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

Вариант 2

Студент **Абдуллаев Ш. В.**

Группа **ИУ7-34Б**

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

Студент _____ **Абдуллаев Ш. В.**

Преподаватель _____ **Силантьева А. В.**

2024

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 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.

Описание исходных данных

Исходные данные представляют собой матрицы с определенной степенью разреженности. Программа поддерживает несколько способов ввода матриц:

- Чтение данных из файла:

  Файл должен содержать размерность матрицы на первой строке. Далее перечисляются её элементы.
- Ввод данных вручную:

  Пользователь вводит количество строк и столбцов. Затем программа предлагает два способа ввода матрицы:
  - Полный ввод: Пользователь вводит значения всех элементов матрицы.
  - Координатный ввод: Пользователь вводит количество ненулевых элементов. Далее для каждого элемента через пробел указываются индекс строки, индекс столбца и само значение.

Имеются следующие ограничения:

1. Ограничение на формат файла: программа обрабатывает только корректные файлы заданного формата. Если файл не соответствует формату, корректная работа программы не гарантируется. При критических ошибках форматирования программа завершает работу с сообщением об ошибке.
2. Ограничение на размер матрицы: размеры матрицы должны быть положительными целыми числами, не большим 1000. Если размерность матрицы не соответствует допустимым значениям, программа завершает работу с ошибкой.
3. Ограничение на индексы для координатного ввода: индексы строк и столбцов должны быть положительными целыми числами и находиться в пределах размерности матрицы. Если пользователь вводит индекс, превышающий размерность матрицы, программа выведет сообщение об ошибке и завершит работу.
4. Ограничение на количество ненулевых элементов: при координатном вводе количество ненулевых элементов не может превышать общее количество элементов матрицы. В случае превышения программа завершает работу с ошибкой.

Дополнительные ошибки:

Несоответствие типа данных при вводе (например, ввод букв вместо чисел) приводит к завершению программы с сообщением о некорректных данных.

Допустимый ввод	**Недопустимый ввод**
Выберите количество строк матриц: 3	Выберите количество строк матриц: q
Выберите количество столбцов матриц: 4	Выберите количество столбцов матриц: -3
Введите количество ненулевых элементов: 3	Введите количество ненулевых элементов: 0
Введите через пробел индексы строки, столбца и значение: 0 0 1	Введите через пробел индексы строки, столбца и значение: -1 a 99

Таблица 1. Примеры ввода

Описание результатов

1. Выход из программы: пользователь может выйти из программы, завершив выполнение.
2. Инициализация двух матриц: пользователь может инициализировать две стандартные матрицы.
3. Вывод первой матрицы: программа позволяет вывести на экран первую стандартную матрицу.
4. Вывод характеристики первой разреженной матрицы: программа выводит информацию о характеристиках первой разреженной матрицы.
5. Вывод первой разреженной матрицы: пользователь может просмотреть первую разреженную матрицу в формате, удобном для анализа.
6. Вывод второй матрицы: вывод второй стандартной матрицы на экран.
7. Вывод характеристики второй разреженной матрицы: программа отображает характеристики второй разреженной матрицы.
8. Вывод второй разреженной матрицы: пользователь может вывести вторую разреженную матрицу для просмотра.
9. Сложение первых двух обычных матриц: программа производит операцию сложения первых двух стандартных матриц и выводит результат.
10. Сложение первых двух разреженных матриц: сложение первых двух разреженных матриц с выводом результата.
11. Сравнение времени выполнения суммирования для разреженных и стандартных матриц: программа выводит время, затраченное на сложение стандартных и разреженных матриц, для сравнения производительности.

Описание задачи, реализуемой программой

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.

Способ обращения к программе

Обращения к программе пользователем происходит с помощью вызова исполняемого файла (app.exe).

Описание возможных аварийных ситуаций и ошибок пользователя

1. Не передан датасет. Программа вернет NO_DATA_ERROR.
2. Неудачный ввод команды. Программа вернет INPUT_COMMAND_ERROR.
3. Ошибка при чтении выбора команды. Программа вернет READ_INPUT_CHOICE_ERROR.
4. Некорректный выбор команды. Программа вернет INVALID_INPUT_CHOICE_ERROR.
5. Ошибка открытия файла для чтения данных. Программа вернет OPEN_FILE_ERROR.
6. Ошибка выделения памяти. Программа вернет ALLOC_ERROR.
7. Ошибка чтения количества строк матрицы. Программа вернет READ_ROWS_MATRIX_ERROR.
8. Некорректное количество строк матрицы. Программа вернет INCORRECT_ROWS_MATRIX_ERROR.
9. Ошибка чтения количества столбцов матрицы. Программа вернет READ_COLS_MATRIX_ERROR.
10. Некорректное количество столбцов матрицы. Программа вернет INCORRECT_COLS_MATRIX_ERROR.
11. Ошибка чтения значения матрицы. Программа вернет READ_VALUE_BY_INPUT_ERROR.
12. Ошибка чтения количества ненулевых элементов матрицы. Программа вернет READ_NON_ZERO_CNT_ERROR.
13. Некорректное количество ненулевых элементов матрицы. Программа вернет INCORRECT_NON_ZERO_CNT_ERROR.
14. Ошибка чтения координат ненулевых элементов матрицы. Программа вернет READ_COORDS_ERROR.

15. Некорректные координаты ненулевых элементов матрицы. Программа вернет INCORRECT_COORDS_ERROR.
16. Ошибка ввода тестового количества строк матрицы. Программа вернет INPUT_TEST_ROWS_ERROR.
17. Ошибка ввода тестового количества столбцов матрицы. Программа вернет INPUT_TEST_COLS_ERROR.
18. Ошибка выделения памяти при тестировании. Программа вернет TEST_ALLOC_ERROR.

Описание внутренних структур данных

Программа содержит в себе структуру, которая используется для хранения стандартной матрицы, которая представлена в Листинге 1.

```
typedef struct
{
    size_t rows;
    size_t cols;
    int **values;
} standard_matrix_t;
```

Листинг 1. Структура standard_matrix_t

Рассмотрим каждое поле структуры:

1. rows: целочисленное значение типа size_t, указывающее количество строк в матрице.
2. cols: целочисленное значение типа size_t, указывающее количество столбцов в матрице.
3. values: указатель на указатель типа int, который представляет собой двумерный массив целых чисел. Этот массив используется для хранения значений стандартной матрицы, где каждое значение является целым числом.

Также программа содержит в себе структуру sparse_matrix_t, которая представляет собой разреженную матрицу, которая представлена в Листинге 2.

```
typedef struct
{
    size_t rows;
    size_t cols;
    size_t A_len;
    int *A;
    int *IA;
    size_t JA_len;
    int *JA;
} sparse_matrix_t;
```

Листинг 2. Структура sparse_matrix_t

Рассмотрим каждое поле структуры:

1. rows: целочисленное значение типа size_t, указывающее количество строк в разреженной матрице.
2. cols: целочисленное значение типа size_t, указывающее количество столбцов в разреженной матрице.
3. A_len: целочисленное значение типа size_t, указывающее количество ненулевых элементов в разреженной матрице.
4. A: указатель на массив целых чисел типа int, содержащий все ненулевые элементы разреженной матрицы.
5. IA: указатель на массив целых чисел типа int, содержащий индексы строк для каждого элемента из массива A.
6. JA_len: целочисленное значение типа size_t, указывающее длину массива JA. Это значение соответствует количеству столбцов плюс один.
7. JA: указатель на массив целых чисел типа int, который содержит индексы в массиве A, где начинаются элементы для каждого столбца.

Описание алгоритма

- Инициализация системы и выделение памяти.
- Основной цикл программы начинается. Пользователю предоставляется меню с различными опциями, и программа ожидает ввода выбора пользователя.
- В зависимости от выбора пользователя, программа выполняет следующие действия:
  - Выйти из программы
  - Инициализировать две матрицы
  - Вывести первую матрицу
  - Вывести характеристику первой разреженной матрицы
  - Вывести первую разреженную матрицу
  - Вывести вторую матрицу
  - Вывести характеристику второй разреженной матрицы
  - Вывести вторую разреженную матрицу
  - Посчитать сложение первых двух обычных матриц и вывести результат
  - Посчитать сложение первых двух разреженных матриц и вывести результат. Алгоритм сложения разреженных матриц можно описать следующим образом:

1. Инициализация: создаем временные массивы для хранения ненулевых элементов и индексов строк и столбцов для результирующей матрицы. Мы запрашиваем память для временных массивов, основываясь на количестве ненулевых элементов в обеих матрицах (матрицы A и B).
2. Итерация по столбцам: для каждой колонки, начиная с первой:
   a. Для каждой колонки определяем количество ненулевых элементов в обеих матрицах (n1 для матрицы A и n2 для матрицы B).
   b. Инициализируем указатели для чтения ненулевых элементов в обоих входных массивах, а также указатель для записи результата.
3. Сравнение строк: сравниваем строки ненулевых элементов:

a. Если строки одинаковые, складываем соответствующие элементы и сохраняем результат в новый массив, увеличивая указатели для обеих матриц.
b. Если строки различаются, мы копируем элемент из той матрицы, где строка меньше (или если ненулевые элементы в другой матрице закончились), и увеличиваем соответствующий указатель.

4. Обновление указателей: после обработки всех ненулевых элементов в колонке мы записываем новую длину ненулевых элементов в результирующую матрицу.
5. Копирование результата: после завершения обработки всех колонок копируем значения из временных массивов в результирующие массивы результирующей матрицы.
6. Очистка: освобождаем память, выделенную для временных массивов, и возвращаем состояние выполнения операции.

- Сравнить время выполнения суммы для разреженной и стандартной матриц
- После выполнения каждой операции программа возвращает пользователя в главное меню, где он может выбрать следующее действие.

Набор тестов

Описание	**Результат**
Чтение данных из файла Передается DATA_1 и DATA_2 DATA_1: DATA_2: 3 3 3 3 1 0 1 0 2 0 2 2 0 0 0 1 0 2 0 1 0 0	Успешное считывание данных из файла
Вывод матрицы Храниться матрица:	Успешный вывод матрицы Вывод:

3 3 1 0 1 2 2 0 0 2 0	1 0 1 2 2 0 0 2 0
Вывод разреженной матрицы Матрица вида: 4 5 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3	Успешный вывод разреженной матрицы Вектор A содержит значения ненулевых элементов A: 1 3 Вектор IA содержит номера строк для элементов вектора A IA: 0 3 Вектор JA, в элементе Nk которого находится номер компонент в A и IA, с которых начинается описание столбца Nk матрицы A JA: 0 1 1 1 1 2
Вывод разреженной матрицы в виде обычной Матрица вида: 4 5 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3	Успешный вывод разреженной матрицы в виде обычной Вывод: 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3
Посчитать сложение первых двух обычных матриц Матрицы вида: 4 5 4 5	Успешное сложение обычных матриц Вывод: 2 1 1 1 1 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3	1 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 3	1 0 0 0 0 0 0 0 0 6

Таблица 2. Набор позитивных тестов

Описание	**Результат**
Попытка не передать данные	Возврат ошибки NO_DATA_ERROR
Попытка передать несуществующий файл	Возврат ошибки OPEN_FILE_ERROR
Попытка ввода некорректного выбора команды	Возврат ошибки INVALID_INPUT_CHOICE_ERROR
Попытка ввести недопустимое количество строк матрицы	Возврат ошибки INCORRECT_ROWS_MATRIX_ERROR
Попытка ввести недопустимое количество столбцов матрицы	Возврат ошибки INCORRECT_COLS_MATRIX_ERROR
Попытка ввода некорректного количества ненулевых элементов матрицы	Возврат ошибки INCORRECT_NON_ZERO_CNT_ERROR
Попытка ввода некорректных координат для ненулевых элементов матрицы	Возврат ошибки INCORRECT_COORDS_ERROR
Попытка выделения памяти для тестирования не удалась	Возврат ошибки TEST_ALLOC_ERROR

Таблица 3. Набор негативных тестов

Оценка эффективности

При запуске программы N = 500 раз, были получены следующие данные:

Размер матрицы	Время, нс		Объем памяти, байт	
	Разреженная	Стандартная	Разреженная	Стандартная
10x10	24368	48388	372	1200
15x15	81397	191732	696	2700
20x20	128475	270371	1116	4800

Таблица 4. Объем занимаемой памяти при 10% заполненности матриц

Эффективность при 10% = (50% + 58% + 52%) / 3 ≈ 53%

Размер матрицы	Время, нс		Объем памяти, байт	
	Разреженная	Стандартная	Разреженная	Стандартная
10x10	41216	54150	852	1200
15x15	88738	101205	1776	2700
20x20	141743	220983	3036	4800

Таблица 5. Объем занимаемой памяти при 30% заполненности матриц

Эффективность при 30% = (24% + 12% + 36%) / 3 ≈ 24%

Размер матрицы	Время, нс		Объем памяти, байт	
	Разреженная	Стандартная	Разреженная	Стандартная
10x10	50506	45936	1164	1200
15x15	37308	37452	2472	2700
20x20	45009	47842	4284	4800

Таблица 6. Объем занимаемой памяти при 43% заполненности матриц

Эффективность при 43% = (-9% + 1% + 5%) / 3 ≈ -1%

Размер матрицы	Время, нс		Объем памяти, байт	
	Разреженная	Стандартная	Разреженная	Стандартная
10x10	78450	65362	1332	1200
15x15	133691	115067	2856	2700

20x20	264365	202939	4956	4800

Таблица 6. Объем занимаемой памяти при 50% заполненности матриц

Эффективность при 50% = (-20% + -16% + -22%) / 3 ≈ -22%

Размер матрицы	Время, нс		Объем памяти, байт	
	Разреженная	Стандартная	Разреженная	Стандартная
10x10	79657	51811	1812	1200
15x15	170159	101551	3936	2700
20x20	278515	178201	6876	4800

Таблица 7. Объем занимаемой памяти при 70% заполненности матриц

Эффективность при 70% = (-54% + -68% + -56%) / 3 ≈ -59%

Размер матрицы	Время, нс		Объем памяти, байт	
	Разреженная	Стандартная	Разреженная	Стандартная
10x10	108265	46002	2532	1200
15x15	221531	110655	5544	2700
20x20	381372	165284	9756	4800

Таблица 8. Объем занимаемой памяти при 100% заполненности матриц

Эффективность при 100% = (-135% + -100% + -131%) / 3 ≈ -122%

Выводы

В данной лабораторной работе было проведено сравнение времени выполнения операций умножения и объема памяти, используемого разреженными и стандартными матрицами. Мы исследовали различные уровни разреженности матриц и оценили их влияние на эффективность разреженных матриц. Время выполнения операций над разреженными и стандартными матрицами зависит от размеров матрицы и уровня ее разреженности. Разреженные матрицы занимают гораздо меньше памяти по сравнению со стандартными матрицами, особенно при высоких уровнях разреженности. Однако, исходя из выполненных замеров, можно увидеть, что приведённый способ хранения разреженных матриц

целесообразен только при заполнении матрицы ниже 43 процентов, иначе эффективность становится и вовсе хуже, особенно при высоких значениях размерностей.

Контрольные вопросы

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

Разреженная матрица – это матрица, содержащая большое количество нулей. Способы хранения:

- Связная схема хранения – каждый ненулевой элемент хранится в виде узла в связном списке. Обычно каждая строка матрицы представлена своим собственным списком, где каждый узел содержит информацию о номере столбца и значении элемента.
- Строчный формат (CSR - Compressed Sparse Row)
- Столбчатый формат (CSR - Compressed Sparse Column)
- Линейный связный список – все ненулевые элементы хранятся в одном списке, где каждый элемент имеет указатели на свои координаты (строка и столбец) и значение
- Кольцевой связный список – в отличие от линейного связного списка, здесь все элементы объединяются в кольцо, что может быть полезно для циклических операций с матрицей (например, итерация по строкам и столбцам).

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

Для хранения разреженной матрицы выделяется значительно меньше памяти по сравнению с обычной матрицей при определенной степени разреженности (43% при наших замерах), т.к. хранятся только ненулевые элементы. Количество памяти, которое выделяется для хранения разреженной матрицы, зависит от выбранной схемы хранения и структуры самой матрицы. Обычная матрица, в свою очередь, выделяет память под каждый элемент матрицы вне зависимости от его значения n*m*sizeof(int).

3. Каков принцип обработки разреженной матрицы?

Обработка разреженной матрицы отличается от обработки обычной матрицы. Основной принцип обработки разреженной матрицы - исключить операции с нулевыми элементами, чтобы снизить вычислительные затраты и использование памяти.

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

Стандартные алгоритмы обработки матриц эффективнее применять, когда матрица имеет почти все ненулевые элементы и размеры матрицы не очень велики. Это зависит от плотности разреженной матрицы – чем больше ненулевых элементов, тем менее эффективным будет использование схемы хранения разреженной матрицы.

Если матрица плотная или имеет малое количество ненулевых элементов, использование стандартных алгоритмов обработки матриц может быть более эффективным.