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 5

Студент **Ильченко Е. А.**

Группа **ИУ7-34Б**

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

Студент _____ **Ильченко Е. А.**

Преподаватель _____ **Силантьева А. В.**

2024 г.

Описание условия задачи

Первая разреженная (содержащая много нулей) матрица хранится в форме 3-х объектов:

- 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.

Вторая разреженная матрица хранится в форме 3-х объектов:

- вектор B содержит значения ненулевых элементов;
- вектор JB содержит номера столбцов для элементов вектора A;
- вектор IB, в элементе Nk которого находится номер компонент в B и JB, с которых начинается описание строки Nk матрицы B.

1. Смоделировать операцию умножения двух матриц, хранящихся в указанной форме, с получением результата в форме хранения первой матрицы.
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 исходных данных и результатов работы программы

Входные данные:

Пользовательская команда из доступных и необходимые аргументы определенного сценария:

1: Ввести матрицы
2: Вывести матрицы в стандартном виде
3: Вывести матрицы в разреженном виде
4: Умножить матрицы в стандартном виде
5: Умножить матрицы в разреженном виде
6: Меню
7: Сравнение эффективности матриц
0: Выйти

При вводе матрицы пользователь может выбрать случайное создание матрицы или ручной ввод матрицы.

При случайном вводе матрицы вводятся размеры матрицы и процент заполнения

матрицы.
При ручном вводе матрицы пользователь вводит размеры матрицы и количество ненулевых элементов. Далее координатным методом вводятся координаты элемента и его значение.

Выходные данные:
Матрицы в стандартном и разреженном виде, результат умножения матриц в стандартном и разреженном виде. Сравнение эффективности двух алгоритмов умножения матриц.

2. Описание задачи, реализуемой в программе

Задача, реализуемая в программе, заключается в реализации алгоритмов обработки разреженных матриц, сравнении эффективности применения этих алгоритмов со стандартными алгоритмами обработки матриц при различном размере матриц и степени их разреженности.

3. Способ обращения к программе

Запуск исполняемого файла

```
./app.exe
```

Далее выбирается, какой пункт меню выполнить

4. Описание возможных аварийных ситуаций и ошибок пользователя

- Неверный ввод пункта меню: сообщение “Неверная команда”
- Перемножение матриц с размерами, не подходящими для перемножения: сообщение “Ошибка: матрицы не могут быть перемножены”
- Неверный ввод количества ненулевых элементов матрицы: сообщение “Ошибка ввода”
- Неверный ввод количество строк матрицы: сообщение “Ошибка ввода”
- Неверный ввод количества столбцов матрицы: сообщение “Ошибка ввода”
- Неверный ввод ненулевого элемента матрицы: сообщение “Ошибка: некорректные индексы или значение элемента.”
- Неверный ввод индекса элемента матрицы: сообщение “Ошибка: некорректные индексы или значение элемента.”

5. Описание внутренних структур данных

```
typedef struct
{
   double *A;
```

```
   int *IA;
   int *JA;
   int nonzeros;
   int rows;
   int cols;
} SparseMatrixA;

typedef struct
{
   double *B;
   int *JB;
   int *IB;
   int nonzeros;
   int rows;
   int cols;
} SparseMatrixB;

typedef struct
{
   double **data;
   int rows;
   int columns;
   int nonzeros;
} Matrix;
```

6. Описание функций

```
void generate_dense_matrix(Matrix *matrix, int rows, int cols, double
density);
```

Генерация случайной матрицы

```
void input_dense_matrix(Matrix *matrix, int rows, int cols, int density);
```

Координатный ввод матрицы

```
void convert_matrix_to_sparse_matrixA(Matrix *matrix, SparseMatrixA
*sparse_matrix);
```

Конвертация стандартной матрицы в первую форму хранения

```
void convert_matrix_to_sparse_matrixB(Matrix *matrix, SparseMatrixB
*sparse_matrix);
```

Конвертация стандартной матрицы во вторую форму хранения

```
void print_dense_matrix(const Matrix *matrix);
```

Стандартный вывод матриц

```
void print_sparse_matrixA(const SparseMatrixA *matrix);
```

Вывод разреженной матрицы в первой форме хранения

```
void print_sparse_matrixB(const SparseMatrixB *matrix);
```

Вывод разреженной матрицы во второй форме хранения

```
SparseMatrixA *multiply_sparse_matrices(const SparseMatrixA *matrixA, const
SparseMatrixB *matrixB);
```

Умножение разреженных матриц

```
void multiply_dense_matrices(Matrix *matrixA, Matrix *matrixB, Matrix
*result_matrix);
```

Стандартное умножение матриц

```
void free_sparse_matrixA(SparseMatrixA *matrix);
```

Освобождение памяти из под матрицы в первой форме хранения

```
void free_sparse_matrixB(SparseMatrixB *matrix);
```

Освобождение памяти из под матрицы во второй форме хранения

```
void free_matrix(Matrix *matrix);
```

Освобождение памяти из под стандартной матрицы

7. Описание алгоритмов

Алгоритм умножения стандартных матриц

Проверка совместимости матриц

1. Проверяется условие: количество столбцов матрицы `A` должно быть равно количеству строк матрицы `B`
2. Если условие не выполняется, алгоритм завершает выполнение с ошибкой.

Инициализация результирующей матрицы

1. Устанавливаются размеры результирующей матрицы: количество строк `rows=A->rows` и количество столбцов `columns=B->columns`
2. Для хранения данных результирующей матрицы выделяется память, инициализированная нулями.

Перебор строк и столбцов результирующей матрицы

1. Внешний цикл проходит по всем строкам результирующей матрицы

2. Вложенный цикл проходит по всем столбцам результирующей матрицы

Расчёт элемента результирующей матрицы

1. Для каждого элемента результирующей матрицы `R[i][j]` выполняется вычисление суммы произведений элементов:
    a. Внутренний цикл проходит по всем индексам столбцов матрицы `A k`
    b. К накопленной сумме добавляется произведение элементов `A[i][k]` и `B[k][j]`
2. Таким образом, `R[i][j]` становится равным сумме всех произведений.

Повторение для всех элементов результирующей матрицы

1. Шаги из пункта 4 повторяются для каждой комбинации строк матрицы `A` и столбцов матрицы `B` до полного заполнения результирующей матрицы

Возврат результата

1. После завершения всех вычислений возвращается результирующая матрица, содержащая произведение двух входных матриц
2. Убедитесь, что память, выделенная для матриц, освобождается после их использования

Алгоритм умножения разреженных матриц

Проверка совместимости матриц

1. Условие для умножения: количество столбцов матрицы A должно быть равно количеству строк матрицы B
2. Если условие не выполняется, выводится ошибка, и алгоритм завершается.

Инициализация результирующей матрицы

1. Создаётся результирующая матрица `R` с параметрами:
    a. количество строк равно количеству строк матрицы A
    b. количество столбцов равно количеству столбцов матрицы B
    c. массивы `A, IA, JA`, где:
        i. `A` – массив для хранения ненулевых элементов;
        ii. `IA` – массив индексов строк ненулевых элементов;
        iii. `JA` – массив указателей на начало каждого столбца.
2. Инициализируется счётчик ненулевых элементов `nonzeros_count=0`

Инициализация временных массивов

1. Создаются вспомогательные массивы:
    a. `temp_values` для накопления промежуточных результатов умножения;
    b. `is_nonzero` для отслеживания строк, которые содержат ненулевые значения в текущем столбце.

Перебор столбцов результирующей матрицы

1. Внешний цикл перебирает столбцы результирующей матрицы `R` от `0` до `B->cols-1`
2. В каждом цикле:

a. Сбрасываются массивы `temp_values` и `is_nonzero`, обнуляя их значения.

Обход строк матрицы `B` для текущего столбца `j`

Вложенный цикл перебирает строки `k` матрицы `B`

1. Для каждого ненулевого элемента в строке `k` матрицы `B`:
    a. Перебираются ненулевые элементы строки `k` матрицы `A`
    b. Если столбец текущего элемента матрицы `B` совпадает с текущим столбцом `j`:
        i. Вычисляется произведение значений элементов из матриц `A` и `B`
        ii. Результат добавляется в соответствующую строку массива `temp_values`
        iii. Строка отмечается как ненулевая в `is_nonzero`

Добавление ненулевых элементов в результирующую матрицу

1. После вычислений для столбца `j`:
    a. Ненулевые значения из массива `temp_values` добавляются в массив `R->A`
    b. Индексы строк этих значений записываются в массив `R->IA`
    c. Счётчик `nonzeros_count` увеличивается для каждого добавленного элемента.
    d. Обновляется массив `R->JA[j+1]`, указывающий на конец текущего столбца.

Освобождение временных массивов

1. После обработки всех столбцов результирующей матрицы освобождаются массивы `temp_values` и `is_nonzero`.

Возврат результата

1. Возвращается результирующая разреженная матрица `R`, содержащая:
    a. массив ненулевых значений `R->A`
    b. массив индексов строк `R->IA`
    c. массив указателей на начало столбцов `R->JA`

Тесты

Тест	**Входные данные**	**Выходные данные**
Заполнить матрицы и вывести на экран в стандартном формате, ввод с клавиатуры	Количество строк, столбцов и количество ненулевых элементов для матрицы A: 2 2 2 Элемент 1: 0 1 1 Элемент 2: 1 0 1 Количество строк,	Матрица (2x2): 0.00 1.00 1.00 0.00 Матрица (2x2): 5.00 3.00 0.00 0.00

	столбцов и количество ненулевых элементов для матрицы B: 2 2 2 Элемент 1: 0 1 3 Элемент 2: 0 0 5	
Вывести матрицы в разреженном виде	Матрица A: 0.00 1.00 1.00 0.00 Матрица B: 5.00 3.00 0.00 0.00	Матрица A Значения ненулевых элементов (A): 1.00 1.00 Индексы строк (IA): 1 0 Индексы начала столбцов (JA): 0 1 2 Матрица B Значения ненулевых элементов (B): 5.00 3.00 Индексы столбцов (JB): 0 1 Индексы начала строк (IB): 0 2 2
Умножить матрицы и вывести в стандартном виде	Матрица A: 0.00 1.00 1.00 0.00 Матрица B: 5.00 3.00 0.00 0.00	Матрица (2x2): 0.00 0.00 5.00 3.00
Умножить матрицы и вывести в разреженном виде	Матрица A: 0.00 1.00 1.00 0.00 Матрица B: 5.00 3.00 0.00 0.00	Значения ненулевых элементов (A): 5.00 3.00 Индексы строк (IA): 1 1 Индексы начала столбцов (JA): 0 1 2
Ввод неверной команды	Команда: “a”	Неверная команда
Ввод числа 4 при выборе	4	Ошибка ввода

способа задания матрицы		
Неверный ввод количество строк матрицы	-1	Ошибка ввода
Неверный ввод количества столбцов матрицы	-1	Ошибка ввода
Перемножение матриц с размерами, не подходящими для перемножения	Матрица 2x2 и матрица 3x3	Ошибка: матрицы не могут быть перемножены
Неверный ввод количества ненулевых элементов	-1	Ошибка ввода
Неверный ввод ненулевого элемента	0	Ошибка: некорректные индексы или значение элемента.
Неверный ввод индекса элемента матрицы	-1	Ошибка: некорректные индексы или значение элемента.

Оценка эффективности

Эффективность по времени/памяти считалась путем замера 100 раз методов умножения стандартных матриц и разреженных матриц и усреднения результатов Измерения проводились на MacBook Pro 13 2019

Матрица 30x30

Заполненность 1%

Вид матрицы	Время, нс	Память, байт	Процентное соотношение времени	Процентное соотношение памяти
Стандартные	164950.00	21600	92.19 %	97.11 %
Разреженные	12880.00	624		

Заполненность 10%

Вид матрицы	Время, нс	Память, байт	Процентное соотношение времени	Процентное соотношение памяти
Стандартные	147530.00	21600	57.01 %	73.83 %
Разреженные	63430.00	5652		

Заполненность 20%

Вид матрицы	Время, нс	Память, байт	Процентное соотношение времени	Процентное соотношение памяти
Стандартные	147230.00	21600	-26.97 %	43.22 %
Разреженные	186940.00	12264		

Заполненность 25%

Вид матрицы	Время, нс	Память, байт	Процентное соотношение времени	Процентное соотношение памяти
Стандартные	146230.00	21600	-76.11 %	30.33 %
Разреженные	257530.00	15048		

Заполненность 30%

Вид матрицы	Время, нс	Память, байт	Процентное соотношение времени	Процентное соотношение памяти
Стандартные	140450.00	21600	-135.62 %	21.50 %
Разреженные	330930.00	16956		

Заполненность 40%

Вид матрицы	Время, нс	Память, байт	Процентное соотношение времени	Процентное соотношение памяти
Стандартные	140770.00	21600	-275.41 %	8.83 %
Разреженные	528460.00	19692		

Заполненность 50%

Вид матрицы	Время, нс	Память, байт	Процентное соотношение времени	Процентное соотношение памяти
Стандартные	164090.00	21600	-433.02 %	-1.72 %
Разреженные	874640.00	21972		

Матрица 70x70

Заполненность 1%

Вид матрицы	Время, нс	Память, байт	Процентное соотношение времени	Процентное соотношение памяти
Стандартные	2333910.00	117600	96.24 %	97.84 %
Разреженные	87640.00	2544		

Заполненность 10%

Вид матрицы	Время, нс	Память, байт	Процентное соотношение времени	Процентное соотношение памяти
Стандартные	2204140.00	117600	36.25 %	63.69 %
Разреженные	1405060.00	42696		

Заполненность 20%

Вид матрицы	Время, нс	Память, байт	Процентное соотношение времени	Процентное соотношение памяти
Стандартные	1836650.00	117600	-115.74 %	31.92 %
Разреженные	3962420.00	80064		

Заполненность 25%

Вид матрицы	Время, нс	Память, байт	Процентное соотношение времени	Процентное соотношение памяти
Стандартные	1793990.00	117600	-220.45 %	24.81 %

Разреженные	5748840.00	88428		

Заполненность 30%

Вид матрицы	Время, нс	Память, байт	Процентное соотношение времени	Процентное соотношение памяти
Стандартные	2068210.00	117600	-272.85 %	19.33 %
Разреженные	7711240.00	94872		

Заполненность 40%

Вид матрицы	Время, нс	Память, байт	Процентное соотношение времени	Процентное соотношение памяти
Стандартные	1810520.00	117600	-624.90 %	9.28 %
Разреженные	13124540.00	106692		

Заполненность 50%

Вид матрицы	Время, нс	Память, байт	Процентное соотношение времени	Процентное соотношение памяти
Стандартные	2259870.00	117600	-828.10 %	-0.72 %
Разреженные	20973930.00	118452		

Матрица 100x100

Заполненность 1%

Вид матрицы	Время, нс	Память, байт	Процентное соотношение времени	Процентное соотношение памяти
Стандартные	5362370.00	240000	96.50 %	97.94 %
Разреженные	187580.00	4944		

Заполненность 10%

Вид матрицы	Время, нс	Память, байт	Процентное соотношение	Процентное соотношение

			времени	памяти
Стандартные	6674330.00	240000	-15.20 %	57.57 %
Разреженные	7689140.00	101832		

Заполненность 20%

Вид матрицы	Время, нс	Память, байт	Процентное соотношение времени	Процентное соотношение памяти
Стандартные	8830430.00	240000	-90.13 %	30.41 %
Разреженные	16789540.00	167016		

Заполненность 25%

Вид матрицы	Время, нс	Память, байт	Процентное соотношение времени	Процентное соотношение памяти
Стандартные	8948420.00	240000	-201.16 %	24.57 %
Разреженные	26949160.00	181044		

Заполненность 30%

Вид матрицы	Время, нс	Память, байт	Процентное соотношение времени	Процентное соотношение памяти
Стандартные	10467510.00	240000	-259.92 %	19.49 %
Разреженные	37674670.00	193212		

Заполненность 40%

Вид матрицы	Время, нс	Память, байт	Процентное соотношение времени	Процентное соотношение памяти
Стандартные	5339220.00	240000	-947.76 %	9.49 %
Разреженные	55942140.00	217212		

Заполненность 50%

Вид матрицы	Время, нс	Память, байт	Процентное соотношение времени	Процентное соотношение памяти
Стандартные	5197090.00	240000	-1534.38 %	-0.50 %
Разреженные	84940120.00	241212		

Вывод

В ходе лабораторной работы были реализованы алгоритмы обработки разреженных матриц, также было проведено сравнение эффективности применения этих алгоритмов со стандартными алгоритмами обработки матриц при различном размере матриц и степени их разреженности. Было выяснено, что при малой степени заполненности матриц хранение их в формате разреженных более эффективно, чем хранение в стандартном формате. Так при одном проценте заполненности матрицы алгоритм умножения для разреженных матриц работает быстрее примерно на 90% по сравнению с алгоритмом умножения для стандартных матриц. Однако при увеличении заполненности эффективность как по времени, так и по объему занимаемой памяти снижается. В данной работе при заполненности 20% и размере матрицы 30x30 алгоритм умножения в формате разреженных матриц по времени становится неэффективным (-26.97 %%), а по объему - при заполненности 50% и малых размерах матриц (-1.72 %).

Ответы на контрольные вопросы

1. **Что такое разреженная матрица, какие схемы хранения таких матриц Вы знаете?**
   Разреженная матрица характеризуется преобладанием нулевых элементов. Она используется в таких областях, как обработка изображений, анализ графов и решение систем уравнений. Для хранения таких матриц применяются различные методы, включая линейные связные списки, диагональные схемы для симметричных матриц и другие связные структуры. Одним из самых популярных методов является "разреженный строчный формат" Чанга и Густавсона. Этот метод требует минимальных затрат памяти и эффективен при выполнении различных операций, таких как сложение и умножение матриц, перестановка строк и столбцов, транспонирование и решение систем уравнений.
2. **Каким образом и сколько памяти выделяется под хранение разреженной и обычной матрицы?**

Для стандартной матрицы требуется память объемом n * m ячеек, где n — это количество строк, а m — количество столбцов. Разреженная матрица обычно требует меньше памяти, поскольку сохраняет только ненулевые элементы, и объем используемой памяти зависит от выбранной схемы хранения. Например, для схемы хранения CSC необходимо 2 * k + n + 1 ячеек памяти, где k — количество ненулевых элементов, а n — количество столбцов.

3. **Каков принцип обработки разреженной матрицы?**
   Работа с разреженной матрицей отличается от работы с обычной. Главный подход в обработке разреженной матрицы заключается в исключении операций с нулевыми элементами, что снижает вычислительные и памятные затраты.
4. **В каком случае для матриц эффективнее применять стандартные алгоритмы обработки матриц? От чего это зависит?**
   Стандартные алгоритмы обработки матриц наиболее эффективны, когда матрицы содержат преимущественно ненулевые элементы и имеют небольшие размеры. Это зависит от "плотности" разреженной матрицы: чем больше ненулевых элементов, тем менее эффективным становится использование схемы хранения разреженной матрицы. В случае плотной матрицы или когда количество ненулевых элементов невелико, применение стандартных алгоритмов может быть более эффективным.