|  |  |
| --- | --- |
|  | **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 Программное обеспечение ЭВМ и информационные технологии

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3**

Название предмета: Типы и структуры данных

Студент: Гриценко Алексей Михайлович

Группа: ИУ7-31Б

*2020г.*

1. **Описание условия задачи**

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 сложения двух матриц, хранящихся в этой форме, с получением результата в той же форме.
2. Произвести операцию сложения, 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.
4. **Техническое задание**
5. ***Описание исходных данных***
6. Целое число, представляющее собой номер команды:целое число в диапазоне от **0** до **6**.
7. Командно**-**зависимыеданные**:**
   * количество строк/столбцов матрицы, количество элементов матрицы, элементы матрицы в формате **“**индекс строки - индекс столбца – значение элемента**”**;
   * количество строк/столбцов матрицы, процент ненулевых элементов матрицы.
8. ***Описание результата программы***
9. Исходные и результирующая матрицы в стандартном виде или разреженном столбцовом виде.

* Количественная характеристика сравнения вариантов сложения матриц

1. ***Задача, реализуемая программой***
2. Ввод матрицы вручную
3. Генерирование матрицы случайно (по проценту заполненности).
4. Сложение матриц, используя стандартный способ хранения.
5. Сложение матриц, используя разреженный столбцовый вид хранения.
6. Вывод исходных и результирующую матрицы в стандартном виде.
7. Вывод исходных и результирующую матрицы в разреженном столбцовом виде
8. ***Обращения к программе***

Способ обращения к программе происходит через консоль.

1. ***Описание возможных аварийных ситуаций и ошибок пользователя***
2. Некорректный ввод номера команды.  
   На входе: число, большее чем 6 или меньшее, чем 0.   
   На выходе: сообщение «Введена недопустимая команда! Повторите попытку.»
3. Некорректный ввод количества строк или столбцов матрицы.  
   На входе: неположительное целое число или буква.  
   На выходе: сообщение «Введено недопустимое значение! Повторите попытку.»
4. Некорректный ввод индекса строки или столбца матрицы.  
   На входе: число , выходящее за границы луча [0;количество\_столбцов(строк)), или буква.  
   На выходе: сообщение «Введено недопустимое значение! Повторите попытку.»
5. Некорректный ввод элемента матрицы.  
   На входе: число, выходящее за границы условия, или буква.  
   На выходе: сообщение «Введено недопустимое значение! Повторите попытку.»
6. **Описание внутренних структур данных**

За стандартное хранение матрицы отвечает именованная структура **matrix\_t**, описанная как:

typedef struct

{

*type\_t* \*\*matrix;

*int* rows;

*int* columns;

} matrix*\_t***;**

Поля структуры:

* *type\_t* \*\*matrix **–** массив указателей на строки матрицы (**type\_t - int**);
* *int* rows ***–***количество строк матрицы;
* *int* columns ***–*** количество столбцов матрицы.

За хранение матрицы в разреженном столбцовом виде отвечает именованная структура **sparse\_t**, описанная как:

typedef struct

{

*type\_t* \*elems;

*int* \*row\_entry;

*int* \*col\_entry;

*int* elems\_amount;

*int* cols\_amount;

} matrix*\_t*;

Поля структуры:

* *type\_t* \*elems **–** массив элементов матрицы, заполняемый проходом по столбцам;
* *int \**row\_entry ***–***массив, каждый элемент которого равен номеру строки соответствующего элемента из **elems**;
* *int \**col\_entry ***–*** массив, каждый элемент которого указывает на индекс элемента из **elems**, с которого начинается описание столбца;
* *int* elems\_amount **–** количество элементов матрицы;
* *int* cols\_amount **–** количество столбцов матрицы.

1. **Описание алгоритма**
2. Пользователь вводит номер команды из меню.
3. Пока пользователь не введет 0 (выход из программы), ему будет предложено выполнять действия с матрицами.
4. При вводе (или генерации) матрицы, матрица сразу хранится двумя способами хранения (стандартном и разреженном столбцовом).
5. В случае выбора стандартного сложения, матрицы складываются в стандартном виде “элемент к элементу”.
6. В случае выбора разреженного столбцового сложения, действия производятся непосредственно над разреженными столбцовыми матрицами. Сравнивается каждый столбец, при этом каждый из них сначала рассматривается как массив построчных вхождений. Если в массивах нет одинаковых вхождений, то в итоговую матрицу элементы записываются по порядку, в порядке возрастания по двум массивам. Если же одинаковые вхождения есть, то соответствующие элементы складываются, после все элементы вновь записываются в порядке возрастания индексов вхождения.

**Функции, использующиеся в программе**

void classic\_sum(const matrix\_t matrix\_a, const matrix\_t matrix\_b, matrix\_t \*const matrix\_res, uint64\_t \*const ticks) - функция суммирования матриц в классическом виде, получает их сумму и количество тиков

int sdots(const sparse\_t sparse\_a, const sparse\_t sparse\_b, int \*const dots) – функция подсчета ненулевых точек

int arrays\_min(type\_t \*a\_arr, const int a\_len, type\_t \*b\_arr, const int b\_len) – функция нахождения меньшего из двух массивов

void sparse\_sum(const sparse\_t sparse\_a, const sparse\_t sparse\_b, sparse\_t \*const sparse\_res, uint64\_t \*const ticks) – функция складывания двух матриц в разряженном виде, получает их сумму и количество тиков

void gorandom(matrix\_t \*const matrix, int percent) – рандомизатор

int input\_interval(int \*const num, const int left, const int right) – функция понимания, входит ли число в итервал

int screate(sparse\_t \*const sparse, const int dots, const int cols) – функция создания разреженной матрицы

int sinput(sparse\_t \*const sparse, const matrix\_t matrix) – функция ввода разреженной матрицы

int soutput(const sparse\_t sparse) – функция вывода разреженной матрицы

int create(matrix\_t \*const matrix) – функция создания стандартной матрицы

int input(matrix\_t \*const matrix, const int dots) – функция ввода стандартной матрицы

int output(const matrix\_t \*const matrix) – функция вывода стандартной матрицы

int deletee(matrix\_t \*const matrix) – функция “удаления” стандартной матрицы

int sdelete(sparse\_t \*const sparse) – функция “удаления” разреженной матрицы

1. **Тестирование**

**Негативные тесты:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Тест | Пользовательский ввод | Результат |
| 1 | Некорректный̆ ввод команды | 7 | Введена недопустимая команда! Повторите попытку. |
| 2 | Некорректный̆ ввод количества строк | 0 | Введено недопустимое значение! Повторите попытку |
| 3 | Некорректный ввод количества строк | A | Введено недопустимое значение! Повторите попытку |
| 4 | Некорректный ввод количества столбцов | -2 | Введено недопустимое значение! Повторите попытку |
| 5 | Некорректный ввод количества столбцов | B | Введено недопустимое значение! Повторите попытку |
| 6 | Некорректный ввод индекса строки | При матрице 4х4: 4 | Введено недопустимое значение! Повторите попытку |
| 7 | Некорректный ввод индекса строки | A | Введено недопустимое значение! Повторите попытку |
| 8 | Некорректный ввод индекса столбца | При матрице 3х3: 5 | Введено недопустимое значение! Повторите попытку |
| 9 | Некорректный ввод индекса столбца | C | Введено недопустимое значение! Повторите попытку |
| 10 | Некорректный ввод элемента матрицы | D | Введено недопустимое значение! Повторите попытку |
| 11 | Некорректный ввод элемента матрицы | 2.1 | Введено недопустимое значение! Повторите попытку |

1. **Оценка эффективности**

Измерения эффективности способов сложения будут производиться в единицах измерения – тактах процессорв. Для измерения была специально написана ассемблерная функция, поэтому погрешность измерений минимальна. При записи результатов использовалось среднее количество тактов, полученное по результатам 10 измерений.

**Время сложения:**

**5% заполнения**

|  |  |  |
| --- | --- | --- |
| Размеры | Разреженная матрица | Обычная матрица |
| 10х10 | 1060 | 11900 |
| 100х100 | 33800 | 1549820 |
| 200х200 | 119080 | 6947080 |

**10% заполнения**

|  |  |  |
| --- | --- | --- |
| Размеры | Разреженная матрица | Обычная матрица |
| 10х10 | 1160 | 8860 |
| 100х100 | 66920 | 579780 |
| 200х200 | 227680 | 1590860 |

**20% заполнения**

|  |  |  |
| --- | --- | --- |
| Размеры | Разреженная матрица | Обычная матрица |
| 10х10 | 1540 | 7620 |
| 100х100 | 310940 | 388600 |
| 200х200 | 437380 | 3946040 |

**30% заполнения**

|  |  |  |
| --- | --- | --- |
| Размеры | Разреженная матрица | Обычная матрица |
| 10х10 | 7200 | 20120 |
| 100х100 | 432400 | 1293560 |
| 200х200 | 648280 | 1544520 |

**40% заполнения**

|  |  |  |
| --- | --- | --- |
| Размеры | Разреженная матрица | Обычная матрица |
| 10х10 | 11300 | 5820 |
| 100х100 | 11847640 | 388600 |
| 200х200 | 46828840 | 1608580 |

**50% заполнения**

|  |  |  |
| --- | --- | --- |
| Размеры | Разреженная матрица | Обычная матрица |
| 10х10 | 25680 | 8020 |
| 100х100 | 6765340 | 1816860 |
| 200х200 | 52933140 | 1568680 |

**100% заполнения**

|  |  |  |
| --- | --- | --- |
| Размеры | Разреженная матрица | Обычная матрица |
| 10х10 | 40380 | 6180 |
| 100х100 | 31208440 | 1293960 |
| 200х200 | 221170380 | 1763540 |

**Объём занимаемой памяти (в байтах):**

**5% заполнения**

|  |  |  |
| --- | --- | --- |
| Размеры | Разреженная матрица | Обычная матрица |
| 10х10 | 112 | 496 |
| 100х100 | 4432 | 40816 |
| 200х200 | 16832 | 161616 |

**10% заполнения**

|  |  |  |
| --- | --- | --- |
| Размеры | Разреженная матрица | Обычная матрица |
| 10х10 | 152 | 496 |
| 100х100 | 8832 | 40816 |
| 200х200 | 32832 | 161616 |

**20% заполнения**

|  |  |  |
| --- | --- | --- |
| Размеры | Разреженная матрица | Обычная матрица |
| 10х10 | 232 | 496 |
| 100х100 | 16432 | 40816 |
| 200х200 | 66456 | 161616 |

**30% заполнения**

|  |  |  |
| --- | --- | --- |
| Размеры | Разреженная матрица | Обычная матрица |
| 10х10 | 312 | 496 |
| 100х100 | 24432 | 40816 |
| 200х200 | 98336 | 161616 |

**40% заполнения**

|  |  |  |
| --- | --- | --- |
| Размеры | Разреженная матрица | Обычная матрица |
| 10х10 | 392 | 496 |
| 100х100 | 32432 | 40816 |
| 200х200 | 130664 | 161616 |

**45% заполнения**

|  |  |  |
| --- | --- | --- |
| Размеры | Разреженная матрица | Обычная матрица |
| 10х10 | 432 | 496 |
| 100х100 | 37360 | 40816 |
| 200х200 | 146304 | 161616 |

**50% заполнения**

|  |  |  |
| --- | --- | --- |
| Размеры | Разреженная матрица | Обычная матрица |
| 10х10 | 472 | 496 |
| 100х100 | 43328 | 40816 |
| 200х200 | 162584 | 161616 |

1. **Выводы по проделанной работе**

Использование алгоритмов хранения и обработки разреженных матриц в случае сложения выгодно при малом количестве ненулевых элементов, примерно до 40% заполненности матриц. Хранение матриц в разреженном столбцовом формате проигрывает по памяти практически во всех случаях, так как структура для хранения разреженных матриц довольно нагружена целочисленными полями для хранения различных компонент матриц. Помимо занимаемой памяти, разреженные матрицы гораздо дольше обрабатываются при заполненности матриц от 40%, так как требуется много времени для составления портрета матрицы.

1. **Ответы на вопросы**

**1. Что такое разреженная матрица, какие способы хранения вы знаете?**

Разреженная матрица – это матрица, содержащая большое количество нулей. Способы хранения: связная схема хранения, строчный формат, столбцовый формат, линейный связный список, кольцевой связный список, двунаправленные стеки и очереди.

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

Под обычную матрицу выделяет N \* M ячеек памяти, где N – строки, а M – столбцы. Для разреженной матрицы – зависит от способа. В случае разреженного формата, требуется 3 \* K ячеек памяти, где K – количество ненулевых элементов.

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

Алгоритмы обработки разреженных матриц предусматривают действие только с ненулевыми элементами, и, таким образом, количество операций будет пропорционально количеству ненулевых элементов.

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

Стандартные алгоритмы обработки матриц эффективнее применять при большом количестве ненулевых элементов (от 40%). Cтоит отметить, что если не так важна память, занимаемая матрицами, но важно время, то в случае сложения лучше так же воспользоваться стандартными алгоритмами сложения матриц.