Отчет по лабораторной работе № 3 по дисциплине

“Типы и структуры данных”

Сушина А.Д. ИУ7-31б

Работа № 3

Обработка разреженных матриц

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.

# Описание условия задачи

Разработать программу умножения 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 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 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.

# Описание ТЗ

### Общая концепция системы

Программа предназначена для умножения строки на матрицу. Матрицу можно ввести самостоятельно, считать из файла или создать рандомно.

Матрицы умножаются с помощью алгоритма умножения разреженных матриц или с помощью обычного умножения. Также программы выводит данные о времени выполнения работы.

### Требования к функциональным характеристикам

Программа должна выполнять следующие функции:

* Производить умножение строки на матрицу в разреженном виде
* Производить умножение строки на матрицу в обычном виде
* Замерять время выполнения операций и выводить информацию о нем
* Предоставлять пользователю возможность ввести данные самостоятельно, считать из файла или сгенерировать автоматически.
* Программа должна сообщать о выполняемой функции
* При выводе результата должны быть представлены пояснения к нему
* Программа должна сообщать пользователю диапазон вводимых данных
* Программа должна сообщать об ошибке при вводе некорректных данных
* Программа должна сообщать об ошибке, если невозможно совершить умножение.

На **вход** программа получает номер действия. При необходимости на вход так же поступают длина строки, количество ее ненулевых элементов и сами элементы или размеры матрицы, количество ее ненулевых элементов и сами элементы.. А также имена файлов.

**Выход** должен быть представлен в виде строки. Также программа должна вывести данные о времени выполнения алгоритма.

Программа должна выдавать корректный ответ при вводе любых данных. Если произошла ошибка ввода, программа должна сообщить об этом.

Программа завершается после любой ошибки или при получении результата.

## Аварийные ситуации

1. Ввод строки вместо числа.  
   Программа выведет сообщение “Input error”.
2. Ввод несуществующего пункта меню  
   Программа выведет сообщение “Input error”.
3. Ввод значений, не удовлетворяющих условиям.

(Индекс больше введенных ранее n и m, количество ненулевых элементов больше количества возможных элементов матрицы, процент заполнения матрицы превышающий 100 и тд.)

Программа выведет сообщение “Input error”.

1. Невозможность открыть файл   
   Программа выведет сообщение “File can’t be open”
2. Невозможно считать данные из файла   
   Программа выведет сообщение “input error in file ”

## Способ обращения к программе

Программа представляет собой файл app.exe. Запускается в консоли. для запуска достаточно команды ./app.exe. Если файл отсутствует можно собрать его с помощью утилиты make.

# Описание структур данных

struct j - список указателей на первый элемент столбца

{

int i; - номер элемента в массиве значений

struct j \*next; - указатель на следующий элемент

};

int \*\*matrix; - матрица в обычном представлении(динамическая)

int n, - количество столбцов матрицы

m, - количество строк матрицы

count, - количество ненулевых элементов матрицы

n1, - количество элементов строки

count1; - количество ненулевых элементов строки

int \*a, - массив ненулевых элементов матрицы(динамический)

\*ai, - массив индексов строк ненулевых элементов матрицы (динам)

\*b, - массив ненулевых элементов строки(динам)

\*bi; - массив индексов ненулевых элементов строки(динам)

struct j \*head; - указатель на начало списка указателей на начало столбцов

int action; - номер действия

char filename[30 + 1]; - имя файла

char buf[30+1]; - буфер для хранения лишних данных и обработки ошибок

int rc; - код ошибки

int \*line; - массив (строка)(динам)

int \*c; - строка с ответом (динам)

long int t1, t2; - время выполнения алгоритма

# Описание алгоритма

* Ввод строки. Выбор действия

1. Самостоятельный ввод   
   Ввести n  
   Ввести count   
   Пока i < count   
    считать элемент  
    записать в строку   
    i++
2. Ввод из файла   
   Ввести имя файла  
   Считать n, count   
   Считать элементы
3. Сгенерировать автоматически   
   Ввести n

Ввести процент заполнения

Посчитать count   
Пока i < count   
 создать элемент  
 записать в строку   
 i++

* Ввод матрицы. Выбор действия

1. Самостоятельный ввод   
   Ввести n  
   Ввести m   
   Ввести count   
   Пока i < count   
    считать элемент  
    записать в матрицу   
    i++
2. Ввод из файла   
   Ввести имя файла  
   Считать n, m, count   
   Считать элементы
3. Сгенерировать автоматически   
   Ввести n, m

Ввести процент заполнения

Посчитать count   
Пока i < count   
 создать элемент  
 записать в матрицу   
 i++

* Напечатать строку
* Напечатать матрицу
* Перенести матрицу в другой способ хранения

1. Пока не конец матрицы
2. Пока не конец строки
3. Если элемент не нулевой

Записать значение в a   
Записать номер строки в ai

Если элемент первый в столбце   
 запомнить индекс  
запомнить индекс последнего

* Перенести строку в другой способ хранения

1. Пока не конец строки
2. Если элемент не нулевой

Записать значение в b   
Записать номер строки в bi

* Перемножить матрицы в разреженном виде
* Перемножить матрицы в обычном виде
* Вывести ответ

# Тесты

1. Несуществующий пункт меню  
   Вход:

Способ: 4

Вывод: Input error

1. Некорректный ввод n (n < 0)

Вход:

n: 0

Вывод: Input error

1. Некорректный ввод count(больше m\*n)

Вход:

n: 5

m: 5

count: 30

Вывод: Input error

1. ВВод процентов больше 100

Вход:

p: 100

Вывод: Input error

1. Ввод некорректного имени файла

Вход:

filename : 12\\\32u

Вывод: file can’t be open

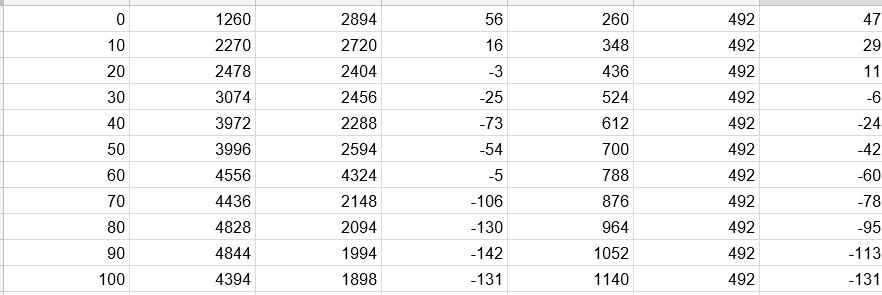
1. Невозможно выполнить умножение   
   Вход:  
   n1: 5  
   n2: 10  
   m2: 10  
   Выход:  
   Can’t make multiply!
2. Возможно выполнить умножение  
   Вход:  
   1 0 0 5 0  
     
   1 0 0 0 0   
   0 0 0 0 0   
   0 0 0 0 0   
   0 0 7 0 0  
   0 0 0 0 0  
     
   Выход:   
   1 0 35 0 0

# Оценка эффективности

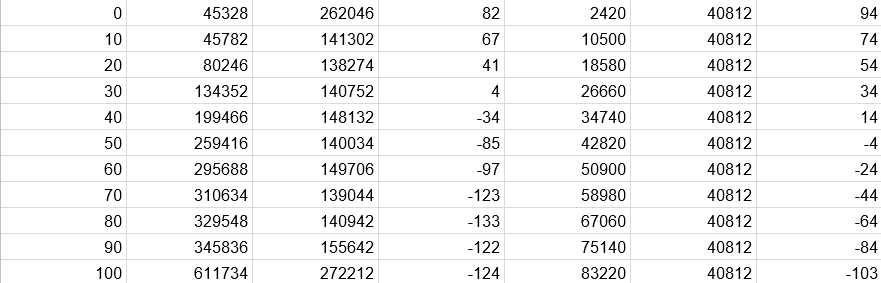
Строка 100 элементов

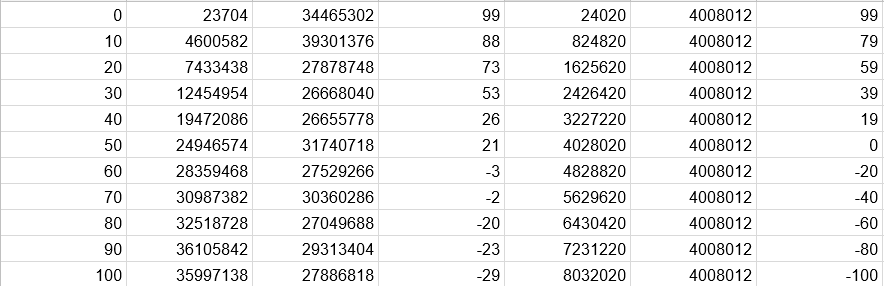
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Процент заполнения | Умножение в разреженном виде t, такты | Умножение в обычном виде  t, такты | %  время | Память в разреженном виде  байты | память в обычном виде  байты | %  память |

10х10



100х100

1000х1000



Метод умножения матриц в разреженном виде выгоднее, при проценте заполнения матрицы < 30%. По памяти этот метод эффективнее при заполнении матрицы до 40%.

# Выводы по проделанной работе

В результате работы написана программа, умножающая строку на матрицу. Реализовано умножение двумя способами: в разреженном виде и в обычном представлении.

Для матрицы 10х10 алгоритм работает быстрее при заполненности меньше 10% и эффективнее по памяти при заполненности до 20%.

Для матрицы 100х100 алгоритм работает быстрее при заполненности меньше 30% и эффективнее по памяти при заполненности до 40%.

Для матрицы 1000х1000 алгоритм работает эффективнее при заполненности до 40% и эффективнее по памяти при заполненности до 50%.

Таким образом, в зависимости от размерности матрицы меняется процент заполнения, при переходе через который метод перестает быть эффективным.   
Чем матрица больше, тем больше такой процент заполнения.

# Ответы на вопросы

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

Разреженная матрица - это матрица, которая содержит большое количество нулей и небольшое количество значащих элементов.

Можно хранить матрицу как связный список.

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

Можно хранить матрицу как три массива: массив элементов и два массив аиндексов этого элемента.

Можно хранить матрицу как два массива(элемент и один из индексов) и список связывающий начало каждого столбца/строки.

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

В обычной матрице хранится каждый элемент.

В разреженной матрице хранятся только ненулевые элементы, один из их индексов и список начала каждого столбца/строки.

3. Каков принцип обработки разреженной матрицы?

Разреженная матрица хранится в памяти как три массива. Хранятся только ненулевые элементы, номер строки/столбца и номер с которого начинается описание столбца/строки в матрице. При обработке матрицы обращение происходит только к ненулевым элементам. За счет этого тратится меньше времени для получения результата.

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

Эффективнее применять стандартные алгоритмы, когда матрицы содержат большое количество ненулевых значений. Граничный процент заполнения, при превышении которого метод умножения разреженных матриц становится неэффективным, определяется в зависимости от самого метода, а также способа хранения разреженной матрицы и размерности этой матрицы. В данной лабораторной это примерно 30%.