МГТУ им. Н. Э. Баумана

**Дисциплина типы и структуры данных**

**Лабораторный практикум №3**

**по теме: «Обработка разреженных матриц»**

**Вариант №1**

Работу выполнил:

студент группы ИУ7-35

Платонова Ольга

Работу проверил:

Москва, 2019 г.

***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.

***Условие задачи***

Разработать программу сложения разреженных матриц. Предусмотреть возможность ввода данных, как с клавиатуры, так и использования заранее подготовленных данных. Матрицы хранятся и выводятся в форме трех объектов. Сравнить эффективность стандартных алгоритмов обработки с алгоритмами обработки разреженных матриц при различной степени разреженности и различной размерности матриц.

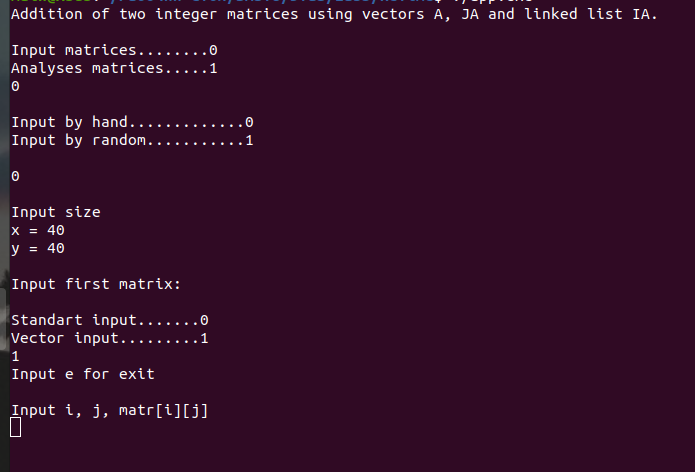
***Техническое задание***

1. Смоделировать операцию сложения двух матриц, хранящихся в виде динамического массива и в форме трех объектов: A, JA, IA.

2. Произвести операцию сложения, применяя стандартный алгоритм.

3. Сравнить затраченные время и память при использовании этих двух алгоритмов.

*Входные данные:*



Пользователь может выбрать

1. Input matrices........0 (Ввод матрицы)

Analyses matrices.....1 (Анализ методов по памяти, времени)

2. Input by hand.............0 (Задание матрицы с клавиатуры)

Input by random.........1 (Задание матрицы функцией random())

3. Input size

x = (количество строк)

y = (количество столбцов)

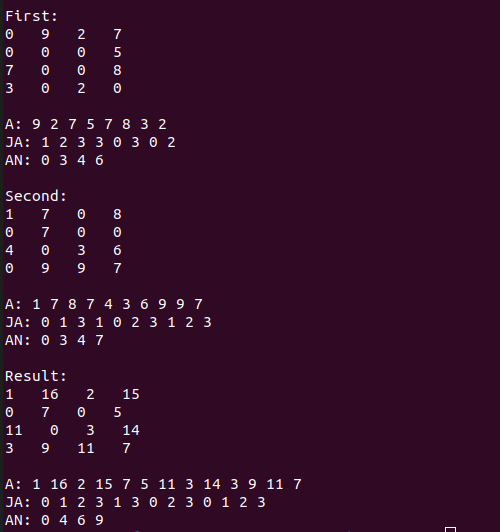
Pocentage = (процент заполненности матрицы, т. е. Число ненулевых элементов)

4. Standart input.......0 (построчное заполнение матрицы)

Vector input......….1 (векторное заполнение матрицы)

5. Input i, j, matr[i][j] (задание индексов и значения элемента)

Значение элемента и размеры матриц — целые числа

*Выходные данные:*

В том случае, когда размеры матрицы небольшие (до 10), на экран выводится стандартное представление результирующей матрицы, вектор JA и список AN, содержащие описания ненулевых элементов. Если размеры превышают 10, то пользователь увидит только специальную форму представления.

Программист должен рассмотреть возможный формат ввода данных, ошибки, связанные с ошибочным вводом, ошибочным выделением памяти. Необходимо реализовать форму хранения разреженных матриц. Также необходимо рассмотреть аварийные ситуации, включающие неверный ввод данных, невозможность выделения памяти.

***Структура данных***

Матрицы хранятся в виде динамического массива.

int \*matr1, \*matr2, \*matr3;

Вектора JA представлен в виде динамического вектора, содержащего номера ненулевых столбцов матрицы.

int \*JA1, \*JA2, \*JA3;

Вектор IA представляет собой односвязный список, содержащий номер компонент в матрице и в векторе JA.

struct IA \*IA1, \*IA2, \*IA3;

***Описание алгоритма***

1. Считать размер матриц.

2. Создать динамический массив.

3. Заполнить матрицы указанным способом (с клавиатуры или функцией random ()).

4. Конвертировать матрицы в форму представления через вектор и список.

Вектор JA: (если ненулевой элемент матрицы, то JA[i] = j)

Список IA: (если первый элемент в строке, то IA->Nk = i;

IA->j = j;

IA = IA->next;)

5. Сложить матрицы, используя стандартный алгоритм.

6. Сложить матрицы, используя особую форму хранения.

7. Вывести на экран результат сложения в двух формах.

8. Вывести на экран время обработки и объем занимаемой памяти.

9. Освободить память и очистить список.

***Тесты***

1. Неверный ввод размера матрицы (не числовой ввод или отрицательное значение).

2. Неверное заполнение матрицы (не числовые значения).

3. Сложение нулевых матриц.

4. Сложение единичных матриц.

5. Задание матриц больших размеров.

***Вопросы***

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

Матрица называется разреженной, если для нее верно , где *g* лежит в интервале от 0.2 до 0.5, n – порядок матрицы.

Такие матрицы удобно хранить в виде линейного связного списка, кольцевого связного списка, двунаправленного стека, очереди.

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

В случае разреженной матрицы необходимо выделить память для вектора JA и линейного связного списка I; память выделяется динамически, поскольку эти компоненты заполняются за один проход матрицы. В случае обычной матрицы память мажет быть выделена как статически, так и динамически, для всех элементов независимо от их значения.

3. Каков принцип обработки разреженной матрицы?

Мы обрабатываем только ненулевые элементы. В случае сложения мы изменяем как значения исходных элементов, так и вектора, хранящие информацию о них. Дополнительную матрицу создавать не требуется.

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

Стандартные алгоритмы работают эффективней, если матрица является плотной, не зависимо от порядка. Также при небольшом порядке и высокой степени разреженности стандартные алгоритмы работают быстрее, но занимают больше памяти на порядок. Так что выбор во много зависит от самого программиста.

***Вывод***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Количество элементов | Процент заполнености | Время | | | Память | |
| Стандартный метод | Метод «разреженных матриц» | Стандартный метод | | Метод «разреженных матриц» |
| 100 | 10 | 0.7716  0.6056 | | | 1200  512 | |
| 100 | 16 | 0.7528  0.7561 | | | 1200  693 | |
| 100 | 20 | 0.7463  0.7680 | | | 1200  832 | |
| 2500 | 10 | 11.7824  6.3075 | | | 30000  10896 | |
| 2500 | 20 | 10.8922  9.5735 | | | 30000  16744 | |
| 2500 | 35 | 10.7240  10.9803 | | | 30000  27394 | |
| 2500 | 50 | 10.9455  15.7762 | | | 30000  35576 | |
| 10000 | 10 | 43.1206  20.8941 | | | 120000  35240 | |
| 10000 | 20 | 43.7385  40.7485 | | | 120000  63856 | |
| 10000 | 22 | 43.4390  43.3950 | | | 120000  67312 | |
| 10000 | 50 | 43.1862  96.9753 | | | 120000  133848 | |

Заметим, что в случае стандартного метода обработки матриц, объем памяти у нас не изменяется, что также важно при выборе метода.

При большом количестве нулевых элементов (от 10 до 30%) второй метод работает быстрее и занимает меньше памяти, а уже при 35% уступает в скорости стандартному методу.