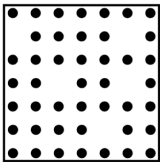


Разреженные матрицы и маскирование в GraphBLAS

Рустам Азимов

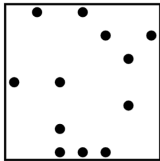
Разреженность матриц

A_{dense}



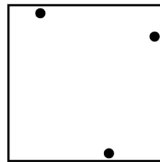
$$\text{nnz}(A_{\text{dense}}) \sim N^2$$

A_{sparse}



$$\text{nnz}(A_{\text{sparse}}) \sim N$$

$A_{\text{hypersparse}}$



$$\text{nnz}(A_{\text{hypersparse}}) \ll N$$

Разреженные форматы для матриц

- ▶ COO
- ▶ CSR, CSC
- ▶ Блочные, древовидные и т.д.

COO (coordinate list)

	0	1	2	3	4
0	2			7	
1		3		5	5
2					
3	1	1		6	9
4			2		3

{<row, col, val>}

{(0, 0, 2), (0, 3, 7),
 (1, 1, 3), (1, 3, 5), (1, 4, 5),
 (3, 0, 1), (3, 1, 1), (3, 3, 6), (3, 4, 9)
 (4, 2, 2), (4, 4, 3)}

nnz = 11
 (number of nonzero)

	0	1	2	3	4	5	6	7	8	9	10	11
cols:	0	3	1	3	4	0	1	3	4	2	4	
rows:	0	0	1	1	1	3	3	3	3	4	4	
vals:	2	7	3	5	5	1	1	6	9	2	3	

Память: $3 \times \text{nnz}$
 Доступ к ряду: $\log(\text{nnz})$

CSR (compressed sparse row)

COO row - major

	0	1	2	3	4	5	6	7	8	9	10	11
cols:	0	3	1	3	4	0	1	3	4	2	4	
rows:	0	0	1	1	1	3	3	3	3	4	4	
vals:	2	7	3	5	5	1	1	6	9	2	3	

CSC (compressed sparse column)

COO column - major

	0	1	2	3	4	5	6	7	8	9	10	11
cols:	0	0	1	1	2	3	3	3	4	4	4	
rows:	0	3	1	3	4	0	1	3	1	3	4	
vals:	2	1	3	1	2	7	5	6	5	9	3	

	0	1	2	3	4	5	6	7	8	9	10	11
rows_ptr:	0	2	5	5	9	11						
cols:	0	3	1	3	4	0	1	3	4	2	4	
vals:	2	7	3	5	5	1	1	6	9	2	3	

Память: $2 \times \text{nnz} + (\text{nrows} + 1)$
Доступ к ряду: $O(1)$

	0	1	2	3	4	5	6	7	8	9	10	11
rows_ptr:	0	2	4	5	8	11						
rows:	0	3	1	3	4	0	1	3	1	3	4	
vals:	2	1	3	1	2	7	5	6	5	9	3	

Память: $2 \times \text{nnz} + (\text{ncols} + 1)$
Доступ к столбцу: $O(1)$

Пример CSR

A	①	②	③	④	⑤	⑥	⑦
①		.3		.8			
②					.1		.7
③						.5	
④	.2		.4				
⑤						.1	
⑥			.5				
⑦			.1	.5	.9		

CSR representation of **A**:

row ptr	1	3	5	6	8	9	10	13				
col index	2	4	5	7	6	1	3	6	3	3	4	5
value	.3	.8	.1	.7	.5	.2	.4	.1	.5	.1	.5	.9

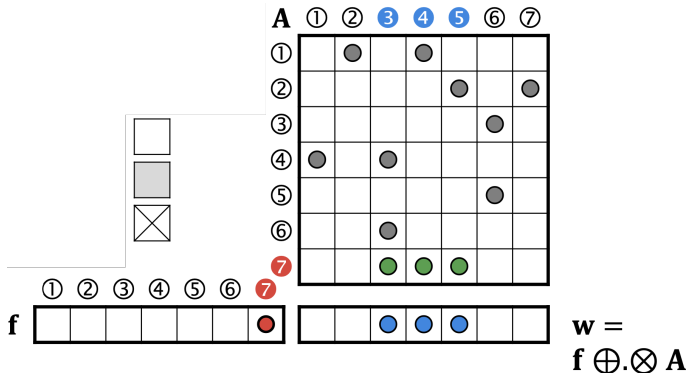
Разреженные матрицы в GraphBLAS

- ▶ Стандартно используется CSR, так как в большинстве алгоритмов интересуют выходящие рёбра $edge(i, j)$ и ячейки $A(i, j)$
- ▶ Также в некоторых случаях можно использовать CSC формат
- ▶ Для CSR и CSC также имеются hypersparse версии для очень разреженных матриц
- ▶ Для матрицы A размера $n \times m$
 - ▶ CSR — $O(n + e)$ памяти
 - ▶ CSC — $O(m + e)$
 - ▶ hypersparse CSR и CSC — $O(e)$

- ▶ Использование масок может существенно снизить сложность алгоритмов
- ▶ При правильных масках мы можем не делать кучу ненужных вычислений
- ▶ Например, в некоторых графах нам достаточно исследовать лишь небольшую их часть, но без масок нам придётся обойти весь граф

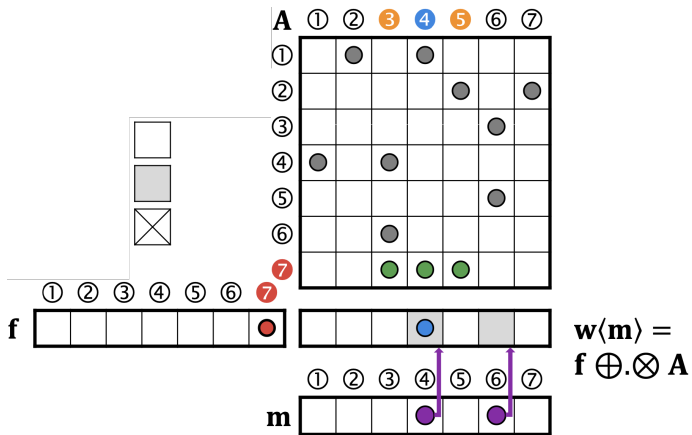
Маскирование в GraphBLAS

- ▶ Без маски
- ▶ С обычной маской
- ▶ С комплементарной маской



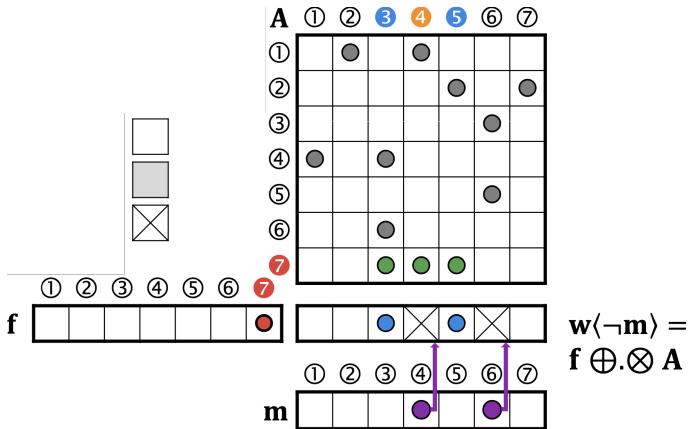
Маскирование в GraphBLAS

- ▶ Без маски
- ▶ С обычной маской
- ▶ С комплементарной маской

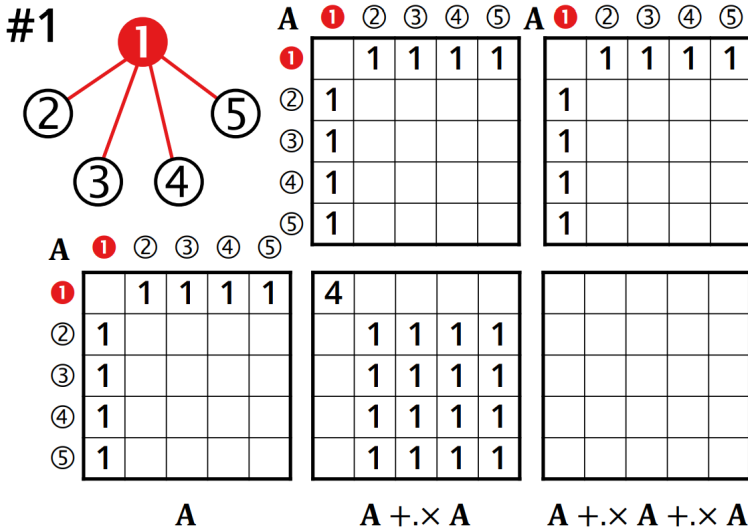


Маскирование в GraphBLAS

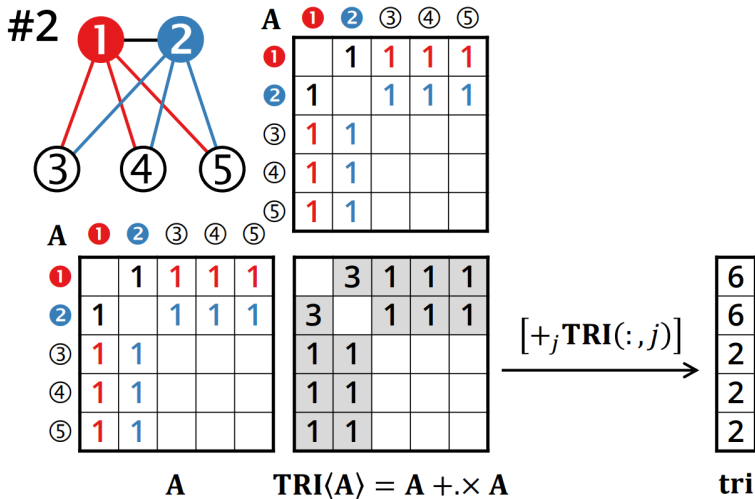
- ▶ Без маски
- ▶ С обычной маской
- ▶ С комплементарной маской



Важность маскирования



Важность маскирования



Модификаторы операций в GraphBLAS

- ▶ Транспонирование входных матриц
- ▶ Использование аккумулятора (accumulator operator)
- ▶ Маскирование (обычное или комплементарное)
- ▶ Очистка матрицы перед записью результата (replace)

- ▶ Пусть мы хотим вычислить $C = A \otimes B$
- ▶ Бывают ситуации, когда мы не хотим перезаписывать содержимое матрицы C , а хотим добавить новую информацию с помощью некоторого бинарного оператора \odot
- ▶ Аккумулятор позволяет задать такой бинарный оператор и при выполнении $C = A \otimes B$ на самом деле выполнится $C = C \odot (A \otimes B)$
- ▶ Пусть T — результат вычисления $A \otimes B$, тогда фаза использования аккумулятора выглядит следующим образом

Accumulator Phase: compute $\mathbf{Z} = \mathbf{C} \odot \mathbf{T}$:

if accum is NULL

$$\mathbf{Z} = \mathbf{T}$$

else

$$\mathbf{Z} = \mathbf{C} \odot \mathbf{T}$$

Replace

- ▶ После фазы использования аккумулятора следует фаза, в которой используются маска и replace опция

Mask/Replace Phase: compute $C\langle M \rangle = Z$:

if (GrB_REPLACE) delete all entries in C

if Mask is NULL

if (GrB_COMP)

C is not modified

else

$C = Z$

else

if (GrB_COMP)

$C\langle \neg M \rangle = Z$

else

$C\langle M \rangle = Z$

$$\mathbf{C} \langle \neg \mathbf{M} \rangle \odot = \oplus . \otimes (\mathbf{A}^T, \mathbf{B}^T)$$

mxm(Matrix *C, Matrix M, BinaryOp accum, Semiring op, Matrix A, Matrix B, Descriptor desc)



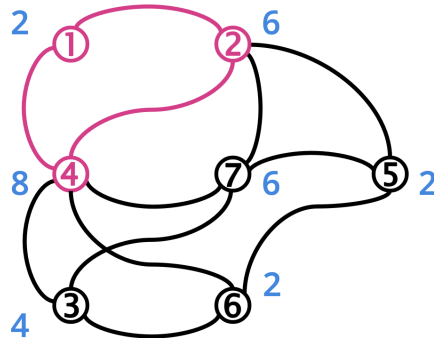
A. Buluç et al.: *Design of the GraphBLAS C API*, GABB@IPDPS 2017

Нотация GraphBLAS

symbol	operation	notation
$\oplus \otimes$	matrix-matrix multiplication	$\mathbf{C}\langle \mathbf{M} \rangle \odot = \mathbf{A} \oplus \otimes \mathbf{B}$
	vector-matrix multiplication	$\mathbf{w}\langle \mathbf{m} \rangle \odot = \mathbf{v} \oplus \otimes \mathbf{A}$
	matrix-vector multiplication	$\mathbf{w}\langle \mathbf{m} \rangle \odot = \mathbf{A} \oplus \otimes \mathbf{v}$
\otimes	element-wise multiplication	$\mathbf{C}\langle \mathbf{M} \rangle \odot = \mathbf{A} \otimes \mathbf{B}$
	(set intersection of patterns)	$\mathbf{w}\langle \mathbf{m} \rangle \odot = \mathbf{u} \otimes \mathbf{v}$
\oplus	element-wise addition	$\mathbf{C}\langle \mathbf{M} \rangle \odot = \mathbf{A} \oplus \mathbf{B}$
	(set union of patterns)	$\mathbf{w}\langle \mathbf{m} \rangle \odot = \mathbf{u} \oplus \mathbf{v}$
f	apply unary operator	$\mathbf{C}\langle \mathbf{M} \rangle \odot = f(\mathbf{A})$
		$\mathbf{w}\langle \mathbf{m} \rangle \odot = f(\mathbf{v})$
$[\oplus \dots]$	reduce to vector	$\mathbf{w}\langle \mathbf{m} \rangle \odot = [\oplus_j \mathbf{A}(:, j)]$
	reduce to scalar	$s \odot = [\oplus_{ij} \mathbf{A}(i, j)]$
\mathbf{A}^T	transpose matrix	$\mathbf{C}\langle \mathbf{M} \rangle \odot = \mathbf{A}^T$
-	extract submatrix	$\mathbf{C}\langle \mathbf{M} \rangle \odot = \mathbf{A}(i, j)$
		$\mathbf{w}\langle \mathbf{m} \rangle \odot = \mathbf{v}(i)$
-	assign submatrix with submask for $\mathbf{C}(\mathbf{I}, \mathbf{J})$	$\mathbf{C}\langle \mathbf{M} \rangle(i, j) \odot = \mathbf{A}$
		$\mathbf{w}\langle \mathbf{m} \rangle(i) \odot = \mathbf{v}$
-	assign submatrix with mask for \mathbf{C}	$\mathbf{C}(i, j)\langle \mathbf{M} \rangle \odot = \mathbf{A}$
		$\mathbf{w}(i)\langle \mathbf{m} \rangle \odot = \mathbf{v}$
-	apply select operator (GxB)	$\mathbf{C}\langle \mathbf{M} \rangle \odot = f(\mathbf{A}, k)$
		$\mathbf{w}\langle \mathbf{m} \rangle \odot = f(\mathbf{v}, k)$
-	Kronecker product	$\mathbf{C}\langle \mathbf{M} \rangle \odot = \text{kron}(\mathbf{A}, \mathbf{B})$

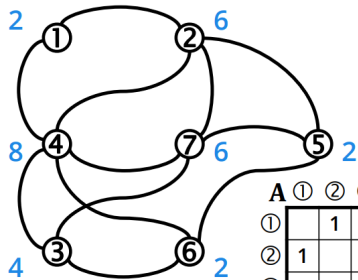
Задача подсчёта треугольников

- ▶ Такая задача была поставлена в IEEE GraphChallenge 2017
- ▶ По заданному графу требуется определить количество "треугольников"
- ▶ "Треугольник" — множество, состоящее из трёх попарно смежных вершин графа
- ▶ Какой ответ на задачу для данного примера?



Задача подсчёта треугольников: наивный подход

$$\text{tri2} = \text{diag}^{-1}(\mathbf{A} +. \times \mathbf{A} +. \times \mathbf{A})$$



A

	①	②	③	④	⑤	⑥	⑦
①		1		1			
②	1			1	1		1
③				1		1	1
④	1	1	1			1	1
⑤		1				1	1
⑥			1	1	1		
⑦		1	1	1	1		

A

	①	②	③	④	⑤	⑥	⑦
①		1		1			
②	1			1	1		1
③				1		1	1
④	1	1	1			1	1
⑤		1				1	1
⑥			1	1	1		
⑦		1	1	1	1		

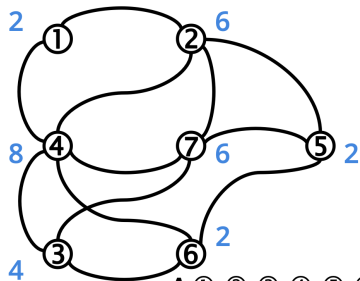
A

	①	②	③	④	⑤	⑥	⑦
①		1		1			
②	1			1	1		1
③				1		1	1
④	1	1	1			1	1
⑤		1				1	1
⑥			1	1	1		
⑦		1	1	1	1		

tri2

2	6	4	7	4	3	4	2
6	6	6	11	8	5	9	6
4	6	4	8	4	7	9	4
7	11	8	8	5	10	12	8
4	8	4	5	2	8	9	2
3	5	7	10	8	2	4	2
4	9	9	12	9	4	6	6

Задача подсчёта треугольников: поэлементное умножение



A	①	②	③	④	⑤	⑥	⑦
①		1		1			
②	1			1	1		1
③				1		1	1
④	1	1	1			1	1
⑤		1				1	1
⑥			1	1	1		
⑦		1	1	1	1		

$$\mathbf{TRI} = \mathbf{A} \oplus \cdot \otimes \mathbf{A} \otimes \mathbf{A}$$

$$\mathbf{tri2} = [\oplus_j \mathbf{TRI}(:, j)]$$

A	①	②	③	④	⑤	⑥	⑦
①		1		1			
②	1			1	1		1
③				1		1	1
④	1	1	1			1	1
⑤		1				1	1
⑥			1	1	1		
⑦		1	1	1	1		

2	1	1	1	1	1	1	2
1	4	2	2	1	2	2	
1	2	3	2	2	1	1	
1	2	2	5	3	1	2	
1	1	2	3	3		1	
1	2	1	1		3	3	
2	2	1	2	1	3	4	

TRI

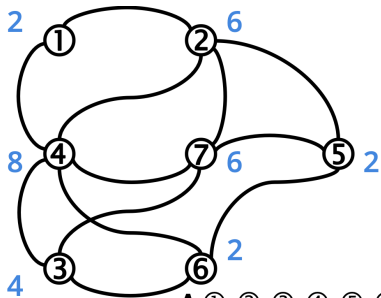
	1		1				
1			2	1		2	
			2		1	1	
1	2	2			1	2	
	1					1	
		1	1				
	2	1	2	1			

$[\oplus_j \dots]$

tri2

2
6
4
8
2
2
6

Задача подсчёта треугольников: маскирование



A	①	②	③	④	⑤	⑥	⑦
①		1		1			
②	1			1	1		1
③				1		1	1
④	1	1	1			1	1
⑤		1				1	1
⑥			1	1	1		
⑦		1	1	1	1		

Masking limits where the operation is computed. Here, we use A as a mask for $A \oplus \cdot \otimes A$.

$$\text{TRI}\langle A \rangle = A \oplus \cdot \otimes A$$

$$\text{tri2} = [\oplus_j \text{TRI}(:, j)]$$

A	①	②	③	④	⑤	⑥	⑦
①		1		1			
②	1			1	1		1
③				1		1	1
④	1	1	1			1	1
⑤		1				1	1
⑥			1	1	1		
⑦		1	1	1	1		

	1		1				
1			2	1		2	
			2		1	1	
1	2	2			1	2	
	1					1	
		1	1				
	2	1	2	1			


$[+_j \dots]$

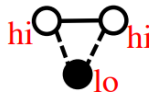
tri2

2
6
4
8
2
2
6

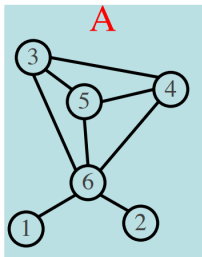
Cohen's algorithm

 - Count triangles by lowest-degree vertex.

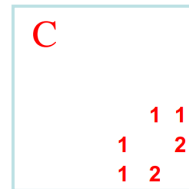
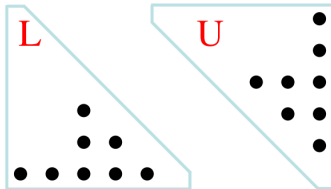
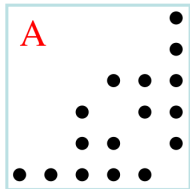
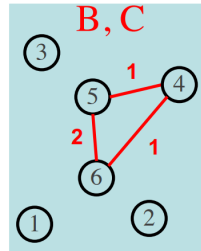
 - Enumerate "low-hinged" wedges.

 - Keep wedges that close.

Cohen's algorithm



$$\begin{aligned} A &= L + U && (\text{hi} \rightarrow \text{lo} + \text{lo} \rightarrow \text{hi}) \\ L \times U &= B && (\text{wedge, low hinge}) \\ A \wedge B &= C && (\text{closed wedge}) \\ \text{sum}(C)/2 &= && \mathbf{4 \text{ triangles}} \end{aligned}$$



Azad, B., Gilbert. "Parallel triangle counting and enumeration using matrix algebra". *IPDPSW, 2015*

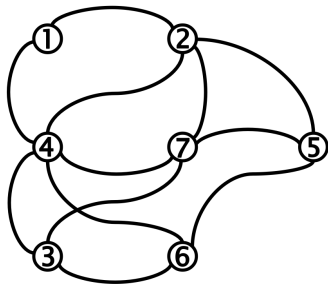
Input: adjacency matrix A

Output: triangle count t

Workspace: matrices L, U, B, C

1. $L = \text{tril}(A)$ extract the lower triangle from A
2. $U = \text{triu}(A)$ extract the upper triangle from A
3. $B = L \oplus \cdot \otimes U$
4. $C = B \otimes A$
5. $t = \sum C / 2$ sum the values in C and divide by 2

Cohen's algorithm: пример



L

	①	②	③	④	⑤	⑥	⑦
①							
②	1						
③							
④	1	1	1				
⑤		1					
⑥			1	1	1		
⑦		1	1	1	1		

U

	①	②	③	④	⑤	⑥	⑦
①		1		1			
②				1	1		1
③				1		1	1
④						1	1
⑤						1	1
⑥							
⑦							

B

	①	②	③	④	⑤	⑥	⑦
①							
②		1		1			
③							
④		1		3	1	1	2
⑤				1	1		1
⑥				1		3	3
⑦				2	1	3	4

A

	①	②	③	④	⑤	⑥	⑦
①		1		1			
②	1			1	1		1
③				1		1	1
④	1	1	1			1	1
⑤		1				1	1
⑥			1	1	1		
⑦		1	1	1	1		

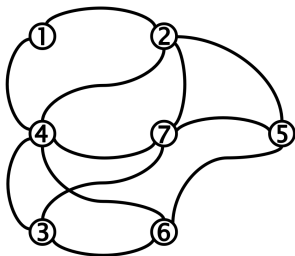
⊗

C

	①	②	③	④	⑤	⑥	⑦
①							
②				1			
③							
④		1				1	2
⑤							1
⑥				1			
⑦				2	1		

$$t = \sum c/2$$

Cohen's algorithm: пример с маской



U	①	②	③	④	⑤	⑥	⑦
①		1		1			
②				1	1		1
③				1		1	1
④						1	1
⑤						1	1
⑥							
⑦							

L	①	②	③	④	⑤	⑥	⑦
①							
②	1						
③							
④	1	1	1				
⑤		1					
⑥			1	1	1		
⑦		1	1	1	1		

C	①	②	③	④	⑤	⑥	⑦
①							
②				1			
③							
④		1				1	2
⑤							1
⑥				1			
⑦				2	1		

$$C(A) = L \oplus \cdot \otimes U$$

$$t = \sum C / 2$$

Cohen's algorithm: SuiteSparse:GraphBLAS

```
/*
 * Given, L, the lower triangular portion of n x n adjacency matrix A (of and
 * undirected graph), computes the number of triangles in the graph.
 */
uint64_t triangle_count(GrB_Matrix L)                // L: NxN, lower-triangular, bool
{
    GrB_Index n;
    GrB_Matrix_nrows(&n, L);                          // n = # of vertices

    GrB_Matrix C;
    GrB_Matrix_new(&C, GrB_UINT64, n, n);

    GrB_Monoid UInt64Plus;                             // integer plus monoid
    GrB_Monoid_new(&UInt64Plus, GrB_PLUS_UINT64, 0ul);

    GrB_Semiring UInt64Arithmetic;                     // integer arithmetic semiring
    GrB_Semiring_new(&UInt64Arithmetic, UInt64Plus, GrB_TIMES_UINT64);

    GrB_Descriptor desc_tb;                             // Descriptor for mxm
    GrB_Descriptor_new(&desc_tb);
    GrB_Descriptor_set(desc_tb, GrB_INP1, GrB_TRAN); // transpose the second matrix

    GrB_mxm(C, L, GrB_NULL, UInt64Arithmetic, L, L, desc_tb); //  $C \triangleleft L = L * L'$ 

    uint64_t count;
    GrB_reduce(&count, GrB_NULL, UInt64Plus, C, GrB_NULL); // 1-norm of C

    GrB_free(&C); // C matrix no longer needed
    GrB_free(&UInt64Arithmetic); // Semiring no longer needed
    GrB_free(&UInt64Plus); // Monoid no longer needed
    GrB_free(&desc_tb); // descriptor no longer needed

    return count;
}
```

<http://graphblas.org>

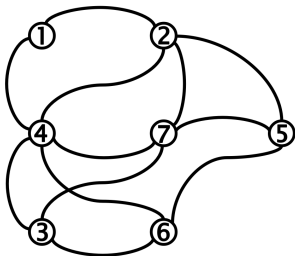
Input: adjacency matrix \mathbf{A}

Output: triangle count t

Workspace: matrices \mathbf{L} , \mathbf{U} , \mathbf{B} , \mathbf{C}

1. $\mathbf{L} = \text{tril}(\mathbf{A})$ extract the lower triangle from \mathbf{A}
2. $\mathbf{C}(\mathbf{L}) = \mathbf{L} \oplus . \otimes \mathbf{L}$ multiply matrices \mathbf{L} and \mathbf{L} using mask \mathbf{L}
3. $t = \sum \mathbf{C}$ sum the values in \mathbf{C}

Sandia algorithm: пример



L	①	②	③	④	⑤	⑥	⑦
①							
②	1						
③							
④	1	1	1				
⑤		1					
⑥			1	1	1		
⑦		1	1	1	1		

L	①	②	③	④	⑤	⑥	⑦
①							
②	1						
③							
④	1	1	1				
⑤		1					
⑥			1	1	1		
⑦		1	1	1	1		

C	①	②	③	④	⑤	⑥	⑦
①							
②							
③							
④	1						
⑤							
⑥			1				
⑦	2	1					

$$C\langle L \rangle = L \oplus \cdot \otimes L$$

$$t = \sum C$$

- ▶ Книга: Кернер J., Гилберт J. — "Graph algorithms in the language of linear algebra"
- ▶ Презентация: Gábor Szárnyas — "Introduction to GraphBLAS"
- ▶ Презентация: Aydın Buluç — Sparse Matrices Beyond Solvers: Graphs, Biology, and Machine Learning