|  |  |
| --- | --- |
| 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 (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04** Программная инженерия

**Отчет**

|  |  |
| --- | --- |
| **По лабораторной работе №** | 3 |

**Название:**

Обработка разреженных матриц

**Дисциплина:** Типы и структуры данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ7-36Б |  | | А.А. Жаворонкова |
|  | (Группа) | |  | (И.О. Фамилия) |
|  |  | |  |  |
| Преподаватель: | Никульшина Т. А. | |  |  |

Москва, 2022

Описание условия задачи

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.

Описание ТЗ

*Описание исходных данных и результатов*

*Исходные данные:* пункт меню (см. описание задачи), матрица, вектор-столбец. *Результаты:* матрица, вектор-столбец, результирующий столбец, таблица эффективности.

*Описание задачи, реализуемой программой*

Меню:

0) Выйти из программы

1) Ввести матрицу и вектор вручную

2) Сгенерировать матрицу и вектор автоматически

3) Классическое умножение

4) Умножение с помощью разреженных векторов

5) Сравнение эффективности

6) Вывести матрицу

7) Вывести вектор

8) Вывести меню

*Способ обращения к программе*

Вызов программы происходит через терминал (main.exe). Дальнейшая работа с программой выполняется при помощи меню, выводимого на экран.

*Описание возможных ошибок пользователя*

* Ввод некорректного пункта меню
* Ввод некорректных элементов матрицы и/или вектора
* Ввод некорректного количества строк и/или столбцов матрицы
* Ввод некорректного индекса элемента

Описание внутренних структур данных

Структура **simple\_matrix\_t** предназначена для хранения матрицы в виде массива указателей на строки.  
**rows** – количество строк  
**columns** – количество столбцов  
**data** – массив указателей на строки матрицы

typedef struct

{

    int rows;

    int columns;

    int \*\*data;

} simple\_matrix\_t;

Структура **matrix\_t** предназначена для хранения матрицы в разреженном виде.  
**A** – массив ненулевых значений матрицы  
**IA** – массив номеров строк для ненулевых значений  
**JA\_head** – указатель на начало связного списка, содержащего элементы, с которых начинается каждый столбец матрицы.  
**rows** – количество строк  
**columns** – количество столбцов  
**elems** – количество ненулевых элементов

typedef struct

{

    int \*A;

    int \*IA;

    JA\_t \*JA\_head;

    int rows;

    int columns;

    int elems;

} matrix\_t;

Структура **JA\_t** описывает элемент односвязного списка.  
**JA** – столбец матрицы  
**next** – указатель на следующий элемент

struct JA\_t

{

    int JA;

    struct JA\_t \*next;

};

Набор тестов с указанием проверяемого параметра

|  |  |  |
| --- | --- | --- |
| **Ввод** | **Вывод** | **Что проверяется** |
| 19 | «Введены некорректные данные» | Неверная команда |
| -1 -1 | «Введены некорректные данные» | Неверные размеры матрицы |
| a | «Введены некорректные данные» | Неверный элемент матрицы |
| Пустой ввод | «Сначала нужно инициализировать матрицу» | Умножение неинициализированной матрицы |
| 25 | «Введены некорректные данные» | Введенное количество ненулевых элементов больше количества элементов матрицы |
| {{1, 0, 3}, {0, 0, 0}, {0, 5, 0}} и {7, 8, 0} | {7, 0, 40} | Умножение квадратной матрицы |
| {5} и {5} | {25} | Умножение матрицы 1х1 |
| {{2, 0, 0, 4}, {0, 5, 8, 0}} и {10, 0, 6, 7} | {48, 30} | Умножение прямоугольной матрицы |

Описание алгоритма

Умножение реализовано двумя способами: с использованием классического и векторного представления матриц.

При классическом умножении матрица и вектор-столбец умножаются при помощи 3х циклов, в которых вычисляются элементы итогового столбца, как сумма произведений соответствующих элементов строки матрицы и столбца.

При векторном умножении происходит обработка только ненулевых элементов. Формируется массив, содержащий индексы соответствующих элементов в векторе-столбце, если элемент нулевой – (-1). Затем заполняется столбец результат: обработка происходит по столбцам, итоговое произведение вычисляется постепенно, т.е. сумма произведений «накапливается» в каждом элементе столбца результата.

Временная эффективность и затраты памяти

Таблица сравнения для матриц размером 100х100:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Процент  заполненности | Обычное представление | | Разреженное представление | |
| Время, нс | Память, б | Время, нс | Память, б |
| 1 | 42 216 | 40 000 | 3 032 | 1 204 |
| 10 | 42 500 | 40 000 | 7 496 | 8 404 |
| 20 | 42 964 | 40 000 | 12 272 | 16 404 |
| 30 | 41 548 | 40 000 | 17 336 | 24 404 |
| 40 | 42 328 | 40 000 | 23 868 | 32 404 |
| 50 | 41 596 | 40 000 | 29 612 | 40 404 |
| 60 | 41 820 | 40 000 | 37 676 | 48 404 |
| 70 | 41 756 | 40 000 | 46 180 | 56 404 |
| 80 | 41 936 | 40 000 | 54 752 | 64 404 |
| 90 | 42 108 | 40 000 | 64 396 | 72 404 |
| 100 | 44 104 | 40 000 | 79 448 | 80 404 |

Таблица сравнения для матриц 1000х1000:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Процент  заполненности | Обычное представление | | Разреженное представление | |
| Время, нс | Память, б | Время, нс | Память, б |
| 1 | 3 487 784 | 4 000 000 | 76 416 | 84 004 |
| 10 | 3 545 856 | 4 000 000 | 401 348 | 804 004 |
| 20 | 3 567 964 | 4 000 000 | 862 272 | 1 604 004 |
| 30 | 3 836 856 | 4 000 000 | 1 471 544 | 2 404 004 |
| 40 | 3 515 848 | 4 000 000 | 2 067 780 | 3 204 004 |
| 50 | 3 595 004 | 4 000 000 | 2 883 340 | 4 004 004 |
| 60 | 3 565 540 | 4 000 000 | 3 549 040 | 4 804 004 |
| 70 | 3 559 548 | 4 000 000 | 4 305 972 | 5 604 004 |
| 80 | 3 231 204 | 4 000 000 | 4 746 752 | 6 404 004 |
| 90 | 2 604 988 | 4 000 000 | 4 529 448 | 7 204 004 |
| 100 | 2 496 100 | 4 000 000 | 5 174 808 | 8 004 004 |

Данные таблицы были получены путем измерения среднего времени работы алгоритма.  
Из таблиц видно, что использование разреженного представления матриц дает преимущество по времени перед обычным представлением до процента заполненности 40 (при размерах 1000х1000) и 60 (при размерах 100х100); по памяти – до процента заполненности 50.  
При этом, если рассматривать матрицу, заполненную менее чем на 50%, выигрыш по времени падает с ростом процента заполненности.  
Также из таблиц видно, что при обычном представлении затрачиваемое время никак не зависит от процента заполненности, только от размеров матриц, в отличие от векторного представления. Так происходит потому, что стандартный алгоритм обрабатывает все элементы матрицы, вне зависимости от того, нулевые они или нет; тогда как векторный метод производит вычисления только с ненулевыми элементами матрицы.

Вывод

При обработке разреженных матриц векторный способ их представления дает преимущества перед стандартным, если матрица больше чем на 70% состоит из нулей. В таком случае использование векторного представления оправдано и по времени, и по памяти. В случаях большей заполненности (меньшей разреженности) следует использовать стандартный алгоритм.

Ответы на вопросы

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

Разреженная матрица — это матрица, содержащая большое количество нулей. Схемы хранения матрицы: связанная схема хранения (с помощью линейных связных списков), кольцевая связанная схема хранения, двунаправленные стеки и очереди, строчной формат, столбцовый формат.

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

Под обычную матрицу (N – количество строк, M – количество столбцов) выделяет N\*M ячеек памяти. Для разреженной матрицы количество ячеек памяти завит от способа. В случае разряженного формата требуется количество ячеек в размере:  
K \* sizeof(<тип элементов матрицы>) + K \* sizeof(int) + (M + 1) \* sizeof(int).

1. Каков принцип обработки разреженной матрицы?

При обработке разреженной матрицы мы работаем только с ненулевыми элементами. Тогда количество операций будет пропорционально количеству ненулевых элементов (прямая зависимость).

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

Эффективнее применять стандартные алгоритмы при большом количестве ненулевых элементов. Стандартный алгоритм будет использоваться либо при большом количестве ненулевых элементов, либо при небольших размерах матриц.