Лабораторная работа №3 "Обработка разреженных матриц"

Студент Варин Дмитрий Владимирович
Группа ИУ7-36Б
Цель работы: реализация алгоритмов обработки разреженных матриц, сравнение эффективности использования этих алгоритмов со стандартными алгоритмами обработки матрицпри различном размере матриц и степени их разреженности.

Описание условия задания

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

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

Техническое задание

Входные данные:

1. **Целое число - номер команды** :

- Число от 0 до 3.

2. **Данные, зависимые от команды**:

- **1** - кол-во строк и столбцов матрицы, элементы вектор-столбца.
- **2** - имя файла с матрицами в заданном формате: *строка - размер, матрица, строка-размер, вектор.*
- **3** - размер матриц, процент разреженности.

Выходные данные:

1. При вызове команды **1** и **2** - результат умножения матрицы на вектор в формате CSR(Column Sparse Rows), а также их разреженность;
2. При вызове команды **3** - время выполнения умножения в нормальном виде и разреженном.
3. При вызове команды **0** - сообщение о завершении выполнения.

Функция программы:

Программа выполняет ряд функций, указанных при её запуске.
Она запускает:
0.Выход из программы.
1.Умножение матрицы на вектор столбец в *CSR* виде.
2.Умножение матрицы на вектор столбец считанных из файла в *CSR* виде.
3.Замер времени выполнения умножения матрицы на вектор-столбец в виде *CSR* и обычном.

Обращение к программе

Запускается из терминала.

Аварийные ситуации

1. Некорректный ввод номера команды.
   На входе: число, не входящее в диапазон команд.
   На выходе: сообщение «Такого режима нет в программе, повторите попытку...»
2. Неверное кол-во строк/столбцов матрицы.
   На входе: число, меньше нуля или не число.
   На выходе: сообщение «Некорректный ввод.»
3. Некорректный ввод столбца/строки матрицы.
   На входе: число, не входящее в диапазон индексации матрицы.
   На выходе: сообщение «Введена несуществующая(ий) строка / столбец.»
4. Ввод некорректного элемента матрицы.
   На входе: не число.
   На выходе: сообщение «Неверный ввод числа.»
5. Превышен размер длины имени файла.
   На входе: строка длины более 32 символов, без символа конца строки.
   На выходе: сообщение «Введено много символов. Ошибка.»
6. Открытие несуществующего файла.
   На входе: имя несуществующего файла.
   На выходе: сообщение «Не удалось открыть файл.»
7. Ввод некорректного процента ненулевых элементов матрицы.
   На входе: x: x < 0 || x > 100.
   На выходе: сообщение «Введен некорректный процент.»
8. Группа ситуаций ошибки выделения памяти для данных.
   На входе: ---
   На выходе: сообщения вида: Не удалось создать/инициализировать матрицу/вектор.»

Структуры данных

Матрица заполнена числами типа `my_num_t`

```
typedef double my_num_t;
```

Для хранение матрицы в стандартном виде используется структура `matrix_t`, состоящая из полей `rows`,`cols`,`matrix`

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

Поле `matrix` - массив указателей на строки матрицы.
Для хранения разреженной матрицы используется структура `sparse_matrix_t`

```
typedef struct
{
    int rows;
    int cols;
    int count;
    my_num_t *values;
    int *rows_ind;
    int *cols_index;
} sparse_matrix_t;
```

- `int rows` - количество строк;
- `int cols` - количество столбцов;
- `int count` - количество ненулевых эл-в;
- `my_num_t *values` - массив ненулевых чисел из матрицы. Размер - **count**;
- `int *rows_ind` - массив индексов строк ненулевых элементов. Размер - **count**;
- `int *cols_index` - массив индексов ненулевых элементов, с которых начинается описание `i-ro` столбца. Размер массива - **cols + 1**.

Алгоритм

1. Пользователь вводит номер команды из меню.
2. Пока пользователь не введет 0 (выход из программы), ему будет предложено выполнять действия в соответствии с ТЗ.
3. При вводе (или генерации) матрицы, матрица сразу хранится двумя способами хранения (стандартном и разреженном столбцовом).
4. При выборе умножения матриц из файла, матрица и вектор загружаются из файла, затем происходит умножения и вывод результата в формате **CSR**.
5. При выборе умножения из консоли, вводятся матрица и вектор, производится умножение, выводится результат в формате **CSR**.

Оценка эффективности

Измерения времени умножения будут производиться в *микросекундах*.
Для измерения используется структура `timeval` из библиотеки `<sys/time.h>`.
При записи результатов использовалось среднее время, полученное по результатам *10 измерений*.
Для подсчёта памяти, занимаемой матрицей в традиционном формате и CRS используются функции

`get_size_normalize` и `get_size_sparse`, подсчитывающие занимаемую память.

Время умножения

1% заполнения

Размеры	Обычный формат хранения	CRS формат
10x10	1.7	1.3
100x100	17.0	2.6
500x500	432.8	25.5
1000x1000	1816.5	172.1
10000x10000	297439.3	189914.9

5% заполнения

Размеры	Обычный формат хранения	CRS формат
10x10	1.7	1.3
100x100	23.0	8.5
500x500	480.5	318.5
1000x1000	2254.5	5930
10000x10000	315720.5	4063447.6

10% заполнения

Размеры	Обычный формат хранения	CRS формат
10x10	2.3	1.7
100x100	24.5	16.4
500x500	687.3	1940.6
1000x1000	3711.5	25091.8
10000x10000	317787.6	16256299.4

15% заполнения

Размеры	Обычный формат хранения	CRS формат
10x10	1.3	1.2
100x100	18.0	25.4
500x500	486	6192.2
1000x1000	3501.6	37550.4
5000x5000	46769.8	2610152.6

30% заполнения

Размеры	Обычный формат хранения	CRS формат
10x10	1.9	1.5
100x100	21.6	181.6
500x500	670.8	28336.5
1000x1000	3266.4	177992.0
5000x5000	45889.1	10817204.2

50% заполнения

Размеры	Обычный формат хранения	CRS формат
10x10	1.2	1.5
100x100	17.4	295.4
500x500	435.5	30489.8
1000x1000	1785.2	234164.8
5000x5000	48667.4	28616878.9

100% заполнения

Размеры	Обычный формат хранения	CRS формат
10x10	0.7	2.6
100x100	9.7	698.6
500x500	306.9	69736.4
1000x1000	1068.7	531548.8
5000x5000	26419.9	67139730.6

Объём занимаемой памяти (в байтах) *1% заполнения*

Размеры	Обычный формат хранения	CRS формат
10x10	928	152
100x100	80848	1712
500x500	2004048	32160
1000x1000	8008048	124220
10000x10000	800080048	12041300

5% заполнения

Размеры	Обычный формат хранения	CRS формат
10x10	928	200
100x100	800848	6560
500x500	2004048	152400
1000x1000	8008048	604700
10000x10000	800080048	60046100

10% заполнения

Размеры	Обычный формат хранения	CRS формат
10x10	928	272
100x100	80848	12620
500x500	200048	302700
1000x1000	8008048	1205300
10000x10000	800080048	120052100

15% заполнения

Размеры	Обычный формат хранения	CRS формат
10x10	928	332
100x100	80848	18680
500x500	2004048	45300
1000x1000	8008048	1805900
5000x5000	200040048	45029100

30% заполнения

Размеры	Обычный формат хранения	CRS формат
10x10	928	536
100x100	80848	36860
500x500	2004048	903900
1000x1000	8008048	3607700
5000x5000	200040048	90038100

50% заполнения

Размеры	Обычный формат хранения	CRS формат

Размеры	Обычный формат хранения	CRS формат
10x10	928	800
100x100	80848	61100
500x500	2004048	1505100
1000x1000	8008048	6010100
5000x5000	200040048	150050100

100% заполнения

Размеры	Обычный формат хранения	CRS формат
10x10	928	1460
100x100	80848	121700
500x500	2004048	3008100
1000x1000	8008048	12016100
5000x5000	200040048	300080100

Контрольные вопросы

1. **Что такое разреженная матрица, какие способы хранения вы знаете?**
   Разреженная матрица - это матрица, содержащая большое количество нулей.
   Способы хранения:

- обычное хранение;
- строчный формат **CRS** *Compressed row storage* или **CSR** *Compressed sparse row*;
- столбцовый формат **CSC** *Compressed sparse column* или **CCS** *Compressed column storage*;
- координатный формат **COO** *Coordinate list*;
- словарь по ключам **DOK** *Dictionary of Keys*;
- список списков **LIL** *List of Lists*.

2. **Каким образом и сколько памяти выделяется под хранение разреженной и обычной матрицы?**
   Под обычную матрицу выделяет `N * M` ячеек памяти, где N - строки, а M -столбцы. Для разреженной матрицы - зависит от способа.
   В случае разреженного формата, требуется `2 * K + (C + 1)` ячеек памяти, где `K` - количество ненулевых элементов, `C` - количество столбцов.
3. **Каков принцип обработки разреженной матрицы?**
   Алгоритмы обработки разреженных матриц предусматривают действие только с ненулевыми элементами.
4. **В каком случае для матриц эффективнее применять стандартные алгоритмы обработки матриц? От чего это зависит?** Стандартные алгоритмы обработки матриц эффективнее применять при большом количестве ненулевых элементов (от 40%). Стоит отметить, что если не

так важна память, занимаемая матрицами, но важно время, то в случае сложения лучше так же воспользоваться стандартными алгоритмами сложения матриц.

Вывод

Алгоритмы хранения и обработки разреженных матриц эффективны по памяти при заполнении матрицы до 60 %.
При большем проценте данный способ невыгоден, если рассматривать эфективность по времени, то алгоритм эффективен при заполнении до 10% и размерности менее 500x500
элементов, в остальных случаях при умножении лучше пользоваться стандартным способом, который в сравнении с рассмотренным выше, эффективнее по всем параметрам:
простота реализации, память, скорость.