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 «ОБРАБОТКА РАЗРЕЖЕННЫХ МАТРИЦ»

Студент Козырнов Александр Дмитриевич

Группа ИУ7 – 32Б

Преподаватель Барышникова Марина Юрьевна

Вариант 6

2023 г.

Оглавление

Описание условия задачи

Разработать программу умножения или сложения разреженных матриц. Предусмотреть возможность ввода данных, как с клавиатуры, так и использования заранее подготовленных данных. Матрицы хранятся и выводятся в форме трех объектов. Для небольших матриц можно дополнительно вывести матрицу в виде матрицы. Величина матриц - любая (допустим, 1000*1000). Сравнить эффективность (по памяти и по времени выполнения) стандартных алгоритмов обработки матриц с алгоритмами обработки разреженных матриц при различной степени разреженности матриц и различной размерности матриц

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.

Описание технического задания

Входные данные:

Количество строк и столбцов матриц, количество ненулевых элементов матриц, их индексы и значения в матрицах,

Выходные данные:

Результат сложения двух матриц, сравнение двух способов обработки. Вывод содержания двух введеных матриц.

Обращение к программе:

Запускается через терминал командой: ./app.exe.

Аварийные ситуации:

1. Ввод несуществующего пункта меню
2. Некорректный ввод размера матриц
3. Некорректные значения индекса матрицы
4. Некорректно введено значение матрицы
5. Попытка вывести на экран непроинициализированную матрицу

НАБОР ТЕСТОВ

№	Название теста	Пользовательский ввод	Вывод
1	Сложение двух матриц	1 1 1 3 1 1 1 1 4 1 1 1 1 5	1
2	Вывод двух матриц	1 1 1 3 1 1 1 1 4 1 1 1 1 2	1 ========== 1
3	Некорректно заданы строчки матрицы	1 -1 1	Вы неверно ввели количество строк

4	Некорректно заданы столбцы матрицы	1 1 -1	Вы неверно ввели количество столбцов
5	Некорректно выбран индекс элемента или неверно введено число	3 1 0 -1 199	Вы неверно указали индекс матрицы или число!
6	Попытка вывести несозданные матрицы	2	Пустые матрицы.

Описание структуры данных

```c
typedef struct spr_matrix_t
{
    // Разряженная матрица
...
    /*
    * rows - Количество строк в матрице
    * cols - Количество столбцов в матрице
    * size - Количество ненулевых элементов
    * */
    int rows, cols, size;
...
    /*
    * *A  - Вектор ненулевых значений. Его размер равен size
    * *JA - Вектор столбцов для ненулевых значений. Его размер равен size
    * *IA - Сжатый вектор строчек. Его размер rows + 1
    * */
    int *A, *JA, *IA;
} spr_matrix_t;
```

```c
typedef struct std_matrix_t
{
    //Простая матрица
...
    /*
    * rows - Количество строк в матрице
    * cols - Количество строк в матрице
    * */
    int rows, cols;
—
    /*
    * **A - Матрица, представленная в виде указателей на массив, размером rows*cols, расположенный в одном месте
    * */
    int **A;
} std_matrix_t;
```

Временные замеры (мс)

Размер	Заполненность, %	Разряженная, мс	Простая, мс	Соотношение, в долях
10	10	0.2	0.08	0.4
	12	0.44	0.04	0.09
	20	0.68	0.2	0.29
	25	0.32	0.04	0.125
	30	0.52	0.16	0.3
	40	0.48	0.24	0.5
	50	0.72	0.04	0.06
	75	0.76	0.12	0.16
	85	0.84	0.12	0.14
	100	0.96	0.16	0.17
50	10	2.32	1.44	0.62
	12	2.8	1.44	0.51
	20	3.68	1.44	0.39
	25	3.96	1.24	0.31
	30	5.12	1.2	0.23
	40	5.92	1.28	0.21
	50	7.20	1.28	0.18
	75	9.40	1.08	0.11
	85	10.64	1.0	10.64
	100	12.08	1.08	0.09
100	10	8.92	4.84	0.54
	12	8.48	6.0	0.7
	20	13.52	4.16	0.3

	25	**15.76**	**4.04**	**0.25**
	30	**19.36**	**4.2**	**0.21**
	40	**24.36**	**4.08**	**0.17**
	50	**28.88**	**4.00**	**0.14**
	75	**39.44**	**4.08**	**0.1**
	85	**40.72**	**4.08**	**0.1**
	100	**45.56**	**4.0**	**0.09**
500	**10**	**181.60**	**403.32**	**2.22**
	12	**214.60**	**392.84**	**1.83**
	20	**343.48**	**389.12**	**1.13**
	25	**441.12**	**409.56**	**0.92**
	30	**510.16**	**409.4**	**0.8**
	40	**674.0**	**427.68**	**0.63**
	50	**815.08**	**429.96**	**0.52**
	75	**1111.12**	**453.12**	**0.41**
	85	**1164.32**	**427.32**	**0.37**
	100	**1310.56**	**438.28**	**0.33**

Память (байт)

Размер	Наполненность, %	Разряженная, байт	Простая, байт	Соотношение, в долях
10	10	124	400	3.22
	12	140	400	2.85
	20	204	400	1.96
	25	244	400	1.64
	30	284	400	1.41
	40	364	400	1.1
	50	444	400	0.9
	75	644	400	0.62
	85	724	400	0.55
	100	844	400	0.47
50	10	2204	10000	4.54
	12	2604	10000	3.84
	20	4204	10000	2.38
	25	5204	10000	1.92
	30	6204	10000	1.61
	40	8204	10000	1.22
	50	10204	10000	0.98
	75	15204	10000	0.66
	85	17204	10000	0.58
	100	20204	10000	0.49
100	10	8404	40000	4.75
	12	10404	40000	3.84
	20	16404	40000	2.44
	25	20404	40000	1.96
	30	24404	40000	1.64
	40	32404	40000	1.23
	50	40404	40000	0.99
	75	60404	40000	0.66
	85	68404	40000	0.58
	100	80404	40000	0.5 (0.497)
500	10	202004	1000000	4.95
	12	242004	1000000	4.13
	20	402004	1000000	2.49
	25	502004	1000000	1.99
	30	602004	1000000	1.66
	40	802004	1000000	1.25
	50	1002004	1000000	1 (0.998)
	75	1502004	1000000	0.67
	85	1702004	1000000	0.59
	100	2002004	1000000	0.5 (0.499)

Результаты были получены с помощью дополнительно написанной программы, не связанной напрямую с основной. Как можем заметить, разреженные матрицы довольно плохо соотносятся по скорости с обычными матрицами на маленьких (до 500 на 500) размерах и/или более заполненных ненулевыми элементами. Однако такие матрицы эффективнее используют память компьютера, если их разряженность составляет более 50% (наполненность менее 50%), на ощутимое количество. При довольно больших матрицах (500-х, где $x > 500$, на моих данных) с разреженностью в 70-100% (наполненность в 0-30%) становятся эффективнее и по памяти, и по скорости работы. Однако при увеличении наполненности они достаточно быстро теряют в скорости и становятся малоэффективными.

Описание алгоритма

1. После запуска программы пользователю предлагается или ввести две матрицы вручную
2. Пользователь указывает размерность матриц
3. Введённые матрицы хранятся в двух разных структурах: «обычной» матрице и разряженной.
4. Две матрицы складываются друг с другом, поочерёдно записывая сумму соответствующих элементов в результирующий массив, хранящийся в обоих видах. При сложении разреженных матриц поиск соответственных элементов реализуется с помощью прохода по массиву со сжатыми значениями IA двух матриц.
5. В случае возникновения аварийной ситуации программа оповещает пользователя о возникшей ошибке и продолжает работу.

Ответы на контрольные вопросы

1. Что такое разреженная матрица, какие способы хранения вы знаете?

Разреженная матрица — матрица, содержащая большое кол-во нулевых элементов. Хранить такую можно как обычную матрицу, с помощью линейных связных списков, кольцевых связных списков, двунаправленных стеков и очередей.

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

Под обычную матрицу выделяется **N * M** ячеек памяти. А для разреженной, в моём случае, **2 * L + (M + 1)**, где L — кол-во ненулевых элементов матрицы, а M – кол-во строк.

3. Каков принцип обработки разреженной матрицы?

Т.к. разреженные матрицы содержат большое кол-во нулей, то и хранятся они в таких структурах, которые «запоминают» только ненулевые элементы. Поэтому алгоритмы обработки оперируют лишь значащими данными, что даёт выигрыш по памяти и скорости по сравнению с алгоритмами обработки обычных матриц в некоторых случаях.

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

Чем менее разряжена матрица, тем меньше смысла ее использовать. Также не рекомендуется использовать разреженные матрица на маленьких матрицах, так как сильного прироста они не дадут. При 50% заполненности всегда лучше всего использовать обычные матрицы, так как на этом значении они эффективнее и по памяти, и по скорости вычисления. Чем больше матрица, тем больше может быть разряжена матрица. То есть для матриц (по тестам с 500x500) нужна наполненность в 20%, когда для матриц большего размера может доходить, чисто в теории, и до 30%.

Вывод

Использовать специальные структуры данных для (разреженных) матриц имеет смысл лишь при большом кол-ве элементов, т.к. тогда выигрыш по памяти будет существенен, и лишь при заполненности до 20-30% – иначе стандартные алгоритмы обработки матриц будут эффективнее во всех случаях, начиная с размерности 500x500. Тем более, чем бОльшая размерность у матриц, тем меньше процент заполненности ненулевыми элементами необходим для того, чтобы стандартный способ сложения матриц превосходил по скорости способ обработки разреженных. В любом случае разреженные матрицы превосходят обычные в том смысле, что они занимают довольно мало места при разреженности более 50%. Однако при разреженности менее 50% они занимают даже больше памяти. При 50% и более заполненности лучше всего использовать обычные матрицы. Разреженные матрицы лучше всего использовать при крайне больших и

разреженных матрицах, так как в этом случае работают они быстрее и эффективнее обычных.