»:

1. count_row – целочисленная переменная, содержащая в себе количество строк матрицы. Максимально возможно количество строк равно 20000;
2. count_col -- целочисленная переменная, содержащая в себе количество столбцов матрицы. Максимально возможно количество столбцов для матрицы равно 20000, а для вектора-столбца равно 1;
3. count_no_zero – целочисленная переменная, содержащая в себе количество ненулевых элементов матрицы;
4. A – целочисленный массив, содержащий в себе только ненулевые элементы матрицы;
5. JA -- целочисленный массив, содержащий в себе индексы столбцов ненулевых элементов матрицы;

6. IA -- односвязный список, элементами которого являются индексы из массивов A и JA, указывающие на первые ненулевые элементы строк.

Односвязный список IA представлен в виде структуры «list_t», содержащей индекс из массивов A и JA, указывающие на первый ненулевой элемент строки, а также содержащей указатель на следующий элемент списка.

```
typedef struct s_list
{
    int index_start_row;
    struct s_list *next_elem;
} list_t;
```

Листинг 3. Структура типа «list_t»

Поля структуры «std_matrix_t»:

1. index_start_row – целочисленная переменная, указывающая на первый ненулевой элемент строки;
2. next_elem – указатель на следующий элемент связного списка.

4.1. Описание алгоритма

Взаимодействие пользователя с программой осуществляется через консоль.

В начале информационного сообщения пользователю кратко излагается суть данной программы. Далее пользователь предлагается ввести целое число (0 или 1) для того, чтобы либо создать матрицу и вектор-столбец, заполненные случайным образом ненулевыми элементами, либо создать и заполнить матрицу и вектор столбец вручную соответственно.

Далее пользователь вводит размерность матрицы и вектора столбца, количество ненулевых элементов в матрице и векторе столбце. В случае,

если выбран ввод данных вручную, то далее предлагается вводить индексы строк и столбцов и значение, соответствующее этим индексам.

После получения всех данных происходит процесс умножения матрицы и вектора столбца, и выводится краткая числовая оценка эффективности работы стандартных алгоритмов умножения и алгоритмов умножения для разряженных матриц.

```
Данная программа умножает матрицу на вектор-столбец, обрабатывая их
а) С помощью алгоритмов для обработки разреженных матриц.
б) С помощью стандартных алгоритмов обработки матриц.

В разреженном виде матрица хранится в двух массивах и одном списке:
    A - массив ненулевых элементов;
   JA - массив номеров столбцов для каждого элемента A;
   IA - односвязный список, который содержит индекс;
        каждого первого элемента очередной строки в массивах A и JA.

Введите 0, если хотите сгенерировать матрицу и вектор-столбец случайно,
или 1, если хотите ввести матрицу и вектор-строку с клавиатуры:
```

Листинг 4. Система взаимодействия с пользователем

4.2. Ограничения на входные данные

Максимально возможное количество строк и столбцов матрицы равно 20000, минимальное – 1.

Нумерация индексов строк и столбцов матрицы начинается с нуля.

При правильном вводе размерностей матрицы и вектора-столбца количество столбцов в матрице должно равняться количеству строк в векторе-столбце. Иначе их просто невозможно умножить.

В случае ввода некорректных данных программа завершает свою работу.

5. Оценка эффективности работы алгоритмов

5.1. Зависимость времени от процента заполнения

При оценке эффективности зависимости времени умножения от процента заполнения матрицы предполагалось, что вектор столбец всегда заполнен ненулевыми элементами на 5%.

1 % заполнения

Размерность матрицы	Время выполнения (обычная матрица), микросекунды	Время выполнения (разреженная матрица), микросекунды
1000x1000	1814	1074
10000x10000	170150	174910
20000x20000	614472	1110423

2 % заполнения

Размерность матрицы	Время выполнения (обычная матрица), микросекунды	Время выполнения (разреженная матрица), микросекунды
1000x1000	1416	1171
10000x10000	158109	300302
20000x20000	632037	2102557

3 % заполнения

Размерность матрицы	Время выполнения (обычная матрица), микросекунды	Время выполнения (разреженная матрица), микросекунды
1000x1000	1606	1253
10000x10000	157159	411353
20000x20000	625090	2958321

4 % заполнения

Размерность матрицы	Время выполнения (обычная матрица), микросекунды	Время выполнения (разреженная матрица), микросекунды
1000x1000	1519	1627
10000x10000	152081	520269
20000x20000	601961	3845199

20 % заполнения

Размерность матрицы	Время выполнения (обычная матрица), микросекунды	Время выполнения (разреженная матрица), микросекунды
1000x1000	1757	5130
10000x10000	170267	2521898
20000x20000	655093	19116262

50 % заполнения

Размерность матрицы	Время выполнения (обычная матрица), микросекунды	Время выполнения (разреженная матрица), микросекунды
1000x1000	1542	11892
10000x10000	166423	6212476
20000x20000	625630	46363902

При оценке первой зависимости можно заметить, что при увеличении заполнения матрицы ненулевыми элементами эффективность алгоритма обработки матрицы как разреженной стремительно падала. Но при использовании такого алгоритма в матрицах не слишком больших размеров, когда в матрице около 1-3% ненулевых элементов очень эффективно по времени (обработка алгоритма умножения для разреженных матриц в 1,5 раза быстрее по времени, чем обработка алгоритмом для стандартного представления матриц).

5.2. Зависимость размера матрицы от процента заполнения

При оценке эффективности зависимости размера матрицы от процента заполнения матрицы предполагалось, что вектор столбец всегда заполнен ненулевыми элементами на 5%.

1 % заполнения

Размерность матрицы	Размер матрицы (обычная матрица), байты	Размер матрицы (разреженная матрица), байты
1000x1000	4000008	84012
10000x10000	400000008	8040012
20000x20000	1600000008	32080012

2 % заполнения

Размерность матрицы	Размер матрицы (обычная матрица), байты	Размер матрицы (разреженная матрица), байты
1000x1000	4000008	164012
10000x10000	400000008	16040012
20000x20000	1600000008	64080012

3 % заполнения

Размерность матрицы	Размер матрицы (обычная матрица), байты	Размер матрицы (разреженная матрица), байты
1000x1000	4000008	244012
10000x10000	400000008	24040012
20000x20000	1600000008	96080012

4 % заполнения

Размерность матрицы	Размер матрицы (обычная матрица), байты	Размер матрицы (разреженная матрица), байты
1000x1000	4000008	324012
10000x10000	400000008	32040012
20000x20000	1600000008	128080012

20 % заполнения

Размерность матрицы	Размер матрицы (обычная матрица), байты	Размер матрицы (разреженная матрица), байты
1000x1000	4000008	1604012
10000x10000	400000008	160040012
20000x20000	1600000008	640080012

50 % заполнения

Размерность матрицы	Размер матрицы (обычная матрица), байты	Размер матрицы (разреженная матрица), байты
1000x1000	4000008	4004012
10000x10000	400000008	400040012
20000x20000	1600000008	1600080012

При оценке второй зависимости можно также увидеть, что с увеличением заполнения матрицы ненулевыми элементами память, выделяемая для хранения матрицы как разреженной существенно увеличивается. Но при

использовании такого способа представления матрицы, когда в матрице порядка 20% ненулевых элементов можно неплохо сэкономить на выделении памяти.

6. Выводы

Разреженный способ представления матриц очень удобно использовать при большом количестве нулей в матрице (порядка 4-5%), так как тратится меньший объем памяти (в 5-10 раз меньше, чем в стандартном представлении матрицы), а также уменьшается время умножения (при 1% заполнения порядка 1,5 раза меньше, по сравнению со стандартным умножением).

Но уже при ~4% стандартный алгоритм выигрывает по времени работы, а затраченная память либо равняется размеру разреженного представления, либо меньше разреженного представления.

Хранение матрицы в разреженном виде выгодно использовать только если матрица содержит большое количество нулей, во всех остальных случаях такой вид хранения проигрывает по памяти, так как структура данного типа довольно сильно нагружена целочисленными полями, которые содержат информацию о компонентах матрицы (занимают большой объем памяти при сравнении с обычным представлением матрицы).

7. Набор тестов

№ Теста	Входные данные	Выходные данные	Результат
01	Выбор схемы создания матрицы и вектора Выбор - 0	Предложение ввести размерности матрицы и вектора столбца	Ожидание следующего действия.
02	Выбор схемы создания матрицы и вектора Выбор - 1	Предложение ввести размерности матрицы и вектора столбца	Ожидание следующего действия.
03	Выбор схемы создания матрицы и вектора Выбор - a	Сообщение о том, что вывод опции некорректный	Завершение программы с ненулевым кодом возврата.
04	Выбор схемы создания матрицы и вектора Выбор - 10	Сообщение о том, что вывод опции неверный	Завершение программы с ненулевым кодом возврата.
05	Ввод количества строк матрицы Количество строк = 10	Предложение ввести количество столбцов	Ожидание следующего действия.
06	Ввод количества строк матрицы Количество строк = 20000	Предложение ввести количество столбцов	Ожидание следующего действия.
07	Ввод количества строк матрицы Количество строк = 20001	Сообщение о том, что ввод размерности неверный	Завершение программы с ненулевым кодом возврата.
08	Ввод количества строк матрицы Количество строк = 1	Предложение ввести количество столбцов	Ожидание следующего действия.
09	Ввод количества строк матрицы Количество строк = 0	Сообщение о том, что ввод размерности неверный	Завершение программы с ненулевым кодом возврата.
10	Ввод количества столбцов матрицы Количество столбцов = 10	Предложение ввести количество строк вектора-столбца	Ожидание следующего действия.
11	Ввод количества столбцов матрицы Количество столбцов = 20000	Предложение ввести количество строк вектора-столбца	Ожидание следующего действия.
12	Ввод количества столбцов матрицы Количество столбцов = 20001	Сообщение о том, что ввод размерности неверный	Завершение программы с ненулевым кодом возврата.
13	Ввод количества столбцов матрицы Количество столбцов = 1	Предложение ввести количество строк вектора-столбца	Ожидание следующего действия.
14	Ввод количества столбцов матрицы Количество столбцов = 0	Сообщение о том, что ввод размерности неверный	Завершение программы с ненулевым кодом возврата.
15	Ввод индекса строки или столбца Индекс = 0	Предложение ввести либо следующий индекс, либо значение по индексу	Ожидание следующего действия.
16	Ввод индекса строки или столбца Индекс = -1	Сообщение о том, что ввод индекса неверный	Завершение программы с ненулевым кодом возврата.

17	Ввод индекса строки или столбца Индекс больше максимально допустимого по размерности	Сообщение о том, что ввод индекса неверный	Завершение программы с ненулевым кодом возврата.
18	Ввод количества ненулевых символов Количество ненулевых символов <= количеству элементов матрицы	Предложение пользователю вводить данные дальше	Ожидание следующего действия.
19	Ввод количества ненулевых символов Количество ненулевых символов больше количества элементов матрицы	Сообщение о том, что ввод количества ненулевых элементов неверный	Завершение программы с ненулевым кодом возврата.
20	Ввод количества ненулевых символов Количество ненулевых символов меньше нуля	Сообщение о том, что ввод количества ненулевых элементов неверный	Завершение программы с ненулевым кодом возврата.
21	Ввод размерностей матрицы и вектора столбца Количество столбцов матрицы не равно количеству строк в векторе столбце	Сообщение о том, что размерность матрицы и вектора столбца не соответствует операции умножения	Завершение программы с ненулевым кодом возврата.
22	Ввод значения ненулевого элемента Значение = 0	Сообщение о том, что введенное значение для ненулевого элемента является нулевым	Завершение программы с ненулевым кодом возврата.

8. Ответы на контрольные вопросы

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

Разреженная матрица – матрица, содержащая много нулей.

Схемы хранения матриц: связная схема хранения (с помощью линейных связанных списков), двунаправленные стеки и очереди, столбцовый формат, строчный формат, диагональная схема хранения.

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

Под матрицу, хранящуюся в стандартном виде, выделяют n * m ячеек памяти, где n – количество строк матрицы, m – количество столбцов матрицы.

Количество ячеек памяти, выделяемой под разреженную матрицу, зависит от способа ее представления. К примеру, в разреженном строчном формате (CSR) под хранение ненулевых элементов выделяют k ячеек памяти (k – количество ненулевых элементов), под хранение индексов ненулевых элементов также выделяют k ячеек памяти, а под хранение компонент строк выделяют n + 1 ячеек памяти, где n -- количество строк матрицы.

8.3. Каков принцип обработки разреженной матрицы?

При обработке разреженных матриц все операции и вычисления проводятся только с ненулевыми ее элементами, то есть количество операций будет прямо пропорционально количеству ненулевых элементов в матрице.

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

Эффективнее применять стандартные алгоритмы выгоднее при большом количестве ненулевых элементов. Стоит отметить, что если расход памяти в программе не так важен, но важно время выполнения программы, то в случае умножения матрицы на вектор столбец лучше воспользоваться стандартным алгоритмом при большом количестве ненулевых элементов, и умножение специального (разряженного) в случае небольшого количества ненулевых элементов.