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
КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

ОТЧЕТ ПО УЧЕБНОЙ ДИСЦИПЛИНЕ “ТИПЫ И СТРУКТУРЫ ДАННЫХ” ЛАБОРАТОРНАЯ РАБОТА №3

Студент Ляпина Наталья Викторовна
фамилия, имя, отчество

Группа ИУ7-32Б

Вариант 1

Студент _____ Ляпина Н.В.
подпись, дата *фамилия, и.о.*

Преподаватель _____ Силантьева А.В.
подпись, дата *фамилия, и.о.*

2021 г.

Оглавление

Задание

1. **Общее задание**
   Реализовать алгоритмы обработки разреженных матриц, сравнить этих алгоритмов со стандартными алгоритмами обработки матриц при различном размере матриц и степени их разреженности.

2. **Задание по варианту**
   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 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

3. **Входные данные**

- Команда для выбора действия (от 0 до 3)
  1. Смоделировать операцию сложения двух матриц, хранящихся в особой форме, с получением результата в той же форме.
  2. Произвести операцию сложения, используя стандартный алгоритм работы с матрицами.
  3. Сравнить время выполнения операции и объем памяти при использовании этих 2-х алгоритмов при различном проценте заполнения матриц.
  0. Выйти из программы.

- Размер матрицы (кол-во строк и столбцов)
- Матрица (при ручном заполнении)
- Элементы матрицы, их координаты и количество (при заполнении по координатам)
- Процент заполнения матриц (при автоматическом заполнении)

- **Требования к входным данным**

1) Количество строк и столбцов не может не натуральным
2) Координаты элементов должны быть валидным (не выходить за пределы матрицы)
3) Процент заполнения матриц находится в промежутке от 0 до 100

4. Выходные данные
- Первая матрица
- Вторая матрица
- Результат сложения

5. Действие программы

Программа выполняет сложение плотных и разреженных матриц, сравнение эффективности этих двух подходов

5. Обращение к программе

Программа может быть вызвана через консоль компилятора с помощью команды “./app.exe”

6. Аварийные ситуации

В случае аварийной ситуации выводится сообщение об определенной ошибке.

Неверный ввод:

- Ошибка при вводе команды
- Ошибка при вводе кол-ва столбцов
- Ошибка при вводе кол-ва строк
- Ошибка при вводе координат матрицы
- Ошибка при вводе процента заполнения матрицы

Структура данных

Для реализации данной задачи была создана структура *matrix_t ,* которая используется для хранения плотных матриц. Эта структура содержит , матрицу matrix, количество строк и количество столбцов матрицы.

Хранение плотной матрицы

```
typedef struct
{
        int **matrix;
        int rows;
        int columns;
} matrix_t;
```

Хранение разреженной матрицы в особой форме осуществляется с помощью структуры *sparce_matrix_t*. Она содержит три массива: *vect_a* - массив ненулевых элементов матрицы, *vect_ja* - массив индексов столбцов ненулевых элементов матрицы и *linked_list_ia* - массив индексов строк ненулевых элементов матрицы. Также структура содержит размер матрицы *rows columns* и количество ненулевых элементов *elems*.

Хранение записи таблицы

```
typedef struct
{
        int *vect_a;
        int *vect_j;
        int *linked_list_ia;
        int rows;
        int columns;
        int elems;
} sparce_matrix_t;
```

Интерфейс модулей

Для обработки типа matrix_t используются функции:

```
int simple_addition(void)
// Функция объединяющая в себе 2 пункт меню
//

void print_matrix(matrix_t *matrix)
// Функция печати плотной матрицы
// matrix - матрица

void addition_simple(matrix_t *res_matrix, matrix_t *matrix, matrix_t *matrix_2)
// Функция непосредственно сложения двух матриц
// res_matrix - результирующая матрица, matrix - первая матрица, matrix_2 - вторая матрица
```

int random_simple_matrix(matrix_t *matrix, int fill_size)
// Функция автоматического заполнения матрицы
// matrix - матрица , fill_size - процент заполнения

int read_elem_matrix(matrix_t *matrix)
// Функция чтения матрицы по ненулевым элементам
// matrix - матрица

int read_matrix(matrix_t *matrix)
// Функция чтения матрицы по строкам
// matrix - матрица

int get_method_simple(int *method, int *fill_size)
// Функция получающая метод ввода матрицы
// method - метод ввода, fill_size - процентное заполнение

int create_simple_matrix(matrix_t *matrix)
// Функция выделения памяти для матрицы
// matrix - матрица

int allocate_matrix(int row, int column)
// Функция выделения памяти
// row - количество строк , column - количество столбцов

int free_matrix(int **arr, int n)
// Функция освобождения памяти матрицы
// arr - указатель на матрицу , n - количество строк

Для обработки типа sparce_matrix_t используются функции:
int sparce_addition(void)
// Функция объединяющая в себе 1 пункт меню
//

int res_size(sparce_matrix_t *mat_1, sparce_matrix_t *mat_2)
// Функция получения размера результирующей матрицы
// mat_1 - первая матрица, mat_2 - вторая матрица.

void addition_matrix_sparce(sparce_matrix_t *res_mat, sparce_matrix_t *mat_1, sparce_matrix_t *mat_2)
// Функция непосредственно сложения двух матриц

// res_mat - результирующая матрица, mat_1 - первая матрица, mat_2 - вторая матрица.

void print_sparce_matrix(sparce_matrix_t *matrix)
// Функция вывода на экран матрицы
// matrix - матрица.

void print_zero_matrix(sparce_matrix_t *matrix)
// Функция вывода на экран нулевой матрицы
// matrix - матрица.

int random_sparce_matrix(sparce_matrix_t *matrix)
// Функция автоматического заполнения матрицы
// matrix - матрица.

int read_sparce_matrix(sparce_matrix_t *matrix)
// Функция чтения всей матрицы с клавиатуры
// matrix - матрица.

void free_sparce_matrix(sparce_matrix_t *matrix)
// Функция освобождения памяти
// matrix - матрица.

int create_sparce_matrix(sparce_matrix_t *matrix)
// Функция выделения памяти под матрицу
// matrix - матрица.

int read_sparce_no_zero(sparce_matrix_t *matrix)
// Функция чтения количества ненулевых элементов матрицы
// matrix -матрица .

int get_method_simple(int *method, int *fill_size)
// Функция получающая метод ввода матрицы
// method - метод ввода, fill_size - процентное заполнение.

int read_matrix_size(int *rows, int *columns)
// Функция получения размера матрицы
// rows - количество строк, columns - количество столбцов.

Описание алгоритма

1. Вводится команда для выполнения определенной функции программы.
2. Выполняется введенная функция
    - Смоделировать операцию сложения двух матриц, хранящихся в особой форме, с получением результата в той же форме.
    - Произвести операцию сложения, используя стандартный алгоритм работы с матрицами.
    - Сравнить время выполнения операции и объем памяти при использовании этих 2-х алгоритмов при различном проценте заполнения матриц.
    - Выйти из программы.
3. При ошибке выполнения функции выводится сообщение об ошибке и программа завершается с ненулевым кодом возврата.

Тесты

Негативные тесты

№	Описание	Вводимая структура	Результат
1	Недопустимый символ команде меню	a2	Ошибка
2	Несуществующая команда меню	5	Ошибка
3	Недопустимый размер матрицы	-10 на 0	Ошибка
4	Недопустимый метод заполнения матрицы	5	Ошибка

5	Недопустимый процент заполнения матрицы	120	Ошибка
6	Количество ненулевых элементов матрицы больше размера матрицы или меньше	0	Ошибка
7	При вводе матрицы по координатам номера столбцов или строк больше допустимых	матрица 3x3 Элемент:3 Номер строки: 4 Номер столбца: -1	Ошибка
8	Повторный ввод элемента, который уже был заполнен	матрица 3x3 Элемент:3 Номер строки: 1 Номер столбца: 1 Элемент:4 Номер строки: 1 Номер столбца: 1	Ошибка
9	Вместо значения элемента буква	Элемент:a Номер строки: 4 Номер столбца: -1	Ошибка
10	Недопустимый символ в команде	5g	Ошибка

Позитивные тесты

№	Описание	Входные данные	Результат
1	Правильный пункт меню	comand = 1	Выбор действия
2	Чтение обычной матрицы построчно	size = 2x2 1 1 1 1	Успешное чтение матрицы
3	Чтение матрицы поэлементно	size = 2x2 Элемент:3 Номер строки: 1 Номер столбца: 1	Успешное чтение матрицы

4	Заполнение матрицы автоматически	method = 2 fill_size = 25	Успешное заполнение матрицы
5	Сложение двух одинаковых плотных матриц	size = 2x2 1 1 1 1 size = 2x2 1 1 1 1	size = 2x2 2 2 2 2
6	При вводе двух матриц в особой форме обе матрицы нулевые	size = 2x2 0 0 0 0 size = 2x2 0 0 0 0	size = 2x2 0 0 0 0
7	При вводе двух матриц в особой форме одна их них нулевая	size = 2x2 0 0 0 0 size = 2x2 1 1 1 1	size = 2x2 1 1 1 1

Оценка эффективности

```
                    Матрицы заполнены на 10%

Обычные матрицы (10x10 и 10x10)                 712 тактов, 0.0000003096 секунд
Затраченная память под массив 10x10 - 400 байт

Матрицы в особой форме (10x10 и 10x10)          375 тактов, 0.0000001630 секунд
Затраченная память под массив 10x10 - 124 байт


Обычные матрицы (50x50 и 50x50)                 17917 тактов, 0.0000077900 секунд
Затраченная память под массив 50x50 - 10000 байт

Матрицы в особой форме (50x50 и 50x50)          6291 тактов, 0.0000027352 секунд
Затраченная память под массив 50x50 - 2204 байт


Обычные матрицы (100x100 и 100x100)                 69628 тактов, 0.0000302730 секунд
Затраченная память под массив 100x100 - 40000 байт

Матрицы в особой форме (100x100 и 100x100)          31312 тактов, 0.0000136139 секунд
Затраченная память под массив 100x100 - 8404 байт
```

```
                        Матрицы заполнены на 15%

Обычные матрицы (10x10 и 10x10)                 739 тактов, 0.0000003213 секунд
Затраченная память под массив 10x10 - 400 байт

Матрицы в особой форме (10x10 и 10x10)          440 тактов, 0.0000001913 секунд
Затраченная память под массив 10x10 - 164 байт


Обычные матрицы (50x50 и 50x50)                 16469 тактов, 0.0000071604 секунд
Затраченная память под массив 50x50 - 10000 байт

Матрицы в особой форме (50x50 и 50x50)          8901 тактов, 0.0000038700 секунд
Затраченная память под массив 50x50 - 3204 байт


Обычные матрицы (100x100 и 100x100)                 62673 тактов, 0.0000272491 секунд
Затраченная память под массив 100x100 - 40000 байт

Матрицы в особой форме (100x100 и 100x100)          46702 тактов, 0.0000203052 секунд
Затраченная память под массив 100x100 - 12404 байт
```

```
                        Матрицы заполнены на 20%

Обычные матрицы (10x10 и 10x10)                 650 тактов, 0.0000002826 секунд
Затраченная память под массив 10x10 - 400 байт

Матрицы в особой форме (10x10 и 10x10)          522 тактов, 0.0000002270 секунд
Затраченная память под массив 10x10 - 204 байт


Обычные матрицы (50x50 и 50x50)                 16191 тактов, 0.0000070396 секунд
Затраченная память под массив 50x50 - 10000 байт

Матрицы в особой форме (50x50 и 50x50)          11720 тактов, 0.0000050957 секунд
Затраченная память под массив 50x50 - 4204 байт


Обычные матрицы (100x100 и 100x100)                 62842 тактов, 0.0000273226 секунд
Затраченная память под массив 100x100 - 40000 байт

Матрицы в особой форме (100x100 и 100x100)          65106 тактов, 0.0000283070 секунд
Затраченная память под массив 100x100 - 16404 байт
```

```
                        Матрицы заполнены на 25%

Обычные матрицы (10x10 и 10x10)                 661 тактов, 0.0000002874 секунд
Затраченная память под массив 10x10 - 400 байт

Матрицы в особой форме (10x10 и 10x10)          615 тактов, 0.0000002674 секунд
Затраченная память под массив 10x10 - 244 байт


Обычные матрицы (50x50 и 50x50)                 15696 тактов, 0.0000068243 секунд
Затраченная память под массив 50x50 - 10000 байт

Матрицы в особой форме (50x50 и 50x50)          15647 тактов, 0.0000068030 секунд
Затраченная память под массив 50x50 - 5204 байт


Обычные матрицы (100x100 и 100x100)                 65717 тактов, 0.0000285726 секунд
Затраченная память под массив 100x100 - 40000 байт

Матрицы в особой форме (100x100 и 100x100)          82779 тактов, 0.0000359909 секунд
Затраченная память под массив 100x100 - 20404 байт
```

```
                        Матрицы заполнены на 50%

Обычные матрицы (10x10 и 10x10)                  681 тактов, 0.0000002961 секунд
Затраченная память под массив 10x10 - 400 байт

Матрицы в особой форме (10x10 и 10x10)           1069 тактов, 0.0000004648 секунд
Затраченная память под массив 10x10 - 444 байт


Обычные матрицы (50x50 и 50x50)                  16513 тактов, 0.0000071796 секунд
Затраченная память под массив 50x50 - 10000 байт

Матрицы в особой форме (50x50 и 50x50)           30477 тактов, 0.0000132509 секунд
Затраченная память под массив 50x50 - 10204 байт


Обычные матрицы (100x100 и 100x100)                 63792 тактов, 0.0000277357 секунд
Затраченная память под массив 100x100 - 40000 байт

Матрицы в особой форме (100x100 и 100x100)          159744 тактов, 0.0000694539 секунд
Затраченная память под массив 100x100 - 40404 байт
```

```
                        Матрицы заполнены на 75%

Обычные матрицы (10x10 и 10x10)                   655 тактов, 0.0000002848 секунд
Затраченная память под массив 10x10 - 400 байт

Матрицы в особой форме (10x10 и 10x10)            1535 тактов, 0.0000006674 секунд
Затраченная память под массив 10x10 - 644 байт


Обычные матрицы (50x50 и 50x50)                   16679 тактов, 0.0000072517 секунд
Затраченная память под массив 50x50 - 10000 байт

Матрицы в особой форме (50x50 и 50x50)            36707 тактов, 0.0000159596 секунд
Затраченная память под массив 50x50 - 15204 байт


Обычные матрицы (100x100 и 100x100)                 64766 тактов, 0.0000281591 секунд
Затраченная память под массив 100x100 - 40000 байт

Матрицы в особой форме (100x100 и 100x100)          202724 тактов, 0.0000881409 секунд
Затраченная память под массив 100x100 - 60404 байт
```

```
                        Матрицы заполнены на 95%

Обычные матрицы (10x10 и 10x10)                  713 тактов, 0.0000003100 секунд
Затраченная память под массив 10x10 - 400 байт

Матрицы в особой форме (10x10 и 10x10)           1325 тактов, 0.0000005761 секунд
Затраченная память под массив 10x10 - 804 байт


Обычные матрицы (50x50 и 50x50)                  15427 тактов, 0.0000067074 секунд
Затраченная память под массив 50x50 - 10000 байт

Матрицы в особой форме (50x50 и 50x50)           33274 тактов, 0.0000144670 секунд
Затраченная память под массив 50x50 - 19204 байт


Обычные матрицы (100x100 и 100x100)                 63808 тактов, 0.0000277426 секунд
Затраченная память под массив 100x100 - 40000 байт

Матрицы в особой форме (100x100 и 100x100)          141924 тактов, 0.0000617061 секунд
Затраченная память под массив 100x100 - 76404 байт
```

Вывод

С увеличением плотности заполнения матрицы снижается преимущество по занимаемой разреженной матрицей памяти по сравнению с плотной. Так, при увеличении процента наполненности от 0% до 50% превосходство объема памяти для плотной матрицы над объемом памяти для разреженной снижается от 475% до 3%.

При 18% и меньше наполненности матрицы эффективнее по памяти обрабатывать разреженные матрицы, чем плотные.

Ответы на контрольные вопросы

1. Что такое разреженная матрица, какие способы хранения вы знаете?

Разреженная матрица - это матрица, содержащая большое количество нулей.
Способы хранения:

- Словарь по ключам (DOK - Dictionary of Keys) строится как словарь, где ключ это пара (строка, столбец), а значение это соответствующий строке и столбцу элемент матрицы
- Список списков (LIL - List of Lists) строится как список строк, где строка это список узлов вида (столбец, значение)
- Список координат (COO - Coordinate list) хранится список из элементов вида (строка, столбец, значение)
- Сжатое хранение строкой (CSR - compressed sparse row, CRS - compressed row storage, Йельский формат)
- Сжатое хранение столбцом(CSC - compressed sparse column, CCS - compressed column storage) То же самое что и CRS, только строки и столбцы меняются ролями - значения храним по столбцам, по второму массиву можем определить строку, после подсчётов с третьим массивом - узнаём столбцы.

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

Под хранение обычной матрицы выделяется $n \times m \times sizeof(type)$ памяти. Под хранение разреженной матрицы выделяется $3 \times k \times sizeof(type)$, где k - количество ненулевых элементов в матрице.

3. Каков принцип обработки разреженной матрицы?

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

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

Для матриц эффективнее по времени применять стандартные алгоритмы при проценте заполненности от 75%. По памяти эффективно применять стандартные алгоритмы при проценте заполненности от 25%. При этом, чем выше становится размер матрицы, тем стандартные алгоритмы эффективнее