Лабораторная работа №3 по дисциплине

“Типы и структуры данных”

Сорокин Антон ИУ7-32Б

Номер по списку – 24

Вариант 1

***Условие 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;

- связный список IA, в элементе Nk которого находится номер компонент

в A и JA, с которых начинается описание строки 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.

***Техническое задание***

*Исходные данные*

На вход программе могут быть поданы файлы с матрицами как в стандартном виде, так и в векторном. Если файлы программы на вход не подаются, пользователь самостоятельно вводит матрицы. Если размер матрицы не превышает 50, то пользователю предлагается ввести матрицу в стандартном. В противном случае будет предложено сгенерировать случайную матрицу с указанным числом ненулевых элементов или векторно ввести матрицу.

Формат записи матрицы в файле (стандартный вид): кол-во строк, кол-во столбцов, элементы.

Формат записи матрицы в файле (векторный вид): кол-во строк, кол-во столбцов, кол-во ненулевых элементов, 3 вектора.

*Результат*

Результат сложения двух матриц в векторном виде и, если размер не превышает 50, в стандартном виде.

*Описание задачи*

Программа выполняет сложение двух разреженных матриц, которые хранятся в виде трёх векторов, а также матриц, которые хранятся в стандартном виде.

*Аварийное завершение работы программы*

Программа завершается аварийно при:

* ошибке открытия файла;
* вводе некорректных данных матрицы (символы, вещественные числа, нули);
* при нехватке вводимых данных;
* вводе недопустимых размерностей и кол-ва ненулевых;
* при разных размерах двух матриц.

***Обращение к программе***

Исполняемый файл app.exe создается путем автоматической сборки проекта с помощью файла makefile. Для выполнения работы следует запустить данный исполняемый файл с указанием или без указания файлов с данными в качестве аргументов.

***Алгоритм***

* Проинициализировать исходные матрицы и результирующую;
* Чтение матрицы в стандартном виде:
  + прочитать размер матрицы, при неположительных значениях вернуть ошибку;
  + выделить память под вектора: под вектора A и JA выделить память под 20 элементов, под вектор IA – по количеству строк; при невозможности выделить память, вернуть код ошибки;
  + прочитать в двойной цикле по строкам и столбцам элементы матрицы, в начале обработки каждой строки i элементу IA[i] присвоить -1 :
    - если элемент символ или вещественное, вернуть код ошибки;
    - если элемент равен 0, пропустить;
    - если элемент не равен 0:
      * перевыделить память под вектора, если кол-во ненулевых элементов кратно 20 – увеличить ещё на 20;
      * записать элемент в конец вектора A;
      * записать текущий индекс столбца в конец вектора JA;
      * если элемент IA[i] = -1, то есть ещё не обозначен первый элемент в строке, присвоить элементу IA[i] текущее кол-во ненулевых элементов;
      * увеличить счётчик кол-ва ненулевых элементов;
    - в конце обработки строки, если элемент IA[i] остался равным -1, присвоить этому элементу текущее кол-во ненулевых элементов.
* Чтение матрицы в векторном виде:
  + прочитать размер матрицы и количество ненулевых элементов, при неположительных значениях размера, при отрицательном кол-ве ненулевых или если оно превосходит размер, вернуть ошибку;
  + последовательно ввести значения всех полей добавляемой записи, в случае некорректных данных, вернуть код ошибки;
  + прочитать последовательно три вектора:
    - если на ввод вектора A введён 0, вернуть код ошибки;
    - если на ввод вектора JA введены отрицательные значения или значения, большие кол-ва столбцов, вернуть код ошибки;
    - если на ввод вектора IA введены отрицательные значения или значения, большие кол-ва ненулевых, вернуть код ошибки;
    - если при вводе вектора IA текущий введённый меньше предыдущего, вернуть код ошибки.
* Вывод матрицы, которая хранится в виде трёх векторов, в стандартном виде:
  + в цикле по количеству строк:
    - определить последний элемент в строке;
    - в цикле от первого элемента в строке до последнего:
      * до текущего элемента вывести нули;
      * вывести текущий элемент;
    - если конец строки не достигнут, вывести недостающее количество нулей.
* Сложение в векторном виде:
  + определить в каждой из матриц первую ненулевую строку;
  + элементы вектора IA результирующей матрицы проинициализировать -1;
  + в цикле пока не будут пройдены все ненулевые элементы хотя бы в одной из матриц:
    - если текущие строки в матрицах равны и совпадают значения векторов JA текущих элементов векторов A:
      * если сумма элементов не равна 0:
        + записать её в конец вектора A результата;
        + текущее значение вектора JA любой матрицы записать в конец вектора JA результата;
        + если элемента вектора IA текущей строки результата равен -1, присвоить ему значение текущего кол-ва ненулевых элементов результата;
      * увеличить счётчики текущих элементов исходных матриц;
    - если текущая позиция в первой матрице (по столбцу и по строке) меньше текущей позиции во второй:
      * записать текущий элемент первой матрицы в конец вектора A результата;
      * текущее значение вектора JA первой матрицы записать в конец вектора JA результата;
      * если элемента вектора IA текущей строки результата равен -1, присвоить ему значение текущего кол-ва ненулевых элементов результата;
      * увеличить счётчик текущего элемента первой матрицы;
    - если текущая позиция во второй матрице (по столбцу и по строке) меньше текущей позиции в первой:
      * записать текущий элемент второй матрицы в конец вектора A результата;
      * текущее значение вектора JA второй матрицы записать в конец вектора JA результата;
      * если элемента вектора IA текущей строки результата равен -1, присвоить ему значение текущего кол-ва ненулевых элементов результата;
      * увеличить счётчик текущего элемента второй матрицы;
    - если достигнут конец строки в матрицах, найти следующую ненулевую строку;
  + пока не будет достигнут конец вектора A матрицы, в которой он не был достигнут предыдущим циклом:
    - * записать текущий элемент матрицы в конец вектора A результата;
      * текущее значение вектора JA второй матрицы записать в конец вектора JA результата;
      * если элемента вектора IA текущей строки результата равен -1, присвоить ему значение текущего кол-ва ненулевых элементов результата;
      * увеличить счётчик текущего элемента матрицы;
  + в цикле с конца вектора IA каждое неинициализированное значение (-1) заменить на значение следующего по порядку элемента; значение запоследнего элемента принять равным количеству элементов.
* Генерация случайной матрицы (на вход подаётся размер матрицы)
  + прочитать количество ненулевых элементов, при отрицательном количестве или при количестве, превышающем размер матрицы, вернуть код ошибки;
  + выделить под вектора A и JA память в размере количества ненулевых элементов (если оно равно 0, то выделить один элемент), под вектор IA – в размере количества строк;
  + заполнить нулями вектора A и JA, вектор IA - -1;
  + если количество ненулевых равно 0, закончить генерацию;
  + создать массив jset, который отвечает за то, использована ли позиция в матрице;
  + в цикле от 0 до количества ненулевых
    - пока не получим новую позицию j (проверяется jset[j]=1), генерировать случайно позицию в диапазоне от 0 до размера матрицы – 1;
  + в цикле по массиву jset:
    - если текущий элемент jset равен 1:
      * остаток от деления текущего индекса на кол-во столбцов записать в вектор JA;
      * частное от деления текущего индекса на кол-во строк записать в вектор A;
      * сгенерировать случайное значение и записать в вектор A;
      * если не определён элемент вектора IA текущей строки, определить его как текущий элемент вектора;
  + в цикле с конца вектора IA каждое неинициализированное значение (-1) заменить на значение следующего по порядку элемента; значение запоследнего элемента принять равным количеству элементов.

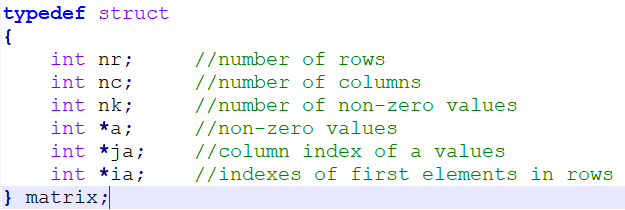
***Тестовые данные***

*Примечание: тесты для ввода аналогичны для второй матрицы*

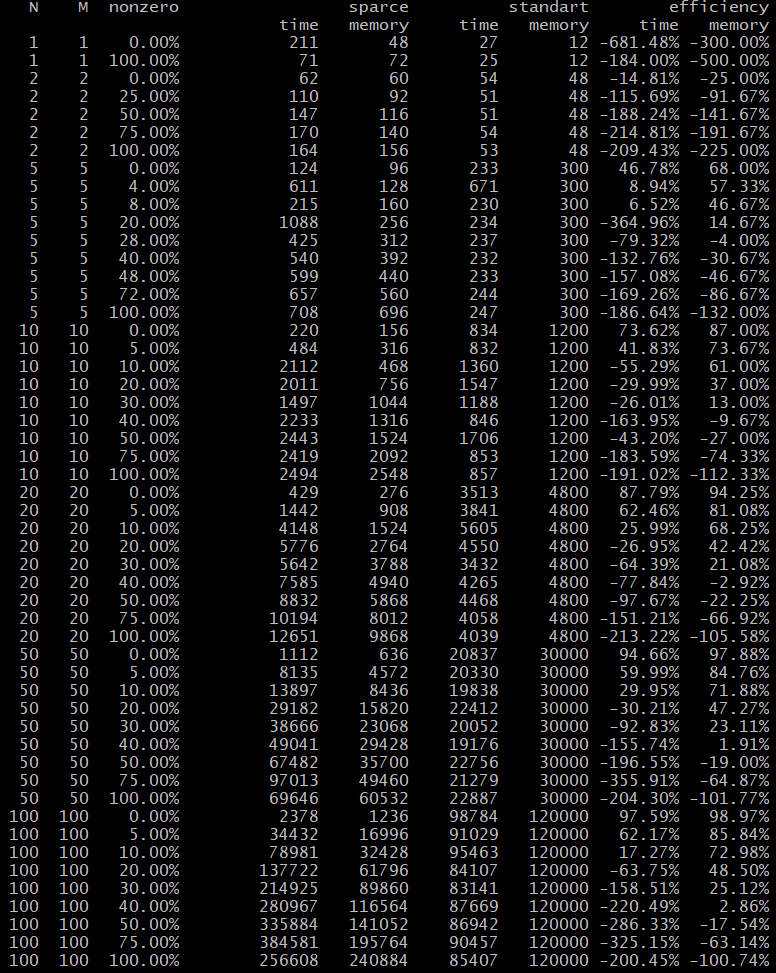
|  |  |  |  |
| --- | --- | --- | --- |
| **1ая матрица** | **2ая матрица** | **Результат** | **Класс** |
| 1 a 1 |  | Сообщение об ошибке | Символьный размер |
| -1 1 1 |  | Сообщение об ошибке | Отрицательный размер |
| 1 0 1 |  | Сообщение об ошибке | Нулевой размер |
| 1 1 a |  | Сообщение об ошибке | Символьное кол-во ненулевых |
| 1 1 -1 |  | Сообщение об ошибке | Отрицательное кол-во ненулевых |
| 2 2 2  a 1 |  | Сообщение об ошибке | Символьный элемент |
| 2 2 2  1.2 1 |  | Сообщение об ошибке | Вещественный элемент |
| 2 2 2  1 2  -1 1 |  | Сообщение об ошибке | Отрицательный индекс столбца |
| 2 2 2  1 2  a 1 |  | Сообщение об ошибке | Символьный индекс столбца |
| 2 2 2  1 2  2 1 |  | Сообщение об ошибке | Индекс, превышающий размер |
| 2 2 2  1 2  0 0  a 1 |  | Сообщение об ошибке | Символьный номер начала строки |
| 2 2 2  1 2  0 0  -1 1 |  | Сообщение об ошибке | Отрицательный номер начала строки |
| 2 2 2  1 2  0 0  1 0 |  | Сообщение об ошибке | Следующий номер меньше предшествующего |
|  |  |  |  |
| 2 2 2  1 2  0 0  1 |  | Сообщение об ошибке | Нехватка данных |
| 2 2 2  1 2  0 0  2 0 |  | Сообщение об ошибке | Номер превосходит количество ненулевых |
| 1 1 1  1  0  0 | 1 2 1  1  0  0 | Сообщение об ошибке | Несоответствие размеров |
| 1 1 1  1  0  0 | 1 1 1  2  0  0 | 1 1 1  3  0  0 | Сложение матриц 1x1 |
| 2 2 2  2 3  0 1  0 2 | 2 2 2  -2 -3  0 1  0 2 | 2 2 0 | Результат – нулевая матрица |
| 2 2 2  2 3  0 1  0 2 | 2 2 2  -2 3  0 1  0 2 | 2 2 1  3  1  0 1 | Образование новых ненулевых элементов |
| 2 2 4  1 2 3 4  0 1 0 1  0 2 | 2 2 0 | 2 2 4  1 2 3 4  0 1 0 1  0 2 | Сложение с нулевой |
| 2 2 3  1 -2 4  0 1 1  0 2 | 2 2 2  2 5  0 0  0 1 | 2 2 4  3 -2 5 4  0 1 0 1  0 2 | Произвольные матрицы |

***Внутренние структуры данных***

Для хранения матрицы используется структура из трёх векторов A, JA и IA и размерностей (строки, столбцы, количество ненулевых). Для хранения матрицы в стандартном виде используются двумерные массивы и переменные размерностей.



***Оценка эффективности***



Согласно полученным данным, сложение разреженных матриц в формате трёх векторов по времени, чем сложение матриц в стандартной форме, при плотности, меньшей 20% и при больших размерах матриц (больших 20x20). Эта эффективность падает при увеличении плотности и растёт при увеличении размера матрицы. По памяти сложение матриц в формате трёх векторов эффективнее стандартного при плотности, меньшей 50%.

***Вывод***

Хранение в векторном формате использовать эффективно использовать для разреженных матриц больших размеров, которые заполнены лишь на 20%.

***Ответы на вопросы***

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

Разреженной называется матрица, в которой количество ненулевых элементов значительно меньше размера матрицы. Отношение кол-ва ненулевых элементов к размеру в таких матрицах называют плотностью матрицы.

Разреженные матрицы, как правило, хранят в виде нескольких векторов. Один из них – это вектор ненулевых элементов. Второй задаёт индексы элементов по строкам/столбцам. Третий вектор можно задать по-разному:

1) вектор задаёт индексы элементов по столбцам/строкам;

2) вектор задаёт индекс элемента в первом векторе, с которого начинается описание стоблца/строки.

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

Под обычную матрицу память выделяется в соответствии с размером матрицы (произведение кол-ва строк на кол-во столбцов).

Под разреженную:

под вектора A и JA – по количеству ненулевых элементов

под вектор IA – по количеству строк

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

При обработке разреженной матрицы операции проводятся только с ненулевыми элементами. Нужно использовать особый алгоритм для того, чтобы определить индекс по строке и по столбцу определённого элемента, как если бы матрица была задана в стандартном виде.

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

То, какой алгоритм следует использовать при обработке матриц, зависит от её размера и её плотности заполнения. При малых размерах (<= 10) или плотности заполнения, большей 20-30%, следует использовать стандартные алгоритмы обработки матриц, так как в противном случае можно получить проигрыш в использовании памяти и в затраченном времени.