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»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

Студент **Звягин Даниил Олегович**

Группа **ИУ7-33Б**

Название дисциплины **Типы и структуры данных**
Название предприятия **НУК ИУ МГТУ им. Н. Э. Баумана**

Студент _____ **Звягин Д. О.**

Преподаватель _____ **Барышникова М. Ю.**

Оценка _____

2023 г.

Описание условия задачи

1. Смоделировать операцию умножения матрицы и вектора-столбца, хранящегося в форме вектора A и вектора, содержащего номера строк ненулевых элементов, с получением результата в форме хранения вектора-столбца.
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

Разработать программу для работы с типом данных «Разреженная матрица» и типом данных «матрица».

Разреженная матрица хранится в форме 3-х объектов:

- вектор A содержит значения ненулевых элементов;
- вектор JA содержит номера столбцов для элементов вектора A;
- вектор IA, в элементе N_k которого находится номер компонент в A и JA, с которых начинается описание строки N_k матрицы A.

Провести операцию умножения каждого из этих типов на вектор-столбец и сравнить эффективность алгоритмов умножения при разном проценте заполнения исходной матрицы.

Входные данные

1. Число - Пункт меню (см. далее)

```
Меню:
0: Выход
1: Ввести матрицу с клавиатуры
2: Добавить элемент в матрицу
```

```
3: Удалить элемент из матрицы
4: Ввести столбец для умножения
5: Умножить матрицу на столбец
6: Сравнить умножение по производительности
7: Вывести матрицу
```

В соответствии с каждым пунктом меню, вам может понадобиться (или не понадобиться) ввод дополнительных данных, его содержание можно понять из названия пунктов меню. (Так, например, для того, чтобы добавить число в матрицу, вам будет необходимо ввести индексы этого элемента и его значение)

Выходные данные

В зависимости от пункта меню:

Состояние матриц (всех введённых) в данный момент.

Умноженные матрицы.

Результаты замеров (по времени и памяти) алгоритмов умножения.

Сравнение средних значений времени и памяти, требуемых для умножения.

Меню.

Сообщения об ошибках.

Способы обращения к программе

1. Программа не имеет графического интерфейса, поэтому вызывается из консоли (терминала) вызовом исполняемого файла (например app.exe). Для работы с программой, можно воспользоваться текстовым меню. Входные данные передаются через стандартный поток ввода консоли (ввод с клавиатуры или перенаправление потоков), а выходные – через стандартный поток вывода (вывод в консоль или в файл с помощью перенаправления потока).

Возможные аварийные ситуации

1. Недостаток памяти для записи (или создания) матрицы (любого типа или её копии)

Никакие ошибочные случаи не приводят к прекращению работы программы. Будет либо предложено повторить ввод, если ошибка заключалась в нём, либо запрошенное действие не будет выполнено.

Описание внутренних структур данных

Перечисляемый тип для пунктов меню

```
enum MENU_ITEMS
{
    EXIT,
    INPUT_MATRIX_KEYBOARD,
    ADD_ELEMENT,
    REMOVE_ELEMENT,
    INPUT_COLUMN,
    MULTIPLY,
    MULTIPLICATION_TEST,
    PRINT_MATRIX,
    N_MENU_ITEMS
};
```

Структура для типа данных «матрица» в «обычном» виде

```
typedef struct
{
    int32_t **rowsptr;
    size_t rows;
    size_t columns;
} matrix_t;
```

Перечисляемый тип для ошибок во время использования интерфейса работы с типом «matrix_t»

```
enum matrix_errors
```

```
{
    NOT_ENOUGH_MEMORY = 1,
    OUT_OF_BOUNDS,
    SCANF_ERROR,
    SIZE_MISMATCH,
};
```

Структура для типа данных «матрица» в «разреженном виде»

```
typedef struct
{
    size_t rows;
    size_t columns;
    size_t el_count;

    int32_t *A;
    size_t *JA;
    ssize_t *IA;
} sparce_matrix_t;
```

Перечисляемый тип ошибок во время работы с функциями ввода с ограничениями файла «my_utils.h»

```
enum INPUT_ERRORS
{
    BAD_COORD_INPUT = 1,
    X_TOO_BIG,
    Y_TOO_BIG,
};
```

Описание алгоритмов умножения матриц

1. «Традиционное» умножение матрицы на столбец по алгоритму умножения обычных матриц – «строка на столбец»

2. Умножение разреженной матрицы на столбец
   2.1. Пустые строки исходной матрицы пропускаются
   2.2. Для непустых строк определяется конечный индекс в массиве A
   2.3. Производится умножение ненулевых элементов, индексы в столбце определяются с помощью массива JA
   2.4. Сумма записывается в соответствующую ячейку результирующей матрицы

Замерный эксперимент

Замеры проводились на квадратных матрицах из-за удобства генерации, но программа поддерживает любые формы матриц. Форма матрицы не должна влиять на скорость умножения, поэтому привожу информацию о матрицах размера NxN. Матрицы заполнены (псевдо)случайными числами от -100 до 100, разумеется, за исключением нулевых элементов.

Время приведено в наносекундах:

N	% заполн.	Кол-во замеров	Обычная матрица		Разреженная матрица	
			Время (Нс)	Память (Б)	Время (Нс)	Память (Б)
20	5	10000	6280	1680	372	704
20	20	10000	6717	1680	548	1388
20	50	10000	6517	1680	911	2960
20	100	10000	7193	1680	1698	5348
100	5	10000	110845	40400	2960	8788
100	20	10000	120835	40400	8670	26332
100	50	10000	114610	40400	20267	62140
100	100	10000	117652	40400	37457	122140
500	5	1000	2641814	1002000	51389	164372
500	20	1000	2709121	1002000	189349	611408
500	50	1000	2672533	1002000	434782	1506956
500	100	1000	2716863	1002000	1169008	2998616
1000	5	1000	10754663	4004000	187707	627940
1000	20	1000	11196655	4004000	820709	2416120
1000	50	1000	10684210	4004000	1767468	5995288
1000	100	1000	10672434	4004000	3445027	11966908
10000	5	10	1085600209	400040000	18850362	60019444
10000	20	10	1086343263	400040000	70711034	239151292

10000	50	10	1096215241	400040000	173430100	597331732
10000	100	10	1093930072	400040000	345980877	1194320968

Анализ полученных результатов:

В результате исследования выяснилось, что умножение разреженной матрицы «обгоняет» умножение обычной матрицы вне зависимости от процента заполнения. Я связываю это с тем, что обычная матрица хранится в виде массива указателей на строки (которые «разбросаны по памяти»), а разреженная матрица хранит все три массива едиными блоками.

Однако размер прироста эффективности напрямую зависит от процента заполнения матрицы. Так, при заполнении матрицы на 5%, скорость умножения улучшается от 15 до 50 раз, при заполнении на 20% от 12 до 15 раз, при заполнении на 50% от 3 до 7 раз, а при заполнении на 100% от 2 до 4 раз. (размер прироста также зависит и от размера матрицы, поэтому есть нижнее и верхнее значения)

Что касается объёма памяти, занимаемого матрицами – разреженная матрица выигрывает обычную только при низком заполнении. Уже при заполнении на 50%, объём памяти, занимаемый разреженной матрицей превышает обычную в полтора-два раза, а при полном заполнении - в три.

Выводы по проделанной работе

Для обработки матриц с низким процентом заполнения, выгодно использовать тип данных «разреженная матрица», который позволяет увеличить эффективность хранения и обработки таких матриц в несколько раз, однако нельзя забывать об ограничениях – при увеличении процента заполнения, разреженные матрицы начинают сильно проигрывать обычным матрицам в объёме занимаемой памяти.

Контрольные вопросы

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

Разреженная матрица – это матрица, содержащая большое количество нулевых элементов. Способы хранения таких матриц: связанная схема хранения, строчный формат, линейный связанный список, кольцевой связанный список, двунаправленные стеки и очереди.

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

Для разреженной матрицы нашего способа хранения выделяется 2K+N ячеек памяти, где K – количество ненулевых элементов, а N – количество строк матрицы. (2K т.к. нужен массив значений и столбцов)

Для обычной матрицы выделяется N*M ячеек памяти, где N – количество рядов, а M – количество строк матрицы

3. Каков принцип обработки разреженной матрицы?

При обработке разреженных матриц мы выполняем те же действия, что и с обычными матрицами, но игнорируем нулевые элементы, что позволяет сэкономить время (или избежать выполнения бессмысленных операций).

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

Если мы рассматриваем эффективность по скорости, выгоднее использовать разреженные матрицы, однако если важно количество памяти, занимаемое структурой, обычные матрицы могут начать сильно выигрывать разреженные, при заполнении более чем на 30% (в моём случае)