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 **Рядинский К.В., ИУ7-33Б**

2020 г.

Условие задачи

Реализовать алгоритмы обработки разреженных матриц, сравнить эффективность использования этих алгоритмов (по времени выполнения и требуемой памяти) со стандартным алгоритмами обработки матриц при различном процентном заполнения матриц ненулевыми значениями и при различных размерах матриц.

Техническое задание

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.

Входные данные

Пункт меню:

1. Ввод матрицы — Вводятся целые числа. Кол-во строк и столбцов строго больше нуля
2. Генерация случайной матрицы — Вводятся целые числа. Процент заполнения 0 — 100
3. Ввод вектора — Вводятся целые числа. Кол-во элементов строго больше нуля
4. Генерация случайного вектора — — Вводятся целые числа. Процент заполнения 0 — 100
5. Умножение обычным алгоритмом
6. Умножение в разреженном виде
7. Вывод результата умножения
8. Очистка экрана
9. Печать матрицы
10. Печать вектора на экран

11. Выход из программы

Выходные данные

Результат умножение и время умножение.

Возможные аварийные случаи

1. Неправильный ввод
2. Длина вектора не соответствует кол-ву строк матрицы

Тесты

Ввод	Вывод
Неверно введенный размер матрицы	Неправильно введен размер матриц.
Неправильный элемент матрицы	Неправильно введен элемент матрицы.
Неверно введенный размер вектора	Неправильно введен размер вектора.
Неверно введенный элемент вектора	Неправильно введен элемент вектора.
Размер вектора не соответствует кол-ву строк матрицы	Длина вектора не равна кол-ву столбцов матрицы. Умножение невозможно.
Умножение «обычное»: Вектор: 1 2 3 Матрица: 1 4 6 7 3 5 8 3 5	39 19 31
Умножение «разреженное»: Вектор: 1 2 3 Матрица: 1 4 6 7 3 5 8 3 5	39 19 31 0 1 2 0 3
Введена только матрица	Недостаточно операндов умножения. Не были введены матрица или вектор.
Введен только вектор	Недостаточно операндов умножения. Не были введены матрица или вектор.

Структуры данных

Структура разреженной матрицы:

```
typedef struct sparse_matrix_s
{
    vec_t *non_zero;
    vec_t *clms;
    vec_t *descr;
} sparse_matrix_t;
```

Структура обычной матрицы:

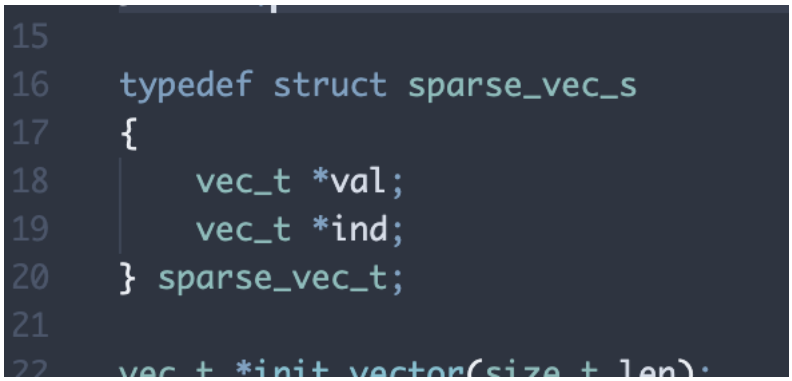
```
typedef struct matrix_s
{
    int x;
    int y;

    int **matrix;
} matrix_t;
```

Структура вектора:

```
typedef struct vec_s
{
    vec_type_t *arr;
    size_t len;
    size_t alloc_len;
} vec_t;
```

Структура разреженного вектора



Алгоритм

Матрица транспонируется. Далее, умножение происходит вектор на вектор столько раз, сколько в исходной матрице столбцов. Для этого, я прохожу по массиву ненулвых элементов, и пользуясь индексами столбцов этих элементов из массива IA, последовательно перемножаю каждый элемент текущий строки на нужный элемент вектора-строки.

sparse_matrix_t *convert_matrix(const matrix_t *matrix) — конвертация обычной матрицы в разреженную

int def_mult(const vec_t *vec, const matrix_t *matrix, vec_t **res) — обычное умножение вектора на матрицу

int sparse_mult(const vec_t *vec, matrix_t *matrix, sparse_matrix_t **res) — умножение разреженного вектора на разреженную матрицу

void print_hello(void) — начальная «приветствующая» печать

void print_help(void) — печать с помощью по командам

int rnd_matrix(matrix_t **matrix) — получение случайной матрицы

Сравнение

Время

Замер времени: замеряется процессорное время с точностью до наносекунды. (См. time.h clock_gettime)

5% нулей

Размеры матрицы	Разреженная матрица	Обычная матрица
10	11000	7000
100	160000	80000
1000	20017000	19069000

10% нулей

Размеры матрицы	Разреженная матрица	Обычная матрица
10	10000	7000
100	140000	80000
1000	19460000	19855000

20% нулей

Размеры матрицы	Разреженная матрица	Обычная матрица
10	10000	9000
100	135000	85000
1000	18366000	19723000

50% нулей

Размеры матрицы	Разреженная матрица	Обычная матрица
10	7000	8000
100	104000	85000
1000	18631000	19723000

90% нулей

Размеры матрицы	Разреженная матрица	Обычная матрица
10	6000	8000
100	81000	85000
1000	15570000	19723000

Занимаемая память

1% заполнение

Размеры матрицы	Разреженная матрица	Обычная матрица
10	168	400
100	2400	40000
500	28000	1000000

10% заполнение

Размеры матрицы	Разреженная матрица	Обычная матрица
10	240	400
100	9600	40000
500	208000	1000000

50% заполнение

Размеры матрицы	Разреженная матрица	Обычная матрица
10	560	400
100	41600	40000
500	1008000	1000000

100% заполнение

Размеры матрицы	Разреженная матрица	Обычная матрица
10	960	400
100	81600	40000
500	2008000	1000000

Вывод

По результатам тестов видно, что разреженный формат матриц выгоден при больших размерах и большом кол-ве нулей.

Использование алгоритмов хранения и обработки разреженных матриц выгодно при большом размере матрицы с большим содержанием нулевых элементов. В таком случае алгоритмы позволяют получить выигрыш как в объеме потребляемой памяти, так и во времени обработки. Иначе мы получаем незначительный выигрыш по времени, но приходится использовать больше памяти.

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, строчный 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 случае разреженного формата, требуется 3 * K 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 (от 50%). Стоит отметить, что если не так важна память, занимаемая матрицами, но важно время, то можно лучше использовать алгоритм обработки разреженных матриц, так как он, хоть и немного, но выигрывает во времени даже на больших % заполненостях матриц (80% - 100%)