**Типы и Структуры Данных**

**Oтчет**

## *Работа № 3: Обработка разреженных матриц*

***Студент: Нгуен Фыок Санг***

***Группa: ИУ7И-36Б***

*2019*

1. **Oписание 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; - 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 умножения вектора-строки и матрицы, хранящихся в этой форме, с получением результата в той же форме. 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.

**II. Oписание ТЗ:**

1. **Исходные данные**: разреженнaя матрицa mA, вектор vB.
2. **Результат**: умножения вектора-строки и матрицы: vC = vB \* mA
3. **Задачи, реализуемой программой**:
   * 1. умножения 2 векторoв
     2. сравнение времени

**III. Описание СД:**

1. **Для ввода:** 
   * Использовать разреженнy матрицy размером 1000 \* 1000 \* 3 mA
   * использовать 2 массива размером 1000 для вектора vB
2. **Для результатa:**
   * **использовать массив размером 1000 для вектора vC**

***Структура разреженной матрицы***

typedef struct mat1

{

double \*AN;

int \*JA;

node\_t IA;

int nia;

} mat1\_t;

где IA - список со структурой узла следующим образом:

struct node

{

int value;

struct node \*next;

};

typedef struct node \*node\_t;

структура вектора

typedef struct vector

{

double \*AN;

int \*JA;

} vector\_t ;

**IV. Oписание алгоритма:**

1. *Умножить 2* векторoв a \* b*:*
   1. Просматривается массив JA и в соответствующий элемент массива IP вписывается индекс массива JA.
   2. Просматривается массив JB. Если соответствующий элемент массива IP ненулевой, т.е. позиции элементов в векторах a и b совпадают, то вычисляются произведения ai \* bi и накапливаем в сумму произведений h до тех пор, пока не будет исчерпан массив JB.

**V. Набор тестов:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N = | 200 | | | | 500 | | | | 1000 | | | |
| % | 1 | 5 | 50 | 100 | 1 | 5 | 50 | 100 | 1 | 5 | 50 | 100 |
| Time1 | 0.1 | 0.7 | 8 | 16 | 3 | 11 | 112 | 224 | 18 | 89 | 976 | 1789 |
| Time2 | 14 | 14 | 15 | 15 | 222 | 221 | 221 | 220 | 1768 | 1760 | 1953 | 1784 |

\*Time1: Время для разреженной матрицы \*Time2: Время для обычной матрицы (ms)

**VI. Ответы на вопросы:**

* ***Что такое разреженная матрица, какие схемы хранения таких матриц Вы знаете?*** 
  + Разрежённая матрица — это [матрица](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D1%80%D0%B8%D1%86%D0%B0_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) с преимущественно нулевыми элементами
  + схемы хранения
    - Линейный связный список: кольцевой связный список, двунаправленные стеки и очереди.
    - Связные схемы разреженного хранения.
* ***Каким образом и сколько памяти выделяется под хранение разреженной и обычной матрицы?***
  + ***Под разреженной матрицы mA(n\*n)***
    - ***Хранение:*** 
      * Значения ненулевых элементов хранятся в массиве AN
      * Соответствующие им столбцовые индексы - в массиве JA.
      * Кроме того, используется массив указателей, например, IA, отмечающих позиции AN и JA, с которых начинаются описание очередной строки.
    - ***Памяти выделяется***

|  |  |  |
| --- | --- | --- |
|  |  |  |
| AN | k \* sizeof(element in A - float) | K: Количество ненулевых элементов |
| JA | k \* sizeof(int) |
| IA (список) | (N + 1) \* sizeof(1 node) | Sizeof(1 node) = 16 |
|  | k \* 4 \* 2 + (n + 1) \* 16 | = 4 \* (2k + 4n + 4) |

* + ***Под хранение обычной матрицы***
    - Использовать двумерные массивы для хранения всех элементов
    - Памяти выделяется

|  |  |  |
| --- | --- | --- |
| A(n \* n) | N \*n \* sizeof(element - float) | = 4 \* (n \* n) |

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

Разреженный вектор - это разреженная матрица-строка или матрица-столбец, поэтому рассмотрим скалярное умножение разреженных векторов (как частный случай работы с матрицей) с использованием так называемого расширенного массива указателей IP. При вычислении стандартным способом нужно N^2 просмотров массива. Для сокращения алгебраических операций удобно во время работы хранить расширенный (по размерности массивов a и b) массив указателей IP (его начальное состояние - нулевое). Этот массив заполняется путем одного просмотра массива JA

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

Матрицы могут быть достаточно большие (больше **1010-20** элементов), а число ненулевых элементов при матрице порядка n может выражаться как **n1+g**, g = 0.2..0.5

Зависит от количества ненулевых элементов, относительно размера матрицы