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

Группа ИУ7-35Б

Тема работы **Обработка разреженных матриц**

Студент Баранов Николай Алексеевич

Преподаватель Барышникова Марина Юрьевна

2022 г.

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.

1. Описание условия задачи

Разработать программу умножения вектора-строки и разреженной матрицы, хранящейся в виде 3-х объектов: вектора значений ненулевых элементов, вектора столбцов для элементов первого вектора и список начал описаний строк. Сравнить эффективность (по памяти и времени выполнения) стандартных алгоритмов обработки с алгоритмами обработки разреженных матриц при различной степени разреженности матриц и различной размерности матриц.

2. Описание ТЗ

2.1. Описание исходных данных и результатов.

При чтении данных программа сначала принимает на вход натуральное число в диапазоне от 1 до 1000: количество строк в матрице и размер вектора-строки. Далее программа считывает ещё одно число в том же диапазоне – количество столбцов в матрице. Далее предлагается один из 3 вариантов ввода – заполнение вручную всей матрицы, заполнение только ненулевых элементов и заполнение случайными числами.

При ручном заполнении всей матрицы программа поочередно считывает все элементы: натуральные числа в диапазоне от -1000 до 1000.

При заполнении только ненулевых элементов программа сначала запрашивает место, в которое надо будет вставить элемент: 2 натуральных числа в диапазоне от 1 до максимального размера матрицы, после чего считывает сам элемент. Ввод заканчивается в одном из трёх случаев: введена позиция за границами матрицы, значение очередного введённого элемента равно 0, введён элемент с максимальными значениями строки и столбца

При заполнении случайными числами вводится только натуральное число от 0 до 100: процент нулевых элементов.

Выбор действия происходит в меню, где вводится число – номер действия.

При сравнении эффективности для каждого способа выводится среднее время работы и максимальная затраченная память на хранение матриц и вектора.

2.2. Описание задачи, реализуемой программой.

Программа позволяет считывать матрицу, содержащую до 1000 строк и столбцов, а также вектор, размер которого равен количеству строк в матрице, выводить матрицу, вектор или результат перемножения, перемножать матрицы обычным способом, а также алгоритмом обработки разреженных матриц, а также сравнивать их эффективность по памяти и времени.

2.3. Способ обращения к программе.

Запуск программы:

./app.exe

В этом случае программа начнёт выполнять свою задачу. Ввод осуществляется вручную с клавиатуры, но чтение матрицы и вектора строки можно сделать и из файла.

2.4. Описание возможных аварийных ситуаций и ошибок пользователя.

2.4.1. Переполнение буфера. Возникает, если строка при чтении оказалась длиннее чем 4 символа (при чтении названия файла буфер увеличивается до 20 элементов).

2.4.2. Неожиданный символ. Возникает, если в строке присутствуют символы, не предусмотренные программой, или из-за неправильного формата входных данных (например, лишний пробел, лишняя точка в числе, пробел посреди числа и т. д.).

2.4.3. Число не входит в диапазон. Возникает, если пользователь ввёл число, которое не входит в диапазон допустимых значений.

2.4.4. Ошибка при выделении памяти. Практически не зависит от действий пользователя. Возникает только при неудачной попытке выделить память на куче, поэтому вероятность возникновения крайне мала.

2.4.5. Ошибка при открытии файла. Возникает, если не найден файл с заданным именем.

2.4.6. Ошибка при чтении файла. Возникает, если в файле содержатся некорректные данные или файл пустой.

3. Описание внутренних структур данных.

Для хранения матриц используются 2 структуры:

```
typedef struct special sp_m;
typedef struct usual us_m;

struct special
{
    int rows, columns;
    int el_cnt, r_cnt;
    int *vals;
    int *cols;
    struct
    {
        int row;
        int start;
    } *rows_st;
};

struct usual
{
    int rows, columns;
    int *vals;
};
```

sp_m используется для хранения разреженной матрицы.

Её поля:

rows – целое число. Хранит количество строк.

columns – целое число. Хранит количество столбцов.

el_cnt – целое число. Хранит количество ненулевых элементов.

r_cnt – целое число. Хранит количество строк с ненулевыми элементами.

vals – указатель на массив целых чисел. Хранит целые числа.

cols – указатель на массив целых чисел. Хранит номера столбцов.

rows_st – указатель на массив пар целых чисел. Хранит номера строк с ненулевыми элементами и их начала.

us_m используется для хранения матрицы в общем случае.

Её поля:

rows – целое число. Хранит количество строк.

columns – целое число. Хранит количество столбцов.

vals – указатель на массив целых чисел. Хранит целые числа.

4. Описание алгоритма.

Для обычного перемножения строки на матрицу: выполняется поэлементное умножение элементов строки и элементов столбца матрицы, произведения складываются и записываются в соответствующий элемент новой матрицы, и так с каждым столбцом. Транспонирование исходной матрицы не проводится. Сложность алгоритма: $O(n^2)$, где n – наибольшая из сторон матрицы.

Для перемножения в случае хранения матрицы в виде 3 массивов: сначала массив элементов вектора, в который будет записываться результат, заполняется нулями, затем для каждой строки матрицы проверяется, есть ли соответствующий ей ненулевой элемент в векторе, и если есть, то этот элемент по очереди умножается на все ненулевые элементы данной строки, а результаты прибавляются к соответствующим значениям в конечном векторе. Далее все ненулевые элементы сдвигаются в начало, а ненужная память освобождается. Обнуление вектора-результата, сдвиг и освобождение выполняются за O(l), где l – число строк в матрице, само умножение выполняется за O(m), где m – число ненулевых элементов, поэтому ожидаемая сложность алгоритма O(max(l, m)). В худшем случае $m = n^2$, поэтому сложность алгоритма в худшем случае $O(n^2)$. Худшим случаем является полное заполнение матрицы ненулевыми элементами.

5. Сравнительный анализ разных способов хранения и перемножения.

Для замера времени и памяти программа запускала 1000 итераций, на каждой из которых при этом заново генерировались матрица и вектор. Память выводится наибольшая использованная, а время работы выводится среднее за все итерации.

В таблицах ниже приведены результаты работы умножения для разных размерностей. (o – обычный способ, p – способ для разреженной матрицы).

Для матрицы 10 на 10:

Процент разреженности матрицы	Время, мкс (o)	Память, байт (o)	Время, мкс (p)	Память, байт (p)
0	1	528	3	1256
10	1	528	2	1160

20	1	528	2	1064
30	1	528	2	968
40	1	528	2	872
50	2	528	2	776
60	2	528	2	680
70	1	528	1	584
80	2	528	1	488
90	1	528	1	384

Эта матрица сравнительно небольшая, поэтому обрабатывалась довольно быстро и сказать про время работы сейчас ничего нельзя из-за низкой точности. Однако уже заметно, что при 80% нулевых элементов уже удалось сэкономить память за счёт того, что мы не стали хранить нулевые элементы.

Для матрицы 100 на 100:

Процент разреженности матрицы	Время, мкс (о)	Память, байт (о)	Время, мкс (р)	Память, байт (р)
0	76	40848	147	83336
10	74	40848	117	75176
20	74	40848	94	67016
30	74	40848	73	58856
40	75	40848	55	50696
50	79	40848	42	42536
60	77	40848	29	34376
70	74	40848	17	26216
80	81	40848	11	18056
90	75	40848	6	9896

Здесь точность времени представлена лучше. Уже на 30% разреженности алгоритмы сравнялись по времени, а на 60% получилось получить выигрыш по памяти.

Для матрицы 1000 на 1000 (количество итераций при замере было снижено до 100):

Процент разреженности матрицы	Время, мкс (о)	Память, байт (о)	Время, мкс (р)	Память, байт (р)
0	6782	4008048	9081	8032136
10	6338	4008048	7188	7230536
20	6362	4008048	5825	6428936
30	6274	4008048	4817	5627336
40	6163	4008048	3577	4825736
50	5999	4008048	3222	4024136
60	6106	4008048	1534	3222536
70	6021	4008048	864	2420936
80	6176	4008048	402	1619336
90	6105	4008048	119	817736

А здесь уже видно, что даже алгоритм для общего случая работает быстрее при увеличении числа нулевых элементов (но это скорее всего из-за более быстрого перемножения – на 0 умножить быстрее чем перемножить 2 ненулевых числа). Алгоритм для разреженных матриц уже на 20% разреженности работает быстрее, так как обрабатывается меньшее число элементов, а память удаётся сэкономить на 60%, так как дополнительная память ну хранение столбцов и начал строк как раз увеличивает используемую для разреженной матрицы память чуть больше чем в 2 раза, из-за чего как раз в интервале 50-60% всегда будет граничный случай, но котором используемая память примерно одинакова.

6. Набор тестов.

Тесты описаны только для ввода значений. Отсутствие выходных данных означает в данном случае что тест позитивный.

В меню проверяется только то что после каждого действия запускается нужная функция, а после неверного ввода просит ввести ещё раз.

Вывод таблицы записей не представлен ниже, так как его сложно записать в эту таблицу.

Входные данные	**Выходные данные**	**Что и где проверялось**
1		Ввод минимального целого числа
3		Ввод целого числа не на границе
1000		Ввод целого числа максимального размера (например, в матрице)
0003		Ввод целого числа с ведущими нулями (без переполнения буфера)
what	Ошибка. Встречен некорректный символ.	Ввод не числа, если нужно целое число.
12345	Ошибка. Переполнение буфера при вводе.	Переполнение буфера (при чтении целого числа).
-13	Ошибка. Число не входит в допустимый диапазон значений.	Число не входит в диапазон. Например, если минимальное допустимое число 1.
	Ошибка. Пустой ввод	Пустой ввод.

7. Ответы на вопросы.

7.1. Что такое разреженная матрица, какие схемы хранения разреженных матриц вы знаете?

Разреженная матрица – матрица с большим количеством нулевых элементов. Схемы хранения: с помощью линейного связного списка, с помощью кольцевого списка, двунаправленного стека или очереди, диагональная схема (для симметричных), а также связные схемы (Кнута, Чанга и Густавсона).

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

Для стандартного способа память выделяется только под хранение элементов, но при этом память тратится и на хранение нулевых элементов. Для способов хранения разреженных матриц память для нулевых элементов не выделяется, но при этом в зависимости от схемы хранения выделяется память под дополнительные данные, по которым мы и определяем, где эти ненулевые элементы находятся.

7.3. Каков принцип обработки разреженной матрицы?

Обрабатываются только ненулевые элементы.

7.4. В каком случае для матриц эффективнее применять стандартные алгоритмы обработки матриц? Отчего это зависит?

Если в матрице не так уж и много ненулевых элементов, то лучше применять стандартные алгоритмы обработки матриц, так как выигрыша по времени от алгоритмов для разреженных матриц практически не будет, зато расход памяти будет намного выше. Также при сравнительно небольших размерах матрицы также лучше применять стандартные алгоритмы, так как в этом случае влияние некоторых дополнительных полей для разреженной матрицы, а также части алгоритма, которые не учитываются при подсчёте асимптотики, здесь будут оказывать большее влияние, из-за чего экономия времени и памяти наступит позже, чем при больших размерах входных данных.