|  |  |
| --- | --- |
| 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)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **ПРОГРАММНАЯ ИНЖЕНЕРИЯ (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04** Программная инженерия

**Отчет**

|  |  |
| --- | --- |
| **По лабораторной работе №** | 3 |

**Название:**

Записи с вариантами. Обработка таблиц

**Дисциплина:** Типы и структуры данных

|  |  |  |  |
| --- | --- | --- | --- |
| Студент | ИУ7-35Б |  | А. В. Толмачев |
|  | (Группа) |  | (И.О. Фамилия) |
|  |  |  |  |

Москва, 2022

**Описание условия 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 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**

**а) Описание исходных данных**

Исходными данными являются целочисленные матрицы. Они могут быть обработаны как при вводе с клавиатуры, так и при передаче пути к файлу с матрицами.

Ограничения к исходным данным:

1. Возможна работа только с целочисленными матрицами
2. Число строк вектора не может превышать 1
3. Число столбцов вектора должно быть равно числу строк матрицы
4. Размер матриц должен задаваться парой положительных целых чисел
5. Путь к открываемому файлу должны быть указан относительно папки с исполняемым файлом
6. Максимальная допустимая длина имени файла (с учетом «.» и расширения) – 30 символов
7. Файл должен содержать вектор-строку и матрицу в следующем формате:
   1. Все числовые данные указываются через пробельные символы. Файл не может содержать символы, отличные от цифр и пробельных символов.
   2. Данные могут быть указаны как в одну строку через пробел, так и с произвольной комбинацией пробельных символов и символов перевода строк (т. е. допускается запись и в «стандартном» формате записи матриц). Это позволяет принимать данные в разных форматах, чтобы не ограничивать пользователя
   3. В связи с предыдущим пунктом отсутствуют проверки на количество чисел в файле и соответствие их размерности.
   4. Возможны два варианта представления матрицы в файле:
      1. Сначала указывается размерность вектора-строки, после перечисляются все ее элементы, выключая нулевые, согласно правилам, описанным выше. После аналогично указывается матрица.
      2. Сначала указывается размерность вектора-строки, после указывается количество вводимых ненулевых элементов. Затем указываются координаты каждого элемента и соответствующие им индексы строк и столбцов. Индексы начинаются с 1. После аналогично указывается матрица.
      3. Комбинация данных двух вариантов в одном файле недопустима
   5. Если файл содержит больше чисел, чем необходимо – лишние значения будут проигнорированы
8. При вводе вектора-строки и матрицы действуют ограничения, указанные для представления данных в файле
9. При вводе матриц координатно
10. Недопустим ввод нулевого вектора-строки или матрицы
11. Размер элемента ограничен размеров переменной типа Int в системе
12. Без успешного считывания матриц невозможно выполнять операции над ними

Выходные данные:

1. Введенные матрицы в стандартном формате
2. Введенные матрицы в сжатом формате
3. Результат умножения матриц стандартным способом
4. Результат умножения сжатых матриц
5. Результат замера эффективности алгоритмов умножения

**b) Описание задачи реализуемой программой**

Программа позволяет следующие способы ввода матриц:

1) Считать вектор и матрицу из файла

2) Ввести вектор и матрицу вручную

3) Считать матрицу из файла координатно

4) Ввести матрицу вручную координатно

Способ генерации матриц:

5) Сгенерировать случайный вектор и матрицу

Оценка эффективности:

6) Сравнить время выполнения при различном проценте заполнения матриц

После успешного чтения матриц доступно выполнять над ними следующие действия:

1) Вывести считанные матрицы

2) Выполнить умножение сжатых матриц

3) Выполнить умножение матриц стандартным способом

1. **Способ обращения к программе**

Для запуска программы необходимо из консоли вызвать исполняемый файл ./app.exe.

1. **Описание возможных аварийных ситуаций и ошибок пользователя**
2. Ошибка при открытии файла
3. Ошибка при выделении динамической памяти
4. Ошибка при перевыделении динамической памяти
5. Ошибка при чтении файла
6. Некорректный ввод данных (не соответствующий ранее указанным ограничениям)

**Описание внутренних структур данных**

Для хранения матрицы в формате сжатой разреженной строке была реализована следующая структура:

**typedef** **struct**

{

**size\_t** rows\_count;

**size\_t** columns\_count;

**size\_t** count\_elements;

**size\_t** count\_memory;

**int** \*values;

**size\_t** \*column\_index;

**node\_t** \*head;

} **sparse\_matrix\_t**;

Данная структура содержит следующие поля:

1. rows\_count – содержит число строк исходной матрицы
2. columns\_count – содержит число столбцов исходной матрицы
3. count\_elements – содержит количество ненулевых элементов матрицы
4. count\_memory – содержит количество выделенной памяти под структуру
5. values – массив ненулевых элементов матрицы
6. column\_index – массив индексов ненулевых элементов в исходной матрице
7. head – указатель на начало связанного списка, содержащего номера компонент в values, с которого начинается i-ая строка

Для реализации связанного списка использована следующая структура:

**typedef** **struct** node

{

**size\_t** index\_start\_row;

**struct** node \*next;

} **node\_t**;

Набор тестов:

**Позитивные тесты:**

1. Считывание файлов с корректными матрицами
2. Корректный ввод матриц
3. Умножение считанных матриц
4. Умножение автоматически сгенерированных матриц
5. Вывод эффективности

**Негативные тесты:**

Тесты с входными данными, не удовлетворяющими ограничениям исходных данных, записанным ранее.

Во всех случаях производился замер времени N\_REPS раз, после чего бралось среднее время, т. е. t/N\_REPS; N\_REPS = 10000;

**Временная эффективность**

Размер матрицы 10x10

|  |  |  |
| --- | --- | --- |
| Процент плотности | Время умножения сжатых матриц, микросекунд | Время умножения стандартных матриц, микросекунд |
| 1% | 0.003 | 0.004 |
| 2% | 0.003 | 0.004 |
| 3% | 0.003 | 0.004 |
| 4% | 0.003 | 0.004 |
| 5% | 0.006 | 0.004 |
| 50% | 0.03 | 0.004 |
| 100% | 1,01 | 0.002 |

**Пространственная эффективность**

|  |  |  |
| --- | --- | --- |
| Процент плотности | Объем при умножение сжатых матриц, байт | Объем при умножение стандартных матриц |
| 1% | 188 | 400 |
| 2% | 200 | 400 |
| 3% | 212 | 400 |
| 4% | 224 | 400 |
| 5% | 236 | 400 |
| 10% | 296 | 400 |
| 20% | 416 | 400 |
| 30% | 536 | 400 |
| 50% | 776 | 400 |
| 100% | 896 | 400 |

Размер матрицы 100x100

**Временная эффективность**

|  |  |  |  |
| --- | --- | --- | --- |
| Процент плотности | Время умножения сжатых матриц, микросекунд | Время транспонирования сжатой матрицы | Время умножения стандартных матриц, микросекунд |
| 1% | 10.57 | 9.46 | 18.34 |
| 2% | 10.72 | 9.38 | 18.05 |
| 3% | 11.04 | 9.89 | 18.18 |
| 4% | 11.86 | 10.47 | 18.21 |
| 5% | 12.39 | 11.14 | 18.36 |
| 10% | 15.88 | 13.61 | 18.09 |
| 15% | 18.55 | 16.02 | 18.87 |
| 16% | 19.73 | 16.97 | 18.33 |
| 50% | 43.95 | 34.09 | 18.38 |
| 100% | 78.51 | 56.39 | 18.21 |

**Пространственная эффективность**

|  |  |  |
| --- | --- | --- |
| Процент плотности | Объем при умножение сжатых матриц, байт | Объем при умножение стандартных матриц |
| 1% | 2816 | 40 000 |
| 2% | 4016 | 40 000 |
| 3% | 5216 | 40 000 |
| 4% | 6416 | 40 000 |
| 5% | 7616 | 40 000 |
| 10% | 13616 | 40 000 |
| 20% | 25616 | 40 000 |
| 30% | 37616 | 40 000 |
| 40% | 49616 | 40 000 |
| 100% | 121616 | 40 000 |

Размер матрицы 500x500

**Временная эффективность**

|  |  |  |  |
| --- | --- | --- | --- |
| Процент плотности | Время умножения сжатых матриц, микросекунд | Время транспонирования сжатой матрицы | Время умножения стандартных матриц, микросекунд |
| 1% | 193.14 | 185.29 | 467.60 |
| 2% | 208.8 | 200.80 | 498.26 |
| 3% | 225.51 | 219.00 | 477.47 |
| 4% | 238.44 | 226.08 | 470.6 |
| 5% | 257.97 | 241.21 | 471.06 |
| 10% | 358.72 | 309.01 | 486.41 |
| 15% | 473.52 | 374.26 | 465.37 |
| 16% | 556.94 | 407.83 | 462.07 |
| 20% | 623.62 | 445.75 | 471.21 |
| 30% | 927.44 | 546.66 | 467.46 |
| 50% | 1643.46 | 788.88 | 460.55 |
| 100% | 2103.58 | 1507.77 | 483.250 |

**Пространственная эффективность**

|  |  |  |
| --- | --- | --- |
| Процент плотности | Объем при умножение сжатых матриц, байт | Объем при умножение стандартных матриц |
| 1% | 38016 | 1 000 000 |
| 2% | 68016 | 1 000 000 |
| 3% | 98016 | 1 000 000 |
| 4% | 128016 | 1 000 000 |
| 5% | 158016 | 1 000 000 |
| 10% | 308 016 | 1 000 000 |
| 20% | 608 016 | 1 000 000 |
| 30% | 908 016 | 1 000 000 |
| 40% | 1 208 016 | 1 000 000 |
| 100% | 3 008 016 | 1 000 000 |

Размер матрицы 1000x1000

**Временная эффективность**

|  |  |  |  |
| --- | --- | --- | --- |
| Процент плотности | Время умножения сжатых матриц, микросекунд | Время транспонирования сжатой матрицы | Время умножения стандартных матриц, микросекунд |
| 1% | 741.91 | 716.57 | 2347.26 |
| 2% | 829.28 | 786.01 | 2282.21 |
| 3% | 872.39 | 839.28 | 2200.33 |
| 4% | 941.38 | 877.96 | 2211.73 |
| 5% | 1040.17 | 980.84 | 2316.98 |
| 10% | 1456.32 | 1218.58 | 2387.57 |
| 15% | 1909.98 | 1436.89 | 2240.98 |
| 16% | 2044.58 | 1519.26 | 2307.03 |
| 17% | 2397.68 | 1835.37 | 2357.62 |
| 20% | 2507.02 | 1872.26 | 2303.39 |
| 30% | 3696.44 | 2216.41 | 2379.33 |
| 50% | 7381.94 | 3395.44 | 2496.24 |
| 100% | 10443.54 | 7917.59 | 2411.71 |

**Пространственная эффективность**

|  |  |  |
| --- | --- | --- |
| Процент плотности | Объем при умножение сжатых матриц, байт | Объем при умножение стандартных матриц |
| 1% | 136016 | 4 000 000 |
| 2% | 256016 | 4 000 000 |
| 3% | 376016 | 4 000 000 |
| 4% | 496016 | 4 000 000 |
| 5% | 616016 | 4 000 000 |
| 10% | 1 216 016 | 4 000 000 |
| 20% | 2 416 016 | 4 000 000 |
| 30% | 3 616 016 | 4 000 000 |
| 40% | 4 816 016 | 4 000 000 |
| 100% | 12 016 016 | 4 000 000 |

**В результате замера эффективности можно сделать следующие выводы:**

1. При маленьких размерах матрицы нет как такового выигрыша по времени. Выигрыш по памяти есть до 20% плотности матрицы. Соответственно, при обработке матриц маленького размера лучше подойдет стандартный алгоритм.
2. При средних и больших размерах матрицы алгоритм обработки сжатых матриц эффективен по времени при плотности матрицы до 15–16% и эффективен по объему при плотности матрицы до 35%.

**Вывод**

В результате выполнения данной работы были рассмотрены и реализованы различные варианты хранения матриц в «стандартном» и сжатом формате.

Сжатый формат хранения более выгоден по памяти при плотности матрицы до 35%, по времени – при плотности матрицы до 15–16%. Т. е. при работе с разреженными матрицами, плотность которых составляет до 15–16%, сжатый формат хранения будет эффективен. При плотности матрицы 16–35% будет получен существенный выигрыш по памяти, однако будет проигрыш по времени. При плотности матрицы более 35% более эффективным является стандартным способ представления и обработки матриц.

В программе был реализован способ хранения сжатых матриц по строкам и алгоритм умножения матриц с их транспонированием в столбцовый формат. Это позволяло облегчить доступ к элементам столбцов и производить умножение как «строка на строку». Однако транспонирование также занимает время.

Особенно сильно чувствуетcя выигрыш по памяти при действительно больших матрицах. Например, при размере матрицы 10000 x 10000, стандартная матрица занимает 400 000 000 байт = 381.47 Мб. В свою очередь сжатое представление занимает всего 12 160 000 байт = 11,6 Мб. Таким образом имеем выигрыш по памяти в 33 раза.

Также при реализации программы неэффективно была выбрана структура данных для хранения результата умножения – вектора строки. Результат хранится в «обще й» структуре – предназначенной для хранения сжатой матрицы. В случае хранения вектора это приводит к дополнительным расходам по памяти на связный список строк, который не имеет смысла для вектора.

Таким образом, если изменить структура для хранения разреженной матрицы, убрав из нее связный список, можно получить еще больший выигрыш по памяти.

**Контрольные вопросы**

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

Разреженная матрица – матрица с преимущественно нулевыми элементами.

Существуют следующие схемы хранения: диагональная схема хранения симметричных матрицы, профильная схема хранения, схема Кнута, кольцевая KMP схема, разреженный строчный формат, разреженный столбцовый формат.

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

Для хранения обычной матрицы выделяется память под все элементы матрицы. При хранении разреженной матрицы выделяется память под хранение ненулевых элементов, а также под хранение индексов строк и столбцов.

1. Каков принцип обработки разреженной матрицы?

Основное принцип обработки разреженной матрицы – это обработка и хранение только ненулевых элементов

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

Эффективнее применять стандартные алгоритмы обработки матриц в том случае, если матрица содержит много ненулевых элементов, или если имеет довольно маленькие размеры.