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

Отчет по лабораторной работе № 3

«Разреженные матрицы»

Работу выполнил:

студент группы ИУ7-32Б

Лемешев А.П.  
Работу проверил:  
Силантьева А.В.

Москва, 2021 г.

**Условие задачи**

Реализовать алгоритмы обработки разреженных матриц, сравнить эффективность использования этих алгоритмов (по времени выполнения и требуемой памяти) со стандартным алгоритмами обработки матриц при различном процентном заполнения матриц ненулевыми значениями и при различных размерах матриц.

**Техническое задание**

Смоделировать операцию умножения вектора строки и матрицы хранящихся

а) в разреженном виде; б) в стандартном виде. Сравнить время выполнения и объем занимаемой памяти при использовании 2-х этих способов хранения.

Разреженная матрица должна хранится в виде:

* 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.

**Входные данные**

Количество строк и столбцов матрицы, количество ненулевых элементов матриц и вектора строки, их индексы в матрице или векторе, вариант заполнения матриц (случайный или с клавиатуры).

**Выходные данные**

Результат умножения вектора строки на матрицу, результаты сравнения 2-х алгоритмов обработки.

**Возможные аварийные ситуации**

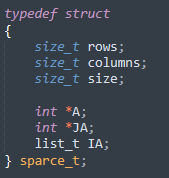
* Некорректный ввод пункта меню
* Некорректный ввод количества строк матрицы
* Некорректный ввод количества столбцов матрицы
* Некорректный ввод количества ненулевых элементов матрицы
* Некорректный ввод количества ненулевых элементов вектора-строки
* Некорректный ввод элемента матрицы
* Некорректный ввод расположения элемента матрицы
* Некорректный ввод элемента вектора-строки
* Некорректный ввод расположения элемента вектора-строки

**Способ обращения к программе**

Запуск через командную оболочку MSYS2 (Windows), терминал sh/bash (Linux).

**Структуры данных**

**Структура разреженной матрицы**

  
**rows** и **columns** — количество строк и столбцов матрицы;

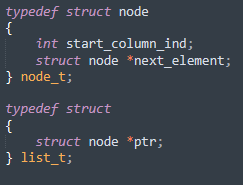
**size** – количество ненулевых элементов;

**A** — указатель на массив, содержащий все ненулевых элементы;

**JA** — указатель на массив, содержащий номера столбцов для каждого элемента массива А;

**IA** – связанный список, который содержит для каждого первого элемента i-ой строки его индекс в массивах A и JA.

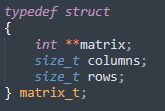
**Структура связанного списка:**

  
**node\_t** — сама структура очередного элемента списка.

**start\_column\_ind** — индекс первого элемента i-ой строки матрицы в массивах А и JA;

**next\_element** — указатель на следующий элемент связанного списка.

**Структура стандартной матрицы**

  
**matrix** — указатель на матрицу;

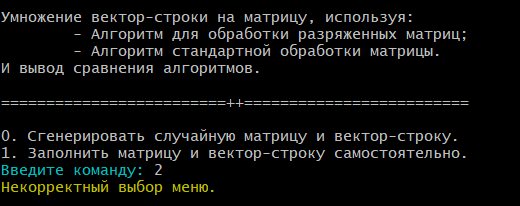
**rows** и**columns** — количество строк и столбцов матрицы.

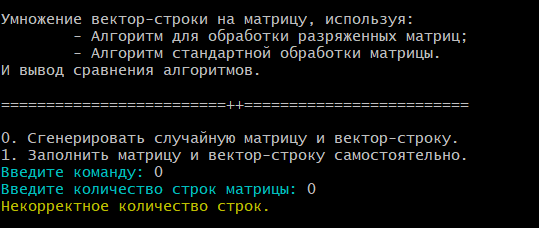
**Алгоритм**

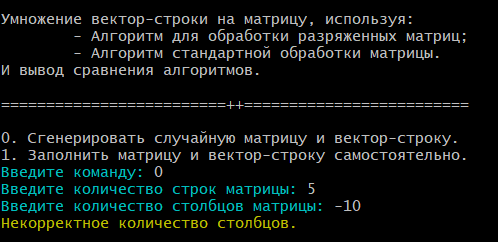
Матрица транспонируется. Далее, умножение происходит вектор на вектор столько раз, сколько в *исходной* матрице столбцов. Для этого, я прохожу по массиву ненулевых элементов, и пользуясь индексами столбцов этих элементов из массива IA, последовательно перемножаю каждый элемент текущей строки на нужный элемент вектора-строки.

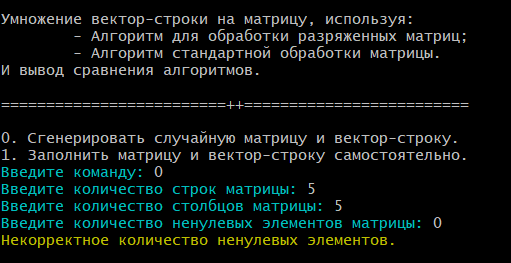
**Тесты**

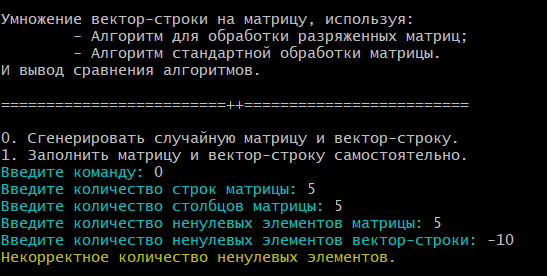
**Негативные тесты**

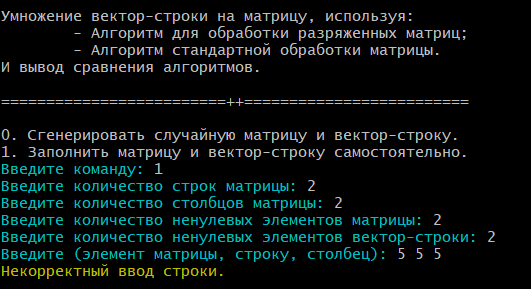
Некорректный ввод меню ****

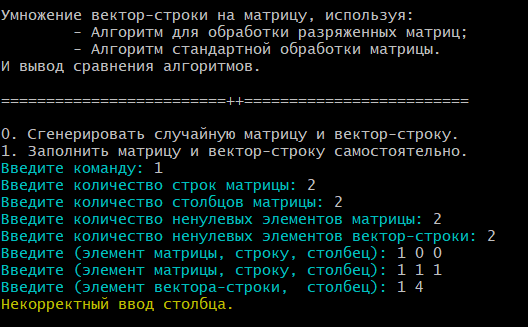
Некорректный ввод количества строк матрицы  


Некорректный ввод количества столбцов матрицы  


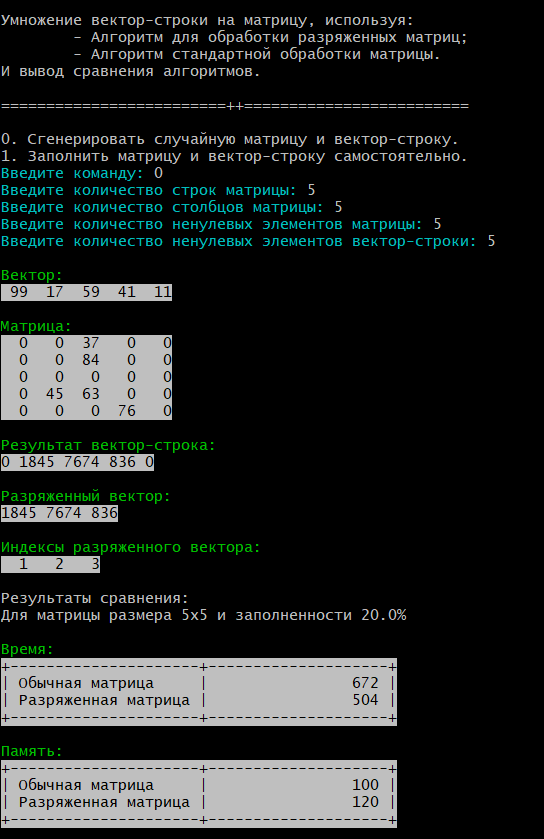
Некорректный ввод количества ненулевых элементов матрицы  


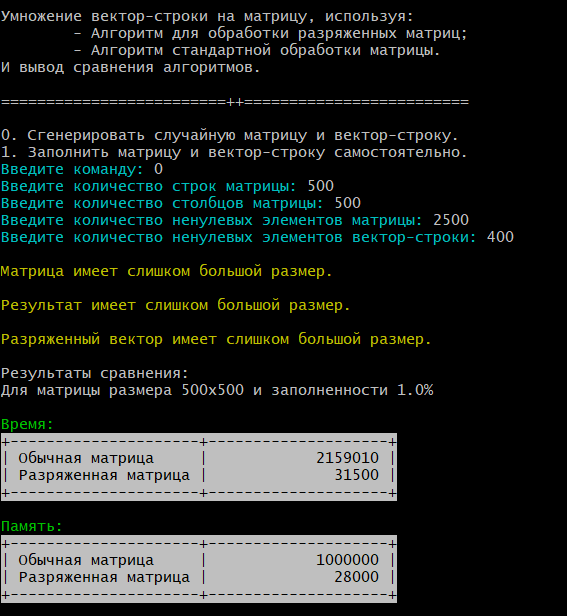
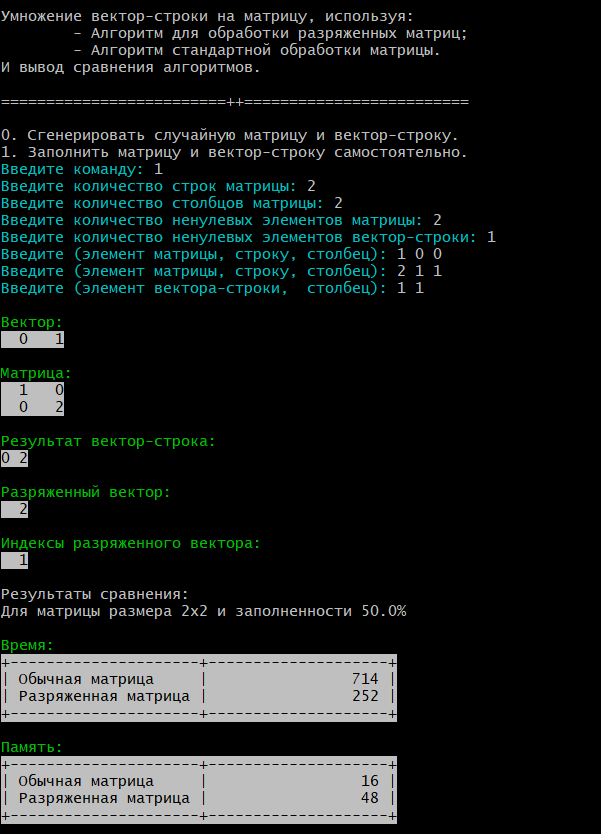
Некорректный ввод количества ненулевых элементов вектора-строки  


Некорректный ввод элемента матрицы  


Некорректный ввод элемента вектора-строки  


**Позитивные тесты**



**Время**

**5% заполнения**

| **Размеры матрицы** | **Разреженная матрица** | **Обычная матрица** |
| --- | --- | --- |
| 10x10 | 504 | 1134 |
| 100x100 | 8988 | 108864 |
| 1000x1000 | 434511 | 10050195 |

**10% заполнения**

| **Размеры матрицы** | **Разреженная матрица** | **Обычная матрица** |
| --- | --- | --- |
| 10x10 | 777 | 1701 |
| 100x100 | 12180 | 108738 |
| 1000x1000 | 628887 | 8339961 |

**20% заполнения**

| **Размеры матрицы** | **Разреженная матрица** | **Обычная матрица** |
| --- | --- | --- |
| 10x10 | 861 | 1512 |
| 100x100 | 19761 | 109767 |
| 1000x1000 | 1204308 | 7971600 |

**50% заполнения**

| **Размеры матрицы** | **Разреженная матрица** | **Обычная матрица** |
| --- | --- | --- |
| 10x10 | 1134 | 1659 |
| 100x100 | 43134 | 106911 |
| 1000x1000 | 2928177 | 7953141 |

**100% заполнения**

| **Размеры матрицы** | **Разреженная матрица** | **Обычная матрица** |
| --- | --- | --- |
| 10x10 | 3066 | 2037 |
| 100x100 | 128205 | 154497 |
| 1000x1000 | 8942913 | 12379941 |

**10x10**

| **Заполнение** | **Разреженная матрица** | **Обычная матрица** |
| --- | --- | --- |
| 5% | 504 | 225% |
| 10% | 154% | 337% |
| 20% | 170% | 410% |
| 50% | 225% | 452% |
| 100% | 300% | 512% |

**Занимаемая память**

**1% заполнения**

| **Размеры матрицы** | **Разреженная матрица** | **Обычная матрица** |
| --- | --- | --- |
| 10x10 | 168 | 400 |
| 100x100 | 2400 | 40000 |
| 1000x1000 | 96000 | 4000000 |

**10% заполнения**

| **Размеры матрицы** | **Разреженная матрица** | **Обычная матрица** |
| --- | --- | --- |
| 10x10 | 240 | 400 |
| 100x100 | 9600 | 40000 |
| 1000x1000 | 832000 | 4000000 |

**50% заполнения**

| **Размеры матрицы** | **Разреженная матрица** | **Обычная матрица** |
| --- | --- | --- |
| 10x10 | 560 | 400 |
| 100x100 | 41600 | 40000 |
| 1000x1000 | 4032000 | 4000000 |

**100% заполнения**

| **Размеры матрицы** | **Разреженная матрица** | **Обычная матрица** |
| --- | --- | --- |
| 10x10 | 960 | 400 |
| 100x100 | 81600 | 40000 |
| 1000x1000 | 8032000 | 4000000 |

**10x10**

| **Заполнение** | **Разреженная матрица** | **Обычная матрица** |
| --- | --- | --- |
| 1% | 100% | 238% |
| 10% | 142% | 238% |
| 50% | 333% | 238% |
| 100% | 571% | 238% |

**Выводы по проделанной работе**

Использование алгоритмов хранения и обработки разреженных матриц выгодно при большом количестве элементов, примерно до 40-45% заполненности матриц. В таком случае, алгоритм выигрывает как и в размерах занимаемой памяти, так и в скорости обработки. Но при заполненности более чем 50%, алгоритм обработки и хранения разреженных матриц начинает проигрывать во времени, но все еще выигрывает во времени. Даже при 100% заполненности матриц этот алгоритм выигрывает во времени у обычного, но занимает места в памяти в более чем в 2 раза больше.

**Контрольные вопросы**

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

Разреженная матрица — это матрица, содержащая большое количество нулей. Способы хранения:

* Словарь по ключам (DOK - Dictionary of Keys) строится как словарь, где ключ — это пара (строка, столбец), а значение это соответствующий строке и столбцу элемент матрицы
* Список списков (LIL - List of Lists) строится как список строк, где строка — это список узлов вида (столбец, значение)
* Список координат (COO - Coordinate list) хранится список из элементов вида (строка, столбец, значение)
* Сжатое хранение строкой (CSR - compressed sparse row, CRS - compressed row storage, Йельский формат)
* Сжатое хранение столбцом (CSС - compressed sparse column, CСS - compressed column storage)

То же самое что и CRS, только строки и столбцы меняются ролями - значения храним по столбцам, по второму массиву можем определить строку, после подсчётов с третьим массивом - узнаём столбцы.

**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 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?**

Стандартные алгоритмы обработки матриц эффективнее применять при большом количестве ненулевых элементов (от 50%). Стоит отметить, что если не так важна память, занимаемая матрицами, но важно время, то можно лучше использовать алгоритм обработки разреженных матриц, так как он, хоть и немного, но выигрывает во времени даже на больших % заполненостях матриц (80% - 100%)