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

«Обработка разреженных матриц»

Студент Маслова Марина Дмитриевна
фамилия, имя, отчество

Группа ИУ7-33Б

2020 г.

Оглавление

Техническое задание 3
Условие задачи 3
Входные данные 3
Выходные данные 3
Задачи, реализуемые программой 3
Возможные аварийные ситуации и ошибки пользователя 4
Описание внутренних структур данных 4
Описание функций 5
Описание алгоритма 10
Тесты 11
Оценка эффективности 13
Контрольные вопросы 16
Вывод 17

Техническое задание

Условие 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 умножения вектора-строки и матрицы, 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 алгоритмов при различном проценте заполнения матриц.

Входные данные

Целое число, задающее выбор пункта меню.

Целые числа, задающие количество строк и столбцов матрицы, элементы матрицы и их положение в ней, процент заполненности матрицы.

Выходные данные

Исходные вектор-строка и матрица в разреженном или стандартном виде. Матрицы, представляющие результат операции умножения исходных матриц. Количественная характеристика сравнения эффективности вариантов выполнения операции умножения матриц.

Задачи, реализуемые программой

1. Ввод разреженной матрицы и вектор-строки с клавиатуры.

2. Генерация разреженной матрицы и вектора-строки по заданным размерам матрицы и проценту её заполненности.

3. Вывод введенных или сгенерированных матриц на экран в разреженном или стандартном виде.

4. Вывод результата операции умножения исходных матриц, а также времени выполнения операции и используемой при этом памяти при каждом из способов умножения.

Возможные аварийные ситуации и ошибки пользователя

Некорректный ввод пункта меню.

Некорректный ввод целочисленных данных.

Попытка вывода или выполнения операции умножения без ввода/генерации исходных матриц.

Описание внутренних структур данных

Для краткости записи использует беззнаковый тип:

```
typedef unsigned int uint;
```

Размеры матрицы представлены в виде структуры `matrix_size_t`:

```
typedef struct
{
    uint rows;      \\ количество строк
    uint columns;   \\ количество столбцов
    uint nonzeros; \\ количество ненулевых элементов
} matrix_size_t;
```

Матрица в стандартном виде хранится в виде структуры `matrix_t`:

```
typedef struct
{
    matrix_size_t sizes; \\ размеры матрицы
    int **matrix;        \\ матрица
} matrix_t;
```

Матрица в разреженном виде хранится в виде структуры `sparse_matrix_t`:

```
typedef struct
{
    matrix_size_t sizes;\\ размеры матрицы
    int *elements;       \\ указатель на массив элементов
    uint *rows;         \\ указатель на массив строк элементов
    uint *columns;      \\ указатель на массив номеров
                        \\ элементов, с которых начинается
                        \\ столбец
```

```
} sparse_matrix_t;
```

Описание функций

◆ create_matrix_by_sparse()

```
void create_matrix_by_sparse ( sparse_matrix_t *  sparse_matrix,
                               matrix_t *         matrix
                             )
```

Создает матрицу в стандартном виде по разреженной матрице

Parameters

sparse_matrix	Указатель на разреженную матрицу
matrix	Указатель на матрицу в стандартном виде

◆ create_sparse_by_matrix()

```
void create_sparse_by_matrix ( matrix_t *         matrix,
                               sparse_matrix_t *  sparse_matrix
                             )
```

Создает разреженную матрицу по матрице в стандартном виде

Parameters

sparse_matrix	Указатель на разреженную матрицу
matrix	Указатель на матрицу в стандартном виде

◆ free_matrix_t()

```
void free_matrix_t ( matrix_t *  matrix )
```

Очищает память, выделенную под матрицу в стандартном виде

Parameters

matrix	Указатель на матрицу в стандартном виде

◆ free_sparse_t()

```
void free_sparse_t ( sparse_matrix_t *  sparse_matrix )
```

Очищает память, выделенную под разреженную матрицу

Parameters

sparse_matrix	Указатель на разреженную матрицу

◆ generate_matrix()

```
void generate_matrix ( matrix_t *        matrix,
                       sparse_matrix_t * sparse_matrix
                     )
```

Генерирует матрицы по их параметрам

Parameters

sparse_matrix	Указатель на разреженную матрицу
matrix	Указатель на матрицу в стандартном виде

Returns

Код ошибки

◆ generate_matrixes()

```
int generate_matrixes ( sparse_matrix_t * sparse_matrix,
                        sparse_matrix_t * sparse_row,
                        matrix_t *        matrix,
                        matrix_t *        row
                      )
```

Считывает параметры генерируемых матриц и вызывает функцию генерации

Parameters

sparse_matrix	Указатель на разреженную матрицу
sparse_row	Указатель на разреженную вектор-строку
matrix	Указатель на матрицу в стандартном виде
row	Указатель на вектор-строку в стандартном виде

Returns

Код ошибки

◆ matrix_init()

```
int matrix_init ( matrix_t * matrix,
                  uint       rows,
                  uint       columns,
                  uint       nonzeros_num
                )
```

Выделяет память и инициализирует нулями матрицу в стандарном виде

Parameters

matrix	Указатель на матрицу в стандартном виде
rows	Количество строк
columns	Количество столбцов
nonzeros_num	Количество ненулевых элементов

Returns

Код ошибки

◆ print_matrix()

int print_matrix (**matrix_t** * matrix)

Печатает матрицу в стандарном виде

Parameters

matrix Указатель на матрицу в стандартном виде

Returns

Код ошибки

◆ print_sparse()

int print_sparse (**sparse_matrix_t** * matrix)

Печатает матрицу в разреженном виде

Parameters

matrix Указатель на матрицу в стандартном виде

Returns

Код ошибки

◆ read_matrix()

```
int read_matrix ( matrix_t *          matrix,
                  sparse_matrix_t *   sparse_matrix,
                  uint *const         rows_num,
                  uint *const         columns_num
                )
```

Считывет матрицу

Parameters

sparse_matrix Указатель на разреженную матрицу

matrix Указатель на матрицу в стандартном виде

rows_num Указатель на количество строк

columns_num Указатель на количество столбцов

Returns

Код ошибки

◆ read_matrix_elements()

```
int read_matrix_elements ( sparse_matrix_t *  sparse_matrix,
                           matrix_t *         matrix,
                           const uint         nonzeros
                         )
```

Считывает элементы матрицы

Parameters

sparse_matrix	Указатель на разреженную матрицу
matrix	Указатель на матрицу в стандартном виде
nonzeros	Количество ненулевый элементов

Returns

Код ошибки

◆ read_matrixes()

```
int read_matrixes ( sparse_matrix_t *  sparse_matrix,
                    sparse_matrix_t *  sparse_row,
                    matrix_t *         matrix,
                    matrix_t *         row
                  )
```

Вызывает функции считывание матрицы и веткора строки

Parameters

sparse_matrix	Указатель на разреженную матрицу
sparse_row	Указатель на разреженную вектор-строку
matrix	Указатель на матрицу в стандартном виде
row	Указатель на вектор-строку в стандартном виде

Returns

Код ошибки

◆ read_row()

```
int read_row ( matrix_t *         row,
               sparse_matrix_t *  sparse_row,
               const uint         columns_num
             )
```

Считывает вектор-строку

Parameters

sparse_row	Указатель на разреженную вектор-строку
row	Указатель на вектор-строку в стандартном виде
columns_num	Количество столбцов матрицы

Returns

Код ошибки

◆ read_row_elements()

```
int read_row_elements ( sparse_matrix_t *  sparse_row,
                        matrix_t *         row,
                        const uint         nonzeros
                      )
```

Считывает элементы веткора-строки

Parameters

sparse_row	Указатель на разреженную вектор-строку
row	Указатель на вектор-строку в стандартном виде
nonzeros	Количество ненулевых элементов

Returns

Код ошибки

◆ sparse_matrix_init()

```
int sparse_matrix_init ( sparse_matrix_t *  matrix,
                         uint               rows,
                         uint               columns,
                         uint               nonzero_num
                       )
```

Выделяет память под разреженную матрицу

Parameters

sparse_matrix	Указатель на разреженную матрицу
rows	Количество строк
columns	Количество столбцов
nonzero_num	Количество ненулевых элементов

Returns

Код ошибки

◆ choice_print()

```
int choice_print ( matrix_t *         matrix,
                   sparse_matrix_t *  sparse_matrix
                 )
```

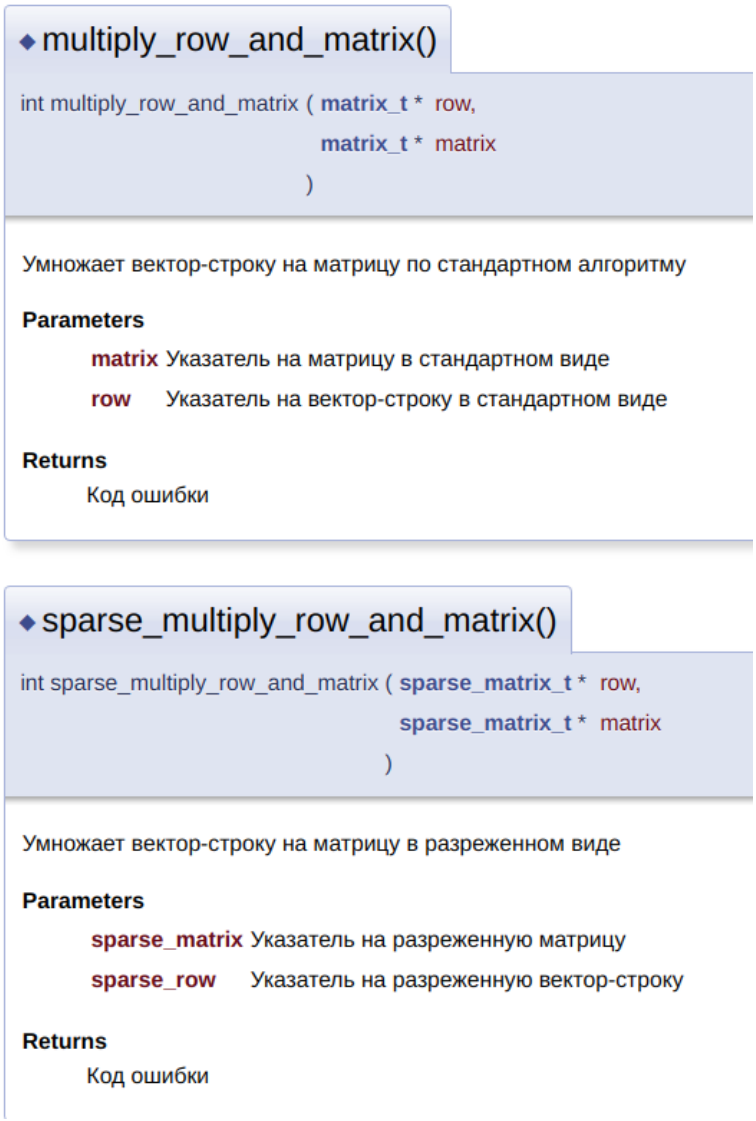
Выбор формата печати матрицы

Parameters

sparse_matrix	Указатель на разреженную матрицу
matrix	Указатель на матрицу в стандартном виде

Returns

Код ошибки



◆ multiply_row_and_matrix()

```
int multiply_row_and_matrix ( matrix_t *  row,
                              matrix_t *  matrix
                            )
```

Умножает вектор-строку на матрицу по стандартном алгоритму

Parameters

matrix Указатель на матрицу в стандартном виде

row Указатель на вектор-строку в стандартном виде

Returns

Код ошибки

◆ sparse_multiply_row_and_matrix()

```
int sparse_multiply_row_and_matrix ( sparse_matrix_t *  row,
                                     sparse_matrix_t *  matrix
                                   )
```

Умножает вектор-строку на матрицу в разреженном виде

Parameters

sparse_matrix Указатель на разреженную матрицу

sparse_row Указатель на разреженную вектор-строку

Returns

Код ошибки

Описание алгоритма

В случае умножения матриц в обычном формате используется стандартный алгоритм умножения матриц, при котором элемент результирующей матрицы C получается из элементов исходных матриц A и B следующим образом: $c_{ij} = \sum_{k=0}^{n-1} a_{ik} b_{kj}$.

При умножении матриц в разреженном формате производится проход по списку матрицы с номерами компонент вектора элементов и вектора их строк, ищется непустой столбец, а также номер элемента, на котором этот столбец заканчивается, поочередно выбираются элементы, соответствующие данному столбцу, умножаются на элементы вектора-строки в столбцах, номер которых совпадает с номером строки элемента матрицы, и добавляются в

сумму. После нахождения такой суммы для всех элементов столбца результат записывается в столбец, номер которого совпадает с номером текущего ненулевого элемента списка с номерами компонент.

Тесты

№	Что проверяется	Входные данные	Результат
1	Некорректный ввод пункта меню	jk	Введенный код не соответствует ни одному действию. Попробуйте ещё раз.
2	Некорректный ввод пункта меню (длинная строка)	hjkl	Некорректный ввод кода действия. Попробуйте ещё раз.
3	Некорректный ввод пункта меню (< 0)	-1	Введенный код не соответствует ни одному действию. Попробуйте ещё раз.
4	Некорректный ввод пункта меню (> 5)	6	Введенный код не соответствует ни одному действию. Попробуйте ещё раз.
5	Неверный ввод количества строк матрицы (не число) (пункты 1-2)	j	Ошибка при чтении числа!
6	Неверный ввод количества строк матрицы (< 0) (пункты 1-2)	-7	Введенное значение выходит за допустимый диапазон значений!
7	Неверный ввод количества столбцов матрицы (не число) (пункты 1-2)	j	Ошибка при чтении числа!
8	Неверный ввод количества строк матрицы (< 0) (пункты 1-2)	-9	Введенное значение выходит за допустимый диапазон значений!

9	Неверный ввод количества ненулевых элементов (не число)	jhk	Ошибка при чтении числа!
10	Неверный ввод ненулевых элементов (< 0)	-8	Введенное значение выходит за допустимый диапазон значений!
11	Неверный ввод ненулевых элементов (> n*m)	10 (при n =2, m=3)	Количество ненулевых элементов больше количества элементов в матрице!
12	Неверный ввод параметров элемента матрицы (не числа)	i 0 1	Неверные параметры элемента матрицы!
13	Неверный ввод параметров элемента матрицы (не числа)	0 j 1	Неверные параметры элемента матрицы!
14	Неверный ввод элемента матрицы (не числа)	0 0 el	Ошибка при чтении числа!
15	Неверный ввод строки элемента матрицы (> n)	10 1 1 (при n=3, m=3)	Неверные параметры элемента матрицы!
16	Неверный ввод столбца элемента матрицы (> m)	1 10 1 (при n=3, m=3)	Неверные параметры элемента матрицы!
17	Неверный ввод строки элемента матрицы (<0)	-1 1 1 (при n=3, m=3)	Неверные параметры элемента матрицы!
18	Неверный ввод столбца элемента матрицы (< 0)	1 -1 1 (при n=3, m=3)	Неверные параметры элемента матрицы!
19	Неверный ввод процента заполненности (не число)	j	Ошибка при чтении числа!

20	Неверный ввод процента заполненности (< 0)	-1	Введенное значение выходит за допустимый диапазон значений!
21	Неверный ввод процента заполненности	101	Процент -- целое число от 0 до 100!
22*	Умножение матриц обычным способом	$(1 \quad 0)\begin{pmatrix}1 & 0\\ 0 & 0\end{pmatrix}$ (ввести матрицы согласно правилам ввода)	Результат умножения вектора-строки на матрицу: 1 0
23*	Умножение матриц в разреженном виде	$(1 \quad 0)\begin{pmatrix}1 & 0\\ 0 & 0\end{pmatrix}$ (ввести матрицы согласно правилам ввода)	Результат умножения вектора-строки на матрицу: Значения элементов: 1 Номера строк этих элементов: 0 Номер элемента, с которого начинается k-ый столбец, где k - порядковый номер столбца: 0 -1 -1 в последнем массиве означает, что в столбце нет элементов

* Результаты 22-23 при любых корректно заданных матрицах должны совпадать

Оценка эффективности

Процент заполненности 10%:

	Стандартное представление		Разреженное представление	
	Время, такты	Память, байт	Время, такты	Память, байт
10x10	6	400	5	216
100x100	177	40800	38	9312
1000x1000	12974	4008000	1783	774584
10000x10000	1447256	400080000	66760	75587840

Процент заполненности 20%:

	Стандартное представление		Разреженное представление	
	Время, такты	Память, байт	Время, такты	Память, байт
10x10	7	480	3	280
100x100	159	40800	77	16624
1000x1000	13165	4008000	4286	1457168
10000x10000	1419833	40008000	116856	143783376

Процент заполненности 30%:

	Стандартное представление		Разреженное представление	
	Время, такты	Память, байт	Время, такты	Память, байт
10x10	7	480	5	424
100x100	174	40800	93	22776
1000x1000	19301	4008000	6109	2072808
10000x10000	1399772	40008000	187812	205488144

Процент заполненности 40%:

	Стандартное представление		Разреженное представление	
	Время, такты	Память, байт	Время, такты	Память, байт
10x10	8	480	7	456
100x100	156	40800	114	28144
1000x1000	14670	4008000	9414	2633704
10000x10000	1406352	40008000	265317	261342008

Процент заполненности 50%:

	Стандартное представление		Разреженное представление	
	Время, такты	Память, байт	Время, такты	Память, байт
10x10	7	480	7	536
100x100	179	40800	167	33392
1000x1000	14036	4008000	8718	3143200
10000x10000	1236972	40008000	352783	311870888

Процент заполненности 60%:

	Стандартное представление		Разреженное представление	
	Время, такты	Память, байт	Время, такты	Память, байт
10x10	8	480	8	576
100x100	159	40800	189	38000
1000x1000	13754	4008000	10533	3595040
10000x10000	1404742	40008000	435347	357571064

Процент заполненности 70%:

	Стандартное представление		Разреженное представление	
	Время, такты	Память, байт	Время, такты	Память, байт
10x10	8	480	9	644
100x100	157	40800	237	42128
1000x1000	11464	4008000	12113	4012312
10000x10000	1437093	40008000	501946	398930200

Процент заполненности 80%:

	Стандартное представление		Разреженное представление	
	Время, такты	Память, байт	Время, такты	Память, байт
10x10	6	480	4	640
100x100	178	40800	226	45904
1000x1000	12911	4008000	12376	4387544
10000x10000	1416699	40008000	547591	436356544

Процент заполненности 90%:

	Стандартное представление		Разреженное представление	
	Время, такты	Память, байт	Время, такты	Память, байт
10x10	8	480	8	736
100x100	177	40800	269	50168
1000x1000	12635	4008000	12807	4725928
10000x10000	1379509	40008000	565781	270188896

Процент заполненности 100%:

	Стандартное представление		Разреженное представление	
	Время, такты	Память, байт	Время, такты	Память, байт
10x10	9	480	11	776
100x100	241	40800	293	52592
1000x1000	13309	4008000	13235	5035072
10000x10000	1335553	40008000	575243	500818088

Контрольные вопросы

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

Разреженная матрица – это матрица с большим количеством нулей.

Схемы хранения разреженных матриц: схема Кнута (избыточна), её модификации, разреженный строчный формат, разреженный столбцовый формат.

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

Для обычной матрицы выделяется NxM ячеек памяти, где N – количество строк, M – количество столбцов, при динамическом выделении памяти есть несколько способов выделить память под стандартную матрицу, способ выбирается программистом под конкретную задачу. Для хранения разреженной матрицы способом, указанным в задании, выделяется память под хранение трех массивов, а количество ячеек памяти, выделяемых подо всю матрицу, можно рассчитать по формуле: 2*K + M, где K — количество ненулевых элементов, а M – количество столбцов.

Каков принцип обработки разреженной матрицы?

Такой алгоритм работает только с ненулевыми элементами матрицы, а, значит, количество операций, выполняемых при его работе, будет пропорционально числу таких элементов.

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

Выбор более эффективного алгоритма зависит от процента заполненности матриц. Случаи, в которых эффективнее выбрать тот или иной алгоритм, описаны в выводе.

Вывод

При проценте заполненности, меньшем 50%, алгоритм хранения и обработки разреженных матриц выигрывает как по памяти, так и по времени, для любых размерностей матриц. При проценте заполненности, большем или равном 50%, такой алгоритм для всех размерностей проигрывает по памяти стандартному алгоритму, что понятно, так как приходится хранить в 3 раза больше информации о каждом элементе. Со временем немного сложнее: с 60% алгоритм проигрывает стандартному для размерностей до 10000 элементов, с 70% эффективность такого алгоритма примерно такая же как и у стандартного для размерностей до 1000000 элементов, при больших размерностях алгоритм разреженных матриц всегда более эффективен, чем стандартный алгоритм, – что объясняется скоростью доступа к конкретному элементу матрицы.