|  |  |
| --- | --- |
| 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 «Программная инженерия»

**ОТЧЁТ**

**«**Работа №3, Обработка разреженных матриц**»**

Выполнил студент: \_\_***Бугаенко Андрей Павлович***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*фамилия, имя, отчество*

Группа: \_\_\_\_***ИУ7-35Б***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверила.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*подпись, дата*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2020 г.*

**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 задания**

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.

**Техническое задание**

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 вектора-строки и матрицы, 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.

**Исходные данные и результат**:

В качестве исходных данных программа получает от пользователя:

1) Номер опции, которую программе необходимо выполнить

2) Имя файла в случае ввода матрицы или вектора из файла

3) Параметры матрицы или вектора в случае ручного ввода или генерации матрицы или вектора

4) Значения элементов в соответствующем способу ввода виде

В качестве результата пользователь может получить:

1) Сообщение об успешном выполнении команды

2) Матрицу в выбранной форме вывода, если её вывод в такой форме возможен

3) Результаты сравнения умножения разными способами

4) Сообщения об ошибке, если таковая была допущена пользователем

**Описание задачи, реализуемой программой:**

Программа должна считывать матрицу с клавиатуры или из файла и переводить её в вид трёх векторов для хранения. Также программа должна позволять производить умножение матрицы на вектор с использованием стандартного матричного умножения и умножения, реализованного на основе компактного хранения данных в виде трёх векторов. Помимо этого программа должна предоставлять возможность сравнить результаты выполнения умножения по памяти и по времени выполнения.

**Способ обращения к программе:**

Для работы с программой пользователь должен использовать консольный клавиатурный ввод. Для запуска программы пользователь использует команду ./app.exe. После этого пользователь попадёт в меню, в котором каждое действие определено как цифра, при вводе которой это действие происходит. Также для выполнения некоторых операций программе могут потребоваться дополнительные данные, о необходимости ввода которых она уведомит пользователя. Данные вводятся в соответствии с запросом программы в формате, описанном в сообщении, выводящимся при выборе той или иной команды. Если данные введены неправильно, то программа выведет сообщение об ошибке.

**Описание возможных аварийных ситуаций и ошибок пользователя:**

В случае ошибки во вводе программа прекратит выполнение команды и выведет описание ошибки, которая произошла.

Типы ошибок ввода:

* Неправильно было введено имя файла
* Имя файла введено правильно, но он не содержит никаких данных
* При ручном вводе введены неправильные параметры создаваемой матрицы/вектора
* При генерации матрицы введены неправильные параметры создаваемой матрицы/вектора
* При ручном вводе неправильно введён элемент матрицы

Типы ошибок при обработке файла:

* Файл пустой
* Файл содержит матрицу в некорректном формате

**Описание внутренних структур данных:**

Для реализации программы была создана структура данных для хранения матрицы в разреженном виде. Выглядит она следующим образом:

// Структура для хранения разреженной матрицы

typedef struct sparse\_matrix

{

    // Число строк и столбцов матрицы

    mat\_index\_t rows;

    mat\_index\_t cols;

    // Число ненулевых элементов (размер двух следующих массивов)

    mat\_index\_t nz\_count;

    // Вектор А содержащий значения ненулевых элементов

    mat\_data\_t \*data\_array; // размер = nz\_amount

    // Вектор JA содержащий номера столбцов для элементов вектора А

    mat\_index\_t \*data\_cols; // размер = nz\_amount

    // Вектор IA содержащий номера компонент с которых начинается описание строк матрицы

    mat\_index\_t \*rows\_array; // размер равен числу строк + 1 = rows + 1

} sp\_mat\_t;

Параметры структуры:

mat\_index\_t и mat\_data\_t - типы элементов структуры, в нашем случае:

typedef unsigned short mat\_index\_t;

typedef long long mat\_data\_t;

mat\_index\_t rows - количество строк в матрице

mat\_index\_t cols - количество столбцов в матрице

mat\_index\_t nz\_count - количество ненулевых элементов в матрице

mat\_data\_t \*data\_array - вектор А, содержащий значения ненулевых элементов

mat\_index\_t \*data\_cols - вектор JA, содержащий номера компонент с которых начинается описание строк матрицы

mat\_index\_t \*rows\_array - вектор IA, содержащий номера компонент с которых начинается описания строк матрицы

**Описание алгоритма:**

Алгоритм простого умножения работает на основе стандартного алгоритма умножения матриц. Сначала умножаемая матрица и вектор переводится разреженный вид, после этого создаётся новый вектор, i-й элемент которого это сумма произведений элементов изначального вектора и i-го столбца матрицы.

Алгоритм быстрого умножения позволяет учитывать только ненулевые элементы матрицы и вектора при умножении. Мы проходим по строке в соответствии с номерами элементов в массиве IA. Если мы находим элементы из матрицы и строки с одинаковыми номерами столбцов, то добавляем их произведение к сумме и ищем следующую пару. Когда строка заканчивается, мы записываем получившийся элемент в вектор. После подсчёта элемент строки записывается в вектор результата, и процесс повторяется, пока мы не просчитаем элементы для всех строк. В конце мы удаляем нулевые элементы из получившегося вектора.

**Набор тестов:**

|  |  |  |
| --- | --- | --- |
| Наименование теста | Входные данные | Выходные данные |
| Проверка ввода матрицы вручную | 1 5 5 3 2 2 1 0 1 2 3 4 5 | Сообщение о том, что матрица введена успешно |
| Проверка ввода матрицы вручную | 1 2 2 7 | Сообщение об ошибке ввода |
| Проверка корректного ввода матрицы из файла | 2/3 data.txt | Сообщение о том, что матрица введена успешно |
| Проверка некорректного ввода матрицы из файла | 2/3 data | Сообщение о том, что имя файла некорректно |
| Проверка генерации матрицы | 4 5 5 20 -4 4 | Сообщение о том, что матрица успешно сгенерирована |
| Проверка генерации матрицы | 4 -5 5 20 | Сообщение об ошибке ввода |
| Проверка вывода матрицы | 5/6/7 (пустая матрица) | Сообщение о том, что нельзя вывести пустую матрицу |
| Проверка вывода матрицы | 5/6/7 | Матрица в требуемом виде |
| Проверка ввода вектора вручную | 8 5 2 1 2 -2 3 | Сообщение о том, что вектор введён корректно |
| Проверка ввода вектора вручную | 8 5 7 | Сообщение об ошибке ввода |
| Проверка ввода вектора из файла | 9/10 data.txt | Сообщение о том, что матрица введена успешно |
| Проверка ввода вектора из файла | 9/10 dat | Сообщение о том, что имя файла некорректно |
| Проверка умножения | 11 | Вывод результатов медленного умножения |
| Проверка умножения | 12 | Вывод результатов быстрого умножения |
| Проверка умножения | 13 | Вывод результатов сравнения способов умножения |
| Проверка генерации вектора | 14 5 20 | Сообщение об успешной генерации вектора |
| Проверка генерации вектора | 14 -5 23 | Сообщение об ошибке |
| Проверка вывода вектора | 15 (вектор не инициализирован) | Сообщение об ошибке |
| Проверка вывода вектора | 15 (вектор существует) | Вывод вектора в файл или на экран |

**Анализ скорости обработки данных:**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Матрица | | | Вектор | | min | max | t f | t s | память f | память s |
| i | j | % | j | % |
| 10 | 10 | 20 | 10 | 20 | -10 | 10 | 0.0017 | 0.0029 | 436 | 968 |
| 10 | 10 | 40 | 10 | 40 | -10 | 10 | 0.0006 | 0.0039 | 696 | 968 |
| 10 | 10 | 60 | 10 | 60 | -10 | 10 | 0.0031 | 0.0045 | 906 | 968 |
| 10 | 10 | 80 | 10 | 80 | -10 | 10 | 0.0008 | 0.0041 | 1136 | 968 |
| 10 | 10 | 100 | 10 | 100 | -10 | 10 | 0.0012 | 0.0049 | 1336 | 968 |
| 25 | 25 | 20 | 25 | 20 | -10 | 10 | 0.0019 | 0.021 | 1706 | 5408 |
| 25 | 25 | 40 | 25 | 40 | -10 | 10 | 0.0039 | 0.036 | 3136 | 5408 |
| 25 | 25 | 60 | 25 | 60 | -10 | 10 | 0.0073 | 0.0479 | 4356 | 5408 |
| 25 | 25 | 80 | 25 | 80 | -10 | 10 | 0.0048 | 0.0505 | 5626 | 5408 |
| 25 | 25 | 100 | 25 | 100 | -10 | 10 | 0.0059 | 0.0556 | 6916 | 5408 |
| 50 | 50 | 20 | 50 | 20 | -10 | 10 | 0.0054 | 0.1581 | 5886 | 20808 |
| 50 | 50 | 40 | 50 | 40 | -10 | 10 | 0.0443 | 0.2209 | 10376 | 20808 |
| 50 | 50 | 60 | 50 | 60 | -10 | 10 | 0.0154 | 0.2697 | 16366 | 20808 |
| 50 | 50 | 80 | 50 | 80 | -10 | 10 | 0.0229 | 0.2987 | 21106 | 20808 |
| 50 | 50 | 100 | 50 | 100 | -10 | 10 | 0.0258 | 0.3351 | 26216 | 20808 |

**Вывод:**

В результате измерений мы можем сделать вывод о том, что использование сжатого вида матриц существенно ускоряет обработку матриц. При тестировании время быстрой обработки всегда меньше времени работы стандартного алгоритма. Причём чем больше матрица, тем больше эта разница. Если же смотреть со стороны памяти, то разреженная структура эффективна только если матрица является разреженной, если же матрица является плотной, то на неё требуется памяти больше, чем на хранение и обработку обычной матрицы. Поэтому подобные структуры лучше всего использовать, если нет ограничений по памяти, или точно известно, что матрица является разреженной. **Контрольные вопросы:**

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

Разрежённая матрица — это матрица с преимущественно нулевыми элементами. В противном случае, если большая часть элементов матрицы ненулевые, матрица считается плотной.

Для хранения матрицы существует множество различных схем, среди которых есть линейный связный список, кольцевой связный список, двунаправленные стеки, очереди. Также существует диагональная схема хранения симметричных матриц и связные схемы разреженного хранения. Связанная схема Кнута предлагает хранить в массиве (например, в AN) в произвольном порядке сами элементы, индексы строк и столбцов соответствующих элементов (например, в массивах I и J), номер (из массива AN) следующего ненулевого элемента, расположенного в матрице по строке (NR) и по столбцу (NC), а также номера элементов, с которых начинается строка (указатели для входа в строку – JR) и номера элементов, с которых начинается столбец (указатели для входа в столбец - JC). Данная схема хранения избыточна, но позволяет легко осуществлять любые операции с элементами матрицы. Самой распространённой является схема предложенная Чангом и Густавсоном, называемая: "разреженный строчный формат". Эта схема предъявляет минимальные требования к памяти и очень удобна при выполнении операций сложения, умножения матриц, перестановок строк и столбцов, транспонирования, решения систем линейных уравнений, при хранении коэффициентов в разреженных матрицах и т.п. В этом случае значения ненулевых элементов хранятся в массиве AN, соответствующие им столбцовые индексы - в массиве JA. Кроме того, используется массив указателей, например, IA, отмечающих позиции AN и JA, с которых начинаются описание очередной строки. Дополнительная компонента в IA содержит указатель первой свободной позиции в JA и AN.

2. Каким образом и сколько памяти выделяется под хранение разреженной и обычной матрицы?   
В случае стандартного хранения обычной матрицы память под матрицу выделяется как массив массивов, т.е. память под матрицу это количество элементов \* размер типа этих элементов.  
Под хранение разреженной матрицы в нашем случае выделяется память под структуру матрицы, т.е. это память, выделяемая на хранение вектора A ненулевых значений, вектора JA индексов столбцов ненулевых значений и вектора IA описывающего начала и концы строк.

3. Каков принцип обработки разреженной матрицы?

Принцип обработки разреженной матрицы заключается в том, что обработку нулевых элементов стараются исключить из алгоритма для экономии памяти и времени выполнения программы.

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

Стандартные методы обработки матриц эффективно применять в случае, когда матрица является плотной и имеет сравнительно небольшие размеры, поскольку алгоритм обработки разреженных матриц в этом случае требует больше памяти и при небольших матрицах по скорости он в принципе может сравняться с эффективным алгоритмом.