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 «ОБРАБОТКА РАЗРЕЖЕННЫХ МАТРИЦ»

Студент Тузов Даниил Александрович

Группа ИУ7 – 32Б

Преподаватель Барышникова Марина Юрьевна
Силантьева Александра Васильевна

2023 г.

Оглавление

ОПИСАНИЕ УСЛОВИЯ ЗАДАЧИ

Реализация алгоритмов обработки разреженных матриц,
сравнение эффективности применения этих алгоритмов со стандартными алгоритмами обработки матриц при различном размере матриц и степени их разреженности.
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;
- вектор 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.

ОПИСАНИЕ ТЕХНИЧЕСКОГО ЗАДАНИЯ

Входные данные:
Сначала пользователь выбирает как вводить матрицу: координатно или полностью. Затем пользователю предлагается выбрать файл, из которого считывать матрицу. Если пользователь хочет ввести матрицу сам, то для этого ему необходимо ввести в консоль «stdin»

Выходные данные:
В консоль выводится время сложения матриц и время сложения векторов, а также сама матрица и сам вектор

Описание программы:
Сложение двух матриц различными способами

Способ обращения к программе:
Взаимодействие через консоль.

Аварийные ситуации:

1. Любой некорректный ввод данных в программе
   Код ошибки — 1
2. Ошибка при работе с файлом
   Код ошибки — 2
3. Ошибка сложения матриц разных размеров
   Код ошибки — 3
4. Ошибка переполнения матриц (матрицы размером до 5000)
   Код ошибки — 4
5. Ошибка работы с памятью
   Код ошибки — 5

ОПИСАНИЕ СТРУКТУР ДАННЫХ

Обычная матрица представлена в виде двумерного массива

```
int **mat_a
```

Структурный тип для вектора
num_row – количество строк в матрице
num_col – количество столбцов в матрице
fnum_elems – количество ненулевых элементов в матрице
A – вектор содержит значения ненулевых элементов
JA – вектор содержит номера столбцов для элементов вектора A
IA – вектор, в элементе Nk которого находится номер компонент

```
typedef struct matrix_vec {
size_t num_row;
size_t num_col;
size_t num_elems;
int *A;
int *JA;
int *IA;
} matrix_vec_t;
```

ОПИСАНИЕ АЛГОРИТМА

1. Пользователю предлагается ввести матрицу одним из двух способов: ввести полную матрицу или ввести матрицу координатно
2. В зависимости от выбора пользователя выбирается соответствующая функция для чтения матрицы
3. По полученной матрице строится вектор
4. Аналогично вводится вторая матрица и составляется второй вектор
5. Программа проверяет, что матрицы одного размера и их можно сложить
6. Сначала складываются матрицы
7. Потом складываются вектора
8. Программа выводит результат пользователю

ОПИСАНИЕ ФУНКЦИЙ

`int check_file(FILE **f, char *mods);` – функция проверяет файл f и открывает его с параметрами mods. Возвращает код ошибки

`void description();` – функция выводит описание работы программы

`int read_matrix(FILE **f, reader_t reader, int ***mat, size_t *n, size_t *m, matrix_vec_t **vec);` – функция читает из файла f функцией reader матрицу mat с рамерами n и m и дополнительно составляет по ней вектор vec. Возвращает код ошибки

`int input_vector(matrix_vec_t **mat_vec, int **data, size_t n, size_t m);` – функция составляет вектор mat_vec по матрице data с размерами n и m. Возвращает код ошибки

`void free_vector(matrix_vec_t *mat_vec);` – функция очищает память по указателю на вектор mat_vec

`matrix_vec_t* sum_mat_vectors(matrix_vec_t *mat_a, matrix_vec_t *mat_b);` – функция складывает два вектора mat_a и mat_b и возвращает их сумму

`void output_vec(FILE *f, matrix_vec_t *mat_vec);` – функция выводит в файл f матрицу по вектору mat_vec

```
void output_vec_coord(FILE *f, matrix_vec_t *mat_vec);
```
– функция выводит в файл f координаты ненулевых элементов в матрице по вектору mat_vec

```
int input_matrix(FILE *f, int ***data, size_t *n, size_t *m);
```
– функция читает из файла f матрицу data с размерами n и m. Возвращает код ошибки

```
int input_matrix_coord(FILE *f, int ***data, size_t *n, size_t *m);
```
– функция читает из файла f матрицу data, заданную координатно, с размерами n и m. Возвращает код ошибки

```
void free_matrix(int **data, size_t n);
```
– функция очищает память по указателю на матрицу data с количеством строк n

```
int** allocate_matrix(size_t n, size_t m);
```
– функция выделяет динамически память под матрицу с размерами n и m и возвращает укаазатель на матрицу

```
int** sum_matrix(int **mat_a, int **mat_b, size_t n, size_t m);
```
— функция скадывает две матрицы mat_a и mat_b с размерами n и m и возвращает новую матрицу

```
void print_matrix(FILE *f, int **data, size_t n, size_t m);
```
– функция печатает матрицу data с размерами n и m в файл f

```
void print_matrix_coord(FILE *f, int **data, size_t n, size_t m);
```
– функция печатает координаты ненулевых элементов в матрице data с размерами n и m в файл f

НАБОР ТЕСТОВ

№	Описание теста	Ввод пользователя	Ответ программы
1	Выбор способа ввода матрицы	1	Ввод полной матрицы
2	Выбор способа ввода матрицы	2	Матрица вводится координатно
3	Выбор способа ввода матрицы	3	Ошибка при выборе способа ввода матрицы
4	Выбор файла для считывания матрицы	stdin	Предложение ввести матрицу
5	Выбор файла для считывания матрицы	tests/1000_50.txt	Введена матрица с размерами 1000*1000 и процентом заполненности 50%
6	Выбор файла для считывания матрицы	Любой некорректный файл	Ошибка файла
7	Выбор файла для считывания матрицы	Файл не соответствует структуре выбранного способа ввода	Ошибка ввода
8	Ввод полной матрицы	2 2 1 2 3 4	Введена матрица 1 2 3 4
9	Ввод полной матрицы	2 x 1 2	Ошибка ввода
10	Ввод полной матрицы	2 2 a 1	Ошибка ввода
11	Сложение матриц	A: 2 2 1 1 1 1 B: 2 2	C: 1 1 1 1

		0 0 0 0	
12	Сложение матриц	A: 2 1 1 1 B: 2 2 0 0 0 0	Ошибка. Матрицы разного размера
13	Ввод матрицы координатно	2 2 2 1 1 1 2 2 2	Введена матрица 1 0 0 2
14	Ввод матрицы координатно	2 2 2 3 3 1 1 1 4	Ошибка ввода. Координаты не соответствуют размеру матрицы
15	Сложение матриц, заданных координатно	A: 2 2 2 1 1 1 2 2 2 B: 2 2 1 1 2 1	C: 1 1 0 2
16	Сложение матриц, заданных координатно	A: 2 3 2 1 1 1 2 2 2 B: 2 2 1 1 2 1	Ошибка. Матрицы разного размера

ЗАМЕРЫ ВРЕМЕНИ РАБОТЫ

В ходе эксперимента для получения результатов проводилось 100 замеров по времени. Усредненный результат представлен в таблице

Процент заполнения	Размер							
	50*50		100*100		500*500		1000*1000	
	M	B	M	B	M	B	M	B
5	0.048	0.005	0.082	0.015	1.526	0.367	6.108	1.475
10	0.027	0.008	0.215	0.083	1.543	0.731	6.155	2.961
15	0.025	0.012	0.065	0.045	1.522	1.104	6.113	4.376
20	0.024	0.015	0.067	0.060	1.550	1.465	6.116	5.817
25	0.021	0.019	0.067	0.074	1.514	1.829	6.123	7.278
30	0.040	0.023	0.215	0.095	1.547	2.208	6.105	8.736
35	0.042	0.027	0.066	0.105	1.513	2.552	6.325	10.782
40	0.045	0.031	0.066	0.118	1.515	2.918	6.114	11.665
45	0.040	0.034	0.072	0.154	1.520	3.270	6.079	13.097
50	0.023	0.039	0.066	0.152	1.524	3.640	6.088	14.535
75	0.044	0.056	0.066	0.220	1.521	5.456	6.114	21.836
100	0.045	0.075	0.066	0.295	1.542	7.283	6.094	29.136

*Значения времени в таблице представлены в миллисекундах

** M – матрица, B – вектор

ОЦЕНКА ЭФФЕКТИВНОСТИ ВРЕМЕНИ ВЫПОЛНЕНИЯ

Процент заполнения	Размер			
	50*50	100*100	500*500	1000*1000
5	9.60	5.46	4.16	4.14
10	3.38	2.59	2.11	2.08
15	2.08	1.44	1.38	1.40
20	1.60	1.12	1.04	1.05
25	1.11	0.91	0.83	0.84
30	1.74	2.26	0.70	0.70
35	1.56	0.63	0.59	0.59
40	1.45	0.56	0.52	0.52
45	1.18	0.47	0.46	0.46
50	0.59	0.43	0.42	0.42
75	0.79	0.30	0.28	0.28
100	0.60	0.22	0.21	0.21

*Числа в таблице определяются отношением времени выполнения сложения матриц к времени выполнения сложения векторов

ОБЪЕМ ИСПОЛЬЗУЕМОЙ ПАМЯТИ

Для процента заполненности 20%

Размер	Матрица	Вектор
50*50	1E4	0.84E4
100*100	4E4	3.28E4
500*500	1E6	0.804E6
1000*1000	4E6	3.208E6

*Значения в таблице представлены в байтах

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

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

Разреженная матрица содержит большое количество нулей. Схемы хранения: методом Кнута, двумерным массивом и схема, предложенная Чангом и Густавсоном.

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

При хранении обычной матрицы целых чисел выделяется N * M * 4 байта памяти. При хранении разреженной матрицы с процентом заполнения X выделяется N * M / 100 * X * 4 * 2 + N * 4 байта памяти.

3. Каков принцип обработки разреженной матрицы?

Разреженная матрица представляется в виде трех векторов: вектор ненулевых элементов, вектор столбцов этих элементов и вектор индексов элементов в первых двух массивах, с которых начинается новая строка. При обработке матрицы оперируют этими тремя векторами

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

Стандартные алгоритмы обработки эффективнее применять при обработке матриц с процентов разреженности больше 20. В таком случае время обработки стандартным способом быстрее из-за меньшего количества операций

ВЫВОД

В рамках этой лабораторной работы я научился работать с разреженными матрицами, представленными методом Кнута. Я оценил эффективность этого метода на примере сложения матриц. Сравнив время сложения матриц стандартным способом и с помощью векторов, я пришел к выводу, что:

1. Сложение с помощью вектором эффективнее по памяти при размерах матрицы вплоть до 50%
2. Сложение с помощью векторов эффективнее по времени вплоть до 20% заполнения
3. Чем меньше размер матрицы, тем эффективнее сложение с помощью векторов
    - Для размера матрицы 50*50 сложение с помощью векторов эффективнее вплоть до 45%
    - Для размера матрицы 100*100 сложение с помощью векторов эффективнее вплоть до 20%
    - Для размера матрицы 500*500 сложение с помощью векторов эффективнее вплоть до 20%
    - Для размера матрицы 1000*1000 сложение с помощью векторов эффективнее вплоть до 20%