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)

Лабораторная Работа №3 «Обработка разреженных матриц» Вариант №3

Студент **Шахнович Дмитрий Сергеевич**

Группа **ИУ7-22Б**

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

Студент **Шахнович Д.С.**

Оценка _____

2023 г.

Описание условия задачи

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 умножения матрицы и вектора-столбца, хранящегося в форме вектора A и вектора, содержащего номера строк ненулевых элементов, с получением результата в форме хранения вектора-столбца.

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.

Техническое задание

Исходные данные:

Файл с матрицей: В файлах хранится матрица, где в начале через пробел указаны количество строк, столбцов и ненулевых элементов, а затем на новых строках указаны все ненулевые элементы координатным способом. Индексация в файле с 1.

Файл с вектором: В файлах хранится вектор, где в начале через пробел указаны количество элементов ненулевых элементов, а затем на новых строках

указаны все ненулевые элементы координатным способом. Индексация в файле с 1.

В меню программы действия задаются числами от 0-3. Ввод матрицы и вектора должен быть реализован как пользовательским вводом в консоль, так и вводом файла.

Выходные данные:

Программа выдает вектор-столбец — результат умножения исходной матрицы на исходный вектор-столбец, в случае выбора пункта умножения. В случае пункта сравнения выдает времена умножения для CRS и стандартного форматов, а также занимаемую ими память.

Описание задания:

Умножение матрицы на вектор-столбец в стандартном хранении и методом CRS, а также сравнение этих вариантов.

Способы обращения к программе:

Запуск программы через терминал, затем управление программой с помощью меню. Пункты меню:

1 — Умножение матрицы на вектор-столбец в формате хранение CRS;

2 — Умножение матрицы на вектор-столбец в стандартном формате хранения;

3 — Сравнение вариантов умножения;

0 — Выход из программы.

Аварийные ситуации:

1) В меню введен код, которого нет;
   Сообщение: «Ошибка: Некорректная команда.»
2) Ошибка ввода/вывода;
   Сообщение: «Ошибка функций ввода/вывода..»

3) Введены строковые литералы при запросе числа или число вне указанного диапазона;

   Сообщение: «Ошибка: Некорректный формат ввода.»

4) Не удалось открыть введенный файл;

   Сообщение: «Ошибка: Неправильное имя файла.»

5) Ошибка при работе с файлом(не удалось считать из него и т. д.)

   Сообщение: «Ошибка при работе с файлом.»

6) Введены объекты, неподходящие для умножения(количество столбцов матрицы не равно размеру вектора)

   Сообщение: «Ошибка: Неподходящие размеры для умножения.»

7) В файле некорректный формат данных.

   Сообщение: «Ошибка: Некорректный формат ввода.»

Описание структур данных

```
/// Структура матрицы в формате CRS.
struct matrix_t
{
    size_t rows; /// Количество строк в матрице.
    size_t cols; /// Количество столбцов в матрице.
    size_t elems; /// Количество ненулевых элементов в матрице.
    int *arr; /// Массив ненулевых элементов в матрице. Имеет размер равный elems.
    size_t *jarr; /// Массив индексов столбцов соответствующих элементов из arr. Имеет размер равный elems.
    size_t *ia; /// Массив индексов, с которых начинается описание i-ой строки. Например описание 2-ой(начиная с 0) строки начинается с элемента с индексом ia[2]. Имеет размер rows + 1.
};
```

Структура матрицы в формате CRS.

```
/// Стандартная матрица.
struct std_matrix_t
```

```
{
   size_t rows; /// Количество строк в матрице.
   size_t cols; /// Количество столбцов в матрице.
   int **matrix; /// Массив указателей на строки матрицы.
};
```

Структура матрицы в станадртном формате.

```
/// Вектор в формате CRS.
struct vector_t
{
   size_t size; /// Размер вектора.
   size_t elems; /// Количество ненулевых элементов в векторе.
   int *arr; /// Массив ненулевых элементов в векторе. Имеет размер elems.
   size_t *iarr; /// Массив индексов ненулевых элементов соответствующих из arr. Имеет размер elems.
};
```

Структура вектора в формате CRS.

```
/// Стандартный вектор
struct std_vector_t
{
   size_t size; /// Размер вектора.
   int *arr; /// Массив элементов вектора.
};
```

Структура вектора в стандартном формате.

Описание алгоритма

1. Вывести пользователю меню и ожидать ввода номера команды;
2. После ввода команды спросить пользователя вид ввода матрицы(файловый или ручной);
3. Ввести название файла или матрицу;
4. Спросить пользователя вид ввода вектора(файловый или ручной);
5. Ввести название файла или вектор;

6. В случае выбора умножения провести умножение в выбранном формате, в случае сравнение — провести замеры умножений;
7. Вывести результаты на экран и завершить программу.

В случае возникновения ошибок на любом этапе, вывести сообщение об ошибки и вернуться в 1-й пункт.

Тестовые данные

Позитивные тесты на умножение подразумеваются для обоих видов хранения матрицы и вектора.

Позитивные тесты			
№	Описание	Вход	Выход
1	Умножение матрицы 1 на 1 на вектор размера 1	Матрица: 1 1 1 1 1 2 Вектор: 1 1 2	Результат: 4
2	Умножение матрицы 3 на 1 на вектор размера 1	Матрица: 3 1 3 1 1 2 2 1 3 3 1 4 Вектор: 1 1 2	Вектор: 4 6 8
3	Умножение матрицы 1 на 3 на вектор размера 3	Матрица: 1 3 3 1 1 2	Вектор: 29

		1 2 3 1 3 4 Вектор: 3 3 1 2 2 3 3 4	
4	Умножение матрицы 3 на 3 на вектор размера 3	Матрица: 3 3 5 1 1 1 2 1 5 2 2 3 3 1 2 3 3 -5 Вектор: 3 3 1 1 2 2 3 3	Вектор: 1 11 -13
5	Умножение матрицы 2 на 3 на вектор размера 3	Матрица: 2 3 3 1 2 -11 1 3 3 2 1 4 Вектор: 3 3 1 2 2 -13 3 11	Вектор: 176 8

6	Умножение нулевой матрицы 3 на 3 на вектор размера 3	Матрица: 3 3 0 Вектор: 3 2 1 2 3 -5	Вектор: 0 0 0
7	Умножение матрицы 3 на 3 на нулевой вектор размера 3	Матрица: 3 3 4 1 2 3 1 3 5 3 1 4 3 2 5 Вектор: 3 0	Вектор: 0 0 0
8	Ввод корректной матрицы в консоль	Матрица: 3 3 4 1 2 3 1 3 5 3 1 4 3 2 5 Вектор: 3 3 1 2 2 2 3 -5	Вектор: -19 0 18
9	Ввод корректной матрицы через файл	Матрица: 3 3 4 1 2 3 1 3 5	Вектор: -19 0 18

		3 1 4 3 2 5 Вектор: 3 3 1 2 2 2 3 -5	
10	Сравнение работы двух умножений	Матрица: 1 1 1 1 1 2 Вектор: 1 1 2	«Информация о скорости работы умножений и о занимаемой памяти»
11	Выход	0	«Выход из программы»
Негативные тесты			
1	Нулевой размер у матрицы	Матрица: 0 3 3	Ошибка: Некорретный формат ввода.
2	Нулевой размер у вектора	Вектор: 0 2	Ошибка: Некорретный формат ввода.
3	Ввод элемента равного 0	Матрица: 2 2 2 1 1 1 1 2 0	Ошибка: Некорретный формат ввода.
4	Ввод элементов с индексами не по возрастанию	Матрица: 3 3 3 2 1 3 1 1 3 3 3 3	Ошибка: Некорретный формат ввода.

5	Неподходящие размеры у матрицы и вектора для умножения	Матрица: 2 1 0 Вектор: 3 0	Ошибка: Неподходящие размеры для умножения.
6	Ввод несуществующего файла	«Несуществующий файл»	Ошибка: Неправильное имя файла.
7	Ввод литерала при вводе команд или матриц/векторов	daasd	Ошибка: Некорректная команда.
8	Ввод некорректной команды	16	Ошибка: Некорректная команда.
9	Ввод недозаполненного файла	Матрица: 3 3 3 1 1 2 1 2 3	Ошибка: Некорретный формат ввода.

Замеры

Замеры проводились следующим образом: с помощью скрипта gen_test.py создавались файлы со случайным образом заполненными матрицами и векторами заданного размера и заполненности. Затем данные считывались в get_statistics.c и 100 раз умножались, замеряя время. В качестве результирующего времени бралось среднее арифметическое по всем 100 проходам. Затем для каждого вида хранения в программе считался объем памяти, занимаемый объектами.

Замеры умножения по времени

Кол-во элементов.	Заполненность матрицы и вектора, %.	Стандартное хранение, мкс	Хранение CRS, мкс	Отношение времени стандартного к CRS
50	10	4	0	inf
	20	5	1	5.0
	40	4	2	2.0
	80	3	6	0.5
100	10	15	2	7.5
	20	15	4	3.75
	40	15	8	1.875
	80	15	20	0.75
500	10	371	46	8.06
	20	380	143	2.66
	40	369	482	0.76
	80	372	787	0.47
1000	10	1474	219	6.73
	20	1516	613	2.47
	40	1540	2021	0.76
	80	1473	3154	0.47

Замеры матриц по памяти

Кол-во элементов.	Заполненность матрицы и вектора, %.	Стандартное хранение матрицы, байт	Хранение CRS матрицы, байт	Стандартное хранение вектора, байт	Хранение CRS вектора, байт
50	10	10024	3456	216	92
	20	10024	6456	216	152
	40	10024	12456	216	272
	80	10024	24456	216	512
100	10	40024	12856	416	152
	20	40024	24586	416	272
	40	40024	48856	416	512
	80	40024	96856	416	992
500	10	1000024	304056	2016	632
	20	1000024	604056	2016	1232
	40	1000024	1204056	2016	2432
	80	1000024	2404056	2016	4832
1000	10	4000024	1208056	4016	1232
	20	4000024	2408056	4016	2432
	40	4000024	4808056	4016	4832
	80	4000024	9608056	4016	9632

Как видно, использование CRS формата выгодно при заполнении матрицы ~20%, причем как по памяти, так и по скорости умножения. При заполнении выше стандартный формат хранения лучше.

Ответы на вопросы

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

   Матрица называется разреженной, если количество её элементов находится на интервале ($n^{1.2}$-$n^{1.5}$), где n — порядок матрицы. Разрешенные матрицы можно хранить в стандартной форме, в форме трех массивов, где есть массив ненулевых элементов, массивы с их столбцами и строками. Также один из массивов с индексами можно

заменить на массив индексов начала строк/столбцов, что может сократить память.

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

   Под обычную матрицу всегда выделяется память объемом всех элементов. Под разреженную матрицу можно выделять память только ненулевые элементы, но при этом также придется выделить дополнительную память для сохранения положения элементов в матрице.

3. *Каков принцип обработки разреженной матрицы?*

   Основным принципом при обработке разреженной матрицы является обработка и хранение только ненулевых элементов матрицы, так как они составляют большую её часть.

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

   Стандартные алгоритмы обработки матриц эффективнее применять при высокой заполненности матрицы ненулевыми матрицами. Граница разреженности определяется в зависимости от конкретной реализации хранения разреженной матрицы.

Выводы

В некоторых задачах перед программистом может возникнуть необходимость обрабатывать и хранить разреженные матрицы. В таком случае в зависимости от степени их разреженности возможно применение моделей хранения разреженных матриц, например как CRS.