|  |  |
| --- | --- |
|  | **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 К ЛАБОРАТОРНОЙ РАБОТЕ №3 ПО «ТИПАМ И СТРУКТУРАМ ДАННЫХ»**

**Тема: «Обработка разреженных матриц»**

Группа: ИУ7-32Б

|  |  |  |
| --- | --- | --- |
| Студент: |  | Сидоров Максим Михайлович |

Преподаватель: Никульшина Татьяна Александровна

**Цель работы:** научиться обрабатывать разреженные матрицы. С эффективность стандартного алгоритма умножения матрицы на вектор-строку и алгоритма для разреженных матриц.

**Условие 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**. Смоделировать операцию умножения вектора-строки и матрицы, хранящихся в форме 3-х векторов, с получением результата в виде вектора. Произвести операцию умножения, применяя стандартный алгоритм работы с матрицами. Сравнить время выполнения операций и объем памяти при использовании этих 2-х алгоритмов при различном проценте заполнения матриц.

**Входные данные:** номер команды из меню, терминальный ввод.

Команды:

1. Заполнить матрицу и вектор-строку рандомно
2. Заполнить вектор-строку и матрицу вручную
3. Вывести объекты матрицы
4. Вывести вектор-строку
5. Вывести исходную матрицу
6. Умножить специальным алгоритмом для разреженных матриц и вывести результат
7. Умножить стандартным алгоритмом и вывести результат
8. Сравнить алгоритмы
9. Выход

**Выходные данные:** в зависимости от выбранной команды программа может вывести:

- исходную матрицу

- матрицу в виде объектов

- вектор-строку

- результат в виде вектора для двух алгоритмов

- сравнение эффективности

**Аварийные ситуации:**

- ввод неправильной команды

- некорректный ввод элементов матрицы и вектора

- некорректный ввод размерности матрицы

**Описание СД:**

1. Для хранения вектора была выбрано следующее представление:

typedef struct

{

int \*arr;

int len;

} arr;

*Листинг 1. Представление вектора.*

1. Для хранения исходной матрицы было выбрано следующее представление:

typedef struct

{

int \*mat;

int n;

int m;

} static\_matrix;

*Листинг 2. Представление исходной матрицы.*

1. Для хранения матрицы в виде объектов было выбрано следующее представление:

typedef struct

{

int n;

int m;

arr a;

arr ia;

arr ja;

} matrix;

*Листинг 3. Представление матрицы в виде объектов.*

**Функции:**

void free\_vec(arr \*a) – освобождение памяти из-под вектора

void free\_mat(matrix \*a) – освобождение памяти из-под объектов матрицы

void free\_stat\_mat(static\_matrix \*a) – освобождение памяти исходной матрицы

void print\_menu(void) – вывод меню

void print\_stat\_mat(static\_matrix \*a) – вывод исходной матрицы

void print\_vec(arr \*a) – вывод вектора

void print\_mat(matrix \*a) – вывод объектов матрицы

void compare(matrix \*mat, arr \*vec, static\_matrix \*stat\_mat) – сравнение алгоритмов умножения по времени и памяти

int mult\_ord(static\_matrix \*mat, arr \*vec, bool f) – умножение стандартным способом и вывод результата

int mult\_pars(matrix \*mat, arr \*vec, bool f) – умножение специальным алгоритмом для разреженных матриц и вывод результата

void create\_mat\_by\_stat(matrix \*a, static\_matrix \*src) – создание объектов матрицы из исходной матрицы

void generate\_mat\_and\_vec(static\_matrix \*a, matrix \*dst, arr \*vec) – сгенерировать случайную матрицу и вектор, а также создать объекты матрицы

void create\_matrix\_and\_vec(static\_matrix \*mat, arr \*vec) – создание матриц, вектора и объектов матрицы при вводе из терминала

void calc\_ord(static\_matrix \*mat, arr \*vec, arr \*res) – умножение стандартным алгоритмом

void calc\_pars(matrix \*mat, arr \*vec, arr \*res) – умножение специальным алгоритмом

void print\_vec\_obj(arr \*a) – вывод объекта вектора

**Описание алгоритма:**

Перебираем объекты разреженной матрицы. Проходимся по всем элементам текущего столбца, их количество JA[i+1]-JA[i]. Далее для каждого номера строки IA[j] находим соответствие в векторе-строке IA[j], если соответствие нашлось, то умножаем и складываем и так для всех элементов каждого столбца. В результате имеем результат в виде вектора, длиной, равной количеству столбцов матрицы.

**Тестовые данные:**

|  |  |  |
| --- | --- | --- |
| **Описание** | **Вход** | **Выход** |
| Некорректная команда | A, -1, 10 | Некорректная команда |
| Некорректный ввод размерностей матрицы | -1, а, 10001 | Неверное количество столбцов/строк |
| Некорректный ввод элемента матрицы или вектора | а | Некорректный элемент |
| Вывод объектов матрицы | - | Вывод на экран 3-х векторов |
| Вывод вектора-строки | - | Вывод вектора на экран |
| Вывод исходной матрицы | - | Вывод исходной матрицы на экран |
| Умножение стандартным методом | - | Вывод результата в виде вектора |
| Умножение специальным алгоритмом для разреженных матрицы | - | Вывод результата в виде вектора |
| Сравнение алгоритмов | - | Вывод времени работы и объем используемой памяти |

**Сравнение эффективности:**

Сложность специального алгоритма для разреженных матриц прямо пропорционален количеству ненулевых элементов. Сложность стандартного алгоритма – О(n\*m), где n и m – размерности матрицы.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Размерность**  **матрицы** | **Процент заполнения, %** | **Время стандартного алгоритма, мкс** | **Время специального алгоритма для разреженных матриц, мкс** | **Память стандартного алгоритма, байты** | **Память специального алгоритма для разреженных матриц, байты** |
| 1х1 | 0 | 0 | 0 | 4 | 8 |
| 100 | 0 | 0 | 4 | 16 |
| 2х2 | 25 | 1 | 0 | 16 | 20 |
| 50 | 1 | 0 | 16 | 28 |
| 75 | 1 | 0 | 16 | 36 |
| 100 | 1 | 0 | 16 | 44 |
| 10х10 | 25 | 1 | 0 | 400 | 244 |
| 50 | 1 | 0 | 400 | 444 |
| 75 | 1 | 0 | 400 | 644 |
| 100 | 1 | 0 | 400 | 844 |
| 20х20 | 25 | 3 | 0 | 1600 | 884 |
| 50 | 4 | 1 | 1600 | 1684 |
| 75 | 4 | 1 | 1600 | 2484 |
| 100 | 4 | 1 | 1600 | 3284 |
| 50х50 | 25 | 10 | 2 | 10000 | 5204 |
| 50 | 11 | 3 | 10000 | 10204 |
| 75 | 9 | 5 | 10000 | 15204 |
| 100 | 10 | 7 | 10000 | 20204 |
| 100х100 | 25 | 35 | 12 | 40000 | 20404 |
| 50 | 34 | 16 | 40000 | 40404 |
| 75 | 33 | 22 | 40000 | 60404 |
| 100 | 35 | 25 | 40000 | 80404 |
| 1000х1000 | 25 | 3222 | 1675 | 4000000 | 2004004 |
| 50 | 3240 | 1890 | 4000000 | 4004004 |
| 75 | 3244 | 2183 | 4000000 | 6004004 |
| 100 | 3261 | 2416 | 4000000 | 8004004 |

Из таблицы можно увидеть, что специальный алгоритм работает быстрее, чем стандартный. При 25% заполненности выигрыш по времени составляет 55.1%, при 50% заполненности – 47.3%, а при 100% - 33.9%.

Из-за того, что затраты по памяти прямо пропорциональны количеству ненулевых элементов, так при 25% заполненности матрицы, затрачиваемая память для специального алгоритма в 2 раза меньше стандартного, а при 100% заполненности – в 2 раза больше стандартного, при 50% заполненности затрачиваемая памяти примерно одинакова.

**Вывод:** специальный алгоритм для обработки разреженных матриц работает быстрее стандартного, но затраты по памяти зависят от заполненности матрицы. Специальный алгоритм следует использовать:

1. если память ограничена, то только при заполненности матрицы <=50%,
2. если разреженность матрицы <75%.

Если в стандартном алгоритме умножения матриц использовать транспонированную матрицу, то стандартный алгоритм будет работать значительно быстрее специального при заполненности матрицы >75%.

**Ответы на вопросы:**

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

Разреженная матрица – матрица, содержащая большое количество нулей.

Схемы хранения: связанная, кольцевая связанная, диагональная, двунаправленные стеки и очереди, столбцовый формат, строковый формат.

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

Под обычную матрицу память, равная количеству строк умноженных на количество столбцов, а также на тип данных. Для разреженной матрицы количество ячеек памяти завит от способа хранения матрицы. В случае разряженного формата необходимо два списка, размер каждого из которых равен количеству ненулевых элементов, помноженных на тип данных, а также список для хранения элементов первых вхождений в строки(столбцы), равный количеству строк (столбцов), помноженных на тип данных

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

При обработке разреженной матрицы мы работаем только с ненулевыми элементами.

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

Стандартные алгоритмы эффективнее применять при большом количестве ненулевых элементов. Если известно, что количество ненулевых элементов в матрице небольшое, то лучше применять специальные алгоритмы обработки разреженных матриц.