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»**

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3 ПО ДИСЦИПЛИНЕ: ТИПЫ И СТРУКТУРЫ ДАННЫХ

Обработка разреженных матриц

Студент _____ Палладий Е.И.

Группа _____ ИУ7-31Б

Название предприятия: НУК ИУ МГТУ им. Н. Э. Баумана

Студент	_____ Палладий Е.И.
Преподаватель	_____ Барышникова М.Ю.

2024 г.

СОДЕРЖАНИЕ

1 Описание условия задачи

Разреженная (содержащая много нулей) матрица хранится в форме 3-х объектов (CSC):

- Вектор A содержит значения ненулевых элементов;
- Вектор IA содержит номера строк для элементов вектора A;
- Вектор JA, в элементе JA_k которого находится номер компонент в A и IA, с которых начинается описание столбца JA_k 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.

2 Техническое задание

2.1 Исходные данные

Матрицы, размер которых ограничен объёмом памяти, доступной на устройстве. Ввод матриц доступен несколькими способами:

1. Из файла
2. Случайным образом
3. Из консоли

Программа выполняет сложение матриц и выводит результат в консоль или файл, на выбор пользователя.

Меню программы:

1. Вывести информацию о программе
2. Считать матрицы
3. Вывести матрицы в консоль
4. Вывести результат сложения матриц в консоль
5. Вывести вектора (A, IA, JA) матриц в консоль
6. Сложить две матрицы обычным алгоритмом
7. Сложить две матрицы усовершенствованным алгоритмом
8. Вывести список стран по списку ключей
9. Записать сумму матриц в файл
10. Произвести исследование
11. Выход

2.2 Описание задачи, реализуемой программой

Чтение, хранение, сложение матриц разными способами. Проведение замерного эксперимента и исследования об эффективности хранения и работы алгоритма сложения разреженных матриц.

2.3 Способ обращения к программе

Программа запускается через терминал: `./app.exe`. Затем необходимо выбрать одну из предложенных опций в меню. Чтобы провести исследование,

в терминале следует написать: `make research`.

2.4 Описание аварийных случаев

1. **Аварийные случаи при выборе опции в меню**
   1.1. Неверный диапазон значений опции меню
   1.2. Попытка вывода матриц в консоль, когда они еще не введены
   1.3. Попытка вывода векторов в консоль, когда они еще не введены
   1.4. Попытка сложение матриц, когда они еще не введены
   1.5. Попытка вывода/записи результирующей матрицы, когда не было сложения
2. **Аварийные случаи считывании матрицы**
   2.1. Считывание матрицы из файла
      - Отсутствие файла для чтения
      - Отсутствие прав доступа на чтения файла

   2.2. Считывание матрицы из консоли
      - Количество строк меньше нуля
      - Количество столбцов меньше нуля
      - Количество ненулевых элементов меньше 0
      - Количество ненулевых элементов больше кол-ва строк $*$ кол-во столбцов

   2.3. Заполнение матрицы случайным образом
      - Количество строк меньше нуля
      - Количество столбцов меньше нуля
      - Процент заполнения меньше 0 или больше 100
3. **Аварийные случаи при сложении матриц**
   - Матрицы разной размерности
4. **Аварийные случаи при записи матрицы в файл**
   - Отсутствие прав доступа на запись в файл
5. **Общий аварийный случай при работе с матрицами:**
   - Отсутствие памяти под матрицу на устройстве

3 Описание структуры данных

```
typedef struct
{
    int *A;
    size_t len_A;

    size_t *IA;
    size_t rows;

    size_t *JA;
    size_t columns;
} csc_matrix_t;
```

Листинг 1: Структура разреженной матрицы

Объяснение полей:

Поле	**Описание**
`A`	Вектор, содержащий значения ненулевых элементов матрицы
`len_A`	Количество элементов
`IA`	Вектор, содержащий номера строк для элементов вектора A
`rows`	Количество строк в матрице
`JA`	Вектор, содержащий индексы элементов вектора A, с которого начинается столбец JA_i
`columns`	Количество столбцов в матрице

4 Описание алгоритма

1. Выбран пункт 1) ”Вывести информацию о программе”:
    - Информация о программе выводится на экран
2. Выбран пункт 2) ”Считать матрицы”:
    2.1. Выбран пункт ”Из файла”
        - Считывается название файла
        - Считываются размеры матрицы из файла
        - Выделяется память и считываются ненулевые элементы
        - Заполняется матрица
    2.2. Выбран пункт ”Случайным образом”
        - Считываются размеры матрицы
        - Считывается процент заполнения ненулевыми элементами
        - Выделяется память и заполняется матрица
    2.3. Выбран пункт ”Из консоли
        - Считываются размеры матрицы
        - Считывается количество ненулевых элементов
        - Выделяется память и заполняется матрица
3. Выбран пункт 3) ”Вывести матрицы в консоль”:
    - Выводится первая матрица в обычном виде
    - Выводится вторая матрица в обычном виде
4. Выбран пункт 4) ”Вывести результат сложения матриц в консоль”:
    - Вводится матрица в обычном виде
5. Выбран пункт 5) ”Вывести вектора (A, IA, JA) матриц в консоль”:
    - Выводятся вектора для первой матрицы
    - Выводятся вектора для второй матрицы
    - Если до этого было сложение разреженных матриц, то выводятся вектора для суммы матриц
6. Выбран пункт 6) ”Сложить две матрицы стандартным алгоритмом ”:

- Две матрицы складываются стандартным алгоритмом

7. Выбран пункт 7) ”Сложить две матрицы усовершенствованным алгоритмом”:
    - Две матрицы складываются усовершенствованным алгоритмом
8. Выбран пункт 8) ”Записать сумму матриц в файл”:
    - Считывается название файла
    - Размеры матрицы и сама матрица записывается в файл

4.1 Основные функции программы

1. `errors_e sum_matrix_standart(matrix_t *summ, csc_matrix_t *pleft, csc_matrix_t *pright);`
    - Складывает две матрицы в обычном представлении
    - **summ**: Результат суммы
    - ***pleft**: Указатель на первую матрицу
    - ***pright**: Указатель на вторую матрицу
    - Возвращает код ошибки
2. `errors_e sum_matrix_fast(csc_matrix_t *summ, csc_matrix_t *pleft, csc_matrix_t *pright);`
    - Складывает две матрицы в разреженном представлении (CSC)
    - **summ**: Результат суммы
    - ***pleft**: Указатель на первую матрицу
    - ***pright**: Указатель на вторую матрицу
    - Возвращает код ошибки
3. `void print_matrix(csc_matrix_t *matrix, matrix_t *default_matrix, FILE *f);`
    - Выводит матрицу в обычном представлении в файл
    - ***matrix**: Структура матрицы в разреженном представлении (CSC)
    - ***default_matrix**: Матрица в обычном представлении

- **f**: Файловая переменная

4. `void print_vectors(csc_matrix_t *matrix);`
    - Выводит вектора матрицы разреженном представлении (CSC)
    - ***matrix**: Структура матрицы
5. `errors_e read_matrix(csc_matrix_t *matrix);`
    - Чтение матрицы в разреженное представление (CSC)
    - ***matrix**: Структура матрица

5 Тестовые данные

Позитивные тесты

Тест	Выходные данные
Чтение матрицы из файла	Матрица успешно считана!
Чтение матрицы из консоли	Матрица успешно считана!
Заполнение матрицы случайным образом	Матрица успешно заполнена!
Вывести матрицы в консоль	Матрицы успешно выведены!
Вывести результат сложения матриц в консоль	Сумма матриц успешно выведена!
Вывести вектора (A, IA, JA) матриц в консоль	Вектора успешно выведены!
Сложить две матрицы стандартным алгоритмом	Матрицы успешно сложены!
Сложить две матрицы усовершенствованным алгоритмом	Матрицы успешно сложены!
Записать сумму матриц в файл	Матрицы успешно записана в файл!
Выход	Пока!

Негативные тесты

Тест	Выходные данные
Неверный диапазон значений опции меню	ERROR WITH CHOOSING ACT
Вывод матриц в консоль, когда они не введены	Две матрицы не введены!

Вывод векторов в консоль, когда матрицы не введены	Две матрицы не введены!
Сложение матриц, когда они не введены	Две матрицы не введены!
Запись суммы матриц, когда не было сложения	Вы до этого не выполняли сложения матриц, чтобы выводить их)!
Отсутствие файла для чтения матрицы	ERR WITH FILE
Некорректные данные в файле	ERR WITH INPUT/OUTPUT
Количество строк или столбцов меньше нуля	ERR WITH RANGE
Количество строк или столбцов меньше или больше допустимой размерности	ERR WITH RANGE
Количество ненулевых элементов меньше нуля	ERR WITH RANGE
Количество ненулевых элементов больше количества всех элементов матрицы	ERR WITH RANGE
Процент заполнения матрицы < 0 или $> 100\%$	ERR WITH RANGE
Сложение матриц с разной размерностью	ERR RANGE
Некорректные данные в файле	ERR WITH INPUT/OUTPUT

Отсутствие прав на запись в файл	ERR WITH INPUT/OUTPUT
Отсутствие памяти под матрицу на устройстве	ERR MEMORY

6 Исследование

Исследование проводилось на основе матриц, размерностью от 50×50 до 600×600 с шагом 25, 50 и 100. Количество итераций сложения: 5. Все данные были сгенрированы случайным образом. В ходе эксперимента применялись два алгоритма сложения: классический, с обычными матрицами и улучшенный, с использованием разреженных матриц. Производились замеры как по потраченному времени, так и по использованной памяти

Данные для 1% заполнения матриц

Таблица 1. Время сложения матриц

Размер матрицы	Обычные матрицы, мкс	CSC матрицы, мкс
50×50	12	0
75×75	27	1
100×100	24	0
150×150	54	1
200×200	99	3
300×300	230	7
400×400	396	19
500×500	609	27
600×600	877	43

Таблица 2. Объём памяти для хранения матриц

Размер матрицы	Обычные матрицы, байт	CSC матрицы, байт
50×50	10000	400
75×75	22500	748
100×100	40000	1200
150×150	90000	2700
200×200	160000	4800
300×300	360000	10800
400×400	640000	19200
500×500	1000000	30000
600×600	1440000	43200

Данные для 3% заполнения матриц

Таблица 3. Время сложения матриц

Размер матрицы	Обычные матрицы, мкс	CSC матрицы, мкс
50×50	5	0
75×75	13	1
100×100	24	2
150×150	54	5
200×200	98	14
300×300	220	29
400×400	389	56
500×500	607	110
600×600	872	178

Таблица 4. Объём памяти для хранения матриц

Размер матрицы	Обычные матрицы, байт	CSC матрицы, байт
50×50	10000	900
75×75	22500	2016
100×100	40000	3600
150×150	90000	8100
200×200	160000	14400
300×300	360000	32400
400×400	640000	57600
500×500	1000000	90000
600×600	1440000	129600

Данные для 5% заполнения матриц

Таблица 5. Время сложения матриц

Размер матрицы	Обычные матрицы, мкс	CSC матрицы, мкс
50×50	5	1
75×75	13	2
100×100	24	4
150×150	54	10
200×200	96	21
300×300	224	54
400×400	393	89
500×500	554	181
600×600	783	289

Таблица 6. Объём памяти для хранения матриц

Размер матрицы	Обычные матрицы, байт	CSC матрицы, байт
50×50	10000	1500
75×75	22500	3372
100×100	40000	6000
150×150	90000	13500
200×200	160000	24000
300×300	360000	54000
400×400	640000	96000
500×500	1000000	150000
600×600	1440000	216000

Данные для 7% заполнения матриц

Таблица 7. Время сложения матриц

Размер матрицы	Обычные матрицы, мкс	CSC матрицы, мкс
50×50	5	1
75×75	12	2
100×100	23	4
150×150	56	12
200×200	96	20
300×300	197	71
400×400	347	150
500×500	547	251
600×600	766	404

Таблица 8. Объём памяти для хранения матриц

Размер матрицы	Обычные матрицы, байт	CSC матрицы, байт
50×50	10000	2100
75×75	22500	4716
100×100	40000	8400
150×150	90000	18900
200×200	160000	33600
300×300	360000	75600
400×400	640000	134400
500×500	1000000	210000
600×600	1440000	302400

Данные для 10% заполнения матриц

Таблица 9. Время сложения матриц

Размер матрицы	Обычные матрицы, мкс	CSC матрицы, мкс
50×50	5	1
75×75	12	3
100×100	22	9
150×150	55	20
200×200	97	34
300×300	200	102
400×400	370	175
500×500	542	387
600×600	770	573

Таблица 10. Объём памяти для хранения матриц

Размер матрицы	Обычные матрицы, байт	CSC матрицы, байт
50×50	10000	3000
75×75	22500	6744
100×100	40000	12000
150×150	90000	27000
200×200	160000	48000
300×300	360000	108000
400×400	640000	192000
500×500	1000000	300000
600×600	1440000	432000

Данные для 12% заполнения матриц

Таблица 11. Время сложения матриц

Размер матрицы	Обычные матрицы, мкс	CSC матрицы, мкс
50×50	5	2
75×75	14	4
100×100	22	9
150×150	54	22
200×200	85	43
300×300	196	125
400×400	344	209
500×500	540	470
600×600	779	690

Таблица 12. Объём памяти для хранения матриц

Размер матрицы	Обычные матрицы, байт	CSC матрицы, байт
50×50	10000	3600
75×75	22500	8100
100×100	40000	14400
150×150	90000	32400
200×200	160000	57600
300×300	360000	129600
400×400	640000	230400
500×500	1000000	360000
600×600	1440000	518400

Данные для 15% заполнения матриц

Таблица 13. Время сложения матриц

Размер матрицы	Обычные матрицы, мкс	CSC матрицы, мкс
50×50	5	2
75×75	12	6
100×100	24	11
150×150	54	26
200×200	87	49
300×300	194	147
400×400	393	261
500×500	554	560
600×600	783	881

Таблица 14. Объём памяти для хранения матриц

Размер матрицы	Обычные матрицы, байт	CSC матрицы, байт
50×50	10000	4500
75×75	22500	10116
100×100	40000	18000
150×150	90000	40500
200×200	160000	72000
300×300	360000	162000
400×400	640000	288000
500×500	1000000	450000
600×600	1440000	648000

Данные для 20% заполнения матриц

Таблица 15. Время сложения матриц

Размер матрицы	Обычные матрицы, мкс	CSC матрицы, мкс
50×50	5	4
75×75	12	9
100×100	23	20
150×150	56	35
200×200	96	71
300×300	197	190
400×400	347	355
500×500	547	791
600×600	766	1205

Таблица 16. Объём памяти для хранения матриц

Размер матрицы	Обычные матрицы, байт	CSC матрицы, байт
50×50	10000	6000
75×75	22500	13500
100×100	40000	24000
150×150	90000	54000
200×200	160000	96000
300×300	360000	216000
400×400	640000	384000
500×500	1000000	600000
600×600	1440000	864000

Данные для 25% заполнения матриц

Таблица 17. Время сложения матриц

Размер матрицы	Обычные матрицы, мкс	CSC матрицы, мкс
50×50	5	5
75×75	12	10
100×100	23	20
150×150	56	50
200×200	96	104
300×300	197	285
400×400	347	458
500×500	547	971
600×600	766	1381

Таблица 18. Объём памяти для хранения матриц

Размер матрицы	Обычные матрицы, байт	CSC матрицы, байт
50×50	10000	7500
75×75	22500	16872
100×100	40000	30000
150×150	90000	67500
200×200	160000	120000
300×300	360000	270000
400×400	640000	480000
500×500	1000000	750000
600×600	1440000	1080000

Данные для 35% заполнения матриц

Таблица 19. Время сложения матриц

Размер матрицы	Обычные матрицы, мкс	CSC матрицы, мкс
50×50	5	5
75×75	12	14
100×100	23	24
150×150	56	71
200×200	96	144
300×300	197	305
400×400	347	574
500×500	547	1417
600×600	766	1933

Таблица 20. Объём памяти для хранения матриц

Размер матрицы	Обычные матрицы, байт	CSC матрицы, байт
50×50	10000	10500
75×75	22500	23616
100×100	40000	42000
150×150	90000	94488
200×200	160000	168000
300×300	360000	377988
400×400	640000	672000
500×500	1000000	1050000
600×600	1440000	1511988

6.1 Выводы исследования

Как видно по полученным данным, использование алгоритма сложения разреженных матриц эффективно, когда содержание ненулевых элементов не

больше 25% от общего числа элементов в матрице. Хранение разреженной матрицы эффективнее вплоть до 35% содержания ненулевых элементов в матрице

7 Ответы на вопросы

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

   Разреженные матрицы — это матрицы, в которых большинство элементов являются нулями. Хранение таких матриц в традиционном виде, где каждый элемент занимает место в памяти, нерационально, так как это приводит к избыточному использованию памяти для хранения большого количества нулевых значений. Для повышения эффективности используются специализированные способы хранения, такие как:

   - **COO (Coordinate List)**: хранение ненулевых элементов в виде списка их значений с указанием соответствующих координат (строка, столбец, значение).
   - **CSR (Compressed Sparse Row)**:хранение только ненулевых элементов построчно с добавлением массива индексов для указания начала каждой строки.
   - **CSC (Compressed Sparse Column)**: аналогичная схема CSR, но используется для столбцов вместо строк.

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

   В обычной матрице каждый элемент хранится в памяти компьютера, поэтому общий размер матрицы составляет:

   $$matrix\_size = rows \times columns \times \text{sizeof}(T) \tag{1}$$

   Где:

   - $rows$ — количество строк
   - $columns$ — количество столбцов
   - T — тип данных

В разреженной матрице хранятся только ненулевые значения и их индексы. Размер такой матрицы складывается из размера трёх векторов. Размер вектора A равен произведению количества ненулевых элементов на их тип:

$$A.size = nonzero \times \text{sizeof}(T_1) \tag{2}$$

Где:

- $nonzero$ — количество ненулевых элементов.
- T_1 — тип данных

Размер вектора IA также равен произведению количества ненулевых элементов на их тип:

$$IA.size = nonzero \times \text{sizeof}(T_2) \tag{3}$$

А размер вектора JA равен произведению количества столбцов на тип:

$$JA.size = columns \times \text{sizeof}(T_3) \tag{4}$$

Тогда, без учёта выравнивания, весь размер будет состоять из сумм размеров векторов (2), (3) и (4):

$$csc\_matrix\_size = A.size + IA.size + JA.size \tag{5}$$

3. **Каков принцип обработки разреженной матрицы** Принцип заключается в том, что происходит обработка только ненулевых элементов, так как нулевые не повлияют на результат. За счёт этого получается улучшить производительность программы
4. **В каком случае для матриц эффективнее применять стандартные алгоритмы обработки матриц? От чего это зависит?** Исходя из моего ис-

следования, стандартные алгоритмы эффективнее применять при количестве ненулевых элементов матриц более чем 30% от общего количества элементов. Это связано с более тяжелым доступом к элементам матрицы через ее разреженное представление

Заключение

На практике разреженные матрицы встречаются довольно часто. Простой пример: матрица смежности для представления графа. Пример из практики: схема метро. Если представить схему московского метрополитена как матрицу смежности, то можно будет заметить, что это очень разреженная матрица

Если в задаче часто требуется складывать матрицы и есть информация, что в основном эти матрицы разрежены $(0 - 25\%)$ ненулевых элементов от общего количества всех элементов, то выгодно использовать алгоритмы по работе с разреженными матрицами

Если в задаче не часто требуется складывать матрицы, но приходится хранить много матриц и есть информация, что в основном эти матрицы разрежены $(0 - 35\%)$ ненулевых элементов от общего количества всех элементов, то выгодно использовать алгоритмы хранения и обработки разреженных матриц