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 структуре данных»

Название: Обработка разреженных матриц

Студент Дремин Кирилл

Группа ИУ7 – 36Б

Тип лабораторной работы: Учебная

Вариант №2

Преподаватель Барышникова Марина Юрьевна

2022 г.

Содержание

Описание условия задачи

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.

Описание ТЗ

Описание исходных данных и результатов:

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 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.

Способ обращения к программе

Обращение происходит посредством вызова заранее скомпилированного файла и ввода чисел через консоль.

Описание входных данных

Программа получает имена двух файлов, содержащих матрицы, через консоль. Последующая работа с программой осуществляется через меню.

```
Choose command:
1. Print sparse matrices
2. Print default matrices
3. Input matrices
4. Sum matrices and measure
5. Sum matrices and print result
0. Exit
```

Корректный ввод - цифра от 0 до 5. При выборе пункта 3 меню необходимо ввести две матрицы в координатном формате:

```
<Кол-во строк> <Кол-во столбцов> <Кол-во ненулевых элементов>
<Номер строки> <Номер столбца> <Ненулевой элемент>
...
```

Описание возможных аварийных ситуаций и ошибок пользователя:

1. Ошибка при открытии файла.
2. Слишком длинная или пустая введенная строка.
3. Некорректные символы при вводе чисел.
4. Неверное количество аргументов командной строки.
5. Пустой файл.
6. Неизвестная команда.
7. Ошибка выделения динамической памяти.
8. Количество ненулевых элементов не соотносится с размером матрицы.
9. Введенный столбец/строка очередного элемента не соотносится с размером матрицы.
10. Не совпадают размеры матриц.

Описание внутренних СД:

Программа оперирует следующим представлением разреженной матрицы:

```
typedef struct sparse_matrix_t {
    /// Количество ненулевых элементов
    size_t el_count;
    /// Количество строк
    size_t rows_count;
```

```
    /// Количество столбцов
    size_t cols_count;
    /// Массив индексов массива строк ненулевых элементов, с которых начинается каждый столбец матрицы. (JA)
    size_t *col_p;
    /// Массив номеров строк ненулевых элементов (IA)
    size_t *row_i;
    /// Массив ненулевых элементов (A)
    int *values;
    } sparse_matrix_t;
```

Описание алгоритма

Сложение матриц, представленных в обычном виде.

1. Проверка размеров матриц: если размеры матриц не совпадают - выдать ошибку.
2. Выделить память под результат, равный размеру обрабатываемых матриц
3. Пройти по каждому элементу первой матрицы по индексам i,j, одновременно идя по элементам второй матрицы, записывая результаты их сложения в соответствующую ячейку матрицы результата

Сложение матриц, представленных в разреженном виде.

1. Проверка размеров матриц: если размеры матриц не совпадают - выдать ошибку.
2. Рассчитать память, необходимую для матрицы результата: проходим по соответствующим столбцам матриц. Для каждого столбца проходим по массиву строк каждой матрицы. Если значение строки первой матрицы < (>) значения строки второй матрицы, то передвинуть указатель на элементы массива строк первой (второй) матрицы на 1 вправо; если же значения равны, то передвинуть оба указателя вправо на 1. К итоговому количеству ненулевых элементов прибавить 1.
3. Выделить память под матрицу результата.

4. Вычисление матрицы результата: проходим по соответствующим столбцам матриц. Для каждого столбца проходимся по массиву строк обеих матриц. Если значение строки первой матрицы < (>) значения строки второй матрицы, то передвинуть указатель на элементы массива строк первой (второй) матрицы на 1 вправо, добавить в конец массива строк результирующей матрицы значение строки первой (второй) матрицы, записать по этому же индексу в массив ненулевых элементов результирующей матрицы значение элемента первой (второй) матрицы; если же значения равны, то передвинуть оба указателя вправо на 1, добавить в конец массива строк результирующей матрицы значение строки любой матрицы, записать по этому же индексу в массив ненулевых элементов результирующей матрицы сумму элементов двух матриц.

Тесты программы

Входные данные	Выходные данные	Что проверяется
a	Error: incorrect symbol.	Некорректный символ в записи числа
6	Error: unknown command.	Несуществующая команда
3 (input) -1 2 1	Error: incorrect number.	Некорректный размер матрицы
3 (input) 2 2 5	Error: incorrect number.	Количество ненулевых элементов не соответствует размеру матрицы
3 (input) 2 2 4 1 3 2	Error: incorrect number.	Номер столбца не соответствует размеру матрицы
3 (input) 2 2 2 1 1 2 1 2 2 2 3 2 1 1 2 1 2 2 5 (sum and print)	Error: incorrect matrix size.	Размеры матриц не совпадают при попытке сложить
./app.exe data/small_10_a.txt data/small_10_b.txt data/another (enter)	Error: incorrect command line.	Неверное количество аргументов
./app.exe data/mid_15_a.txt data/mid_155_b.txt (enter)	Error: opening file.	Несуществующий файл

3 1 1 1 1 1 1 1 1 1 1 1 1 5	1 1 1 1 1 2	Проверяется работа с матрицей размером 1x1
3 3 3 0 3 3 0 5	3 3 0	Проверяется работа с нулевой матрицей
3 2 2 4 1 1 -1 1 2 -2 2 1 -3 2 2 -4 2 2 4 1 1 1 1 2 2 2 1 3 2 2 4 5	2 2 0	Проверяется сложение матриц с отрицательными элементами

Измерение эффективности

(Процент заполнение - отношение числа ненулевых элементов к общему числу элементов).

Рассмотрим результаты измерения эффективности при обработке матриц 1000 x 1000 с разными процентами заполнения при 1000 повторениях.

Процент заполнения	Время обычая, мс	Время обычая, мс	Память обычная	Память разреженная
1	5175	5175	4 000 000	322 916
5	5336	5336	4 000 000	1 562 212
10	5369	5369	4 000 000	3 042 148
15	5427	5427	4 000 000	4 445 908
20	5377	5377	4 000 000	5 755 732
25	5498	5498	4 000 000	6 997 476
30	5426	5426	4 000 000	8 168 916
40	5551	5551	4 000 000	10 242 100
50	5587	5587	4 000 000	11 999 780
75	5093	5093	4 000 000	15 009 540
100	5403	5403	4 000 000	16 004 004

Рассмотрим результаты измерения эффективности при обработке матриц 2500 x 2500 с разными процентами заполнения при 100 повторениях.

Процент заполнения	Время обычая, мс	Время разреженная, мс	Память обычная	Память разреженная
5	5086	1097	25 000 000	9 741 460
10	4962	2115	25 000 000	19 015 108
15	4789	3161	25 000 000	27 754 948
20	4419	4404	25 000 000	36 012 100
25	4499	5338	25 000 000	43 756 308
30	4266	6383	25 000 000	51 014 052

Вывод

Исходя из полученных данных можно сделать вывод, что обработка матрицы в разреженной форме (CSC - Compressed Sparse Column) эффективнее обычной обработки матриц при проценте заполнения ненулевывыми элементами менее 20%-25%. Эффективность по памяти при работе с разреженными матрицами теряется не сильно (и зависит только от процента заполнения), т.к. превышение порога требуемой памяти по сравнению с классическим форматом хранения матрицы проявляется при низкой разреженности, где алгоритмы и так не применимы.

Ответы на вопросы

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

Разреженная матрица - матрица с большим числом нулевых элементов. Существуют различные методы хранения элементов матрицы в памяти: линейный связный список, диагональная схема хранения симметричных матриц, связные схемы разреженного хранения, разреженный строчный/столбцовый формат.

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

Под обычную матрицу выделяется n * m * sizeof(тип данных матрицы) байт памяти, где n - количество строк, m - количество столбцов матрицы, то есть память выделяется под все элементы. Память под хранение разреженной матрицы выделяется только для информации о ненулевых элементах. Для CSC обозначим число ненулевых элементов за e, тогда объём требуемой памяти: e * (sizeof(size_t) + sizeof(тип данных матрицы)) + m (кол-во столбцов) * sizeof(size_t)

3. ***Каков принцип обработки разреженной матрицы?***

Основной принцип алгрритмов обработки разреженных матриц - обрабатвать только ненулевые элементы, не тратя время на обработку нулевых

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

Стандартные алгоритмы обработки матриц эффективнее при низкой разреженности матриц, т.е. при большом количестве ненулевых элементов, также эффективность обработки зависит от размерности матриц, поскольку при низкой размерности обработка матрицы в стандартной форме эффективнее при любом проценте заполнения