|  |  |
| --- | --- |
|  | **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 ПО ПРЕДМЕТУ "ТИПЫ И СТРУКТУРЫ ДАННЫХ" №3**

Студент\_Золотухин Алексей Вячеславович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*фамилия, имя, отчество*

Группа\_\_\_\_\_\_ИУ7-34Б\_\_\_\_\_\_\_\_\_

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_Золотухин А. В.\_\_\_\_

*подпись, дата фамилия, и.о.*

Принял \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_Силантьева А. В. \_

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Москва, 2021**

**Цель работы**

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.

**Задание** 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;

- связный список 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 программы**

Данная программа предназначена для работы с разреженными матрицами. Она представляет собой консольное приложение с последовательными операциями:

1) Демонстрация алгоритма сложения двух матриц, хранящихся в разряженном виде;

2) Анализ быстродействия и затрат памяти разреженного способа хранения по сравнению с классическим способом.

**Руководство по эксплуатации**

Программа принимает на вход две матрицы. Формат ввода указывается пользователем. И выводит третью — сумму введённых, а также время, за которое происходило сложение матриц.

Также, если программе передать имя файл как аргумент командной строки, то она прочитает данные из него. Матрицы вводятся в следующих форматах:

* Сначала вводятся размеры матрицы;
* После этого вводятся непосредственно значения матрицы.

Или

* Сначала вводятся размеры матрицы;
* После вводятся элементы с указанием строки и столбца («строка» «столбец» «значение элемента»);
* Ввод элементов продолжается до некорректного элемента.

**Примечания**

Интерфейс программы прост и понятен.

В случае некорректных действий выводится сообщение об ошибке, и операция не выполняется.

Элементами матрицы являются исключительно целые числа.

**Аварийные ситуации**

В аварийных ситуациях происходит вывод сообщения пользователю и/или выход из программы.

Некорректный ввод матрицы

**Структуры данных**

Для реализации данной задачи были создана структура данных "matrix\_t" и "sparse\_t".

typedef struct

{

int \*\*m;

int \*mp;

size\_t r;

size\_t c;

} matrix\_t;

typedef struct

{

int \*a;

size\_t an;

size\_t \*ia;

size\_t \*ja ;

size\_t r;

size\_t c;

} sparse\_t;

**Описание алгоритма**

1) Ввод матрицы в разреженном виде. После ввода двух размеров происходит проверка их корректности. После чего создаются соответствующие векторы. Вводятся элементы векторов. Структура создаётся. После этого происходит проверка корректности введённой матрицы с точки зрения математики.

2) Сложение двух матриц в разреженном виде. По векторам ненулевых значений слагаемых бежим двумя указателями (по столбцам). На каждом шагу определяем, номера строк рассматриваемых элементов. Если они совпадают, то добавляем в вектор значений результата новое число — их сумму. Номер строки для нового элемента равен номерам строк слагаемых. В векторе, описывающем столбцы, изменяем элемент, описывающий следующий столбец на 1. (Изначально вектор состоит из нулей). Если строки слагаемых не совпали, то в результат добавляются оба слагаемых в порядке увеличения строки.

**Анализ эффективности разрежённого способа хранения (тактах)**

1) 50\*50 элементов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Заполненность | Время классического вида | Время разреженного вида | Память  классического вида | Память разреженного вида |
| 10% | 8 | 9 | 10 000 | 2204 |
| 11% | 9 | 9 | 10000 | 2404 |
| 30% | 9 | 18 | 10 000 | 6204 |
| 50% | 10 | 24 | 10 000 | 10204 |
| 70% | 9 | 27 | 10 000 | 14204 |
| 90% | 9 | 22 | 10 000 | 18204 |

2)100\*100 элементов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Заполненность | Время классического вида | Время разреженного вида | Память классического вида | Память разреженного вида |
| 10% | 33 | 25 | 40000 | 8404 |
| 15 | 33 | 33 | 40000 | 12404 |
| 30% | 33 | 63 | 40000 | 24404 |
| 50% | 33 | 100 | 40000 | 40404 |
| 70% | 33 | 115 | 40000 | 56404 |
| 90% | 33 | 87 | 40000 | 72404 |

4) 500\*500 элементов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Заполненность | Время классического вида | Время разреженного вида | Память классического вида | Память разреженного вида |
| 10% | 779 | 592 | 250000 | 52004 |
| 14% | 815 | 834 | 250000 | 72004 |
| 30% | 807 | 1667 | 250000 | 152004 |
| 50% | 805 | 2557 | 250000 | 252004 |
| 70% | 787 | 2830 | 250000 | 352004 |
| 90% | 794 | 2086 | 250000 | 452004 |

5) 1000\*1000 элементов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Заполненность | Время классического вида | Время разреженного вида | Память классического вида | Память разреженного вида |
| 10% | 2739 | 2269 | 1000 000 | 204004 |
| 13% | 2740 | 2851 | 1000000 | 264004 |
| 30% | 2840 | 6455 | 1000000 | 604004 |
| 50% | 2755 | 9851 | 1000000 | 1004004 |
| 70% | 2769 | 10827 | 1000000 | 1404004 |
| 90% | 2774 | 7861 | 1000000 | 1804004 |

**Полученные результаты позволяют сделать следующие выводы:**

1) Разреженный вид становится эффективней по памяти при заполненности менее чем 50%;

2) Разреженный вид становится эффективней по времени при 15% заполненности.

3) Чем больше матрица, тем сильней проявляется эффективность разряженного метода хранения (при соответствующих заполненностях).

Таким образом:

1) Программист должен использовать разреженный способ хранения матриц при заполненности менее 15% или при 50% если память важнее, чем скорость выполнения.

2) В остальных случаях выбор метода решения остаётся за программистом и зависит от доступных ресурсов и поставленных целей.

**Контрольные вопросы**

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

Разреженная матрица – матрица с преимущественно нулевыми элементами. Число ненулевых элементов в матрице порядка n может выражаться как n^(1+g), где g < 1. Значения g лежат в интервале 0.2 ... 0.5.

Существуют различные методы хранения элементов матрицы в памяти.

Например, линейный связный список, т.е. последовательность ячеек, связанных в определенном порядке. Каждая ячейка списка содержит элемент списка и указатель на положение следующей ячейки.

Можно хранить матрицу, используя кольцевой связный список, двунаправленные стеки и очереди.

Существует диагональная схема хранения симметричных матриц, а также связные схемы разреженного хранения.

Связная схема хранения матриц, предложенная Кнутом, предлагает хранить в массиве (например, в AN) в произвольном порядке сами элементы, индексы строк и столбцов соответствующих элементов (например, в массивах I и J), номер (из массива AN) следующего ненулевого элемента, расположенного в матрице по строке (NR) и по столбцу (NC), а также номера элементов, с которых начинается строка (указатели для входа в строку – JR) и номера элементов, с которых начинается столбец (указатели для входа в столбец - JC).

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

Для хранения обычной матрицы: N \* M \* sizeof(elem). Память под разреженную матрицу выделяется в зависимости от схемы хранения. Память выделяется по мере наполнения ненулевыми элементами.

**3. Каков принцип обработки разреженной матрицы?**

Обработка разреженной матрицы предполагает работу только с ненулевыми элементами (таким образом, количество операций пропорционально количеству ненулевых элементов).

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

Разреженность матрицы следует учитывать только в том случае, если из этого можно извлечь выгоду за счёт игнорирования нулевых элементов.

При достижении определенного процента наполнения ненулевыми элементами происходит значительное падение эффективности по времени.

**Вывод**

Использование разреженной матрицы оправдано при большой размерности и небольшом заполнении. В таком случае можно получить значительный выигрыш по памяти и по времени. Однако в этом случае значительно усложняется алгоритм обработки матриц.

Время выполнения стандартного алгоритма почти линейно зависит от размерности матрицы. Этот алгоритм эффективен при высоком заполнении матрицы. Однако при заполнении матрицы менее 20-15% разреженный алгоритм позволяет добиться более высокой скорости работы при использовании меньшего количества памяти.

Сложность разреженного алгоритма связана с невозможностью доступа к элементам по индексам.

При заполнении матрицы более, чем на 15-20%, и размерности менее 50\*50, безусловно, целесообразнее использовать стандартные способы обработки и хранения матриц.