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» (ИУ)

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии» (ИУ7)

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

«ОБРАБОТКА РАЗРЕЖЕННЫХ МАТРИЦ»

по курсу:

«ТИПЫ И СТРУКТУРЫ ДАННЫХ»

Вариант: 1

Студент:

Авдейкина Валерия Павловна, группа ИУ7-33Б _____

(подпись, дата)

Руководители:

Преподаватель ИУ7

Силантьева Александра Васильевна _____

(подпись, дата)

Преподаватель ИУ7

Барышникова Марина Юрьевна _____

(подпись, дата)

Проверяющий:

_____

(подпись, дата)

Оценка: _____

2022 г.

Оглавление

Описание условий 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.

Вариантное задание (1): CSC

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

1 Входные данные

- Входные данные при вводе номера выбранной опции:
  - Целые беззнаковые числа [0; 1]
- Входные данные на разных этапах работы программы:
  - Заполнение матрицы:
    - размер матрицы — целые числа
    - элементы матрицы — целые числа

2 Выходные данные

- Выходные данные при:
  - выводе матриц — матрицы в обычном формате и в формате, заданном вариантом задания
  - внализе эффективности — данные анализа

3 Задача, реализуемая программой

Программа реализует обработу (сложение) матриц при хранении в стандартном формате и в формате разреженной матрицы (CSC).

4 Способ обращения к программе

Программа вызывается в командной строке без каких-либо аргументов. Ввод данных производится с клавиатуры после соответствующего приглашения к вводу («`Команда...`» / «`Введите...`»).

5 Возможные аварийные ситуации и ошибки со стороны пользователя

1. Некорректные данные при вводе с клавиатуры
2. Некорректный ввод номера опции:
   2.1. Пустой ввод
   2.2. Недопустимые символы (*поведение программы не определено*)
   2.3. Недопустимый номер

Описание внутренних структур данных

В ходе работы были составлены следующие структуры:

1. Матрица в стандартном формате:
   - matrix — указатель на начало данных матрицы (начало одномерного массива)
   - rows — количество строк матрицы
   - cols — количество столбцов матрицы

```
int *matrix;
int rows;
int cols;
```

Итоговый размер структуры: 16 Б

2. Матрица в CSC формате:
   - A — массив ненулевых элементов матрицы
   - JA — массив номеров столбцов для ненулевых элементов матрицы
   - IA — линейный односвязный список, каждый элемент которого содержит:
     - i — индекс строки, которой соответствует элемент списка
     - Nk — индекс первого ненулевого элемента строки, которой соответствует элемент списка

```
int *A;
int *JA;
struct IA *IA;

struct IA
{
   int i;
   int Nk;
   struct IA *next;
};
```

Итоговый размер структуры: 24 Б

Алгоритм

- Алгоритм сложения разреженных матриц, используемый в программе:
  1. Пока список IA не пуст для обоих матриц:
     a) если очередная строка 1-й матрицы, соответствующая текущему элементу списка 1-й матрицы, находится «выше», чем следующая ненулевая строка 2-й матрицы, выполняется копирование этой строки 1-й матрицы в результирующую
     b) в обратном случае копируется строка 2-й матрицы
     c) в случае, когда индексы ненулевых строк совпали, происходит сложение строк 1-й и 2-й разреженных матриц (с записью в третью)
  2. Если список IA 1-й матрицы остался непуст, оставшиеся строки копируются в результирующую матрицу
  3. Если список IA 2-й матрицы остался непуст, оставшиеся строки копируются в результирующую матрицу

- Алгоритм копирования строк разреженных матриц:
  1. Начало копирования — индекс первого ненулевого элемента в массиве A копируемой строки
  2. Конец копирования — количество ненулевых элементов матрицы (если ненулевых строк больше нет) или индекс первого ненулевого элемента в массиве A следующей за копируемой строки
  3. В процессе копирования в массивы A и JA матрицы-«назначения» записываются соответствующие элементы массивов A и JA матрицы-«источника», а также создается новый элемент списка IA

- Алгоритм сложения строк разреженных матриц:
  1. Методом, аналогичен способу нахождения пределов копирования, находимы пределы суммирования элементов для двух строк
  2. В пределах суммирования, пока в обоих строках есть ненулевые элементы, выполняем:

- копирование текущего ненулевого элемента 1-й строки в результирующую строку, если его индекс меньше, чем индекс текущего ненулевого элемента 2-й строки
- аналогичное копирование элемента 2-й строки в обратном случае
- сложение, если индексы ненулевых элементов строк совпали

3. Если в 1-й строке остались ненулевые элементы, копируем их в результирующую строку
4. Аналогично действуем с оставшимися элементами 2-й строки

- Алгоритм сложения матриц в стандартном формате:
  1. Для каждого элемента 1-й матрицы получаем сумму его и соответствующего элемента 2-й матрицы и записываем результат в итоговую матрицу

Для реализации алгоритмов в программе были использованы следующие функции:

`void generate_matrix(int *matr, int n, int m, int fill)`
`void input_matrix(int *matr, int n, int m)`
`void convert_matrix(const int *matr, int n, int m, int *A, int *JA, struct IA *IA)`
`void add_matrices(const int *matr1, const int *matr2, int *matr3, int n, int m)`

`void add_matrices_scr(const int *A1, const int *JA1, struct IA *IA1, int num_of_non_zero_elements_1,`
`const int *A2, const int *JA2, struct IA *IA2, int num_of_non_zero_elements_2,`
`int *A3, int *JA3, struct IA *IA3, int *num_of_non_zero_elements_3)`

`void copy_row(const int *A1, const int *JA1, struct IA *IA1, int max_iel,` `int *A3, int *JA3, struct IA *IA3, int *num_of_non_zero_elements_3)`

`void add_rows(const int *A1, const int *JA1, struct IA *IA1, int max_curr_1_index,`
`const int *A2, const int *JA2, struct IA *IA2, int max_curr_2_index,`
`int *A3, int *JA3, struct IA *IA3, int *num_of_non_zero_elements_3)`

Таблица 1. Основные функции

Тестирование

- **Автоматическое заполнение и вывод**

```
Выполнить сложение двух матриц? (1 - да, 0 - нет): 1
Заполнить матрицы вручную? (1 - да, 0 - нет): 0
Введите размерность (кол-во рядов, кол-во столбцов): 7 7
Введите процент заполнения (0 - 100): 10

Вывести матрицы на экран? (1 - да, 0 - нет): 1

ПЕРВАЯ МАТРИЦА
0 0 0 0 0 8 0
8 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 2 0 4 0 5
0 1 0 0 0 6 0
0 0 0 0 0 0 0
5 0 0 0 3 0 0

A: 8 8 2 4 5 1 6 5 3
JA: 5 0 2 4 6 1 5 0 4
IA:
  Non-zero indexes: 0 1 3 4 6
  ANk: 0 1 2 5 7
```

- **Заполнение вручную и вывод**

```
ВВОД ПЕРВОЙ МАТРИЦЫ
Введите элемент (i, j, значение): 1 1
1
Введите элемент (i, j, значение): -1
-1
-1

ВВОД ВТОРОЙ МАТРИЦЫ
Введите элемент (i, j, значение): 1 1 1
Введите элемент (i, j, значение): 4 4 1
Введите элемент (i, j, значение): -1 -1 -1

Вывести матрицы на экран? (1 - да, 0 - нет): 1

ПЕРВАЯ МАТРИЦА
0 0 0 0 0
0 1 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0

A: 1
JA: 1
IA:
  Non-zero indexes: 1
  ANk: 0

ВТОРАЯ МАТРИЦА
0 0 0 0 0
0 1 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 1

A: 1 1
JA: 1 4
IA:
  Non-zero indexes: 1 4
  ANk: 0 1
```

Оценка эффективности

Оценка эффективности проводилась в трех плоскостях: сложение матриц разных размерностей в разных форматах с разным процентом заполнения. Количество итераций: 100. Время указано в микросекундах.

АНАЛИЗ ЭФФЕКТИВНОСТИ

Элементов	g(%)	Не CSR: Время	Размер (Б)	CSR: Время	Размер (Б)
100	5	0	1200	0	240
100	20	0	1200	0	1016
100	50	0	1200	0	1688
10000	5	29	120000	9	21664
10000	20	29	120000	23	62784
10000	50	29	120000	65	135336
1000000	5	2901	12000000	994	1749832
1000000	20	2924	12000000	3355	5813144
1000000	50	3066	12000000	7952	13051040
1000000	90	2923	12000000	8375	20897160

Из результатов анализа можно сделать следующие выводы:

- При увеличении размера матрицы сложение в обоих форматах занимает больше времени
- При увеличении процента заполнения матрица в стандартном формате использует фиксированный объем памяти (максимальный), а матрица в CSC формате — постепенно увеличивающийся объем, который достигает максимального значения при максимальном проценте заполнения
- Операции сложения матриц в стандартном формате занимает больше времени, чем в CSC формате, при проценте заполнения <=15%
- При проценте заполнения = 50 матрица в CSC формате занимает больший объем памяти, чем матрица в стандартном формате

Контрольные вопросы

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

Разреженная матрица — матрица, для которой отношение количества ненулевых элементов к общему количеству элементов находится в интервале от 0,2 до 0,5.

Схемы хранения разреженных матриц:

- линейный связный список
- кольцевой связный список
- диагональная схема хранения симметричных матриц
- связные схемы разреженного хранения
  - схема Кнута (7 массивов, избыточна)
  - разреженный строчный/столбцовый формат (CSR, CSC)

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

Для хранения обычной матрицы выделяется (независимо от значений элементов) rows*cols*sizeof(element) байт памяти, а для разреженной — в зависимости от количества ненулевых размеров, схемы хранения

3 Каков принцип обработки разреженной матрицы?

Принцип обработки разреженной матрицы заключается в том, что обрабатываются лишь ненулевые элементы матрицы, что сокращает затрачиваемое время.

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

Эффективнее применять стандартные алгоритмы обработки матриц при достижении определенного уровня заполненности матрицы (обычно после 30-50%).

Выводы

В ходе работы были изучены детали обработки разреженных матриц и сделан следующий вывод: при сильно разреженных данных более выгодно использовать CSC или CSR формат хранения матрицы, однако эта выгода пропадает, когда данные становятся менее разреженными.