|  |  |
| --- | --- |
| 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 Программное обеспечение ЭВМ и информационные технологии

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3**

**«ОБРАБОТКА РАЗРЕЖЕННЫХ МАТРИЦ»**

Студент ???

Группа ИУ7 – 3?Б

Преподаватель Силантьева А.В.

*2021 г.*

**Описание условия 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.

Указания к выполнению работы

Все логически завершенные фрагменты алгоритма (ввод, вывод, обработка и т.п.) необходимо оформить как подпрограммы. При разработке интерфейса программы следует предусмотреть:

* указание формата и диапазона вводимых данных,
* указание операции, производимой программой,
* наличие пояснений при выводе результата,
* указание формата выводимых данных
* возможность заполнения разреженных матриц вручную (даже при большой размерности, например, 1000\*1000) и автоматически с разным процентом разреженности.

При тестировании программы необходимо:

проверить правильность ввода o проконтролировать правильность вывода данных (т.е. их соответствие требуемому формату);

* проверить правильность выполнения операций; o обеспечить вывод сообщений при отсутствии входных данных («пустой ввод»); o обеспечить вывод сообщений при нулевых результате или вывод нулевого результата при ненулевом входе;
* обеспечить возможность ввода данных и вывода результата как при малых матрицах, так и при больших (например, 1000 \* 1000).
* сравнить время выполнения стандартного алгоритма обработки матриц и алгоритма обработки разреженных матриц при различной заполненности матриц (от 1 элемента до того количества нулей (в %), при котором становится неэффективно использование алгоритма сокращенного умножения).
* сравнить объем требуемой памяти для реализации стандартного алгоритма обработки матриц и алгоритма обработки разреженных матриц при различном проценте заполнения матриц и при различном их размере.
* следует также протестировать программу при полной загрузке системы, то есть при полном заполнении матриц. Программа должна адекватно реагировать на неверный ввод, пустой ввод и выход за границы матрицы или вектора. Необходим

**Описание технического условия**

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 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.

Входные данные:

1. Целое число – размеры матрицы и вектора, команда программы.
2. Дробное число – элементы матрицы.

Выходные данные:

1. Исходная и результирующая матрица в стандартном, разреженном виде.
2. Сравнение времени умножения матриц разного вида.

Функции программы:

1. Заполнение матрицы вручную;
2. Заполнение матрицы из файла;
3. Создание случайной матрицы;
4. Заполнение матрицы по координатам;
5. Заполнение вектора вручную;
6. Заполнение вектора из файла;
7. Создание случайного вектора;
8. Заполнение вектора по координатам;
9. Умножение матрицы и вектора;
10. Сравнение разреженного умножения и обычного.

Обращение к программе:

Запускается через терминал.

Аварийные ситуации:

1. Некорректный ввод номера команды.

На входе: число, большее чем 1 или меньшее, чем 0.

На выходе: «Invalid input»

2. Некорректный файл.

На входе: файл с матрицей некорректных размеров.

На выходе: сообщение «Sorry, but matrix size is wrang»

3. Неверный ввода размер.

На входе: размер, заданный отрицательным числом.

На выходе: сообщение «Sorry, but input is wrang»

4. Неверный размер.

На входе: количество столбцов матрицы не равно количеству строк вектора.

На выходе: сообщение «Sorry but multiplication is not possible»

Описание структуры данных

*Стандартный вид матрицы:*

typedef struct

{

int col;

int row;

double \*data;

} matrix\_t;

int row – количество строк;

int col – количество столбцов;

double \*data **–** значения элементов матрицы.

*Разреженный вид матрицы:*

typedef struct

{

int size;

double \*A;

int \*JA;

node\_t IA;

} spa\_matrix\_t;

int size – количество ненулевых элементов;

double \*A – элементы матрицы;

int \*JA – столбцы соответствующих элементов;

node\_t IA – связанный список строк.

*Связанный список*

typedef struct node\_t node\_t;

struct node\_t

{

int Nk;

node\_t \*next;

};

Nk – значение индекса, с которого начинается описание столбца;

node\_t \*next – указатель на следующий элемент списка.

*Вектор в разреженном виде*

typedef struct

{

int size;

double \*A;

int \*JA;

} spa\_vector\_t;

int size – длина вектора

double \*A – элементы вектора

int \*JA – столбцы соответствующих элементов

**Описание алгоритма умножения**

1. Программа идет по всем строкам матрицы, в которых есть хоть один ненулевой элемент;
2. В выбранной строке берется ненулевой элемент матрицы;
3. Этот элемент умножается на не нулевой элемент вектора, который находится в соответствующей строке;
4. Пункты 2 и 3 повторяются до тех пор, пока в выбранной строке матрицы есть ненулевые элементы.
5. Полученные значения складываются и записываются в соответствующую строку результирующего вектора.

**Описание алгоритма программы**

1. Пользователь последовательно вводит команду.
2. Программа считывает матрицу и делает расчеты.

**Набор тестов**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название теста** | **Пользовательский ввод** | **Результат** |
| 1 | Некорректный ввод команды | j | Invalid input |
| 2 | Некорректный ввод команды | -45 | Invalid input |
| 3 | Неверный размер | -4 | Size should be a positive integer |
| 4 | Неверный размер | 5.3 | Size should be a positive integer |
| 5 | Неверное количество элементов матрицы | 1 2 3  4 5 6  (матрица 2х2) | Invalid size |
| 6 | Ручной ввод матрицы и вектора | Команды 1, 1 | Матрицы записаны и умножены |
| 7 | Считывание матрицы и вектора из файла | Команды 2, 2 | Матрицы записаны и умножены |
| 8 | Создание матрицы и вектора заданного размера | Команды 3, 3 | Матрицы записаны и умножены |
| 9 | В файле неверный размер матрицы | Команда 2 | Invalid size |
| 10 | Неверные размеры матрицы и вектора | Матрица 3х4  Вектор 5 | Sorry but multiplication is not possible |

**Оценка эффективности**

|  |  |  |  |
| --- | --- | --- | --- |
| Разреженность | Размер | Разреженная матрица | Обычная матрица |
| 5% | 10x10 | 220 | 880 |
| 50x50 | 2324 | 20400 |
| 100x100 | 7660 | 80800 |
| 20% | 10x10 | 424 | 880 |
| 50x50 | 6920 | 20400 |
| 100x100 | 25840 | 80800 |
| 50% | 10x10 | 830 | 880 |
| 50x50 | 16100 | 20400 |
| 100x100 | 62200 | 80800 |
| 65% | 10x10 | 860 | 880 |
| 50x50 | 20384 | 20400 |
| 100x100 | 80380 | 80800 |
| 100% | 10x10 | 1480 | 880 |
| 50x50 | 31400 | 20400 |
| 100x100 | 122800 | 80800 |

*Время (мкс)*

|  |  |  |  |
| --- | --- | --- | --- |
| Размерность | Размер | Разреженная матрица | Обычная матрица |
| 5% | 10x10 | 0 | 0 |
| 50x50 | 0 | 15 |
| 100x100 | 5 | 78 |
| 8% | 10х10 | 0 | 0 |
| 500х500 | 4 | 12 |
| 100х100 | 23 | 62 |
| 10% | 10x10 | 0 | 0 |
| 50x50 | 2 | 15 |
| 100x100 | 34 | 46 |
| 15% | 10x10 | 0 | 0 |
| 50x50 | 14 | 15 |
| 100x100 | 62 | 70 |
| 20% | 10х10 | 2 | 0 |
| 50х50 | 20 | 14 |
| 100х100 | 119 | 62 |
| 100% | 10х10 | 3 | 0 |
| 50х50 | 237 | 17 |
| 100х100 | 1439 | 67 |

Ответы на контрольные 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 связанных списков), кольцевая связанная схема хранения, двунаправленные стеки и очереди, диагональная схема хранения, строчной формат.

*2. 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 разреженного формата требуется количество ячеек в размере (*2 \* К + N)* (К — количество ненулевых элементов, умножаем на 2, так как необходимо 2 списка, для хранения информации об этих элементах; N - количество строк матрицы).

*3. 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).

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

Эффективнее применять стандартные алгоритмы выгоднее при большом количестве ненулевых элементов.  
Стоит отметить, что если расход памяти в программе не так важен, но важно время выполнения программы, то в случае умножения матрицы на вектор столбец лучше воспользоваться стандартным алгоритмом при большом количестве ненулевых элементов, и умножение специального (разреженного) в случае небольшого количества ненулевых элементов.

Вывод

Разреженные матрицы удобно использовать, когда важнее память, а не скорость работы. При работе с разреженной матрицей выигрыш по памяти получается даже при 60% заполнении. Однако время значительно увеличивается уже при 20% заполнении. При этом, если матрица заполнена примерно на 15%, выигрыш по времени также у разреженного алгоритма.

В итоге, если заполненность матрицы около 15%, то в любом случае выгоднее использовать разреженную матрицу. Если же заполненность матрицы больше 20%, то необходимо решить, что важнее: память или скорость обработки матрицы.