|  |  |
| --- | --- |
| 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\_\_\_\_\_\_\_\_

**ТИПЫ И СТРУКТУРЫ ДАННЫХ**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3**

**“Обработка разреженных матриц”**

**ВАРИАНТ 4**

Студент\_\_\_\_\_\_\_\_\_\_ \_Гурова Наталия Алексеевна\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*фамилия, имя, отчество*

Группа\_\_\_\_\_\_ИУ7-34Б\_\_\_\_\_\_\_\_\_

Выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_Гурова Н.А.\_\_\_

*подпись, дата фамилия, и.о.*

Принял \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_Барышникова М.Ю.\_\_\_

*подпись, дата фамилия, и.о.*

*2021 г.*

**Цель работы**

Цель работы - 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.

**Задание**

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-строки, 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**

Для генерации матрицы и вектора необходимо ввести количество строк и столбцов матрицы, а также процент заполненности ее и вектора ненулевыми элементами. Кроме того, предусмотрен ручной ввод.

**Выходные данные**

При выборе соответствующего пункта меню, будут выведены имеющиеся данные либо в стандартной форме (п. 10), либо в разреженной (п. 9). При выборе пунктов 11 или 12 будет посчитано произведение матрицы на вектор-строку классическим или разреженным способом соответственно.

**Способ обращения к программе**

Программа может быть вызвана через консоль с помощью команды app.exe

**Аварийные ситуации**

Могут быть выведены такие ошибки, как:

* Некорректный ввод параметров матрицы или вектора
* Не совпадают соответствующие размерности матрицы и вектора
* Неверно введено имя файла

**Структуры данных**

Для хранения информации о столбцах матрицы (JA) использовался односвязный список:

typedef struct Node\_t  
{  
 struct Node\_t \*next;  
 int data;  
} Node\_t;

Поля структуры:

* next – указатель на следующий узел списка
* data – значение соответствующего узла

Для хранения матрицы в обычном формате использовалась следующая структура:

typedef struct  
{  
 size\_t rows;  
 size\_t columns;  
 int \*\*data;  
} matrix\_t;

Поля структуры:

* rows – количество строк в матрице
* columns – количество столбцов в матрице
* data – элементы матрицы

Для хранения матриц в разреженном формате использовалась следующая структура:

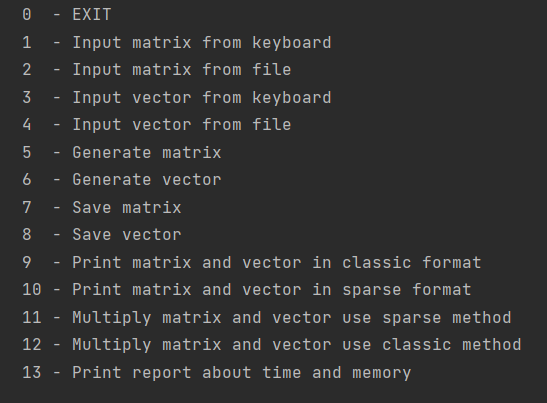
typedef struct  
{  
 size\_t count\_not\_null;  
  
 int \*values;  
 size\_t \*rows;  
 Node\_t \*columns;  
} sparse\_matrix\_t;

Поля структуры:

* count\_not\_null – размер матрицы
* values – массив значений
* rows – массив индексов строк
* columns – односвязный список индексов столбцов

**Описание алгоритма**

Данная программа представляет собой консольное приложение со следующими возможными операциями, представленными в меню:



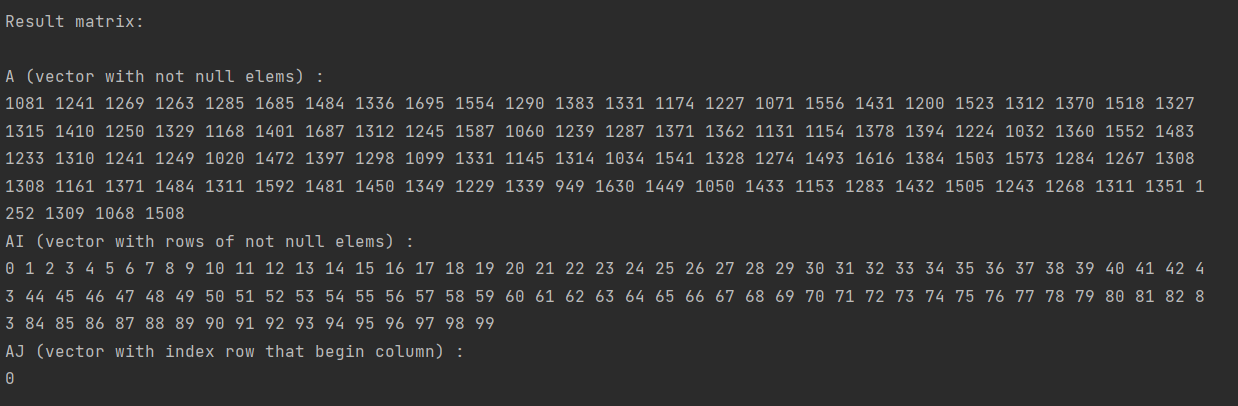
**Реализация умножения**

В данной работе мне требовалось реализовать умножение матрицы и вектора-строки представленных как стандартным способом, так и разреженным.

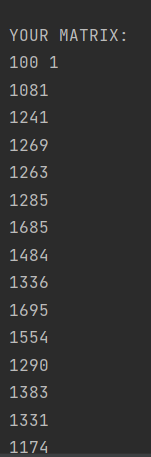
Умножение разреженным способом осуществляется при выборе пункта 11 в первоначальном меню. До этого следует ввести саму матрицу и вектор. Если что-то из этого не будет введено, выведется сообщение об ошибке.

Пример вывода результата умножения.

Разреженное умножение:



Классическое умножение:

****

**Анализ эффективности разрежённого способа хранения**

**Сравнение памяти**

1. 5% заполненность

|  |  |  |
| --- | --- | --- |
| Количество элементов | Стандартная матрица (B) | Разреженная матрица (B) |
| 50\*50 | 10448 | 1848 |
| 100\*100 | 40848 | 5648 |
| 10000\*10 | 440088 | 400188 |

1. 25% заполненность

|  |  |  |
| --- | --- | --- |
| Количество элементов | Стандартная матрица (B) | Разреженная матрица (B) |
| 50\*50 | 10448 | 5848 |
| 100\*100 | 40848 | 21648 |
| 100000\*10 | 440088 | 200088 |

1. 50% заполненность

|  |  |  |
| --- | --- | --- |
| Количество элементов | Стандартная матрица (B) | Разреженная матрица (B) |
| 50\*50 | 10448 | 10848 |
| 100\*100 | 40848 | 41648 |
| 10000\*10 | 440088 | 400016 |

1. 75% заполненность

|  |  |  |
| --- | --- | --- |
| Количество элементов | Стандартная матрица (B) | Разреженная матрица (B) |
| 50\*50 | 10448 | 15848 |
| 100\*100 | 40848 | 61648 |
| 10000\*10 | 440088 | 6000188 |

1. 95% заполненность

|  |  |  |
| --- | --- | --- |
| Количество элементов | Стандартная матрица (B) | Разреженная матрица (B) |
| 50\*50 | 10448 | 20896 |
| 100\*100 | 40848 | 81648 |
| 10000\*10 | 440088 | 8000188 |

**Сравнение времени**

1. 5% заполненность

|  |  |  |
| --- | --- | --- |
| Количество элементов | Стандартная матрица | Разреженная матрица |
| 50\*50 | 0.119 | 0.006 |
| 100\*100 | 0.214 | 0.064 |
| 10000\*10 | 0.456 | 0.107 |

1. 25% заполненность

|  |  |  |
| --- | --- | --- |
| Количество элементов | Стандартная матрица | Разреженная матрица |
| 50\*50 | 0.10 | 0.034 |
| 100\*100 | 0.512 | 0.246 |
| 10000\*10 | 0.686 | 0.340 |

1. 50% заполненность

|  |  |  |
| --- | --- | --- |
| Количество элементов | Стандартная матрица | Разреженная матрица |
| 50\*50 | 0.140 | 0.098 |
| 100\*100 | 0.802 | 0.561 |
| 10000\*10 | 1.256 | 0.879 |

1. 75% заполненность

|  |  |  |
| --- | --- | --- |
| Количество элементов | Стандартная матрица | Разреженная матрица |
| 50\*50 | 0.164 | 0.148 |
| 100\*100 | 1.040 | 0.936 |
| 10000\*10 | 1.399 | 1.554 |

1. 95% заполненность

|  |  |  |
| --- | --- | --- |
| Количество элементов | Стандартная матрица | Разреженная матрица |
| 50\*50 | 0.165 | 0.236 |
| 100\*100 | 0.924 | 1.320 |
| 10000\*10 | 1.302 | 1.860 |

**Контрольные вопросы**

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

Разреженная матрица – матрица с преимущественно нулевыми элементами. Число ненулевых элементов в матрице порядка n может выражаться как n^(1+g), где g < 1. Значения g лежат в интервале 0.2 ... 0.5, т.е. матрица разрежена.

Существуют различные методы хранения элементов матрицы в памяти.

Например, линейный связный список, т.е. последовательность ячеек, связанных в определенном порядке. Каждая ячейка списка содержит элемент списка и указатель на положение следующей ячейки.

Можно хранить матрицу, используя кольцевой связный список, двунаправленные стеки и очереди.

Существует диагональная схема хранения симметричных матриц, а также связные схемы разреженного хранения.

Связная схема хранения матриц, предложенная Кнутом, предлагает хранить в массиве (например, в AN) в произвольном порядке сами элементы, индексы строк и столбцов соответствующих элементов (например, в массивах I и J), номер (из массива AN) следующего ненулевого элемента, расположенного в матрице по строке (NR) и по столбцу (NC), а также номера элементов, с которых начинается строка (указатели для входа в строку – JR) и номера элементов, с которых начинается столбец (указатели для входа в столбец - JC).

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

Для хранения обычной матрицы: N \* M \* sizeof(elem). Память под разреженную матрицу выделяется в зависимости от схемы хранения. Кроме того, память зависит от количества ненулевых элементов.

3. Каков принцип обработки разреженной матрицы?

Обработка разреженной матрицы предполагает работу только с ненулевыми элементами (таким образом, количество операций пропорционально количеству ненулевых элементов).

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

Разреженность матрицы следует учитывать только в том случае, если из этого можно извлечь выгоду за счёт игнорирования нулевых элементов.

При достижении определенного процента наполнения ненулевыми элементами происходит значительное падение эффективности по времени.

**Вывод**

Использование разреженной матрицы имеет смысл, если в матрице много нулей (больше 75%) и, если работа идет с матрицами больших размеров (количество элементов 150 и больше). В этом случае можно получить выигрыш по времени и памяти. (Но нужно помнить, что алгоритм обработки матриц в этом случае значительно усложняется).

Время выполнения стандартного алгоритма зависит от размерности матрицы. Этот алгоритм эффективен при высоком заполнении матрицы (больше 75%). В этом случае получается выигрыш по памяти примерно в 2 раза и выигрыш по времени примерно в 1.4 раза.