# Лабораторная работа №3. "Разреженные матрицы"

## **Студент** *Ивахненко Дмитрий - ИУ7-36Б*

## Описание условия задачи

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 вектора-строки и матрицы, 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.

## Техническое задание

**Входные данные**:

1. Целое число в диапазоне `[0; 8]`, представляющее собой номер команды программы.
2. Командно-зависимые данные:
    - Ввод матрицы
        - количество строк/столбцов матрицы (натуральные числа)
        - количество ненулевых элементов матрицы (натуральное число, не превышающее общее кол-во элементов)
        - элементы матрицы в формате **`индекс_строки;индекс_столбца=значение_элемента`**, где
            - индекс - целое число в диапазоне `[1; количество_столбцов(строк)]`
            - значение - целое число.
    - Ввод вектора
        - длина вектора (натуральное число)
        - количество ненулевых элементов вектора (натуральное число, не превышающее длину вектора)
        - элементы вектора в формате **`индекс_элемента=значение_элемента`**, где
            - индекс - целое число в диапазоне `[1; количество_элементов]`
            - значение - целое число.
    - Генерация матрицы
        - количество строк/столбцов матрицы (натуральные числа)
        - процент ненулевых элементов матрицы (целое числов диапазоне `[0;100]`)
    - Генерация вектора
        - длина вектора (натуральное число)

- процент ненулевых элементов вектора (целое число в диапазоне `[0; 100]`)

**Выходные данные**:

1. Напечатать матрицу/вектор
   - исходная матрица/вектор
2. Умножить вектор на матрицу
   - исходная матрица
   - исходный и результирующий вектора в стандартном и разреженном видах.
3. Произвести сравнительный анализ вариантов умножения вектора на матрицу
   - Количественная характеристика сравнения вариантов умножения вектора на матрицу, представленная в виде таблицы

## Команды программы

### Работа с матрицей

1. Ввести матрицу
2. Сгенерировать случайную матрицу
3. Напечатать матрицу

### Работа с вектором

4. Ввести вектор
5. Сгенерировать случайный вектор
6. Напечатать вектор

### Операция умножения

7. Умножить вектор на матрицу
8. Провести сравнение разных преставлений матриц

## Обращение к программе:

запускается из терминала при помощи комманды `./bin/app`.

## Аварийные ситуации:

1. Некорректный ввод номера команды в меню.

   **На входе**: символ, отличный от целого числа из диапазона, указанного в ТЗ.

   **На выходе**: сообщение о некорректном вводе номера команды.

2. Некорректный ввод количества строк или столбцов матрицы (длины вектора).

   **На входе**: символ, не являющийся целым числом, принадлежащим диапазону, указанному в ТЗ.

   **На выходе**: сообщение некорректном вводе кол-ва строк/столбцов матрицы (длины вектора).

3. Некорректный ввод индекса строки или столбца матрицы (элемента вектора).

**На входе**: символ, не являющийся целым числом, принадлежащим диапазону, указанному в ТЗ.

**На выходе**: сообщение о некорректном вводе индекса строки/столбца матрицы (элемента вектора).

4. Некорректный ввод значения элемента матрицы/вектора.

   **На входе**: символ, не являющийся числом.

   **На выходе**: сообщение о некорректном вводе элемента матрицы/вектора.

## Структуры данных

Для хранение элементов матриц используется тип `mtype_t`, который является псевдонимом `long`

```
typedef long mtype_t;
```

За стандартное хранение матрицы отвечает именованная структура `matrix_t`:

```
typedef struct
{
    size_t rows;        // кол-во строк
    size_t cols;        // кол-во столбцов
    mtype_t **data;     // данные

} matrix_t;
```

За хранение матрицы в разреженном виде отвечает именованная структура `sp_matrix_t`:

```
typedef struct
{
    size_t rows;  // кол-во строк
    size_t cols;  // кол-во столбцов
    size_t count;  // кол-во ненулевых элементов

    mtype_t *values;  // массив значений
    size_t *col_indexes;  // массив индексов столбцов
    size_t *start_rows;  // массив индексов начал строк

} sp_matrix_t;
```

**Вектор представляется в виде вырожденной матрицы, имеющей одну строку.**

# Алгоритм

- **Общий алгоритм**

1. Пользователю на экран выводится меню с доступными командами.
2. Пользователь вводит число - номер команды.
3. Выполняется команда, которую выбрал пользователь.

- **Алгоритм случайной генерации матрицы/вектора `N` элементами**

    1. Первые `N` элементов матрицы, при просмотре по строкам заполняются случайными элементами
    2. Матрица перемешиваетсяы
        - При просмотре матрицы по строкам текущий элемент меняется со случайным элементом, следующим за текущим

- **Стандартный алгоритм умножения вектора-строки на матрицу**

    1. Создается результирующий вектор, длиной равной кол-ву столбцов в матрице.
    2. Каждый элемент `K_n` нового вектора является суммой произведений каждого элемента `J_m` старого вектора на каждый элемент `I_m`, находящийся в `n-м` столбце матрицы.

- **Алгоритм умножения разреженного вектора и на матрицу**

    1. Создается результирующий вектор, длиной равной кол-ву столбцов в матрице.
    2. Матрица просматривается по строкам
    3. Каждый элемент `i-й` строки умножается на элемент вектора с индексом `i-й`, если он существует, и данная сумма добавляется к `j-му` элементу результируещего вектора, где `j` - столбец текущего элемента в матрице. После чего, в случае, если какие-то элементы результирующего вектора остались не заполнены, данный вектор "ужимается".

# Описание основных функций

- `matrix_t *matrix_create(size_t rows, size_t cols);`
    - принимает кол-во строк и столбцов
    - инициализирует матрицу
    - возвращает созданную матрицу
- `void del_matrix(matrix_t *matrix);`
    - принимает матрицу
    - освобождает матрицу
- `sp_matrix_t *sp_matrix_create(size_t rows, size_t cols, size_t count)`
    - принимает кол-во строк, столбцов и ненулевых элементов матрицы
    - инициализирует разреженную матрицу
- `void del_sp_matrix(sp_matrix_t *matrix);`
    - принимает разреженную матрицу
    - освобождает разреженную матрицу
- `matrix_t *multiply(const matrix_t *left, const matrix_t *right, clock_t *time);`
    - принимает две матрицы

    - умножает две матрицы
    - возвращает результат и время выполнения
- `sp_matrix_t *sp_multiply(const sp_matrix_t *vec, const sp_matrix_t *m, clock_t *time);`
    - принимает разреженные вектор-строку и матрицу
    - умножает разреженные вектор-строку и матрицу
    - возвращает результат и время выолнения
- `sp_matrix_t *m_to_sparse(matrix_t *m);`
    - принимает стандартную матрицу
    - конвертирует принятую матрицу в разреженный формат
    - возвращает разреженную матрицу
- `matrix_t *sp_to_matrix(sp_matrix_t *sparse);`
    - принимает разреженную матрицу
    - конвертирует принятую матрицу в стандартный формат
    - возвращает стандартную матрицу

## Тесты

| **Описание теста** | **Вввод** | **Вывод** |
|---|---|---|
| Неправильный номер команды | -1 | Сообщение об ошибке |
| Неправильный номер команды | 11 | Сообщение об ошибке |
| Неправильный размер матрицы | 1 0 0 | Сообщение об ошибке |
| Неправильный номер элемента | 1 1 1 1 0;0=1 | Сообщение об ошибке |
| Неправильный номер элемента | 1 1 1 1 2;2=1 | Сообщение об ошибке |
| Неправильный размер вектора | 4 0 | Сообщение об ошибке |
| Неправильный номер элемента вектора | 4 1 1 0=1 | Сообщение об ошибке |
| Неправильный номер элемента вектора | 4 1 1 2=1 | Сообщение об ошибке |
| Неправильная заполненность матрицы | 2 1 1 101 | Сообщение об ошибке |
| Неправильная заполненность вектора | 5 1 101 | Сообщение об ошибке |
| Неправильные размеры для умножения | 2 1 1 0 5 2 0 7 | Сообщение об ошибке |
| Удачный ввод матрицы | 1 10 10 1 2;2=5 | Матрица размером 10x10 с одним элементом |
| Удачный ввод вектора | 4 10 1 2=8 | Вектор длиной 10 с одним элементом |

| Описание теста | Вввод | Вывод |
|---|---|---|
| Удачное перемножение | 1 2 2 2 1;1=4 2;2=5 4 2 1 2=55 7 | вектор 1x2 с одним элементом 275 |

## Оценка эффективности

Для каждого измерения взято среднее значение по времени для **100** разных случайных пар матрица-вектор.

- Время работы указано в микросекундах (μs)
- Объём матриц указан в байтах.
- **Заполненность** указана в процентах

| Размер матрицы | Заполнености матриц | Время умножения стандарных матриц | Время умножения разреженных матриц | Объём стандартных матриц | Объём разреженных матриц |
|---|---|---|---|---|---|
| 16X16 | 0 | 3 | 2 | 2224 | 272 |
| 16X16 | 10 | 4 | 3 | 2224 | 688 |
| 16X16 | 20 | 4 | 4 | 2224 | 1136 |
| 16X16 | 30 | 3 | 4 | 2224 | 1552 |
| 16X16 | 40 | 4 | 4 | 2224 | 2000 |
| 16X16 | 50 | 4 | 5 | 2224 | 2448 |
| 16X16 | 60 | 3 | 6 | 2224 | 2864 |
| 16X16 | 70 | 3 | 6 | 2224 | 3312 |
| 16X16 | 80 | 3 | 6 | 2224 | 3728 |
| 16X16 | 90 | 3 | 6 | 2224 | 4176 |
| 16X16 | 100 | 4 | 6 | 2224 | 4624 |
| 32X32 | 0 | 11 | 2 | 8496 | 528 |
| 32X32 | 10 | 12 | 4 | 8496 | 2208 |
| 32X32 | 20 | 9 | 7 | 8496 | 3888 |
| 32X32 | 30 | 8 | 9 | 8496 | 5584 |
| 32X32 | 40 | 11 | 12 | 8496 | 7264 |
| 32X32 | 50 | 13 | 12 | 8496 | 8976 |
| 32X32 | 60 | 12 | 14 | 8496 | 10656 |
| 32X32 | 70 | 11 | 13 | 8496 | 12336 |

| Размер матрицы | Заполненость матриц | Время умножения стандарных матриц | Время умножения разреженных матриц | Объём стандартных матриц | Объём разреженных матриц |
|---|---|---|---|---|---|
| 32X32 | 80 | 10 | 15 | 8496 | 14032 |
| 32X32 | 90 | 9 | 15 | 8496 | 15712 |
| 32X32 | 100 | 10 | 16 | 8496 | 17424 |
| 64X64 | 0 | 27 | 2 | 33328 | 1040 |
| 64X64 | 10 | 24 | 7 | 33328 | 7680 |
| 64X64 | 20 | 24 | 8 | 33328 | 14336 |
| 64X64 | 30 | 22 | 26 | 33328 | 20992 |
| 64X64 | 40 | 26 | 28 | 33328 | 27648 |
| 64X64 | 50 | 27 | 31 | 33328 | 34320 |
| 64X64 | 60 | 30 | 34 | 33328 | 40960 |
| 64X64 | 70 | 29 | 38 | 33328 | 47616 |
| 64X64 | 80 | 27 | 39 | 33328 | 54272 |
| 64X64 | 90 | 28 | 42 | 33328 | 60928 |
| 64X64 | 100 | 31 | 49 | 33328 | 67600 |
| 128X128 | 0 | 119 | 3 | 132144 | 2064 |
| 128X128 | 10 | 117 | 17 | 132144 | 28464 |
| 128X128 | 20 | 116 | 34 | 132144 | 54880 |
| 128X128 | 30 | 114 | 116 | 132144 | 81312 |
| 128X128 | 40 | 117 | 121 | 132144 | 107728 |
| 128X128 | 50 | 114 | 127 | 132144 | 134160 |
| 128X128 | 60 | 113 | 132 | 132144 | 160560 |
| 128X128 | 70 | 117 | 139 | 132144 | 186976 |
| 128X128 | 80 | 119 | 147 | 132144 | 213408 |
| 128X128 | 90 | 123 | 156 | 132144 | 239824 |
| 128X128 | 100 | 129 | 161 | 132144 | 266256 |
| 256X256 | 0 | 450 | 5 | 526384 | 4112 |
| 256X256 | 10 | 464 | 51 | 526384 | 109360 |
| 256X256 | 20 | 497 | 124 | 526384 | 214640 |

| Размер матрицы | Заполненость матриц | Время умножения стандарных матриц | Время умножения разреженных матриц | Объём стандартных матриц | Объём разреженных матриц |
|---|---|---|---|---|---|
| 256X256 | 30 | 461 | 217 | 526384 | 319888 |
| 256X256 | 40 | 452 | 459 | 526384 | 425168 |
| 256X256 | 50 | 471 | 496 | 526384 | 530448 |
| 256X256 | 60 | 478 | 548 | 526384 | 635696 |
| 256X256 | 70 | 467 | 563 | 526384 | 740976 |
| 256X256 | 80 | 482 | 574 | 526384 | 846224 |
| 256X256 | 90 | 449 | 581 | 526384 | 951504 |
| 256X256 | 100 | 440 | 591 | 526384 | 1056784 |
| 512X512 | 0 | 1736 | 6 | 2101296 | 8208 |
| 512X512 | 10 | 1757 | 181 | 2101296 | 428448 |
| 512X512 | 20 | 1704 | 445 | 2101296 | 848688 |
| 512X512 | 30 | 1691 | 780 | 2101296 | 1268944 |
| 512X512 | 40 | 1704 | 1382 | 2101296 | 1689184 |
| 512X512 | 50 | 1696 | 1795 | 2101296 | 2109456 |
| 512X512 | 60 | 1745 | 2117 | 2101296 | 2529696 |
| 512X512 | 70 | 1859 | 2180 | 2101296 | 2949936 |
| 512X512 | 80 | 1707 | 2097 | 2101296 | 3370192 |
| 512X512 | 90 | 1748 | 2134 | 2101296 | 3790432 |
| 512X512 | 100 | 1708 | 2191 | 2101296 | 4210704 |
| 1024X1024 | 0 | 6785 | 10 | 8396848 | 16400 |
| 1024X1024 | 10 | 6617 | 694 | 8396848 | 1695744 |
| 1024X1024 | 20 | 6605 | 1773 | 8396848 | 3375104 |
| 1024X1024 | 30 | 6701 | 3091 | 8396848 | 5054464 |
| 1024X1024 | 40 | 6656 | 4832 | 8396848 | 6733824 |
| 1024X1024 | 50 | 6763 | 6884 | 8396848 | 8413200 |
| 1024X1024 | 60 | 6721 | 7743 | 8396848 | 10092544 |
| 1024X1024 | 70 | 6618 | 8158 | 8396848 | 11771904 |
| 1024X1024 | 80 | 6856 | 8479 | 8396848 | 13451264 |

| Размер матрицы | Заполненость матриц | Время умножения стандарных матриц | Время умножения разреженных матриц | Объём стандартных матриц | Объём разреженных матриц |
|---|---|---|---|---|---|
| 1024X1024 | 90 | 6959 | 8465 | 8396848 | 15130624 |
| 1024X1024 | 100 | 6824 | 8557 | 8396848 | 16810000 |
| 2048X2048 | 0 | 26720 | 18 | 33570864 | 32784 |
| 2048X2048 | 10 | 26521 | 2752 | 33570864 | 6746928 |
| 2048X2048 | 20 | 26564 | 6821 | 33570864 | 13461088 |
| 2048X2048 | 30 | 26956 | 12441 | 33570864 | 20175264 |
| 2048X2048 | 40 | 26838 | 19748 | 33570864 | 26889424 |
| 2048X2048 | 50 | 27086 | 27452 | 33570864 | 33603600 |
| 2048X2048 | 60 | 26908 | 31340 | 33570864 | 40317744 |
| 2048X2048 | 70 | 27034 | 31369 | 33570864 | 47031904 |
| 2048X2048 | 80 | 26989 | 31387 | 33570864 | 53746080 |
| 2048X2048 | 90 | 27013 | 31510 | 33570864 | 60460240 |
| 2048X2048 | 100 | 27049 | 32923 | 33570864 | 67174416 |

## Контрольные вопросы

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

   Разреженная матрица – это матрица, содержащая большое количество нулей. Способы хранения: связная схема хранения в строчном либо столбцовом формате, 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 выделяется `N * M` ячеек памяти, где `N` – строки, а `M` – столбцы. Для разреженной матрицы – зависит от способа. В моем случае требуется `2 * K + (N + 1)` ячеек памяти, где `K` – количество ненулевых элементов.

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 обработки матриц? От чего это зависит?

Стандартные алгоритмы обработки матриц эффективнее применять при большом количестве ненулевых элементов (при разреженности ниже 40-50%).

## Вывод

Проанализировав таблицу, можно заметить, что, например, для заполнености 20% матрицы 2048x2048 разреженный формат выполняется на 257% быстрее и при этом объём требуемой памяти на 149% меньше. А для процента 50% заполненности разреженный формат сравним с обычным форматом по времени работы и по объёму занимаемой памяти. Для процента заполненности 100% обычный метод быстрее на 18%, и объём требуемой памяти памяти меньше в 2 раза.

Таким образом, можно сделать вывод, что при заполненности матрицы менее 50% для задачи умножения вектора строки на матрицу целосообразненнее использовать метод хранения разреженных матриц, в ином случае следует использовать стандартное хранение матриц.