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 ЛАБОРОТОРНОЙ РАБОТЕ № 3
«ОБРАБОТКА РАЗРЕЖЕННЫХ МАТРИЦ»

Студент Байрамгалин Ярослав Ринатович
Группа ИУ7-33Б
Преподаватель Силантьева Александра Васильевна

Москва 2021
МГТУ им. Н.Э. Баумана

1. Описание условия задачи

Разреженная (содержащая много нулей) матрица хранится в форме 3-х объектов:

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 технического задания

Входные данные:

Пункт меню: целое число, содержащее соответствующий пункт меню.

Матрица 1: матрица, представленная в разреженном координатном формате (сначала вводится количество строк в матрице, затем количество столбцов, затем количество ненулевых элементов, затем каждый из элементов матрицы (строка, число, значение)). Стоит принять во внимание, что нумерация строк и столбцов в матрице начинается с нуля.

Матрица 2: то же, что матрица 1.

Выходные данные:

Результат сложения двух матриц: разреженная матрица, представленная через 3 вектора.

Обращение к программе:

Запускается командой $./app.exe$ через терминал, находясь в директории, содержащей программу.

Аварийные ситуации:

1. При сложении введены матрицы, для которых не определена операция сложения (отличается количество строк или столбцов).
2. Пустой ввод пункта меню.
3. Неверный пункт меню.
4. При координатном вводе матрицы значение столбца (строки) больше количества столбцов (строк) или меньше нуля.

3. Описание структур данных

Для хранения разреженной матрицы используется следующая структура данных:

```
typedef struct {
    int *values;
    size_t *cols;
    size_t count_cols;
    size_t count_rows;
    size_t count_non_zero;
    until_row_count_list_t row_list;
} sparse_matrix_t;
```

Поля структуры:

`*values` – массив ненулевых элементов;

`cols` – массив, который содержит номера столбцов для элементов массива values;

`count_cols` – количество столбцов матрицы;

`count_rows` – количество строк матрицы;

`count_non_zero` – количество ненулевых элементов;

`*row_list` – связный список в элементе k-ом которого находится номер компонента, с которых начинается описание k-ой строки матрицы.

Для связного списка `*row_list` используется следующая структура данных:

```
struct until_row_count_list_t {
    size_t value;
    until_row_count_list_t *next;
};
```

Поля структуры:

`value` – значение, с которого начинается описание k-ой строки для каждого k-го элемента списка;

`*next` – указатель на следующий элемент связного списка.

Для хранения матрицы в обычном виде используется следующая структура данных:

```
typedef struct {
    int **values;
    int rows;
    int cols;
} matrix_t;
```

Поля структуры:

`*values` – матрица значений;

`rows` – количество строк в матрице;

`cols` – количество столбцов в матрице.

4. Описание алгоритма

1. Пользователь выбирает необходимый пункт меню (1 или 2).
2. При выборе пункта 1 меню пользователь вводить 2 матрицы, затем отображается результат сложения этих матриц.
3. При выборе пункта 2 меню выводится время и память, требуемые для сложения одинаковых матриц размерностью 250x250 и 500x500 при разной степени зполненности (2-30%).

5. Набор тестов

Позитивные тесты.

№	Пункт меню и входные данные	Действия и выходные данные	Результат
1	Пункт меню = 0	Информация о завершении программы	Код возврата = 0
2	Пункт меню = 1 Ввод двух валидных ненулевых матриц	Ввод количества строк, столбцов и ненулевых элементов матрицы. Ввод самих элементов матрицы.	Вывод сложения двух матриц в разреженном формате. Код возврата = 0
3	Пункт меню = 1 Ввод двух нулевых матриц	Ввод количества строк, столбцов и ненулевых элементов матрицы. Ввод самих элементов матрицы.	Вывод сложения двух матриц в разреженном формате: пустой массив A и JA, связный список содержит три нуля. Код возврата = 0
4	Пункт меню = 2	Вывод информации о затрачиваемой памяти и получившийся скорости для двух методов сложения	Код возврата = 0

Негативные тесты

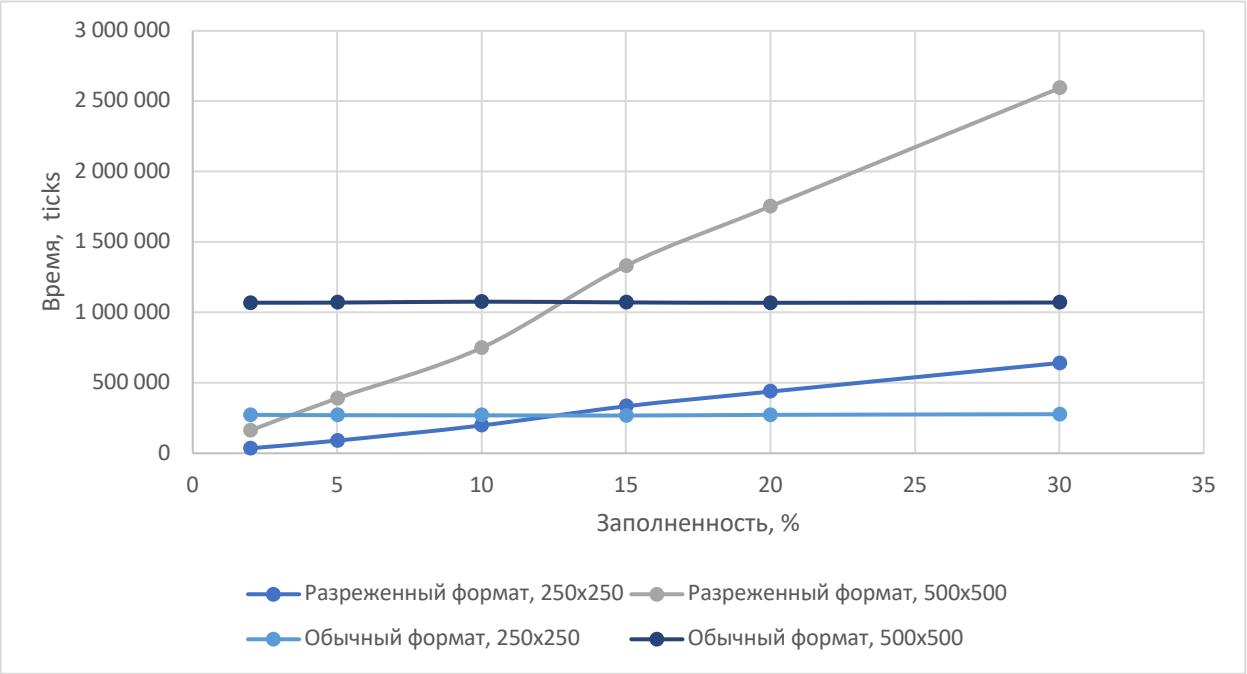
№	Пункт меню	Действия и выходные данные	Результат
1	Пункт меню = 10 Неверный ввод ключа	Сообщение, что пользователь ввёл невалидный ключ.	Код возврата = 1

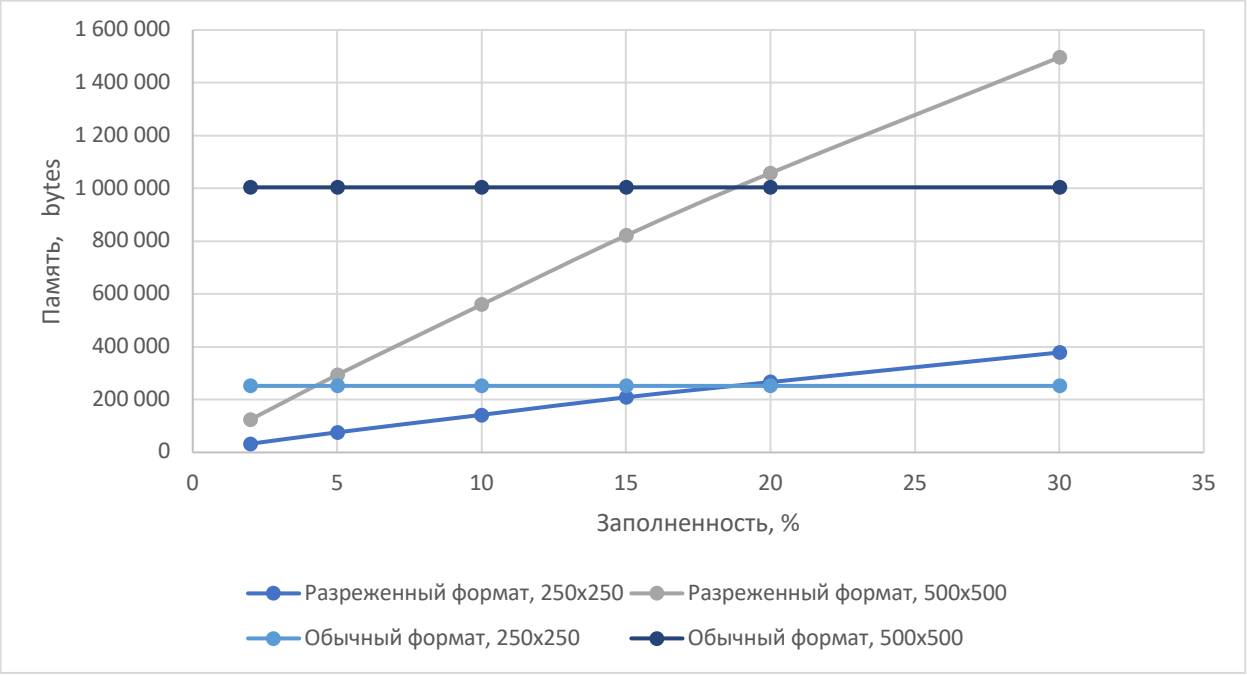
2	Пункт меню = 1 Ошибка выделения памяти	Завершение программы из-за невыделенной памяти	Код возврата = 2
3	Пункт меню = 1 Ввод отрицательных/действительных/буквенных параметров матриц.	Сообщение об ошибке	Код возврата = 3
4	Пункт меню = 1 Валидный ввод параметров матриц. Ввод действительных/буквенных элементов матрицы	Сообщение об ошибке	Код возврата = 4
5	Пункт меню = 1 Валидный ввод параметров матриц и самих элементов, но разная размерность	Сообщение об ошибке	Код возврата = 1

6. Оценка эффективности

В таблице указаны результаты 1000 измерений.

№	Размер матрицы	Заполненность, %	Время, ticks		Память, bytes	
			Разреженная	Обычная	Разреженная	Обычная
1	250x250	2	35 338	273 043	32 856	252 016
2	250x250	5	90 019	270 495	76 020	252 016
3	250x250	10	197 675	269 495	142 292	252 016
4	250x250	15	333 908	268 147	208 800	252 016
5	250x250	20	438 015	272 872	266 424	252 016
6	250x250	30	640 963	277 663	378 517	252 016
7	500x500	2	162 905	1 069 345	124 540	1 004 016
8	500x500	5	390 529	1 070 162	293 452	1 004 016
9	500x500	10	749 093	1 076 790	559 612	1 004 016
10	500x500	15	1 331 987	1 071 519	821 449	1 004 016
11	500x500	20	1 753 505	1 068 801	1 058 092	1 004 016
12	500x500	30	2 592 682	1 070 776	1 496 116	1 004 016


3 000 000
2 500 000
2 000 000
1 500 000
1 000 000
500 000
0
Время, ticks
0
5
10
15
20
25
30
35
Заполненность, %
Разреженный формат, 250x250
Разреженный формат, 500x500
Обычный формат, 250x250
Обычный формат, 500x500


1 600 000
1 400 000
1 200 000
1 000 000
800 000
600 000
400 000
200 000
0
Память, bytes
0
5
10
15
20
25
30
35
Заполненность, %
Разреженный формат, 250x250
Разреженный формат, 500x500
Обычный формат, 250x250
Обычный формат, 500x500

Нетрудно заметить, что при низкой заполненности исходных матриц алгоритм сложения разреженных матриц работает эффективнее как по времени, так и по памяти.

Так при изначальной заполненности исходных матриц в 2% использование разреженных матриц приводит к выигрышу по времени 7.8 раз, по памяти в 7.7 раз.

Однако с ростом количества ненулевых элементов данный подход быстро деградирует. Судя по графику одинаковое время работы должно наблюдаться при заполненности около 13%. Тем не менее, при такой заполненности все еще сохраняется выигрыш по памяти в 5-7%. Окончательно нерелевантным использование разреженных матриц становится при изначальной заполненности более 18%, так как в этом момент затраты памяти также начинают превышать классический алгоритм.

7. Выводы

Способ хранения и алгоритмы обработки разреженных матриц следует использовать только если гарантируется низкая заполненность исходных матриц. При очень низкой (до 10%) заполненности такой подход дает многократный выигрыш как по времени, так и по памяти.

Алгоритм быстро деградирует. Таким образом при заполненности исходных матриц более 18% его использование перестает быть целесообразным. Такая деградация по памяти связана с тем, что например для при сложении двух матриц заполненных на 18% итоговый результат будет заполнен на 18-36%.

8. Контрольные вопросы

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

Разреженная матрица — это матрица, содержащая большое количество нулей.

Схемы хранения матрицы: связанная схема хранения (с помощью линейных связанных списков), кольцевая связанная схема хранения, диагональная схема хранения, строчной формат, столбцовый формат.

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

Под обычную матрицу (N – количество строк, M – количество столбцов) выделяет N*M*sizeof(int) + N*sizeof(int*) байт памяти. Для разреженной матрицы количество ячеек памяти завит от способа. В случае разряженного формата требуется количество ячеек в размере

K* sizeof(int) + K * sizeof(size_t*) +N * sizeof(until_row_count_list_t) + sizeof(size_t) * 3 + sizeof(size_t*)

(K — количество ненулевых элементов, N – количество строк).

3. Каков принцип обработки разреженной матрицы?

При обработке разреженной матрицы мы работаем только с ненулевыми элементами. Тогда количество операций будет пропорционально количеству ненулевых элементов (прямая зависимость).

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

Эффективнее применять стандартные алгоритмы выгоднее при большом количестве ненулевых элементов(~20% от матрицы и больше). Стоит отметить, что если расход памяти в программе не так важен, но важно время выполнения программы, то в случае сложения матриц лучше воспользоваться стандартным алгоритмом при большом количестве (~20% от матрицы и больше) ненулевых элементов в матрице, и сложение специального (разряженного) в случае небольшого количества(до ~15% от матрицы) ненулевых элементов.