МГТУ им. Н.Э. Баумана

**Дисциплина Типы и Структуры данных.**

**Лабораторный практикум №3**

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

Работу выполнил:

студент группы ИУ7-35Б

Прянишников Александр

**Цель работы**: реализация алгоритмов обработки разреженных матриц, сравнение эффективности использования этих алгоритмов со стандартными алгоритмами обработки матриц при различном размере матриц и степени их разреженности.

# Условие задачи. Вариант 3

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 матрицы и вектора-столбца, 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.

# Требования к задаче

**Ввод данных**

Матрица и вектор состоят из элементов типа double. Элементы могут вводиться в двух режимах: ручном и автоматическом. В первом случае пользователь вводит каждое число с новой строки, запись происходит построчно.

Во втором случае программа через генератор случайных чисел составляет матрицу, а процент заполняемости выбирает сам пользователь. В общем случае для матрицы и вектора это могут быть разные значения.

**Вывод данных**

Вывод происходит для следующих данных: обычная матрица, разреженная матрица, сравнение времени работы алгоритмов.

Также в программу был добавлен интерфейс, упрощающий работу для пользователей: есть пояснительные надписи, перед вводом данных написано, как правильно вводить данные.

**Описание задачи, реализуемой программой**

Программа предоставляет пользователю интерфейс для работы с разреженными матрицами. Пользователь может перемножить матрицы двумя алгоритмами: обычным умножением или через разреженные матрицы. В меню есть пункты, позволяющие провести умножение любым выбранным способом. Также пользователь может сравнить время работы алгоритмов на одинаковом массиве.

**Способ обращения к программе.**

Программу можно запустить через MSYS либо через QT Creator и любую среду разработки.

Выполняется следующая последовательность команд: *./main.exe*

**Описание возможных аварийных ситуаций и ошибок пользователя.**

**Ошибки пользователя при работе с меню и массивом**

* Выбор отсутствующего пункта меню (от 0 до 4)  
  *В этом случае программа сообщит пользователю о невозможности совершения операции и попросит ввести новое значение пункта меню.*
* Ввод букв при вводе числа  
  *В этом случае программа сообщит пользователю о невозможности совершения операции и попросит ввести новое значение пункта меню.*
* Пустой ввод  
  *В этом случае программа сообщит пользователю о невозможности совершения операции и попросит ввести новое значение пункта меню.*

**Предусмотренные проблемные ситуации**

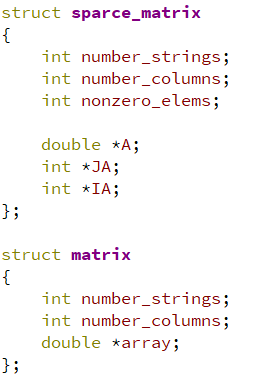
* Слишком длинный вывод  
  *В этом случае программа предложит пользователю не выводить массив, но если он настаивает, то вывод все равно состоится.*

# Описание внутренних структур данных.

Для решения этой задачи я решил использовать схему «разреженного строчного формата». Структура состоит из размера матрицы, количества ненулевых элементов, и трёх массивов – не нулевых значений, индексов столбцов, и номера элементов, которые являются первыми для своей строки. При этом если в строке нет ненулевых элементов, то сохраняется –1.

Также есть тип данных обычной матрицы, для которой реализован алгоритм обычного перемножения матриц.

Вот так описаны структуры:



# Описание алгоритма

У программы есть три режима, каждый отличается своим алгоритмом работы:

**1 – Ввести (сгенерировать) и перемножить разреженную матрицу и вектор-столбец.**Программа предлагает пользователю ввести размеры матрицы (натуральные числа), после чего предоставляет два способа ввода чисел: вручную и автоматически.  
Сначала матрица сохраняется в тип данных matrix, а затем преобразовывается в sparce\_matrix – это сделано потому, что чтение все равно по времени займёт n2, но зато все вычисления дальше будут значительно легче, так как сразу подсчитывается количество ненулевых элементов.  
Затем аналогичным образом вводится вектор–столбец, но для него уже сразу определены размеры.   
Программа затем умножает матрицы друг на друга, используя оптимизацию разреженных матриц. Алгоритм работы – O(N).

**2 – Ввести (сгенерировать) обычные матрицы и перемножить друг с другом**.  
Программа предлагает пользователю ввести размеры матрицы (натуральные числа), после чего предоставляет два способа ввода чисел: вручную и автоматически.  
Матрица сразу сохраняется в тип данных matrix. Затем аналогичным образом вводится вектор–столбец, но для него уже сразу определены размеры.   
Программа затем умножает матрицы друг на друга, используя обычный алгоритм. Алгоритм работы – O(N2).

**3 –**  **Cравнить время работы двух методов перемножения матриц.**Программа предлагает пользователю ввести размеры матрицы (натуральные числа), после чего предоставляет два способа ввода чисел: вручную и автоматически.  
Матрица сохраняется и в тип данных matrix, и в тип данных sparce\_matrix. Затем аналогичным образом вводится вектор–столбец, но для него уже сразу определены размеры.  
Затем программа перемножает вектор и матрицу двумя разными способами, выводя время работы каждого алгоритма.

**0 – Выход.**

Поговорим про алгоритм умножения разреженных матриц. Программа запускает цикл по массиву ненулевых значений, каждый раз проходя по одной строке. Каждый элемент результирующей матрицы рассчитывается построчно, этому помогает массив для строк. Программа смотрит на индекс столбца исходного элемента, а затем смотрит, что находится под тем же индексом в массива для строки вектора. Если значение по строки не –1, значит, программа производит умножение, и заносит промежуточной результат в итоговый массив. Всё происходит в два линейных цикла: в первом высчитывается память, во втором –

# Тесты

Тестирование происходило по такому принципу: каждый пункт тестировался отдельно, после чего контролировалась работа программы целиком. Поэтому здесь указаны только негативные тесты, которые в целом проверяли работоспособность модулей программы.

|  |  |
| --- | --- |
| **123345fd** | **Буквы при задании числа** |
|  | **Пустой ввод** |
| **–1** | **Выбор неправильного пункта меню** |
| **15** | **Выбор неверного значения поля** |

# Оценка эффективности

Пусть N – количество строк, M – количество столбцов, а K – количество ненулевых элементов. Тогда для обычной матрицы размер занимаемой памяти составляет порядок N\* M, а для разреженной – 2 \* K + N. Если нулевых элементов много, то разреженная матрица выигрывает в памяти, если нет – то обычная гораздо выгодней.

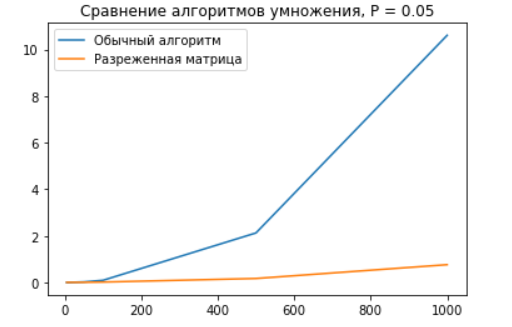
Согласно теоретическим расчётам, алгоритм умножения разреженных матриц имеет зависимость N2, а обычного умножения – N2. Составим таблицу значений при различных N и коэффициенту заполняемости P, чтобы проверить законы, а затем визуализируем. Для удобства будем рассматривать квадратные матрицы:

P = 0.05

|  |  |  |
| --- | --- | --- |
|  | Обычная(время) | Разреженная(время) |
| N = 5 | 0.002 | 0.003 |
| N = 10 | 0.002 | 0.003 |
| N = 50 | 0.023 | 0.006 |
| N = 100 | 0.092 | 0.016 |
| N = 500 | 2.125 | 0.170 |
| N = 1000 | 10.609 | 0.762 |

Аналогичная таблица для памяти (учитываем и вектор, и матрицу)

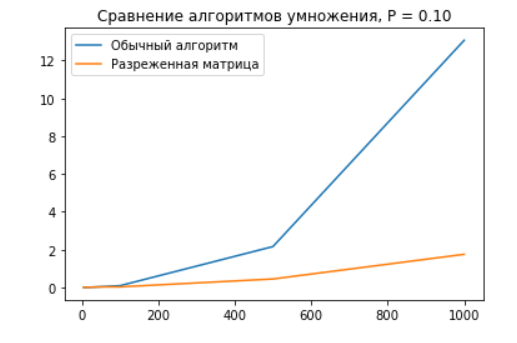
|  |  |  |
| --- | --- | --- |
|  | Обычная(память) | Разреженная(память) |
| N = 5 | 136 | 80 |
| N = 10 | 456 | 126 |
| N = 50 | 10216 | 1464 |
| N = 100 | 40416 | 4920 |
| N = 500 | 1002016 | 103312 |
| N = 1000 | 4004016 | 407304 |



P = 0.10

|  |  |  |
| --- | --- | --- |
|  | Обычная | Разреженная |
| N = 5 | 0.001 | 0.003 |
| N = 10 | 0.001 | 0.003 |
| N = 50 | 0.035 | 0.028 |
| N = 100 | 0.091 | 0.029 |
| N = 500 | 2.161 | 0.446 |
| N = 1000 | 13.07 | 1.75 |

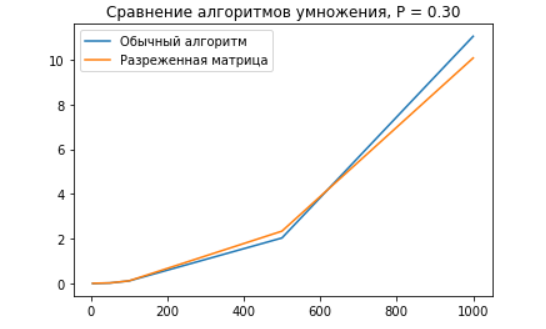
|  |  |  |
| --- | --- | --- |
|  | Обычная(память) | Разреженная(память) |
| N = 5 | 136 | 88 |
| N = 10 | 456 | 176 |
| N = 50 | 10216 | 2296 |
| N = 100 | 40416 | 8224 |
| N = 500 | 1002016 | 205304 |
| N = 1000 | 4004016 | 805728 |



P = 0.3

|  |  |  |
| --- | --- | --- |
|  | Обычная | Разреженная |
| N = 5 | 0.001 | 0.004 |
| N = 10 | 0.003 | 0.007 |
| N = 50 | 0.018 | 0.022 |
| N = 100 | 0.111 | 0.115 |
| N = 500 | 2.033 | 2.338 |
| N = 1000 | 11.054 | 10.08 |

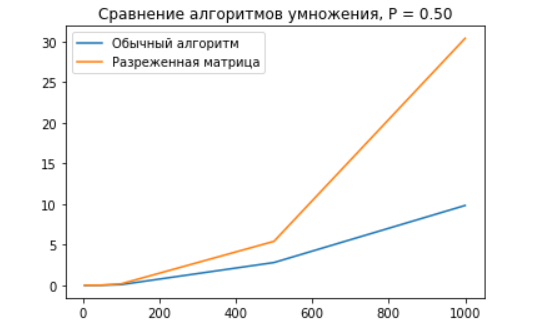
|  |  |  |
| --- | --- | --- |
|  | Обычная(память) | Разреженная(память) |
| N = 5 | 136 | 152 |
| N = 10 | 456 | 376 |
| N = 50 | 10256 | 6112 |
| N = 100 | 40416 | 25746 |
| N = 500 | 1002016 | 606216 |
| N = 1000 | 4004016 | 2414288 |



P = 0.5

|  |  |  |
| --- | --- | --- |
|  | Обычная | Разреженная |
| N = 5 | 0.001 | 0.004 |
| N = 10 | 0.016 | 0.029 |
| N = 50 | 0.028 | 0.033 |
| N = 100 | 0.095 | 0.190 |
| N = 500 | 2.81 | 5.417 |
| N = 1000 | 9.263 | 30.38 |

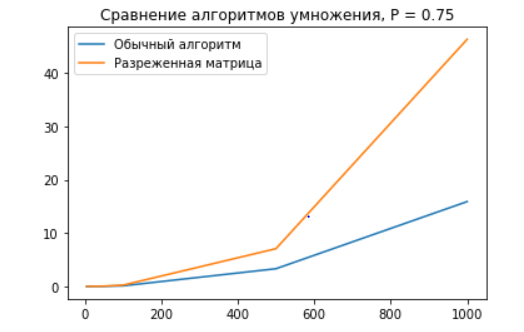
|  |  |  |
| --- | --- | --- |
|  | Обычная(память) | Разреженная(память) |
| N = 5 | 136 | 168 |
| N = 10 | 456 | 126 |
| N = 50 | 10216 | 10744 |
| N = 100 | 40416 | 41992 |
| N = 500 | 1002016 | 1003968 |
| N = 1000 | 4004016 | 4016808 |



P = 0.75

|  |  |  |
| --- | --- | --- |
|  | Обычная | Разреженная |
| N = 5 | 0.001 | 0.003 |
| N = 10 | 0.012 | 0.028 |
| N = 50 | 0.028 | 0.054 |
| N = 100 | 0.110 | 0.233 |
| N = 500 | 3.315 | 7.076 |
| N = 1000 | 15.91 | 46.35 |

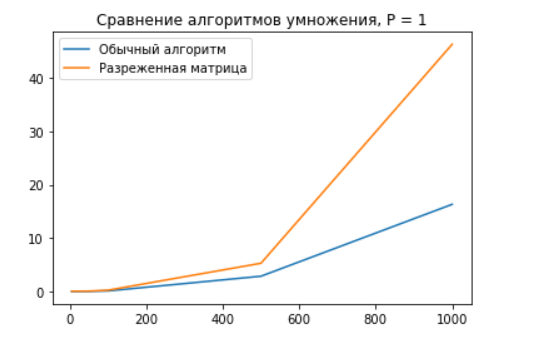
|  |  |  |
| --- | --- | --- |
|  | Обычная(память) | Разреженная(память) |
| N = 5 | 136 | 224 |
| N = 10 | 456 | 808 |
| N = 50 | 10216 | 15624 |
| N = 100 | 40416 | 61528 |
| N = 500 | 1002016 | 1505592 |
| N = 1000 | 4004016 | 6013840 |



P = 1

|  |  |  |
| --- | --- | --- |
|  | Обычная | Разреженная |
| N = 5 | 0.013 | 0.032 |
| N = 10 | 0.019 | 0.047 |
| N = 50 | 0.028 | 0.067 |
| N = 100 | 0.105 | 0.237 |
| N = 500 | 2.871 | 5.291 |
| N = 1000 | 16.34 | 47.36 |

|  |  |  |
| --- | --- | --- |
|  | Обычная(память) | Разреженная(память) |
| N = 5 | 136 | 304 |
| N = 10 | 456 | 984 |
| N = 50 | 10216 | 20824 |
| N = 100 | 40416 | 81624 |
| N = 500 | 1002016 | 2008024 |
| N = 1000 | 4004016 | 8016024 |



Можно сделать несколько выводов по эффективности. Во–первых, хранение разреженных матриц действительно помогает при очень маленькой заполняемости матрицы, но уже при P = 0.30 алгоритмы слабо отличаются друг от друга. По занимаемой памяти разреженная матрица лучше обычной до P = 0.5. При этом на маленьких N обычное умножение матриц работает эффективнее вообще при любых P. При больших P разреженные матрицы проигрывают в пух и прах, как и по памяти, так и по времени.

В целом, разреженные матрицы эффективны только при очень маленьких P: уже при P = 0.30 и по памяти, и по времени работы способы одинаковые.

# Выводы по проделанной работе

Сегодня я познакомился с алгоритмами хранения разреженных матриц, способом их перемножения, оценил их эффективность по сравнению с обычными алгоритмами умножения. Выяснилось, что при небольшом проценте заполнения матрицы «разреженный строчный формат» оправдывает все затраты и существенно сокращает время работы и занимаемую память, но если данных достаточно много, то выгоднее хранить обычную матрицу.

# Ответы на контрольные вопросы

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

Разреженная матрица – матрица, в которой очень мало ненулевых элементов. Способы хранения матриц: линейный связный список, диагональная схема хранения матриц, схема хранения Кнута, «разреженный строчный формат».

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

В обычной матрице хранятся все элементы, то есть всего N\*M ячеек типа int/float. Разреженная матрица хранится в виде трёх массивов: ненулевых значений, индексов столбцов и элементов, с которых начинается строка. То есть, если количество ненулевых элементов выражается как N1 + q , то затраченная память = 3 \* N + Nq. Для обычной матрицы достаточно статической матрицы, если заранее известны её размеры, то для разреженной требуется знать количество ненулевых элементов, поэтому разумнее использовать динамическую память.

Каков принцип обработки разреженной матрицы?

Принцип в том, что мы работаем только с ненулевыми элементами.

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

Если в матрице заполнено хотя бы 50% значений, то разреженные алгоритмы уже не применимы, и стандартные алгоритмы работают эффективнее. Также обычная матрица быстрее, если её размеры маленькие: при N = 50 ещё не сильно заметны различия между алгоритмами.