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 ПО ДИСЦИПЛИНЕ: ТИПЫ И СТРУКТУРЫ ДАННЫХ

Обработка разреженных матриц.

Студент **Батуев А.Г.**

Группа **ИУ7-36Б**

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

Студент _____ **Батуев А.Г.**

Преподаватель _____ **Никульшина Т.А.**

2024 г.

Условие задачи

Цель работы - реализовать алгоритмы обработки разреженных матриц, сравнить эффективность использования этих алгоритмов (по времени выполнения и по требуемой памяти) со стандартными алгоритмами обработки матриц при различном процентном заполнении матриц ненулевыми значениями и при различных размерах матриц.
Вариант 2: 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 Nk которого находится номер компонент в A и IA, с которых начинается описание столбца 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

Техническое задание

На вход **программа получает**:

- Номер действия (от пользователя)
- В зависимости от выбранного действия может запрашивать доп. информацию, необходимую для выполнения действия.

Программа должна реализовывать создание, хранение, вывод, а также операцию сложения матрицы, хранящейся в стандартном формате и формате CSC.

Обращение к программе осуществляется по указанию названия программы.

Необходимо внимательно **обработать возможные аварийные ситуации**, которые могут включать: неправильный ввод данных, неправильный результат сложения, неправильная работа с указателями.

Описание внутренних структур данных

Для удобства обращения с данными реализуется структура, состоящая из 5 полей, которая описывает матрицу формата CSC, где:

- values – ненулевые значения
- row_indices – индексы строк, в который хранятся ненулевые элементы
- nonZ_sum – кумулята ненулевых значений (под кумулятой понимается количественная сумма всех ненулевых элементов в предыдущих строках)

```
typedef struct
{
    int* values; // значения
    int* row_indices;  // номера строк
    int* nonZ_sum;  // Кумулянта не нулевых значений
    int rows;
    int cols;
} CSCMatrix;
```

Описание алгоритмов

Программа реализует следующие программные реализации:

- init_csc_matr(int rows, int cols, int nnz): инициализирует пустую матрицу в формате CSC.
- printCSCMatrix(CSCMatrix matrix): выводит матрицу в формате CSC.
- printCSCMatrix_stdView(CSCMatrix matrix): выводит матрицу в стандартном формате (если размерность <= 30x30).
- printStdMatrix(int** matrix, int rows, int cols): выводит матрицу в стандартном формате.
- show_menu(void): отображает меню пользовательского интерфейса.
- createRandomCSC(CSCMatrix **given_matrix, int rows, int cols, double percentage): создает случайную разреженную матрицу в формате CSC (Compressed Sparse Column).
- Функция createMatrixByCoordinates (CSCMatrix **given_matrix, int rows, int cols): создает матрицу по заданным координатам.
- Функция addCSCMatrices (CSCMatrix A, CSCMatrix B): складывает две матрицы в формате CSC.
- Функция create_std_matrix_from_csc (CSCMatrix* cscMatrix): Преобразует матрицу в формате CSC в стандартный двумерный массив.
- Функция addStdMatrix (int **matrix1, int **matrix2, int rows, int cols): Складывает две стандартные матрицы.

Набор тестов

Входные данные:

- опция из меню (цифра от 0 до 9)
- для создания случайной матрицы 3 числа: кол-во строк и столбцов, процент наполненности
- для создания матрицы по координатам: кол-во строк, столбцов, ненулевых элементов, затем для каждого ненулевого элемента координата и значение.

Выходные данные:

- вывод матрицы в стандартном виде
- вывод матрицы в формате CSC
- результат сложения матриц
- результат измерения времени сложения стандартным и CSC методом

Позитивные тесты:

Описание	Входные данные	Выходные данные
Создание матрицы случайными значениями	2 2 2 50	0 1 2 0
Создание матрицы случайными значениями	2 1 1 100	0 0 1 0
Создание матрицы с координатным вводом	3 3 3 3 1 1 5 0 0 10 2 2 15	10 0 0 0 5 0 0 0 15
Сложение двух разреженных матриц (CSC)	4	*Правильный результат сложения*
Сложение двух разреженных матриц (стандартным способом)	5	*Правильный результат сложения*

Негативные тесты:

Описание	Входные данные	Выходные данные
Неверный ввод размера матрицы	2 -4 5 50	*Сообщение об ошибке*
Неверный ввод процента заполненности	2 4 5 110	*Сообщение об ошибке*
Попытка сложения матриц разных размеров	2 4 5 50	*Сообщение об ошибке*

	1 3 5 25	
Ввод координаты за пределами размерами матрицы	3 2 2 1 3 4 10	*Сообщение об ошибке*

Аналитическая часть

Формат CSC (Compressed Sparse Column) используется для хранения разреженных матриц, где большинство элементов равны нулю. Этот формат эффективен как по времени, так и по памяти, особенно для операций с редкими данными. Давайте рассмотрим основные аспекты формата CSC и его структуру на примере представленного типа CSCMatrix.

Структура CSC:

- values: массив значений, содержащий все ненулевые элементы матрицы.
- row_indices: массив, который указывает индекс строки для каждого ненулевого элемента.
- nonZ_sum: кумулянт (или префиксная сумма) ненулевых элементов. Этот массив содержит индексы, с которых начинается каждая колонка матрицы в массиве values. Таким образом, для любой колонки можно легко найти все ненулевые элементы, находящиеся в ней.
- rows и cols: количество строк и столбцов в матрице соответственно.

Преимущества формата CSC:

- Экономия памяти: хранение только ненулевых элементов, что особенно выгодно при разреженных матрицах.
- Быстрая работа с колонками: операции по столбцам выполняются эффективно, так как все данные внутри колонки легко находятся по кумулянте.
- Оптимизация арифметических операций: благодаря компактному представлению можно ускорить выполнение операций, таких как сложение матриц, умножение и другие линейные вычисления.

Недостатки:

- Ограниченная применимость: при плотных матрицах использование CSC может не дать преимущества, а наоборот, стать излишне сложным и оттого невыгодным со всех сторон.

Однако все преимуществе предоставляемые форматом CSC пропадают с повышением количества ненулевых элементов.

Для хранения плотной матрицы используется двумерный массив фиксированного размера rows * cols, где каждый элемент матрицы занимает место, независимо от того, является ли он нулевым или ненулевым.

Формат CSC использует три массива определяющие положение и значение только ненулевого значения, оттого такой формат сильно превосходит стандартный, когда речь идет о разряженных матрицах.

Так предположим, что мы имеем матрицу 1000 на 1000 элементов при 0.1% заполненности, то есть мы имеем 1000 ненулевых элементов. Память, выделяемую для переменных, указывающих размерности матрицы, опустим.

Метод хранения	Формула для расчета занимаемой памяти	Количество занимаемой памяти в байтах
Стандартный	Rows * cols * sizeof(int)	4.000.000

CSC формат	Len(values) * sizeof(int) + len(row_indicies) * sizeof(int) + len(nonZ_sum) * sizeof(int)	12.004

Теперь предположим, что мы имеем матрицу 1000 на 1000 элементов при 80% заполненности, то есть 800 000 ненулевых элементов. Формулы для расчета остаются теми же.

Метод хранения	Количество занимаемой памяти в байтах
Стандартный	4.000.000
CSC формат	6 404 004

Теперь приведем результаты анализа сложения матриц, хранящихся в двух форматах, и проанализируем скорость выполнения операций на этими матрицами. Замеры происходили на матрице 1000 на 1000 элементов.

Способ хранения матрицы	Процент заполненности матрицы (%)	Фактическое количество элементов	Время сложения (с)	Отношение стандартного к CSC (%)
Стандартный	0.01	100	0.002584	58595
CSC			0.000004	
Стандартный	0.05	500	0.002608	32393
CSC			0.000008	
Стандартный	0.1	1000	0.002621	20624
CSC			0.000013	
Стандартный	1	10000	0.002602	2259
CSC			0.000115	
Стандартный	5	50000	0.002593	391
CSC			0.000662	
Стандартный	10	100000	0.002592	213
CSC			0.001215	
Стандартный	15	150000	0.002612	136
CSC			0.001912	
Стандартный	20	200000	0.002595	98
CSC			0.002637	
Стандартный	25	250000	0.002618	80
CSC			0.003255	

Все замеры происходили не ноутбуке с подключенным питанием и в режиме высокой производительности.
Характеристики ноутбука:

- ОС: Ubuntu Lts 22.04
- Оперативная память: 16 Гб DDR4

- Процессор: Intel core i5-12500H

Каждое измерение проводилось 100 раз для усреднения результирующего времени.

:

Контрольные вопросы

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

Разреженная матрица — это матрица, в которой большая часть элементов равна нулю, и только небольшой процент содержит ненулевые значения.
Основные схемы хранения разреженных матриц:

- Coordinate List
- CSR (Compressed Sparse Row)
- CSC (Compressed Sparse Column)
- Diagonal storage

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

Обычная (плотная) матрица: Память выделяется для хранения всех элементов матрицы, даже если большая их часть — это нули. Если матрица размера M на N, то количество памяти, затрачиваемое на хранение такой матрицы:

$$Memory = M * N * sizeof(type)$$

В случае с CSC потребуется только хранение ненулевых элементов в количестве nnz (предполагается что индексы и кумулянт хранятся в формате int):

$$Memory = nnz * sizeof(type) + nnz * sizeof(int) + (N + 1) * sizeof(int)$$

3. Каков принцип обработки разреженной матрицы?

Принцип работы с разреженными матрицами основан на операциях только с ненулевыми элементами. Это позволяет существенно снизить количество вычислений и затраты памяти.

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

Матрицы, хранящиеся в формате CSC, начинают проигрывать в скорости обработки и затратах памяти на хранение при увеличении процентного соотношения ненулевых элементов ко всем. Поэтому матрицы с высокой плотностью заполнения будет лучше обрабатывать в стандартном формате. В плотных структурах доступ к элементам происходит за O(1), тогда как в разреженных структурах доступ может потребовать поиска индексов, что может увеличивает время до O(N).

Заключение

CSC представление более эффективно по памяти при малом количестве ненулевых элементов (разреженных матрицах), так как хранит только ненулевые значения и индексы. При этом выигрыш в памяти будет значительным при заполнении менее 10-20%. Но CSC-формат оказывается невыгоден для матриц с высокой плотностью ненулевых элементов, так как потребляет больше памяти из-за хранения дополнительных структур.

При очень разреженных матрицах (например, до 0.010%), CSC-формат показал значительное преимущество по скорости — более чем на 20000% быстрее, чем стандартное представление матрицы. И продолжает сохранять преимущество по скорости выполнения вплоть до 20%, где происходит перелом в другую сторону.

Таким образом, CSC-формат оказывается невыгоден для матриц с высокой плотностью ненулевых элементов, так как потребляет больше памяти из-за хранения дополнительных структур и тратит больше времени на их обработку. Но, когда речь заходит о матрицах высокой разрежённости, то несомненное преимущество за форматом CSC.