|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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)** |

**Лабораторная работа №3**

**По дисциплине «Типы и структуры данных»  
  
Тема: «Обработка разреженных матриц»**

**Вариант 9**

* + 1. Выполнил: студент группы ИУ7-36(Б)  
       Ковель Александр Денисович

**Москва, 2021**

**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**

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.

**Техническое задание**

**Общая концепция системы**

Программа предназначена для умножения строки на матрицу. Матрицу можно ввести самостоятельно или создать с помощью случайных чисел. Матрицы умножаются с помощью алгоритма умножения разреженных матриц или с помощью обычного умножения матриц и вывод данных о времени выполнения работы

**Требования к функциональным характеристикам**

Программа должна выполнять следующие функции:

* Производить умножения матриц обычным способом
* Производить умножение строки на матрицу в разреженном
* Замерить и предоставить время выполнения операция приведенных выше.
* Предоставить ввод разреженной матрицы пользователю
* Автоматическая генерация матрицы
* При вводе и выводе выводятся сообщения программы
* При ошибках ввода, программа выводит соответствующее сообщение
* Программа должна предоставлять диапазон вводимых данных

**Вход**

На вход программа получает размеры матрицы и матрицы-строки. Далее выбирается пункт меню. В первом пункте (ручной ввод) требуется ввод количества не нулевых элементов, далее ввод координат элемента и сам элемент.

**Ограничения**

Матрица ограничена 500 элементами на строках и столбцов.

**Выход**

Результат умножения в виде строки и разреженной строки и вывод по затраченному времени и памяти.

**Аварийные ситуации**

1. Ввод строки вместо числа.

Программа выведет сообщение “Input error”.

2. Ввод несуществующего пункта меню

Программа выведет сообщение “Input error”.

3. Ввод координат элемента больше размеров матрицы

Программа выведет сообщение “Input error”.

4. Ввод количества элементов матрицы строки не равно количество элементов столбцов матрицы

Программа выведет сообщение “Input error”.

**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 структуры данных**

int matrix[N][M]; - матрица на 500\*500 элементов

int n, m – кол-во столбцов и строк матрицы.

int number\_of\_nonezero – кол-во не нулевых элементов матрицы.

int vector\_len – количество элементов матрицы строки

int vector[N] – матрица-строка

int JA – массив индексов столбцов ненулевых элементов матрицы

int A – массив ненулевых элементов столбцов

int Ia – связанных список

**Описание алгоритма**

Пока не выбрана команда выход:

1. Ввод размеров матрицы и матрицы строки
2. Ввод матрицы вручную и матрицы строки
   1. Ввод количества не нулевых элементов
   2. Ввод координат элемента
   3. Ввод не нулевого элемента
3. Умножение матрицы на матрицу строку
   1. Создать разреженную матрицу
   2. Перемножить матрицу и матрицу строку (умножив не нулевой элемент, обратившись к ненулевом значению матрицы строки с помощью связанного списка, указав место с помощью массива индексов)
   3. Вывести матрицу-строку
4. Автоматический ввод
   1. Ввод процента не нулевого количества элементов
   2. Заполнение с помощью случайных элементов
5. Сравнение времени
   1. Ввести все элементы автоматически
   2. Провести 10 раз умножение с помощью разреженных матриц
   3. Провести 10 раз умножение с помощью обычного умножения

**Тесты**

|  |  |  |
| --- | --- | --- |
| Пункт | Ввод | Выход |
| До выбора пункта | 2 2 | Введите размер матрицы строки |
| До выбора пункта | 2 | Выберете пункт меню |
| 1 – ввод матрицы строки (Ввод количества не нулевых элементов) | 0 | Input error |
| 1 – ввод матрицы строки (Ввод количества не нулевых элементов) | 3 | Input error |
| 1 – ввод матрицы строки (Ввод количества не нулевых элементов) | 1 | Введите индекс |
| 1 | 0 | Введите не нулевой элемент |
| 1 | 0 | Введите ненулевой элемент |
| 1 | 1 | Введите количество не нулевых элементов матрицы |
| 1 | 1 | Введите индексы строки и столбцов |
| 1 | 0 0 | Введите не нулевой элемент |
| 1 | 7 | 7 0  0 0 |
| 1 - вывод результата |  | 7 0 |
| 2 – Введите процент не нулевых элементов | 50 | 0 17  0 -50 |

**Оценка эффективности**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Процент заполнения | Умножение в разреженном виде, мкс | Умножение в обычном виде, мкс | Память разреженном виде, байты | Память разреженном виде, байты |
| 10 | 37.799999 | 31.500000 | 71584 | 200000 |
| 30 | 29.600000 | 31.500000 | 55584 | 200000 |
| 50 | 21.000000 | 31.400000 | 39584 | 200000 |
| 90 | 4.800000 | 31.400000 | 7584 | 200000 |

|  |  |  |
| --- | --- | --- |
| Процент заполнения | %, время на сколько быстрее разреженный способ | %, память на сколько меньше разреженный способ |
| 10 | -20 | 64 |
| 30 | 6 | 72 |
| 50 | 33 | 80 |
| 90 | 85 | 96 |

**Выводы по проделанной работе**

В ходе проделанной работы была создана программа для работы с таблицей.

Таблицы ключей дают выигрыш в скорости сортировки в 30%. Сортировка таблицы ключей на 1200 байт больше по памяти сортировки всей таблицы. Это существенно уменьшает время работы программы.

Способ сортировки влияет на время сортировки. Быстрая сортировка на 90% быстрее сортировки пузырьком.

Тип данных, по которым сортируется таблица так же имеет значение. Так целочисленные поля сортируются в разы быстрее, чем строчные. При этом выигрыш по способу сортировки уменьшается.

**Ответы на вопросы**

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

Разреженная матрица - это матрица, которая содержит большое количество нуле. Можно хранить матрицу как связный список. Можно хранить матрицу как три массива: массив элементов, массивы индексов этого элемента.

Можно хранить матрицу как три массива: ненулевые элементы, столбцы ненулевых элементов и связанный список.

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. Если матрица имеет примерно 30% нулевых значений, то стоит применять не стандартные способы.