



НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМ. Н.И. ЛОБАЧЕВСКОГО

ЦЕНТР КОМПЕТЕНЦИЙ ONEAPI В ННГУ

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МАТЕМАТИКИ И МЕХАНИКИ



НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ им. Н.И. ЛОБАЧЕВСКОГО
ЦЕНТР КОМПЕТЕНЦИЙ oneAPI в ННГУ
ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МАТЕМАТИКИ И МЕХАНИКИ

***ВВЕДЕНИЕ В АНАЛИЗ ПРОИЗВОДИТЕЛЬНОСТИ
И ОПТИМИЗАЦИЮ ПРОГРАММ***

**Транспонирование разреженных матриц
в формате CRS**

Содержание

- ❑ Цель работы
- ❑ Алгоритмы транспонирования разреженных матриц
- ❑ Тестовая задача

ЦЕЛЬ РАБОТЫ

Цель работы

- ❑ Реализовать и сравнить 2 алгоритма транспонирования разреженных матриц в формате CRS

Задачи:

- ❑ Реализовать транспонирование разреженной матрицы с предварительным формированием столбцов матрицы
- ❑ Реализовать транспонирование разреженной матрицы без использования дополнительной памяти
- ❑ Сравнить два реализованных алгоритма по скорости их работы

ФОРМАТ CRS

Формат CRS

- В формате CRS матрица хранится по строкам в следующем виде:
 - Массив *Value* – содержит все ненулевые значения элементов в матрице, записанные построчно (в строке элементы отсортированы по столбцам)
 - Массив *Column* – содержит значения столбцов всех ненулевых элементов
 - Массив *RowIndex* – содержит индекс начала каждой строки, если строка пустая, совпадает с предыдущим значением

Матрица A

1				2	
		3	4		
			8		5
	7	1			6

Структура хранения:

Value

1	2	3	4	8	5	7	1	6
---	---	---	---	---	---	---	---	---

Column

0	4	2	3	3	5	1	2	5
---	---	---	---	---	---	---	---	---

RowIndex

0	2	4	4	6	6	9
---	---	---	---	---	---	---

АЛГОРИТМЫ ТРАНСПОНИРОВАНИЯ РАЗРЕЖЕННЫХ МАТРИЦ

Алгоритмы транспонирования разреженных матриц

Транспонирование разреженной матрицы

- При транспонировании разреженной матрицы в формате CRS важно уметь выделить столбец исходной матрицы в отдельную сущность

Матрица A

1				2	
		3	4		
			8		5
	7	1			6

Структура хранения:

Value

1	2	3	4	8	5	7	1	6
---	---	---	---	---	---	---	---	---

Column

0	4	2	3	3	5	1	2	5
---	---	---	---	---	---	---	---	---

RowIndex

0	2	4	4	6	6	9
---	---	---	---	---	---	---

Алгоритмы транспонирования разреженных матриц

Алгоритм №1 с формированием столбцов матрицы

- ❑ Сформируем N одномерных векторов для хранения целых чисел, а также N векторов для хранения вещественных чисел.
- ❑ В цикле посмотрим все строки исходной матрицы, для каждой строки – все ее элементы. Пусть текущий элемент находится в строке i , столбце j , его значение равно v . Тогда добавим числа i и v в j -ые вектора для хранения целых и вещественных чисел (соответственно). Тем самым в векторах мы сформируем строки транспонированной матрицы.
- ❑ Последовательно скопируем данные из векторов в CRS-структуру транспонированной матрицы (*Column* и *Value*), попутно формируя массив *RowIndex*.

Алгоритмы транспонирования разреженных матриц

Алгоритм №1. С формированием столбцов матрицы

Матрица A

1				2	
		3	4		
			8		5
	7	1			6

Структура хранения A:

Value:

1	2	3	4	8	5	7	1	6
---	---	---	---	---	---	---	---	---

Column:

0	4	2	3	3	5	1	2	5
---	---	---	---	---	---	---	---	---

RowIndex:

0	2	4	4	6	6	9
---	---	---	---	---	---	---

Матрица A^T

1					
					7
	3				1
	4		8		
2					
			5		6

Структура хранения A^T:

Value:

1	7	3	1	4	8	2	5	6
---	---	---	---	---	---	---	---	---

Column:

0	5	1	5	1	3	0	3	5
---	---	---	---	---	---	---	---	---

RowIndex:

0	1	2	4	6	7	9
---	---	---	---	---	---	---

Вспомогательные вектора:

Value Vectors: *Index Vectors:*

1	
7	
3	1
4	8
2	
5	6

0	
5	
1	5
1	3
0	
3	5

❑ Основные недостатки:

- Требуется дополнительная память
- Неизвестен размер векторов при решении в один проход

Алгоритмы транспонирования разреженных матриц

Алгоритм №2 без использования дополнительной памяти

- ❑ Используется память, выделенная для хранения транспонируемой матрицы
- ❑ При первом проходе по матрице считается количество элементов в каждом столбце матрицы. Результат накапливается в *RowIndex* транспонированной матрицы
- ❑ Перед вторым проходом по матрице в *RowIndex* считается позиция начала каждого столбца в исходной матрице (для транспонированной матрицы – позиция начала каждой строки)
- ❑ При втором проходе по исходной матрице элементы раскладываются на свои места, заполняется *Value* и *Column* для транспонированной матрицы

Алгоритмы транспонирования разреженных матриц

Алгоритм №2. Иллюстрация

Матрица A

1				2	
		3	4		
			8		5
	7	1			6

Структура хранения A:

Value:

1	2	3	4	8	5	7	1	6
---	---	---	---	---	---	---	---	---

Column:

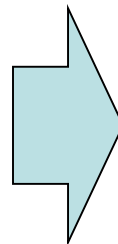
0	4	2	3	3	5	1	2	5
---	---	---	---	---	---	---	---	---

RowIndex:

0	2	4	4	6	6	9
---	---	---	---	---	---	---

RowIndex A^T

0
1
1
2
2
1
2



Алгоритмы транспонирования разреженных матриц

Алгоритм №2. Иллюстрация

Матрица A

1				2	
		3	4		
			8		5
	7	1			6

Структура хранения A:

Value:

1	2	3	4	8	5	7	1	6
---	---	---	---	---	---	---	---	---

Column:

0	4	2	3	3	5	1	2	5
---	---	---	---	---	---	---	---	---

RowIndex:

0	2	4	4	6	6	9
---	---	---	---	---	---	---

RowIndex A^T

0
1
1
2
2
1
2

RowIndex A^T

0
0
1
2
4
6
7

- ❑ После данной итерации в $RowIndex[i+1]$ хранится индекс элемента, в который надо положить элемент i -строки в транспонированной матрице
- ❑ При добавлении элемента к каждой строке соответствующее значение изменяется на $+1$

Алгоритмы транспонирования разреженных матриц

Алгоритм №2. Иллюстрация

Матрица A

1				2	
		3	4		
			8		5
	7	1			6

Структура хранения A:

Value:

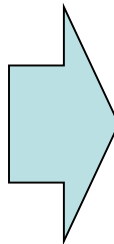
1	2	3	4	8	5	7	1	6
---	---	---	---	---	---	---	---	---

Column:

0	4	2	3	3	5	1	2	5
---	---	---	---	---	---	---	---	---

RowIndex:

0	2	4	4	6	6	9
---	---	---	---	---	---	---



RowIndex A^T

0
0
1
2
4
6
7

Матрица A^T

1					
					7
	3				1
	4		8		
2					
			5		6

Структура хранения A^T:

Value:

1						2		
---	--	--	--	--	--	---	--	--

Column:

0						0		
---	--	--	--	--	--	---	--	--

RowIndex:

0	1	1	2	4	7	7
---	---	---	---	---	---	---

Алгоритмы транспонирования разреженных матриц

Алгоритм №2. Иллюстрация

Матрица A

1				2	
		3	4		
			8		5
	7	1			6

Структура хранения A:

Value:

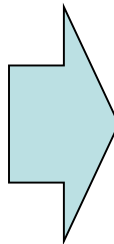
1	2	3	4	8	5	7	1	6
---	---	---	---	---	---	---	---	---

Column:

0	4	2	3	3	5	1	2	5
---	---	---	---	---	---	---	---	---

RowIndex:

0	2	4	4	6	6	9
---	---	---	---	---	---	---



RowIndex A^T

0
0
1
2
4
6
7

Матрица A^T

1					
					7
	3				1
	4		8		
2					
			5		6

Структура хранения A^T:

Value:

1		3		4		2		
---	--	---	--	---	--	---	--	--

Column:

0		1		1		0		
---	--	---	--	---	--	---	--	--

RowIndex:

0	1	1	3	5	7	7
---	---	---	---	---	---	---

Алгоритмы транспонирования разреженных матриц

Алгоритм №2. Иллюстрация

Матрица A

1				2	
		3	4		
			8		5
	7	1			6

Структура хранения A:

Value:

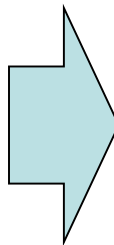
1	2	3	4	8	5	7	1	6
---	---	---	---	---	---	---	---	---

Column:

0	4	2	3	3	5	1	2	5
---	---	---	---	---	---	---	---	---

RowIndex:

0	2	4	4	6	6	9
---	---	---	---	---	---	---



RowIndex A^T

0
0
1
2
4
6
7

Матрица A^T

1					
					7
	3				1
	4		8		
2					
			5		6

Структура хранения A^T:

Value:

1		3		4	8	2	5	
---	--	---	--	---	---	---	---	--

Column:

0		1		1	3	0	3	
---	--	---	--	---	---	---	---	--

RowIndex:

0	1	1	3	6	7	8
---	---	---	---	---	---	---

Алгоритмы транспонирования разреженных матриц

Алгоритм №2. Иллюстрация

Матрица A

1				2	
		3	4		
			8		5
	7	1			6

Структура хранения A:

Value:

1	2	3	4	8	5	7	1	6
---	---	---	---	---	---	---	---	---

Column:

0	4	2	3	3	5	1	2	5
---	---	---	---	---	---	---	---	---

RowIndex:

0	2	4	4	6	6	9
---	---	---	---	---	---	---

Структура хранения A^T:

Value:

1	7	3	1	4	8	2	5	6
---	---	---	---	---	---	---	---	---

Column:

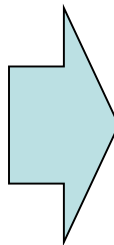
0	5	1	5	1	3	0	3	5
---	---	---	---	---	---	---	---	---

RowIndex:

0	1	2	4	6	7	9
---	---	---	---	---	---	---

RowIndex A^T

0
0
1
2
4
6
7



- ❑ Все массивы в конце работы алгоритма содержат ожидаемые числа

Матрица A^T

1					
					7
	3				1
	4		8		
2					
			5		6

ТЕСТОВАЯ ЗАДАЧА

Тестовая задача

- ❑ В качестве тестовой задачи предлагается использовать разреженную матрицу, созданную предложенным генератором
- ❑ Параметры тестовой задачи:

$$N = 100\,000$$

$$cntInRow = 500$$

$$NZ = 50\,000\,000$$

- ❑ Для реализации обоих алгоритмов возможно использование STL
- ❑ **Задание:** реализовать оба алгоритма, провести вычислительные эксперименты и сравнить время работы. **Сделать выводы.**

Тестовая задача

Код программы (1)

// см. приложенный код

```
crs_matrix.h
struct crsMatrix {
    size_t N; // Размер матрицы (N x N)
    size_t NZ; // Кол-во ненулевых элементов
    double * Value; // Массив значений (размер NZ)
    size_t * Column; // Массив номеров столбцов (размер NZ)
    size_t * RowIndex; // Массив индексов строк (размер N + 1)
};
```

Тестовая задача

Код программы (2)

// см. приложенный код

```
matrix_generate.cpp
#include "crs_matrix.h"
#include <random>
void GenerateRegularCRS(size_t N, size_t cntInRow, crsMatrix& mtx){
    std::random_device rd;
    std::mt19937 gen(rd());
    std::uniform_int_distribution<> idis(0, N - 1);
    std::uniform_real_distribution<> rdis(0, 1);
    size_t NZ = cntInRow * N;
    mtx.N = N;  mtx.Nz = NZ;
    mtx.Value = new double[NZ];
    mtx.Column = new size_t[NZ];
    mtx.RowIndex = new size_t[N + 1];
    for (size_t i = 0; i < N; i++) {
        // Формируем номера столбцов в строке i
        for (size_t j = 0; j < cntInRow; j++) {
            int f = 0;
            do {
                mtx.Col[i * cntInRow + j] = idis(gen);
                f = 0;
                for (size_t k = 0; k < j; k++)
                    if (mtx.Column[i * cntInRow + j] == mtx.Column[i * cntInRow + k])
                        f = 1;
            } while (f == 1);
        }
    }
```

```
// Сортируем номера столбцов в строке i
for (size_t j = 0; j < cntInRow - 1; j++)
    for (size_t k = 0; k < cntInRow - 1; k++)
        if (mtx.Column[i * cntInRow + k]
            > mtx.Column[i * cntInRow + k + 1]) {
            size_t tmp = mtx.Col[i * cntInRow + k];
            mtx.Column[i * cntInRow + k] =
                mtx.Column[i * cntInRow + k + 1];
            mtx.Column[i * cntInRow + k + 1] = tmp;
        }
}
// Заполняем массив значений
for (i = 0; i < cntInRow * N; i++)
    mtx.Value[i] = rdis(gen);
// Заполняем массив индексов строк
mtx.RowIndex[0] = 0;
for (i = 1; i <= N; i++)
    mtx.RowIndex[i] = mtx.RowIndex[i - 1] + cntInRow;
}
```

Литература

1. Джордж А., Лю Дж. Численное решение больших разреженных систем уравнений. — М.: Мир, 1984.
2. Писсанецки С. Технология разреженных матриц. — М.: Мир, 1988.
3. Stathis P., Cheresiz D., Vassiliadis S., Juurlink B. Sparse Matrix Transpose Unit // 18th International Parallel and Distributed Processing Symposium (IPDPS'04) – Papers, vol. 1, 2004.
[http://ce.et.tudelft.nl/publicationfiles/869_1_IPDPS2004paper.pdf]
4. Gustavson F. Two Fast Algorithms for Sparse Matrices: Multiplication and Permuted Transposition // ACM Transactions on Mathematical Software (TOMS), Volume 4 Issue 3, Sept. 1978. – Pp. 250-269.

Авторский коллектив

- ❑ Мееров Иосиф Борисович, к.т.н., доцент, зам. зав. каф. МОСТ
- ❑ Сысоев Александр Владимирович, к.т.н., доцент каф. МОСТ
- ❑ Линев Алексей Владимирович, зав. лаб. интернета вещей, каф. ПРИН
- ❑ Волокитин Валентин Дмитриевич, программист лаборатории СТиВВ, каф. МОСТ
- ❑ Козинов Евгений Александрович, к.т.н., преподаватель каф. МОСТ
- ❑ Панова Елена Анатольевна, инженер лаборатории СТиВВ, каф. МОСТ

Контакты

Нижегородский государственный университет

<http://www.unn.ru>

Центр компетенций oneAPI в ННГУ

<http://hpc-education.unn.ru/ru/центр-компетенций-oneapi-в-ннгу>

Институт информационных технологий, математики и механики

<http://www.itmm.unn.ru>