|  |  |
| --- | --- |
|  | **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*

Отчет по лабораторной работе №3 по предмету

“Типы и структуры данных”

Обработка разреженных матриц.

Выполнила:

Кузнецова Анастасия

Группа:

ИУ7-31Б

Вариант 2

Проверил(а):

Барышникова Марина Юрьевна

**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**

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.

**2. Описание ТЗ:**

Описание исходных данных и результатов

*Входные данные:*

Если пользователь хочет обработать матрицы, записанные в файле, то на вход подаётся имя файла в аргументах командной строки. В файле записана размерность первой матрицы, затем размерность второй матрицы, затем элементы первой матрицы, затем элементы второй матрицы. Элементы и размерность записаны через пробел.

Ограничения для входных данных:

* файл должен существовать
* файл не должен быть пустым
* Элементы должны быть целыми числами
* Количество строк и столбцов матриц должны быть целыми числами
* Количество строк каждой матрицы должно быть от 1 до 1000 включительно
* Количество столбцов каждой матрицы должно быть от 1 до 1000 включительно
* Размерность одной матрицы должна быть равна размерности другой матрицы

Если же пользователь хочет ввести матрицу с клавиатуры вручную, то запускать программу следует без каких-либо дополнительных ключей. В этом случае он должен выбрать:

1 – Ввод по индексам, 2 – Ввод всей матрицы целиком.

В зависимости от выбора пользователя:

1 — Пользователь вводит размерность матриц, затем индексы ненулевого элемента и сам элемента

2 — Пользователь вводит размерность матриц, затем сами матрицы целиком

Ограничения для входных данных:

* Элементы должны быть целыми числами
* Количество строк и столбцов матриц должны быть целыми числами
* Количество строк каждой матрицы должно быть от 1 до 1000 включительно
* Количество столбцов каждой матрицы должно быть от 1 до 1000 включительно
* Размерность одной матрицы должна быть равна размерности другой матрицы

Дополнительные входные данные в зависимости от выбора пользователя (Пользователь вводит целое число от 0 до 5 включительно):

5. Вводится размерность матриц

Ограничения для входных данных:

* Количество строк и столбцов матриц должны быть целыми числами
* Количество строк каждой матрицы должно быть от 1 до 1000 включительно
* Количество столбцов каждой матрицы должно быть от 1 до 1000 включительно
* Размерность одной матрицы должна быть равна размерности другой матрицы

*Выходные данные:*

Программа выводит меню и в зависимости от выбора пользователя выводит:

* Вывод исходных матриц в стандартном виде
* Вывод матриц в разреженном виде
* Вывод результата сложения в разреженном виде
* Вывод результата стандартного сложения в стандартном виде
* Вывод результатов сравнения времени и памяти при работе двух алгоритмов

При любом некорректном вводе программа выводит сообщение об ошибке.

Описание задачи, реализуемой программой

Программа считывает матрицы (либо из файла, либо с клавиатуры) и выполняет задачу, запрошенную пользователем. В программе реализовано меню с выборами действий:

1. Выход
2. Просмотр исходных матриц в стандартном виде
3. Просмотр матриц в разреженном виде
4. Сложение матриц в разреженном виде
5. Стандартного сложение в стандартном виде
6. Сравнение времени и памяти при работе двух алгоритмов

Способ обращения к программе

Программу следует вызывать из командной строки с помощью команды:

./app.exe или ./app.exe input.txt

Описание возможных аварийных ситуаций и ошибок пользователей

* Файл не существует
* Файл пустой
* Некорректные данные в файле (см. 2а)
* Некорректный ввод при выборе 5 действия (см. 2а)

**3. Описание внутренних СД**

Количество строк и столбцов матриц:

int n1, n2, m1, m2

Матрицы в стандартном виде хранятся в виде массива указателей:

int \*\*matrix\_1 = NULL;

int \*\*matrix\_2 = NULL;

Результат в стандартном виде хранится в виде массива указателей:

int \*\*result;

Матрицы в разреженном виде и результат в разреженном виде хранятся в виде структуры:

typedef struct {

int \*A;

int \*IA;

int \*JA;

int amount;

} type\_sparse\_matrix;

**4. Описание алгоритма**

В зависимости от ключей запуска программы:

1. Без ключей:

1.1 Ввод размерности матрицы с клавиатуры

1.2 Программа просит выбрать пользователя 1 — Ввод по индексу, 2 — Ввод матрицы целиком

1.3 В зависимости от выбора пользователя осуществляется соответствующий ввод матрицы

2. С ключом названия файла:

2.1 Осуществляется считывание размерности матриц из файла

2.2 Осуществляется считывание элементов матрицы из файла

3. Пока выбор не 0 печать меню

4. В зависимости от выбора пользователя:

4.0 Выход

4.1 Просмотр исходных матриц в стандартном виде

4.2 Просмотр матриц в разреженном виде

4.3 Сложение матриц в разреженном виде и вывод результата

4.4 Стандартного сложение в стандартном виде и вывод результата

4.5 Сравнение времени и памяти при работе двух алгоритмов и вывод результатов

**5. Набор тестов**

Позитивные тесты:

|  |  |  |  |
| --- | --- | --- | --- |
| № | Описание | Input | Output |
| 1 | Матрицы только с положительными элементами | ./app.exe  3  3  Choice: 2 (whole matrix input)  1 2 3  0 1 0  0 0 0  3  3  Choice: 2 (whole matrix input)  0 1 0  1 0 0  0 0 1 | Выбор 3:  Result in sparse view:  Vector of values of nonzero elements:  1 1 3 1 3 1  Vector of row numbers of nonzero elements:  1 2 1 2 1 3  The numbers of the nonzero elements at which the column starts:  1 3 5  Выбор 4:  Result:  1 3 3  1 1 0  0 0 1 |
| 2 | Матрицы только с отрицательными элементами | ./app.exe  3  3  Choice: 2 (whole matrix input)  -1 -2 -3  0 -1 0  0 0 0  3  3  Choice: 2 (whole matrix input)  0 -1 0  -1 0 0  0 0 -1 | Выбор 3:  Result in sparse view:  Vector of values of nonzero elements:  -1 -1 -3 -1 -3 -1  Vector of row numbers of nonzero elements:  1 2 1 2 1 3  The numbers of the nonzero elements at which the column starts:  1 3 5  Выбор 4:  Result:  -1 -3 -3  -1 -1 0  0 0 -1 |
| 3 | Одна из матриц полностью нулевая | ./app.exe  3  3  Choice: 2 (whole matrix input)  0 0 0  0 0 0  0 0 0  3  3  Choice: 2 (whole matrix input)  0 1 0  1 0 0  0 0 1 | Выбор 3:  Result in sparse view:  Vector of values of nonzero elements:  1 1 1  Vector of row numbers of nonzero elements:  2 1 3  The numbers of the nonzero elements at which the column starts:  1 2 3  Выбор 4:  Result:  0 1 0  1 0 0  0 0 1 |
| 4 | Ввод матрицы 1000 на 1000 по индексам | ./app.exe  1000  1000  Choice: 1 (index input)  i: 100  j: 100  value: 10  i: 1  j: 1  value: a  1000  1000  Choice: 1 (index input)  i: 200  j: 200  value: 20  i: 1  j: 1  value: a | Выбор 3:  Result in sparse view:  Vector of values of nonzero elements:  10 20  Vector of row numbers of nonzero elements:  100 200  The numbers of the nonzero elements at which the column starts:  0..010..20..0 |
| 5 | Ввод из файла | ./app.exe input.txt  3 3  3 3  0 1 0  1 2 3  0 0 0  0 0 0  0 0 0  1 1 1 | Выбор 3:  Result in sparse view:  Vector of values of nonzero elements:  1 1 1 2 1 3 1  Vector of row numbers of nonzero elements:  2 3 1 2 3 2 3  The numbers of the nonzero elements at which the column starts:  1 3 6  Выбор 4:  Result:  0 1 0  1 2 3  1 1 1 |
| 6 | Первый столбец одной из матриц нулевой | ./app.exe  3  3  Choice: 2 (whole matrix input)  0 1 0  0 3 0  0 0 2  3  3  Choice: 2 (whole matrix input)  0 1 0  1 0 0  0 0 1 | Выбор 3:  Result in sparse view:  Vector of values of nonzero elements:  1 2 3 3  Vector of row numbers of nonzero elements:  2 1 2 3  The numbers of the nonzero elements at which the column starts:  1 2 4  Выбор 4:  Result:  0 2 0  1 3 0  0 0 3 |
| 7 | Нулевые столбцы посередине одной из матриц | ./app.exe  3  3  Choice: 2 (whole matrix input)  3 0 0  2 0 0  0 0 2  3  3  Choice: 2 (whole matrix input)  0 1 0  1 0 0  0 0 1 | Выбор 3:  Result in sparse view:  Vector of values of nonzero elements:  3 3 1 3  Vector of row numbers of nonzero elements:  1 2 1 3  The numbers of the nonzero elements at which the column starts:  1 3 4  Выбор 4:  Result:  3 1 0  3 0 0  0 0 3 |
| 8 | Последний столбец одной из матриц нулевой | ./app.exe  3  3  Choice: 2 (whole matrix input)  0 1 0  0 3 0  4 0 2  3  3  Choice: 2 (whole matrix input)  0 1 0  1 0 0  0 0 0 | Выбор 3:  Result in sparse view:  Vector of values of nonzero elements:  1 4 2 3 2  Vector of row numbers of nonzero elements:  2 3 1 2 3  The numbers of the nonzero elements at which the column starts:  1 3 5  Выбор 4:  Result:  0 2 0  1 3 0  4 0 2 |

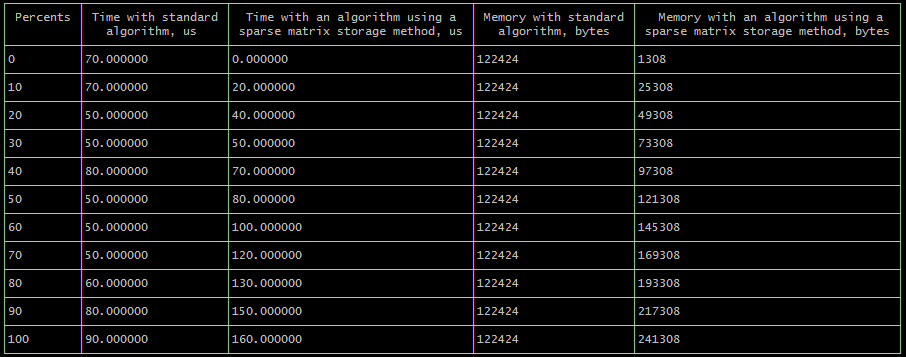
Негативные тесты:

|  |  |  |  |
| --- | --- | --- | --- |
| № | Описание | Input | Output |
| 1 | Аргументов командной строки больше допустимого | ./app.exe input.txt osvhn | Key error. |
| 2 | Файл не существует | ./app.exe ldjvh | File open error. |
| 3 | Файл пустой | ./app.exe input.txt | Input error. |
| 4 | Элемент матрицы не число (ввод с клавиатуры) | ./app.exe  3  3  Choice: 2 (whole matrix input)  0 a 0 | Input error. |
| 5 | Количество строк матрицы не число (ввод с клавиатуры) | ./app.exe  A  3 | Input error. |
| 6 | Количество столбцов матрицы не число (ввод с клавиатуры) | ./app.exe  3  a | Input error. |
| 7 | Количество столбцов матрицы меньше 1 (ввод с клавиатуры) | ./app.exe  10  0 | Input error. |
| 8 | Количество строк матрицы меньше 1 (ввод с клавиатуры) | ./app.exe  -1  10 | Input error. |
| 9 | Количество столбцов матрицы больше 1000 (ввод с клавиатуры) | ./app.exe  10  2000 | Input error. |
| 10 | Количество строк матрицы больше 1000 (ввод с клавиатуры) | ./app.exe  1001  10 | Input error. |
| 11 | Количество строк одной матрицы не равно количеству строк второй матрицы (ввод с клавиатуры) | ./app.exe  3  3  Choice: 2 (whole matrix input)  0 1 0  1 2 3  0 0 0  4  3 | Incorrect dimension of matrices |
| 12 | Количество столбцов одной матрицы не равно количеству столбцов второй матрицы (ввод с клавиатуры) | ./app.exe  3  3  Choice: 2 (whole matrix input)  0 1 0  1 2 3  0 0 0  3  5 | Incorrect dimension of matrices |
| 13 | Количество строк матрицы не число (ввод из файла) | ./app.exe input.txt  A 3 | Input error. |
| 14 | Количество столбцов матрицы не число (ввод из файла) | ./app.exe input.txt  3 a | Input error. |
| 15 | Количество столбцов матрицы меньше 1 (ввод из файла) | ./app.exe input.txt  10 0 | Input error. |
| 16 | Количество строк матрицы меньше 1 (ввод из файла) | ./app.exe input.txt  -1 10 | Input error. |
| 17 | Количество столбцов матрицы больше 1000 (ввод из файла) | ./app.exe input.txt  10 2000 | Input error. |
| 18 | Количество строк матрицы больше 1000 (ввод из файла) | ./app.exe input.txt  1001 10 | Input error. |
| 19 | Элемент матрицы не число (ввод из файла) | ./app.exe input.txt  3 3  3 3  0 a 0  1 2 3  0 0 0  0 0 0  0 0 0  0 0 0 | Input error. |
| 20 | Количество строк одной матрицы не равно количеству строк второй матрицы (ввод из файла) | ./app.exe input.txt  3 3  4 3 | Incorrect dimension of matrices |
| 21 | Количество столбцов одной матрицы не равно количеству столбцов второй матрицы (ввод из файла) | ./app.exe input.txt  3 3  3 5 | Incorrect dimension of matrices |

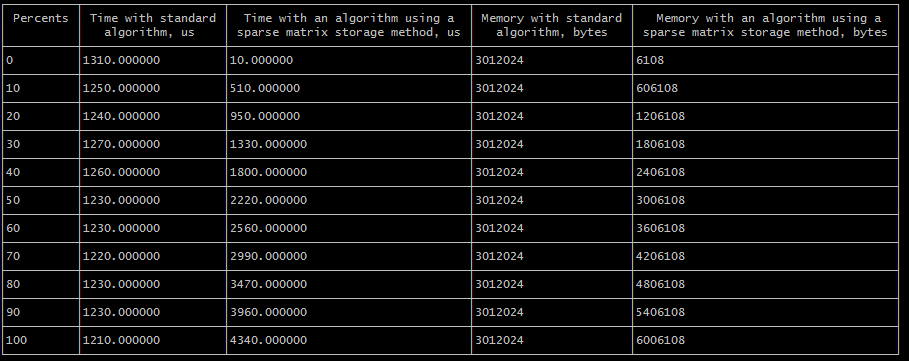
**6. Сравнение времени и памяти при работе двух алгоритмов.**

Я сравнивала время сложения матриц стандартным образом и в разреженном виде при разном процентном заполнении матрицы ненулевыми элементами. Замеры проводила на 100 повторах, брала среднее значение. Результаты представлены ниже:

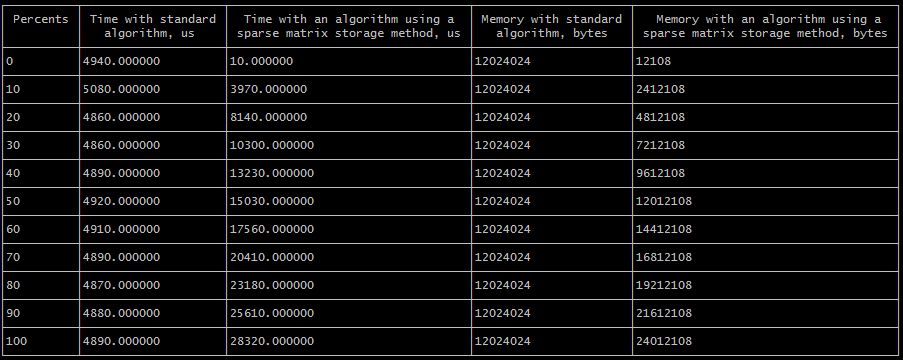
При размерности 100 на 100:



При размерности 500 на 500:



При размерности 1000 на 1000:



При количестве ненулевых элементов меньше 50 % разреженный вид эффективнее по памяти, чем стандартный.

При количестве ненулевых элементов до 30 % разреженный вид эффективнее по времени. С 30% до 50% разреженный вид эффективнее только по памяти

При количестве ненулевых элементов больше 50% неэффективно использовать разреженный способ хранения, так как на него тратится больше памяти и времени.

**6. Выводы по проделанной работе**

В результате выполненной лабораторной работы я приобрела навыки работы c разреженными матрицами, сравнила эффективность использования алгоритмов со стандартными матрицами и с разреженными (по времени и по требуемой памяти) при различном процентном заполнении матриц ненулевыми значениями и при различных размерах матриц.

В результате сравнения можно сделать выводы:

1. При количестве ненулевых элементов меньше 50 % разреженный вид эффективнее по памяти, чем стандартный.
2. При количестве ненулевых элементов до 30 % разреженный вид эффективнее по времени. С 30% до 50% разреженный вид эффективнее только по памяти
3. При количестве ненулевых элементов больше 50% неэффективно использовать разреженный способ хранения, так как на него тратится больше памяти и времени.

**7. Ответы на вопросы**

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

Разреженная матрица – это матрица с преимущественно нулевыми элементами.

Число ненулевых элементов при матрице порядка n может выражаться как n1+g , где g < 1. В матричных задачах для разреженных матриц g лежат в интервале 0.2 … 0.5.

Схемы хранения разреженных матриц:

* Связная схема хранения матриц, предложенная Кнутом: она предлагает хранить в массиве (например, в AN) в произвольном порядке сами элементы, индексы строк и столбцов соответствующих элементов (например, в массивах I и J), номер (из массива AN) следующего ненулевого элемента, расположенного в матрице по строке (NR) и по столбцу (NC), а также номера элементов, с которых начинается строка (указатели для входа в строку – JR) и номера элементов, с которых начинается столбец (указатели для входа в столбец - JC).
* Схема, предложенная Чангом и Густавсоном, называемая: "разреженный строчный формат". В этом случае значения ненулевых элементов хранятся в массиве AN, соответствующие им столбцовые индексы - в массиве JA. Кроме того, используется массив указателей, например, IA, отмечающих позиции AN и JA, с которых начинаются описание очередной строки. Дополнительная компонента в IA содержит указатель первой свободной позиции в JA и AN.

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

В моём случае обычная матрица представлена в виде массива указателей, поэтому для хранения обычной матрицы выделяется:

sizeof (тип эл-та \*\*) + sizeof (тип эл-та \*) \* n + sizeof(тип эл-та) \* m \* n байт, где n – количество строк, m – количество столбцов матрицы.

Количество памяти, выделяемой под разряженную матрицу зависит от количества ненулевых элементов. У меня разряженная матрица представлена в виде структуры, в которой три динамических массива и целое число – количество ненулевых элементов. У меня для хранения разряженной матрицы выделяется:

sizeof (тип структуры) + nonzero \* 2 \* sizeof(тип эл-та) + (m + 1) \* sizeof(тип эл-та)), где nonzero – кол-во ненулевых элементов.

3. Каков принцип обработки разреженной матрицы?

Алгоритмы обработки разреженных матриц предусматривают действия 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?

Использовать стандартные алгоритмы обработки матриц эффективнее, если в матрице больше 50 % ненулевых элементов, так как в этом случае на хранение матрицы в разреженном виде требуется больше памяти и на обработку затрачивается больше времени.

По памяти стандартные алгоритмы становятся эффективнее с 50% заполнения ненулевыми элементами. По времени – с 30%.