|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство образования и науки 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»**

Студент, группа **Пронин А.С. ИУ7-32Б**

*2020 г.*

# 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 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

## Входные данные

1) Целое число от 0 до 7 (для ввода номера команды из меню)

2) Целое число от 1 до 5000 (для ввода кол-ва строк и столбцов)

3) Целое число от 1 до 100 (для ввода процента разреженности)

4) Целое число от 1 до максимального кол-ва элементов в матрице (rows\*columns) (для ввода кол-ва ненулевых элементов разреженной матрицы)

5) Целые числа через пробел (для ввода элементов разреженной матрицы)

6) Целые числа от 0 до кол-ва строк введённых ранее через пробел (для ввода номеров строк для элементов разреженной матрицы)

6) Целые числа через пробел (для ввода элементов связного списка JA)

## Выходные данные

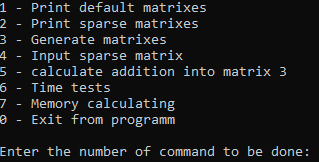
Вывод матриц в обычном или разреженном виде

Время выполнения сложения матриц обычным способом и методом разреженных матриц.

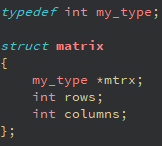
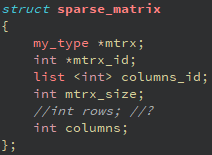
Сравнение объёмов занимаемой памяти в зависимости от способа.

## Функции программы

Программа представляет из себя консольное приложение исполняющее следующие команды:



## Структуры данных

# Описание алгоритма

При запуске программы создаются 3 матрицы с заранее заданными размерами (3 строчки на 4 столбца), которые хранятся в обычном и разреженном видах. Первые две заполняются с заранее заданным процентом разреженности (40%) а в третью сохраняется их сумма. После этого выводится меню.

Команды:

1 – Выводит все 3 матрицы в обычном виде

2 – Выводит все 3 матрицы в разреженном виде

3 – Генерирует матрицы 1 и 2 с заданными размерами и процентом разреженности

4 – Позволяет ввести матрицу в разреженном виде и сохранить её в 1, 2 или 3 матрицу

5 – Складывает обычные матрицы 1 и 2 обычным способом, а матрицы в разреженном виде, разреженным.

6 – Выполняет команду 3 и 5 считая и выводя время сложения разными способами.

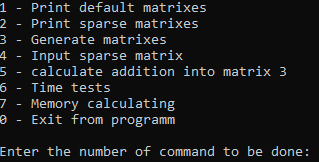
7 – Считает необходимое кол-во памяти для хранения матрицы заданного размера при заданной заполненности для разных методов хранения.

0 – Завершает программу

Алгоритм сложения матриц в разреженном виде реализован в функции int **sparse\_matrix\_sum**(sparse\_matrix a, sparse\_matrix b, sparse\_matrix \*rez), которая складывает матрицы a и b в разреженном виде, сохраняя результат в \*rez. Данная подпрограмма параллельно проходит по разреженным матрицам a и b. Она определяет какие элементы из массивов a.mtrx и b.mtrx нужно взять для вычисления текущего элемента rez->mtrx после чего также сохраняет текущий номер строки в массив rez->mtrx\_id. Попутно с этим, каждый раз когда мы переключаемся на новый столбец, вычисляется текущий элемент связного списка rez->columns\_id хранящий номера элементов с которых начинается каждый столбец. (Перед передачей указателя на разреженную матрицу rez, для неё уже должна быть выделена память. Делается это с помощью функции int **sparse\_matrix\_calloc**(), а кол-во элементов в результирующей матрице определяется предварительно с помощью функции int **get\_elems\_amount**())

# Тесты

### Вот так выглядит меню:



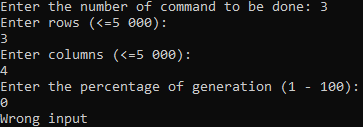
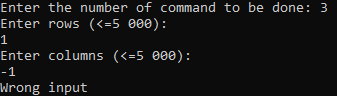
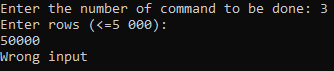
### Некорректный ввод:

1. Некорректная команда

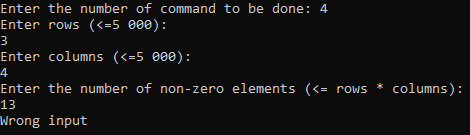




1. Команда 3 (Генерация матриц)



1. Команда 4 (Ввод разреженной матрицы)



Неккоректный ввод в остальных местах аналогичен тому что выше (rows, columns и percantage)

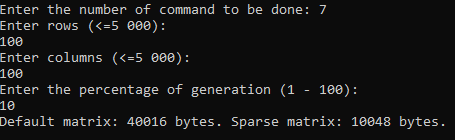
# Оценка эффективности

## Память

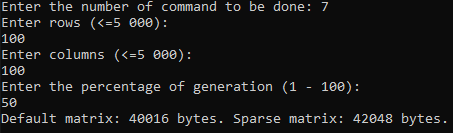
С помощью команды 7 определим память занимаемую матрицами разного типа.

Размер 100X100

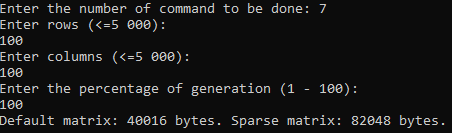
10%



50%

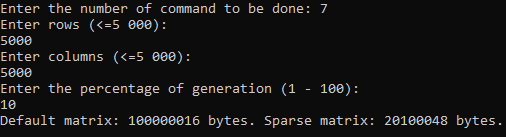


100%

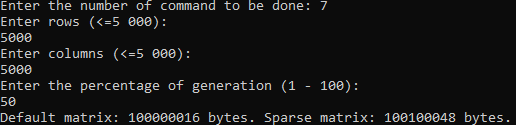


Размер 5000X5000

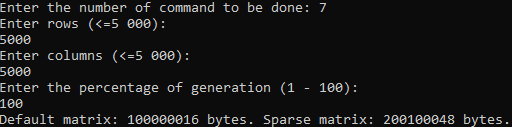
10%



50%



100%

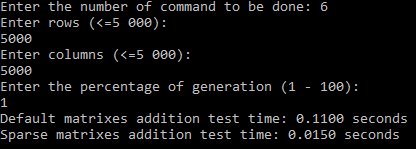


Из тестов можно сделать вывод что при заполненности ~<45% способ хранения матриц в разреженном виде выигрывает у обычного.

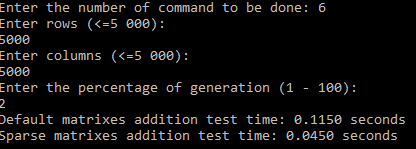
## Эффективность

Размер 5000X5000

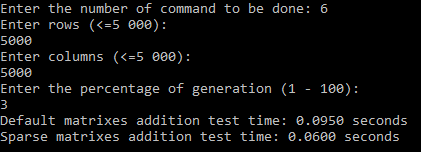
1%



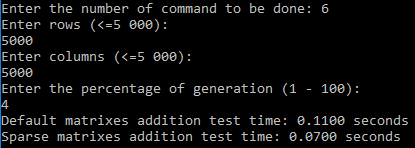
2%



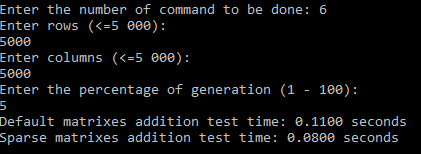
3%



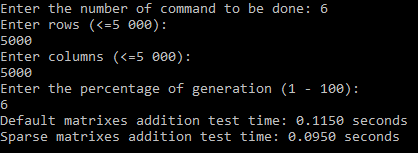
4%



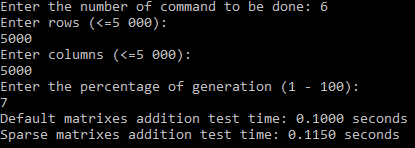
5%



6%



7%



Как мы можем наблюдать, метод сложения разреженных матриц выигрывает по времени у обычного до 6% заполненности

# Контрольные вопросы

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

Разреженная матрица – это матрица, содержащая большое количество нулей. Способы хранения: связная схема хранения, строчный формат, столбцовый формат, линейный связный список, кольцевой связный список, двунаправленные стеки и очереди.

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

Под обычную матрицу выделяет N \* M ячеек памяти, где N – строки, а M – столбцы. Для разреженной матрицы – зависит от способа. В моём случае требуется 2 \* K + M ячеек памяти, где K – количество ненулевых элементов.

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

Алгоритмы обработки разреженных матриц предусматривают действие только с ненулевыми элементами, и, таким образом, количество операций будет пропорционально количеству ненулевых элементов

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

Стандартные алгоритмы обработки матриц эффективнее применять при большом количестве ненулевых элементов. Стоит отметить, что если не так важна память, занимаемая матрицами, но важно время, то в случае сложения лучше так же воспользоваться стандартными алгоритмами сложения матриц.

# Вывод

Использование алгоритмов хранения и сложения разреженных матриц выгодно при небольшом количестве ненулевых элементов, примерно до 6% заполненности матриц. Хранение матриц в разреженном столбцовом формате по сравнению с обычными матрицами выгодно при не более 45% заполненности матриц.