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 **ПРОГРАММНАЯ ИНЖЕНЕРИЯ (ИУ7)**

О Т Ч Е Т

По лабораторной работе № 3

Название: Обработка разреженных матриц

Дисциплина: Типы и структуры данных

Вариант: 3

Студент ИУ7-31Б (Группа) Савинова М. Г. (Фамилия И. О.)

Преподаватель Силантьева А. В.

Москва, 2022

Оглавление

Цель работы

Реализация алгоритмов обработки разреженных матриц, сравнение этих алгоритмов со стандартными алгоритмами обработки матриц при различном размере матриц и степени их разреженности.

Условие 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 матрицы и вектора-столбца, 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.

Описание ТЗ

1. Описание входных данных

Исходными данными являются целочисленные матрицы.

Ввод данных происходит с помощью выбора 1-3 пункта в меню.

```
This programm provides STORAGE, MULTIPLICATION of matrix in
ordinary and sparse form
Also you have the opportunity to compare efficiency of the
different storage forms, size, %fill

1. Manual input
2. File input
3. Random input
4. Check efficiency
```

```
// пример корректного ввода матрицы

MIN value: 0, MAX value: 2000
Input rows and columns: 10 10

*ATTENTION: any wrong symbol means the ending of the input!

Input row / column / value: 1 1 1
Input row / column / value: 3 3 3
Input row / column / value: a
```

```
// пример корректного хранения матрицы в файле
3 3
1 0 3
-88 56 0
0 0 0
```

Некорректный ввод вызывает завершение программы с ненулевым кодом возврата.

Ограничения:

- Матрица:
  - **Размерность:** целое положительное число **от 1 до 2000**
  - **Данные:** целые числа **от -65536 до 65535**
- Пункт меню: целое число **от 1 до 4**

2. Описание выходных данных

Опции 1-3:

В результате выполнения программы будет сформирована результирующая матрица, имеющая представление в виде обычной и разреженной матрицы.

Опция 4:

В результате выполнения программы будет сформирована таблица сравнения времени выполнения операций и объема памяти при использовании этих 2-х алгоритмов при различном проценте заполнения и размерах матриц.

3. Описание задачи, реализуемой в программе

Программа реализует:

- умножение матрицы на столбец, представленных в обычном и разреженном виде
- измерение времени на умножение матриц разными способами.

4. Способ обращения к программе

Обращение к программе происходит через консоль, путём запуска *.exe файла.

5. Описание возможных аварийных ситуаций и ошибок пользователя

Аварийная ситуация	Код завершения	Сообщение
Неверно выбранный пункт меню	1	`Wrong choice!`
Нет входного файла	2	`No input file :(`
Входной файл пустой	3	`Empty input file!`
Некорректная размерность	4	`Incorrect dimensions :(`
Не удалось выделить память	5	`Memory error :(`
Некорректно введенные данные	6	`Incorrect data :(`

Описание внутренних структур данных

- Структура, хранящая матрицу в «обычном» виде

```
typedef struct
{
    size_t rows;    // кол-во рядов
    size_t columns; // кол-во столбцов
    int *pointer;   // указатель на массив значений
} ordinary_mtr_t;
```

- Структура, хранящая матрицу в «разреженном» виде

```
typedef struct
{
    int *A;    // массив ненулевых значений
    int *JA;   // массив столбцов ненулевых значений
    int *IA;   // массив кол-ва ненулевых эл-ов

    size_t elems;  // кол-во ненулевых элементов
    size_t rows;   // кол-во строк + 1
} sparse_mtr_t;
```

- Указатель на функцию, которую мы выбираем в зависимости от пункта меню

```
typedef int (*func_ptr)(FILE *, ordinary_mtr_t *);
```

Тестовые данные

1. Позитивные тесты

1. Ручной ввод: размерности 1*1, 1*1(единичные матрицы); в матрице корректные данные
2. Ручной ввод: размерности 1*10, 10*1(строка на столбец); в матрице корректные данные
3. Ручной ввод: размерности 3*4, 4*1(матрица на столбец); в матрице корректные данные
4. Файловый ввод: размерности 1*1, 1*1(единичные матрицы); в матрице корректные данные
5. Файловый ввод: размерности 1*10, 10*1(строка на столбец); в матрице корректные данные
6. Файловый ввод: размерности 3*4, 4*1(матрица на столбец); в матрице корректные данные
7. Случайный ввод: размерности 1*1, 1*1(единичные матрицы); в матрице корректные данные
8. Случайный ввод: размерности 1*10, 10*1(строка на столбец); в матрице корректные данные
9. Случайный ввод: размерности 3*4, 4*1(матрица на столбец); в матрице корректные данные

2. Негативные тесты

1. Исходный файл не задан
2. Исходный файл пустой
3. Некорректный размер матрицы
4. Кол-во элементов не соответствует размерности матрицы
5. Некорректные данные в матрице
6. Некорректные размеры для умножения матриц

Оценка эффективности

Были проведены тесты с матрицами размерностями:

- 10*10, 10*1
- 50*50, 50*1
- 100*100, 100*1
- 500*500, 500*1

И процентными заполнениями: 5, 10, 15, 25, 30, 35, 45, 50, 55

Количество замеров: 10

Эффективность по времени:

При низком проценте заполняемости (5-10%) и размерности матрицы (< 500*500) умножение разреженных матриц быстрее «обычного» умножения. При 15% заполняемости можно заметить, что обработке матрицы размером 50*50 время выполнения алгоритмов практически совпадает. Поэтому при дальнейшем увеличении размерностей и процента заполняемости выгоднее использовать стандартное умножение матрицы

Эффективность по памяти:

Не сложно заметит, что при заполняемости меньше ~48% разреженная матрица выигрывает по памяти в 7-8 раз. При увеличении количества нулевых элементов умножение «обычной» матрицы оказывается намного выгоднее.

```
*Measurements of time in microseconds
Measurements of memory in bytes

Option: 4
```

FILL%	SIZE	ORDINARY MATRIX		SPARES MATRIX	
		TIME	MEMORY	TIME	MEMORY
5	10	0	480	0	136
	50	14	10400	4	1848
	100	58	40800	22	6056
	500	1803	1004000	2121	124856
10	10	0	480	0	248
	50	16	10400	13	2824
	100	64	40800	61	10136
	500	1744	1004000	7770	225824
15	10	0	480	1	240
	50	15	10400	16	4248
	100	75	40800	141	14120
	500	1749	1004000	14459	325504
25	10	1	480	1	296
	50	15	10400	40	6112
	100	69	40800	295	22272
	500	1637	1004000	39509	521176
30	10	0	480	1	352
	50	15	10400	91	7088
	100	62	40800	463	26472
	500	1639	1004000	52445	621704
35	10	0	480	2	496
	50	17	10400	112	8000
	100	105	40800	660	30432
	500	1604	1004000	69447	719936
45	10	0	480	2	560
	50	30	10400	194	10240
	100	71	40800	1102	38488
	500	1599	1004000	114156	919080
50	10	0	480	2	680
	50	15	10400	156	11104
	100	63	40800	1198	41952
	500	1510	1004000	138887	1017208
55	10	2	480	5	680
	50	19	10400	215	12200
	100	62	40800	1395	46984
	500	1503	1004000	170854	1116320

Выводы

В ходе выполнения данной лабораторной работы я ознакомилась со способами хранения матриц в компьютере.

Оказалось, что хранение разреженных матриц лучше реализовывать в разреженном представлении если плотность матрицы меньше 15%. Быстродействие для такой плотности увеличиться и количество потребляемой памяти снизится.

На отрезке от ~15% до ~45% программист сам должен выбрать способ хранения, так как время выполнения в данном случае у разреженных матриц хуже, но затраты по памяти меньше. В случае с плотностью матрицы более 50% - лучше себя показал обычный способ хранения матрицы.

*Ответы на контрольные вопросы

1) *Что такое разреженная матрица, какие схемы хранения таких матриц вы знаете?*

Разреженная матрица – матрица, большая часть которой заполнена нулями.

Можно хранить схемой Кнута, кольцевой схемой и в разряженном строчном(столбовом) формате.

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

Под обычную матрицу выделяется **n*m*sizeof(element)** байт памяти.

Требуемая память под хранение разреженной матрицы зависит от выбранного типа хранения и количества ненулевых элементов.

3) *Каков принцип обработки разреженной матрицы?*

Обрабатываются только ненулевые элементы, что позволяет сократить время обработки.

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

Разряженная матрица эффективна, только когда в массиве много нулевых элементов. Если их незначительно количество, то целесообразнее использовать стандартные алгоритмы.