A GraphBLAS solution to the SIGMOD 2014 Programming Contest using multi-source BFS

High-Performance Extreme Computing (HPEC) 2020

M. Elekes, A. Nagy, D. Sándor, J.B. Antal, T.A. Davis, G. Szárnyas

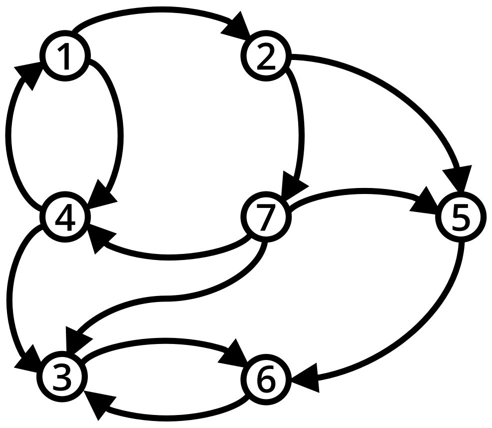
MŰEGYETEM 1782

TEXAS A&M UNIVERSITY

CWI Centrum Wiskunde & Informatica

MOTIVATION

- Graph algorithms are challenging to program
    - irregular access patterns → poor locality
    - caching and parallelization are difficult
- Optimizations often limit portability
- **GraphBLAS** introduces an abstraction layer using the language of linear algebra
    - graph ≡ sparse matrix
    - navigation step ≡ matvec on semirings

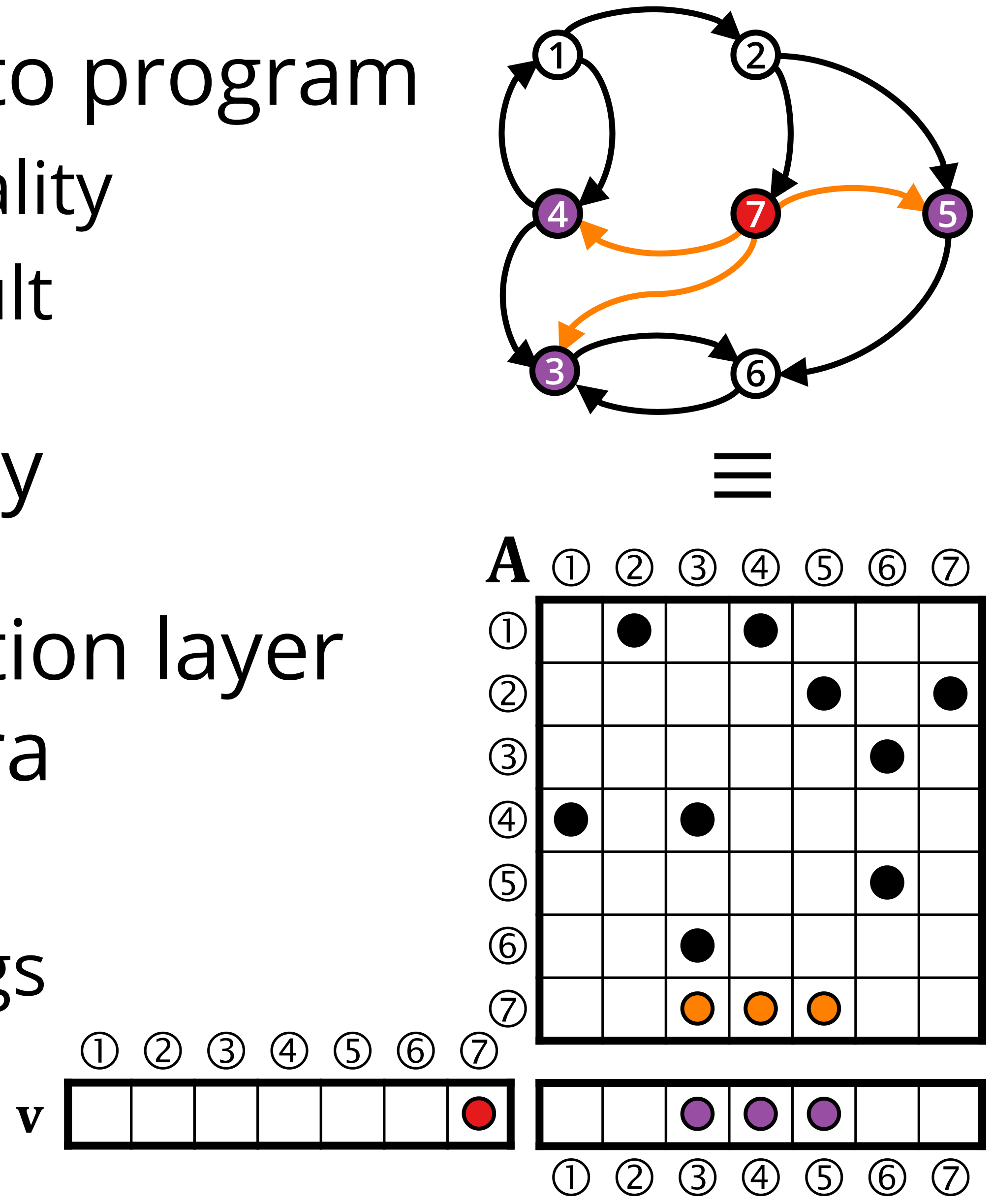


≡

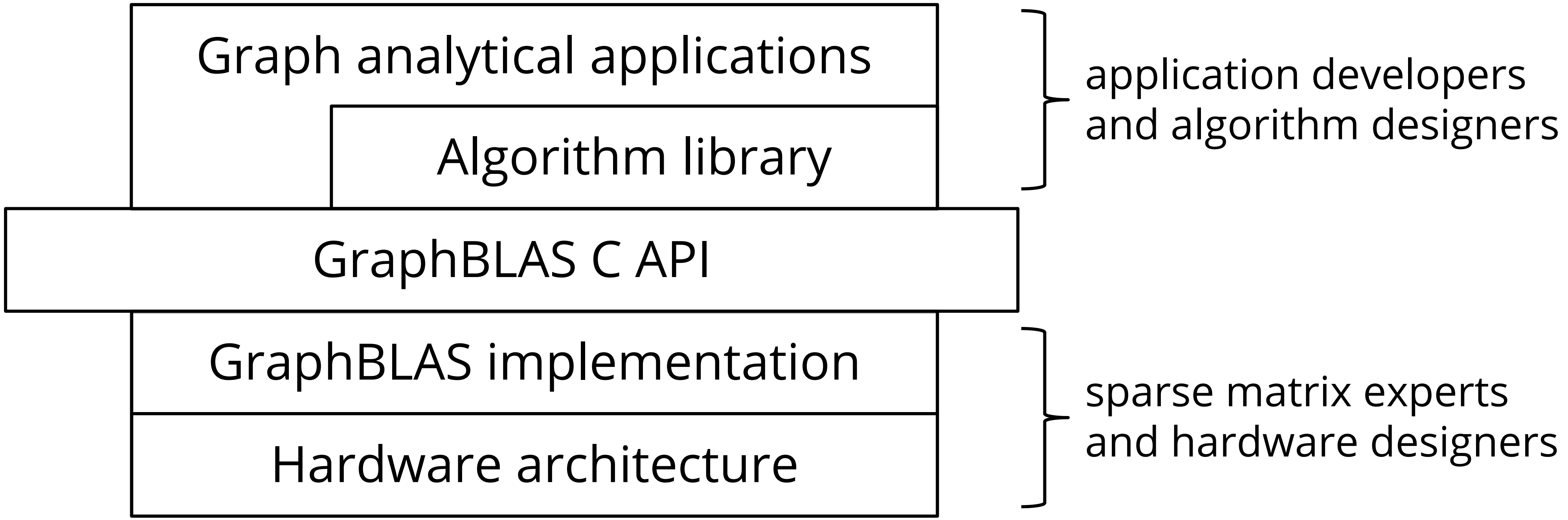
A	①	②	③	④	⑤	⑥	⑦
①		●		●			
②					●		●
③						●	
④	●		●				
⑤						●	
⑥			●				
⑦			●	●	●		

MOTIVATION

- Graph algorithms are challenging to program
  - irregular access patterns → poor locality
  - caching and parallelization are difficult
- Optimizations often limit portability
- **GraphBLAS** introduces an abstraction layer using the language of linear algebra
  - graph ≡ sparse matrix
  - navigation step ≡ matvec on semirings



GRAPHBLAS STACK



Textbook algos: BFS, PageRank, triangle count — untyped graphs

GraphBLAS-based Cypher engine: RedisGraph — property graphs

RQ: How to formulate mixed workloads on property graphs?

SIGMOD 2014 PROGRAMMING CONTEST

<u>Annual contest</u>

- Teams compete on database-related programming tasks
- Highly-optimized C++ implementations

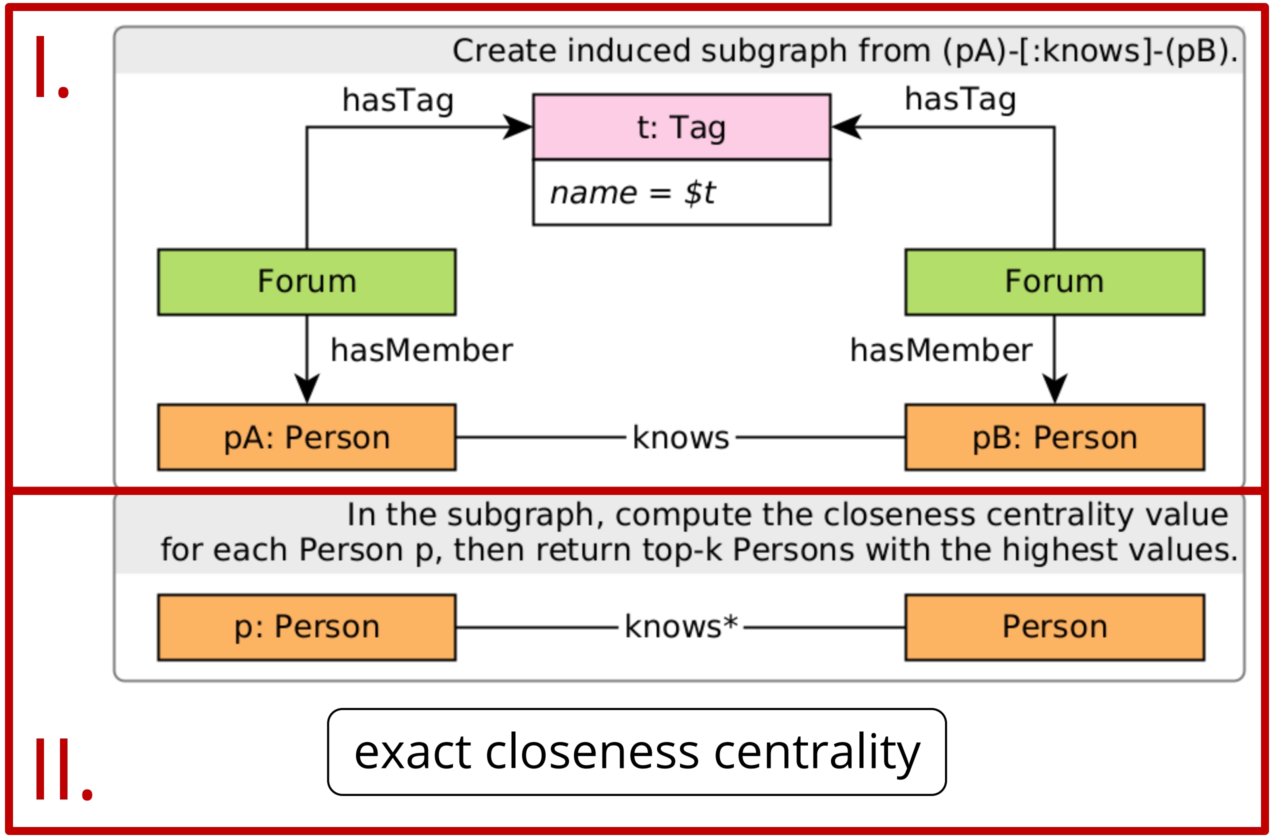
<u>2014 event</u>

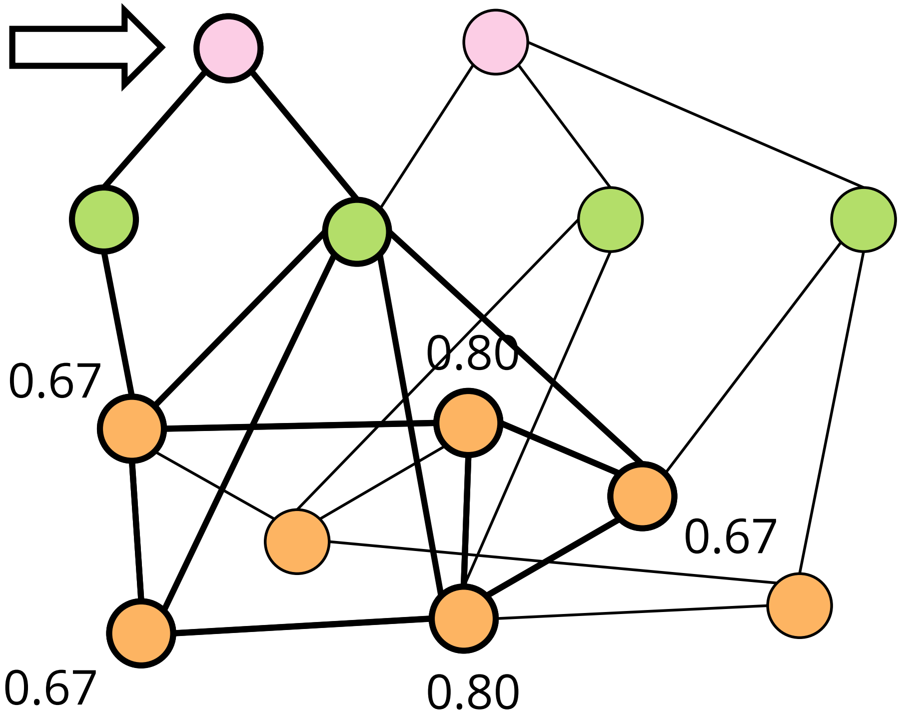
- Tasks on the LDBC social network graph
  - Benchmark data set for property graphs
  - People, forums, comments, hashtags, etc.
- 4 queries
  - Mix of filtering operations and graph algorithms

QUERY TEMPLATE

I. Compute an induced subgraph over Person-knows-Person

II. Run the graph algorithm on the subgraph

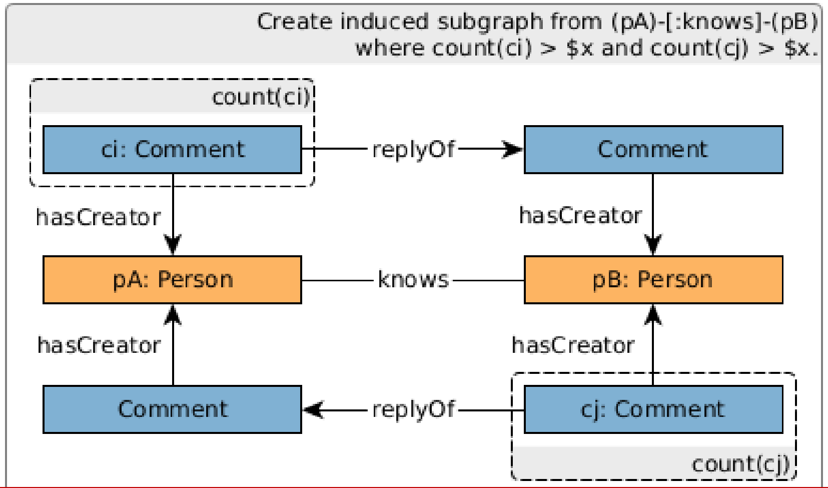


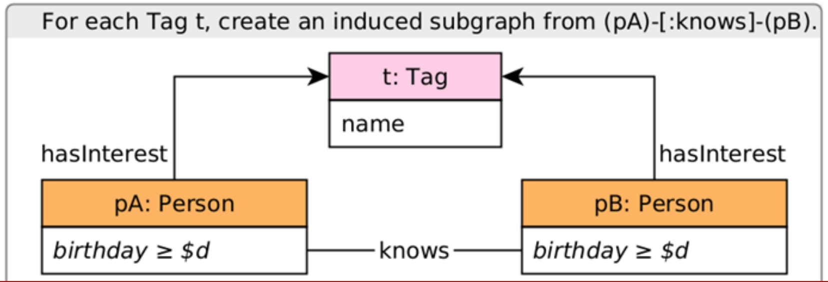


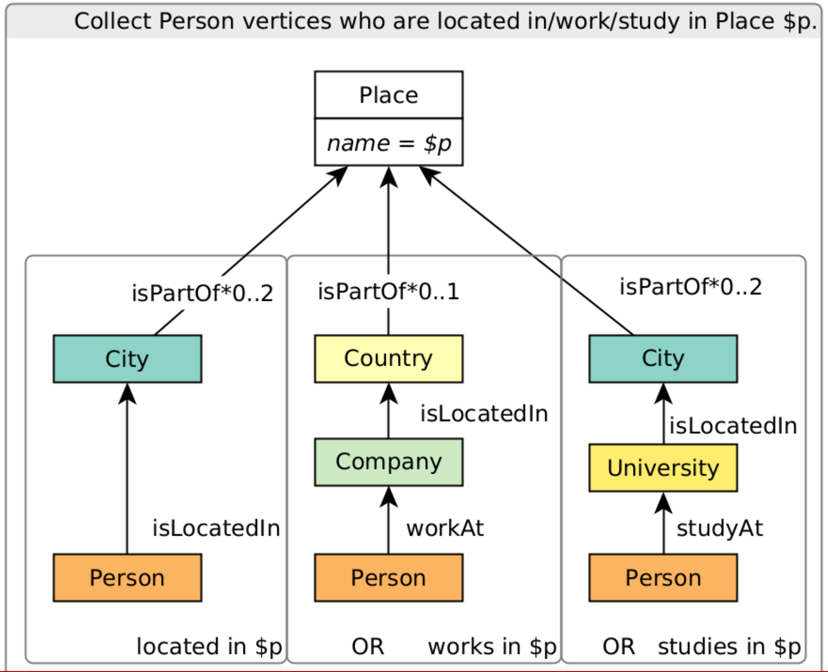
key kernel: **all-source BFS**

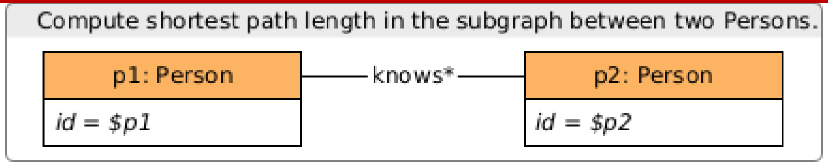
OVERVIEW OF QUERIES 1, 2, 3

I. Filter the induced subgraph(s)

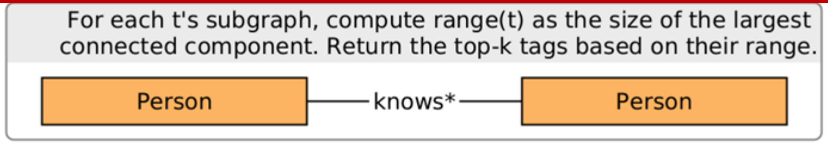




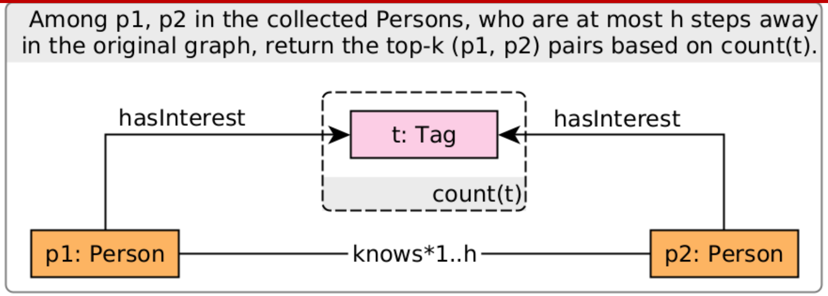




unweighted shortest path



connected components



pairwise reachability

II. Run the graph algorithm

GRAPHBLAS SOLUTION OF THE QUERIES

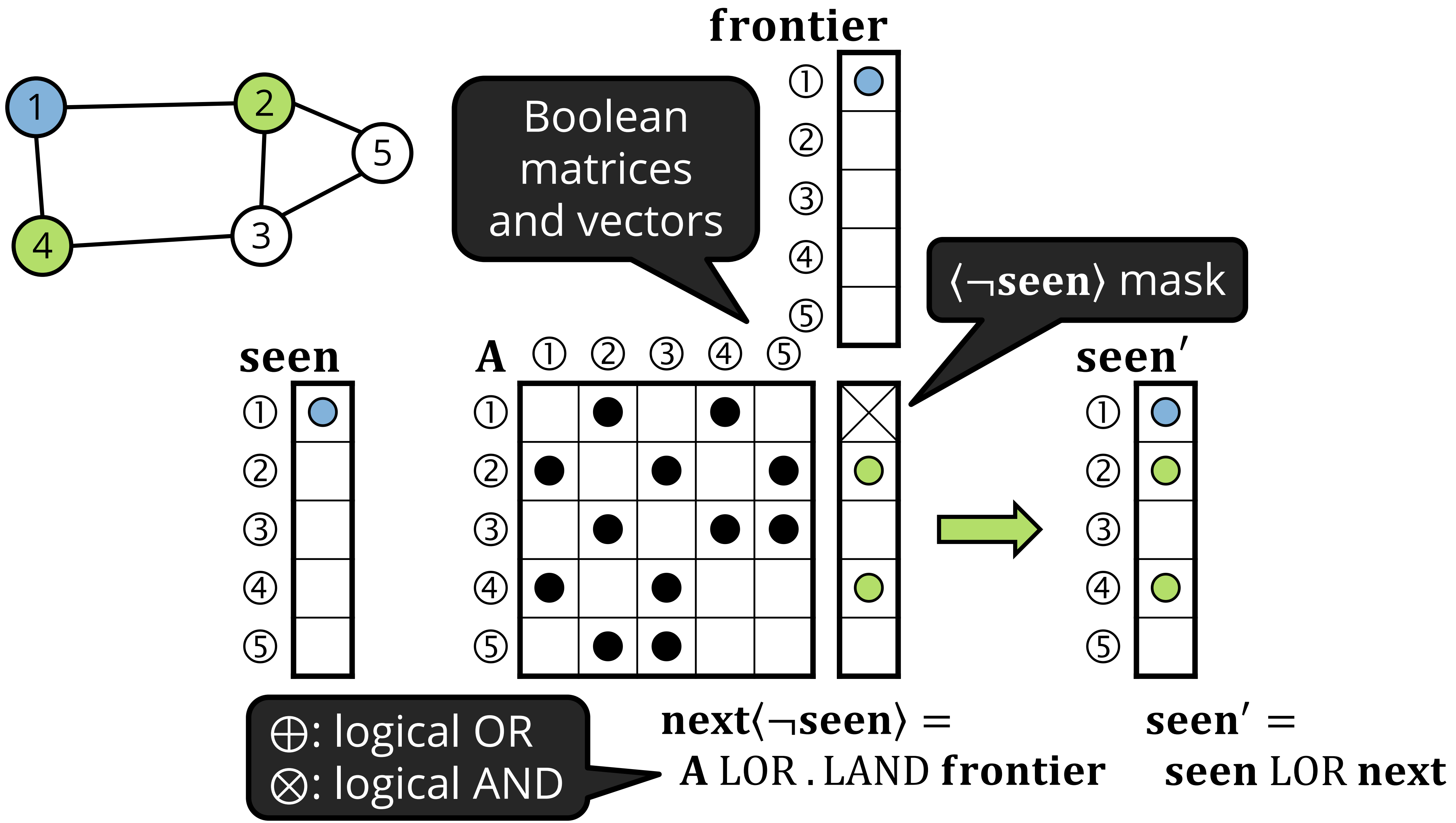
- **Loading** includes relabelling UINT64 vertex IDs to a contiguous sequence $0 \ldots N-1$.

- **Filtering the induced subgraph** from the property graph is mostly straightforward and composable with the algorithms.

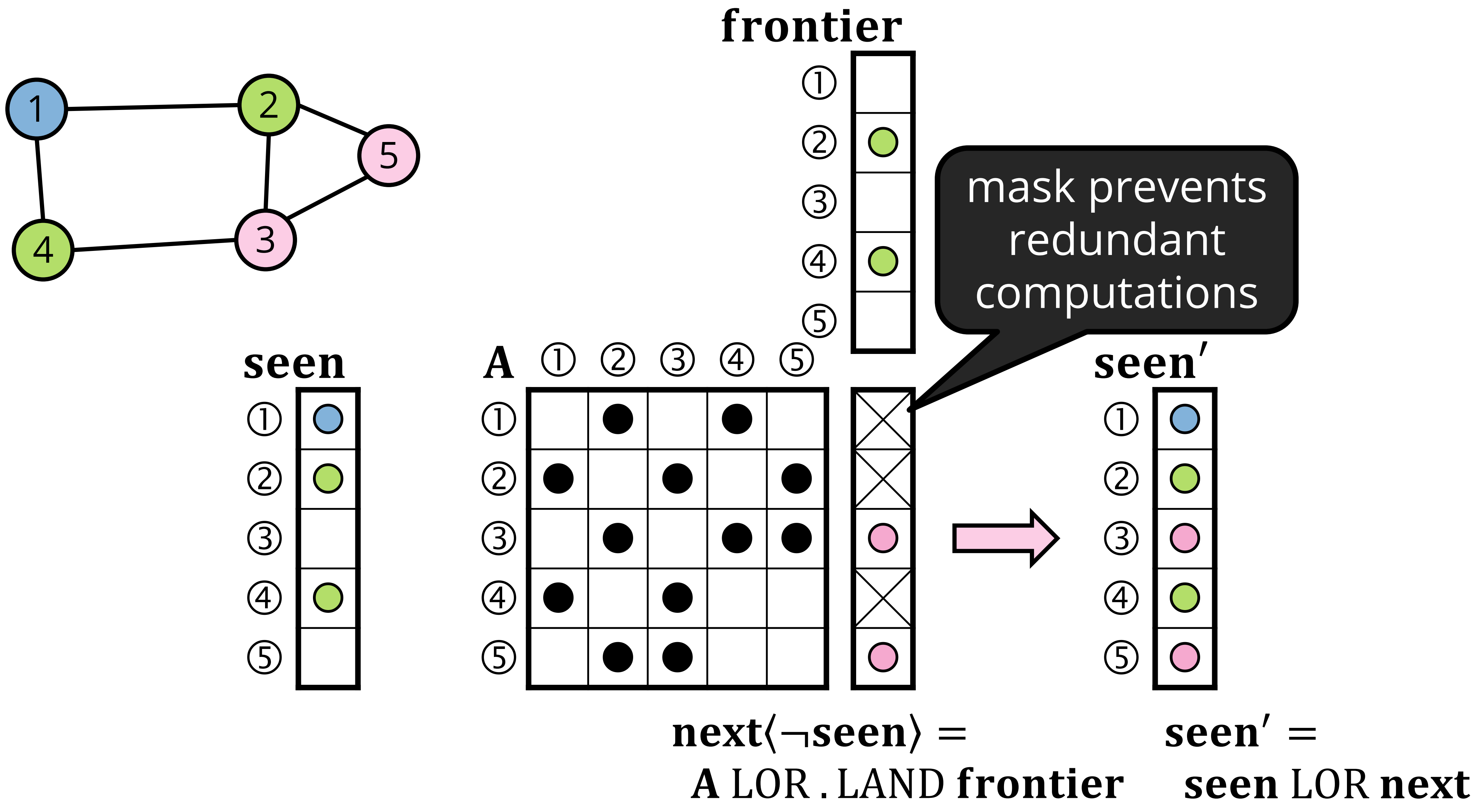
- **The algorithms** can be concisely expressed in GraphBLAS:
  - Connected components ✓ → FastSV [Zhang et al., PPSC'20]
  - BFS ✓
  - Bidirectional BFS
  - All-source BFS + bitwise optimization
  - Multi-source bidirectional BFS

BFS

BFS: BREADTH-FIRST SEARCH



BFS: BREADTH-FIRST SEARCH



All-source BFS

Q4: CLOSENESS CENTRALITY VALUES

Q4 computes the top-*k* Person vertices based on their exact *closeness centrality values*:

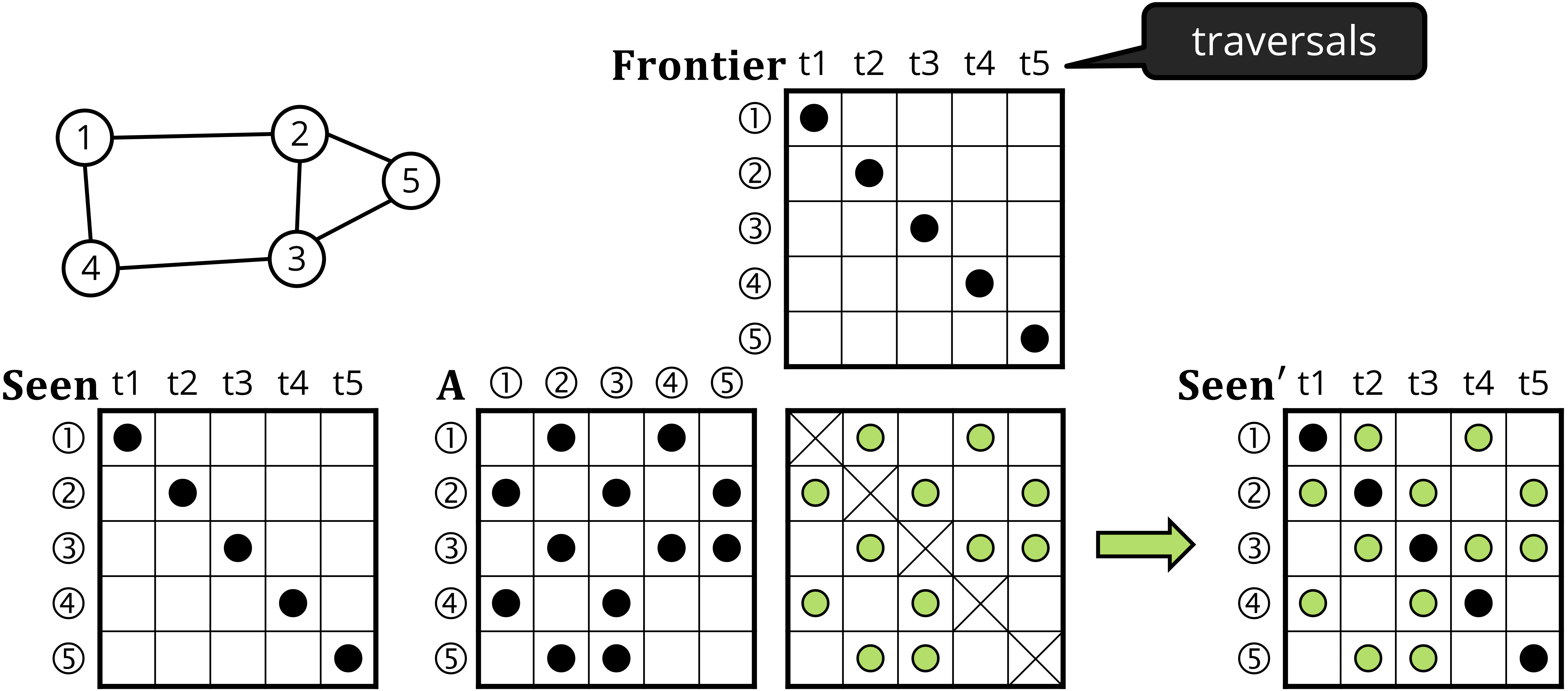
$$CCV(p) = \frac{(C(p)-1)^2}{(n-1)\cdot s(p)}$$

where

- $C(p)$ is the size of the connected component of vertex p,
- n is the number of vertices in the induced graph,
- $s(p)$ is the sum of geodesic distances to all other reachable persons from p.

$s(p)$ is challenging: needs unweighted all-pairs shortest paths.

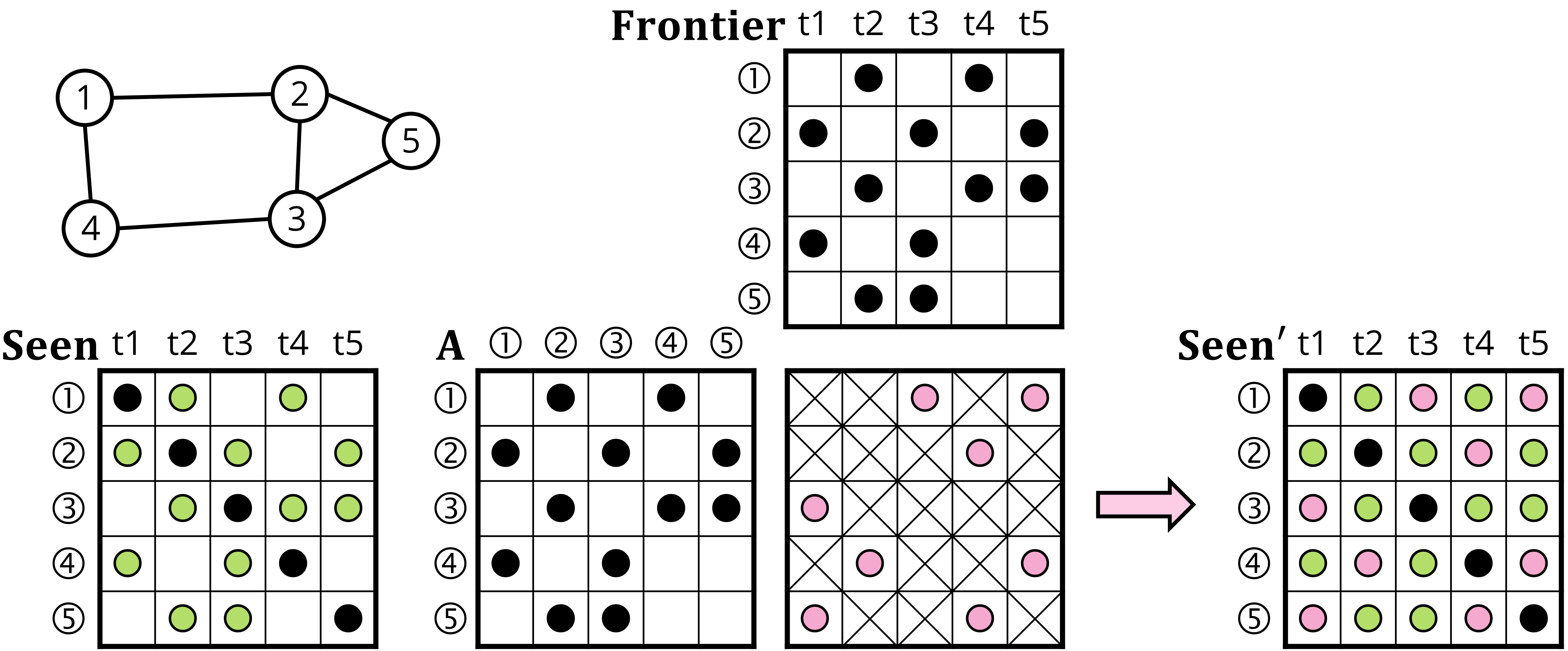
BOOLEAN ALL-SOURCE BFS ALGORITHM



Next$\langle \neg$**Seen**$\rangle =$
A LOR . LAND **Frontier**

Seen′ $=$
Seen LOR **Next**

BOOLEAN ALL-SOURCE BFS ALGORITHM



Next⟨¬**Seen**⟩ =
A LOR . LAND **Frontier**

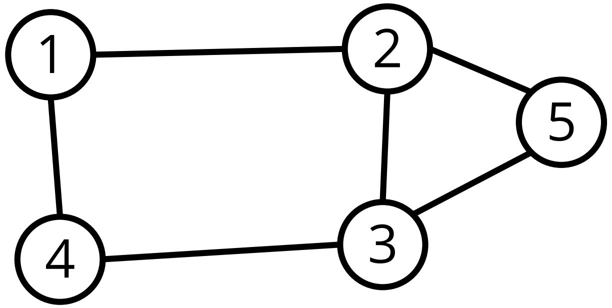
Seen′ =
Seen LOR **Next**

Bitwise all-source BFS

BITWISE ALL-SOURCE BFS ALGORITHM

- For large graphs, the all-source BFS algorithm might need to run 500k+ traversals
- Two top-ranking teams used bitwise operations to process traversals in batches of 64 [Then et al., VLDB'15]
- This idea can be adopted in the GraphBLAS algorithm by
  - using `UINT64` values
  - performing the multiplication on the $\mathrm{BOR}.\mathrm{SECOND}$ semiring, where BOR is "bitwise or" and $\mathrm{SECOND}(x, y) = y$
- 5-10x speedup compared to the Boolean all-source BFS

BITWISE ALL-SOURCE BFS ALGORITHM



Using UINT4s here

Frontier

	t1-t4	t5
①	1000	0000
②	0100	0000
③	0010	0000
④	0001	0000
⑤	0000	1000

Seen

	t1-t4	t5
①	1000	0000
②	0100	0000
③	0010	0000
④	0001	0000
⑤	0000	1000

A

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

0101	0000
1010	1000
0101	1000
1010	0000
0110	0000

$\mathbf{Next} =$
$\mathbf{A}\ \mathrm{BOR.SECOND}\ \mathbf{Frontier}$

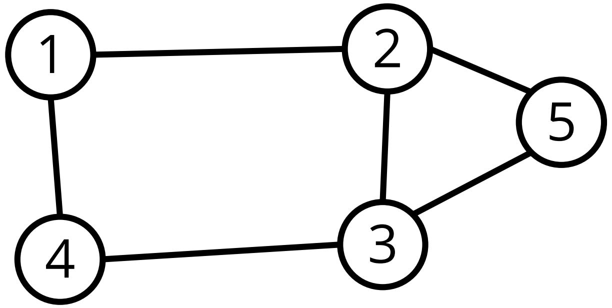
→

Seen′

	t1-t4	t5
①	0101	0000
②	1010	1000
③	0101	1000
④	1010	0000
⑤	0110	0000

$\mathbf{Seen'} =$
$\mathbf{Seen}\ \mathrm{BOR}\ \mathbf{Next}$

BITWISE ALL-SOURCE BFS ALGORITHM



Frontier

	t1-t4	t5
①	0101	0000
②	1010	1000
③	0101	1000
④	1010	0000
⑤	0110	0000

Full VLDB paper on this algorithm

vs.

9 GrB operations

Seen

	t1-t4	t5
①	0101	0000
②	1010	1000
③	0101	1000
④	1010	0000
⑤	0110	0000

A

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

0010	1000
0001	0000
1000	0000
0100	1000
1001	0000

Next = **A** BOR . SECOND **Frontier**

→

Seen′

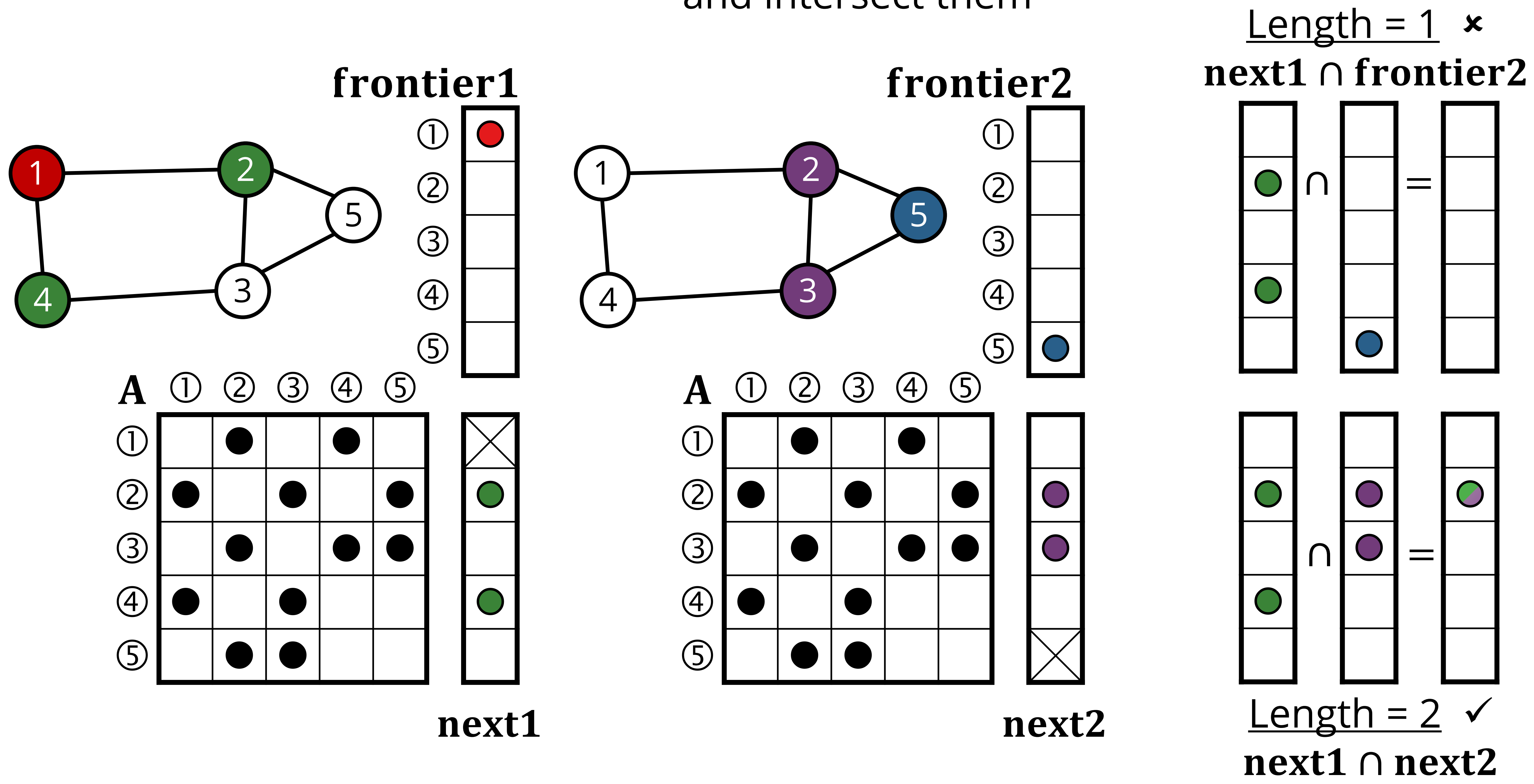
	t1-t4	t5
①	1111	1000
②	1111	1000
③	1111	1000
④	1111	1000
⑤	1111	1000

Seen′ = **Seen** BOR **Next**

Bidirectional BFS

BIDIRECTIONAL BFS

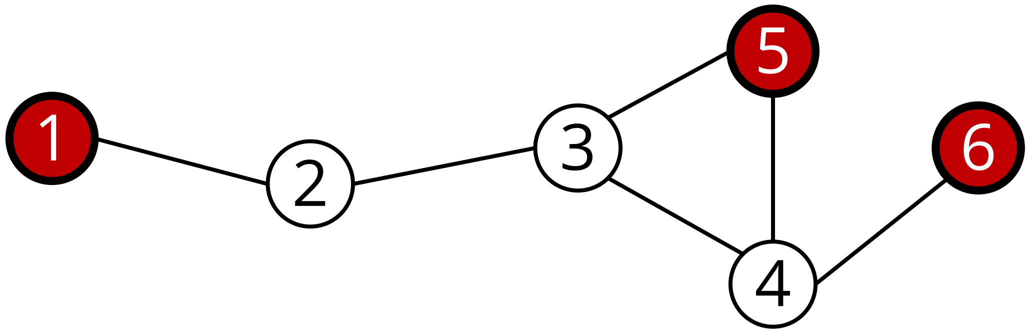
Advance frontiers alternately and intersect them



Bidirectional MSBFS

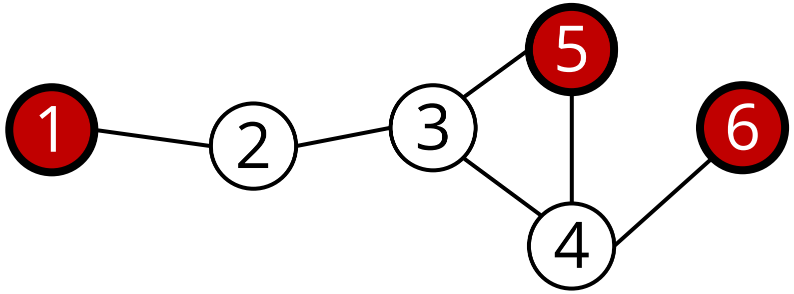
BIDIRECTIONAL MSBFS ALGORITHM

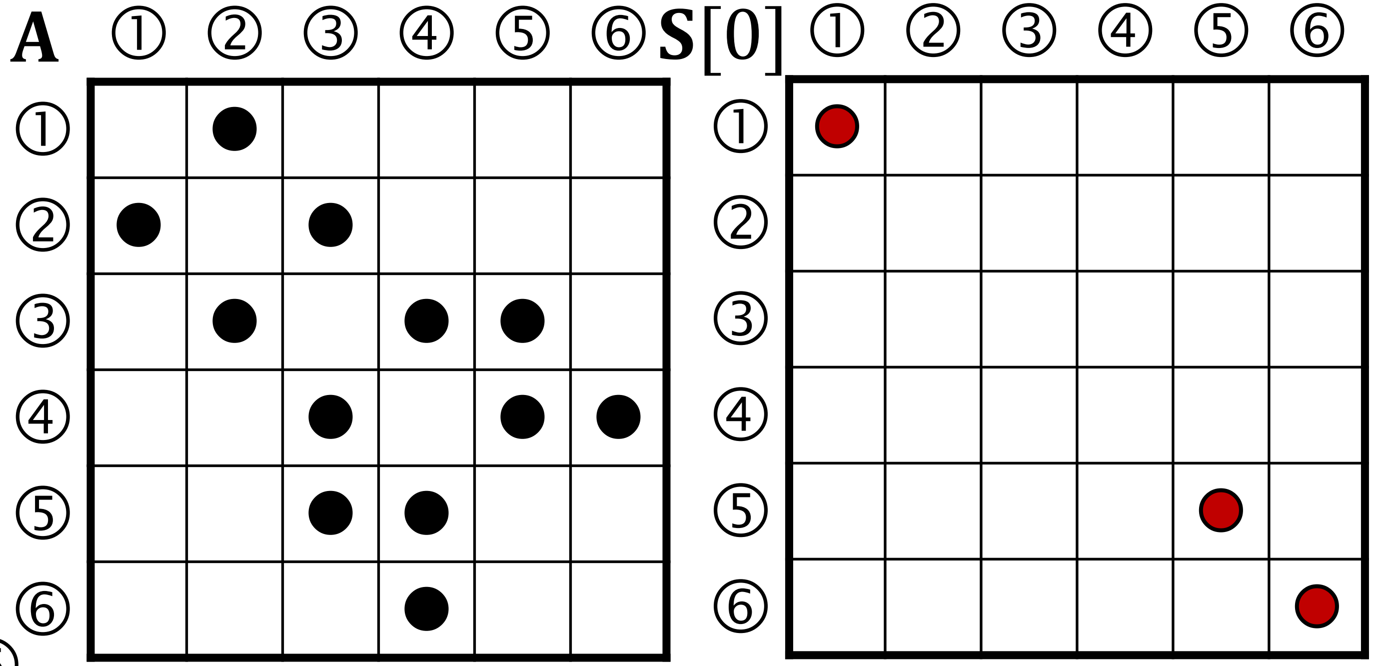
- **Pairwise reachability problem:**
  From a given set of k vertices, which pairs of vertices are reachable from each other with at most h hops?
- **Naïve solution:**
  Run a k-source MSBFS for h steps and check reachability. The frontiers get large as they grow exponentially.
- **Better solution:**
  Advance all frontiers simultaneously for $\lceil h/2 \rceil$ iterations.

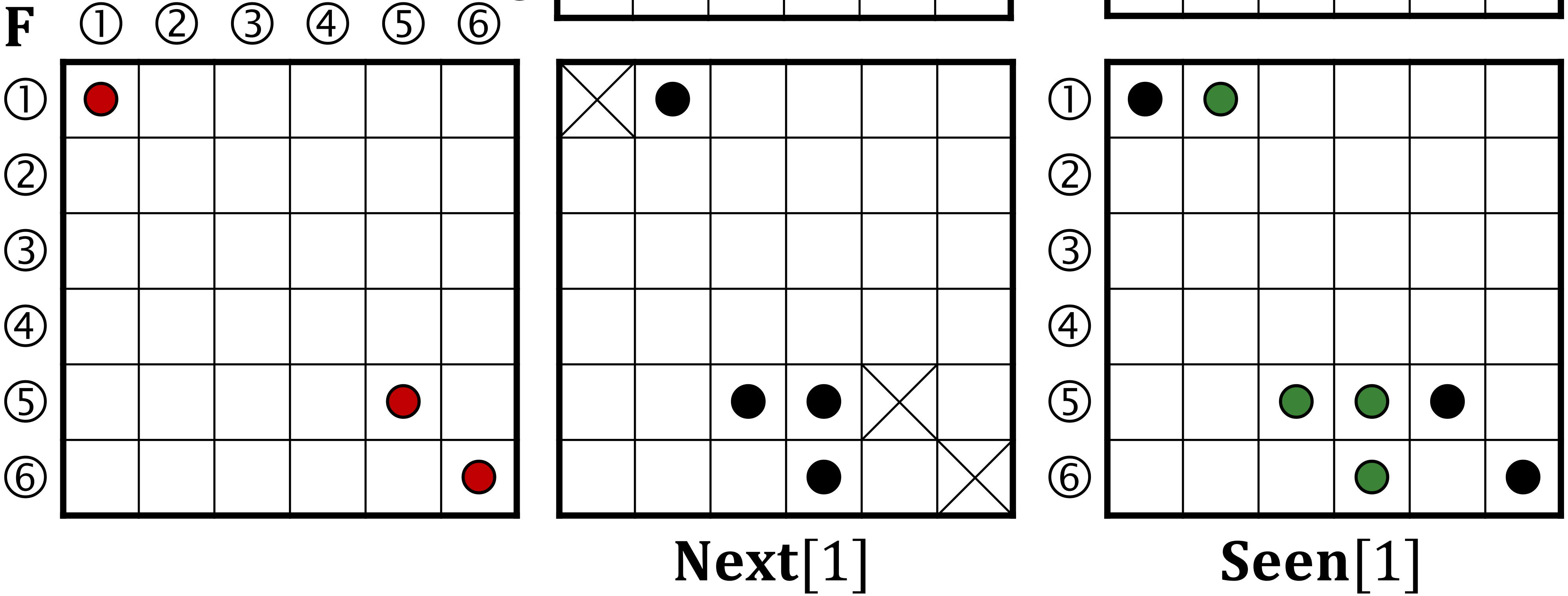


BIDIRECTIONAL MSBFS

Seen[1]: reachability with ≤1 hops

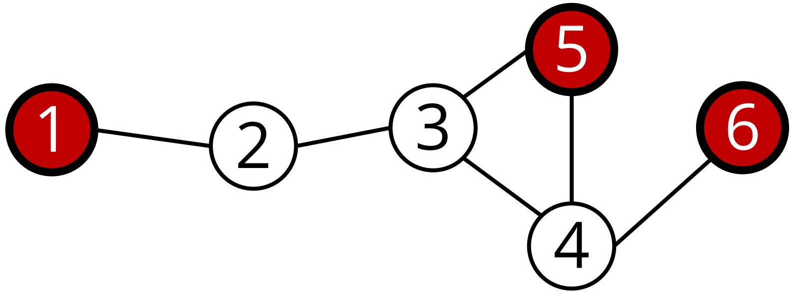


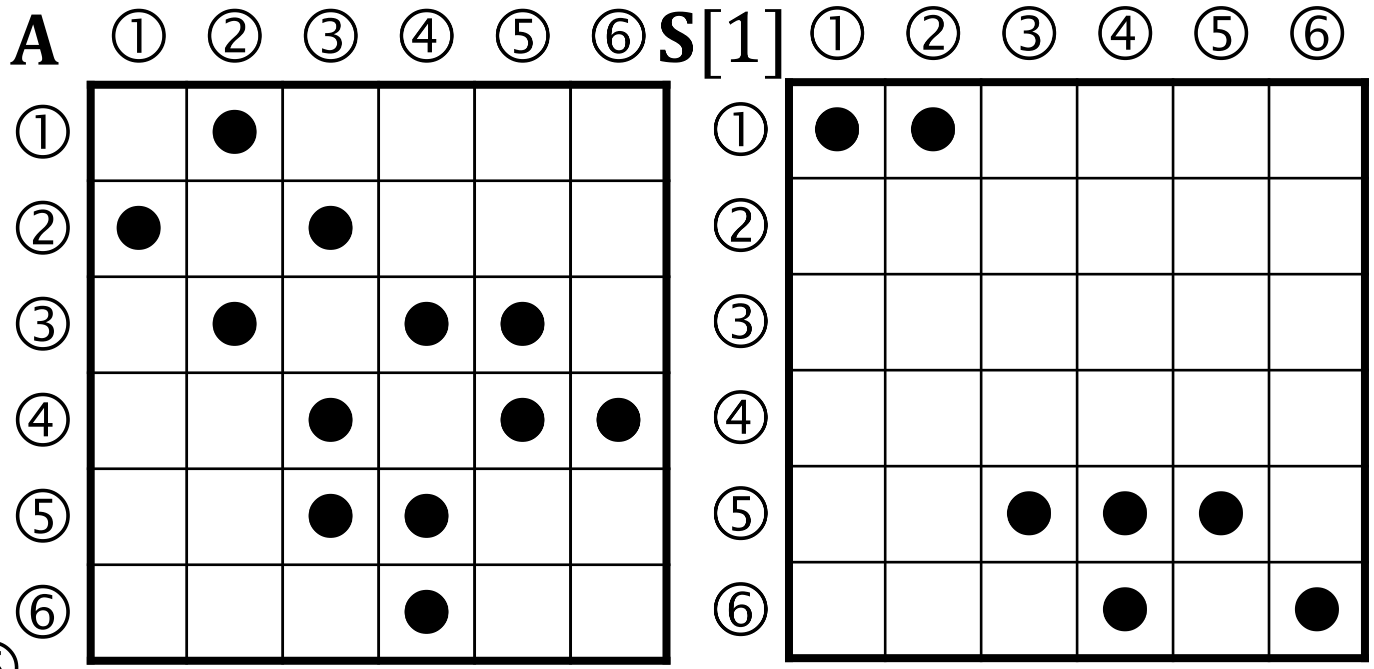


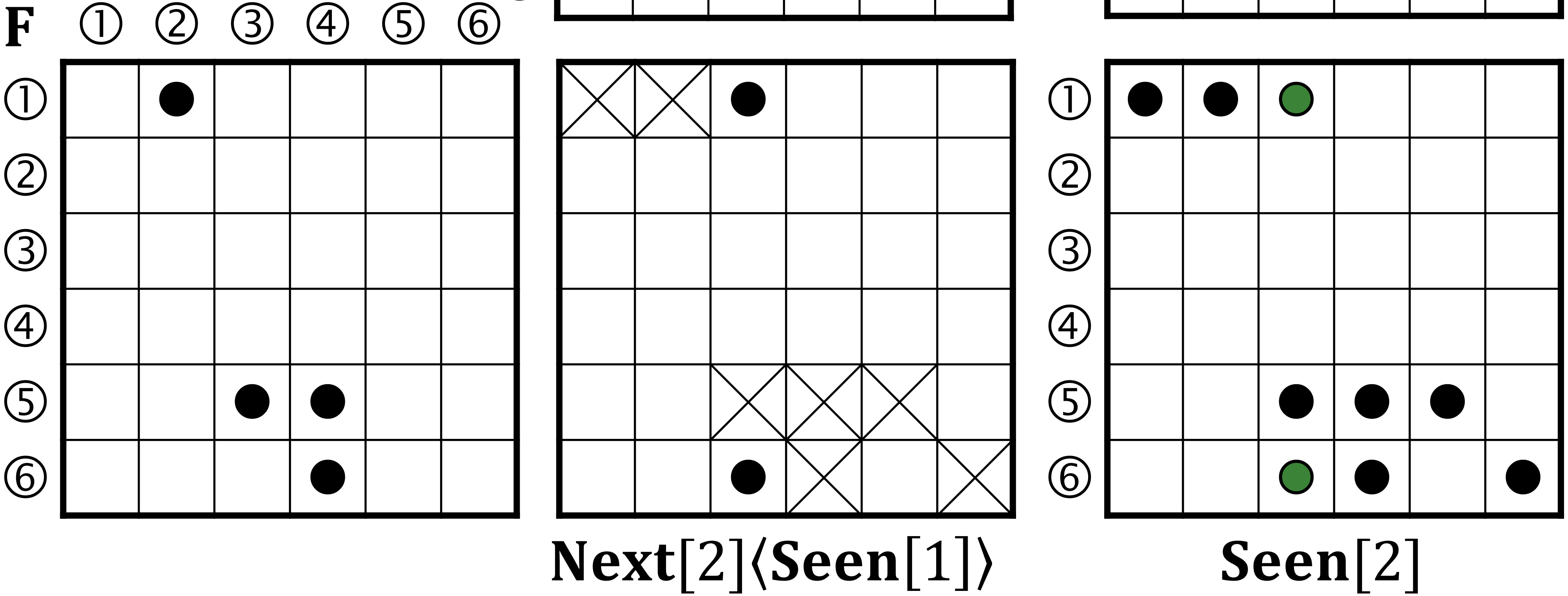


BIDIRECTIONAL MSBFS

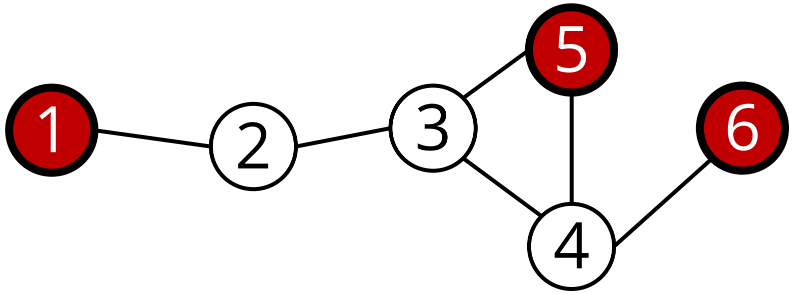
Seen[2]: reachability with ≤2 hops

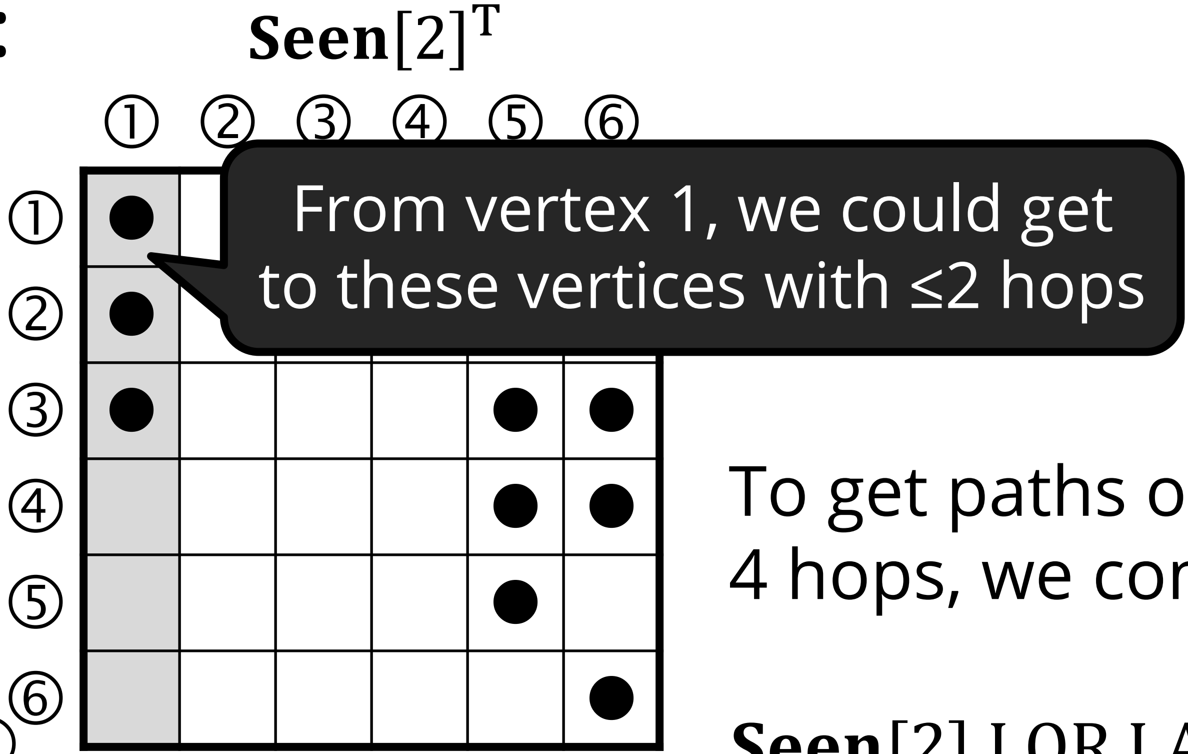


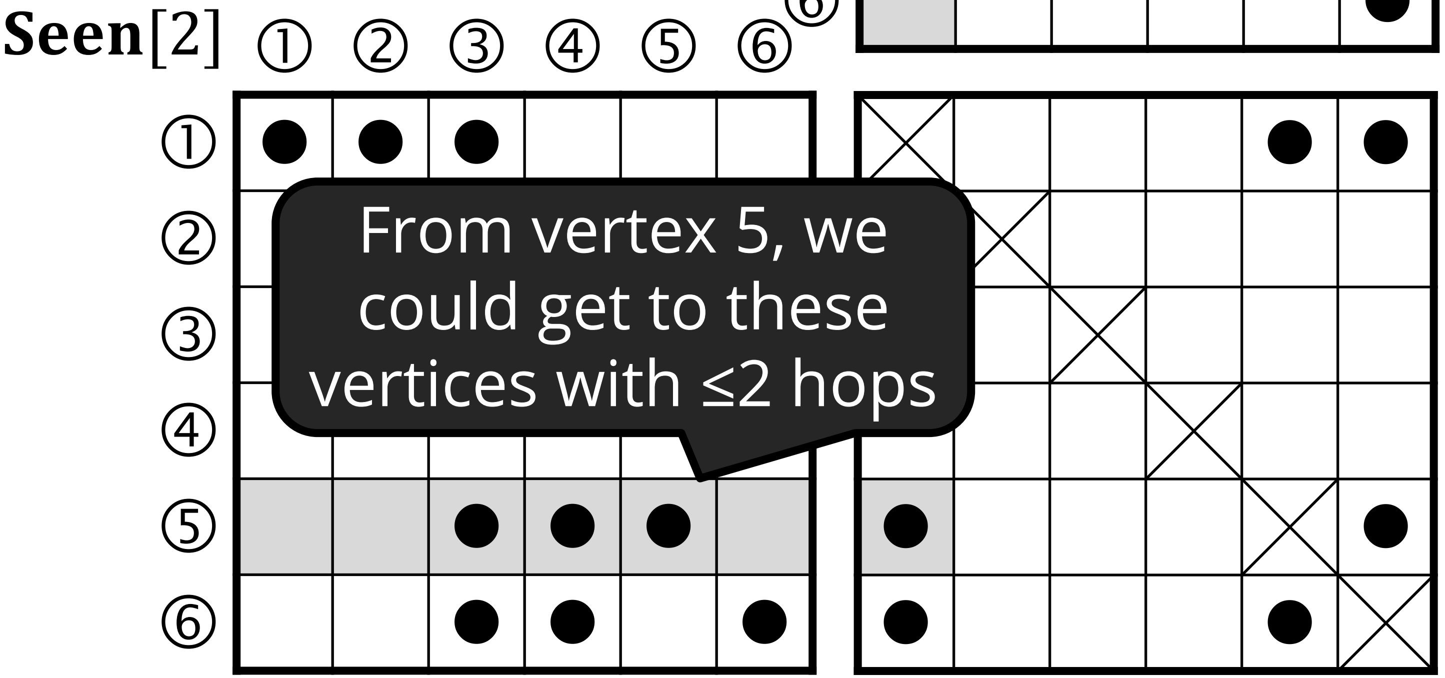




BIDIRECTIONAL MSBFS: PATHS OF LENGTH ≤4







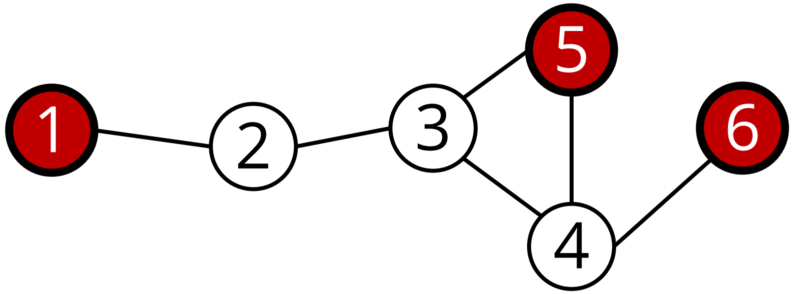
To get paths of at most 4 hops, we compute

Seen[2] LOR.LAND **Seen**[2]^T

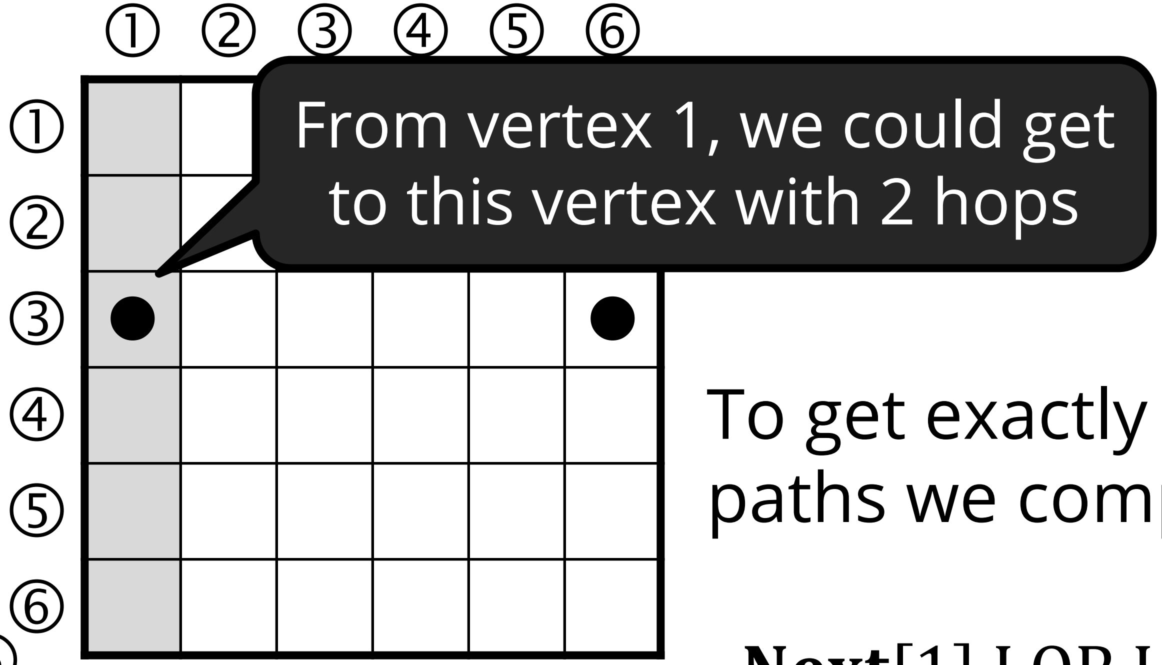
Here, we found paths between all pairs:

- from ① to ⑤,
- from ① to ⑥,
- from ⑤ to ⑥.

BIDIRECTIONAL MSBFS: PATHS OF LENGTH 3



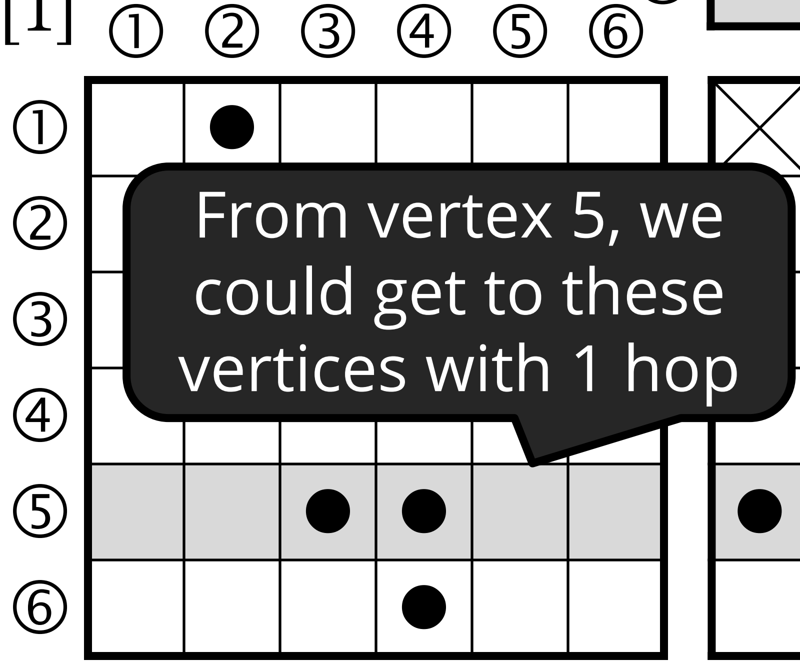
Next[2]^T

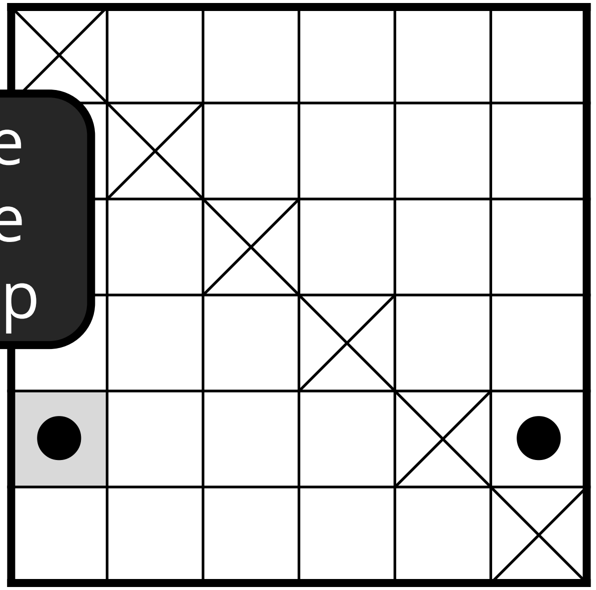


To get exactly 3-length paths we compute

Next[1] LOR.LAND **Next**[2]^T

Next[1]





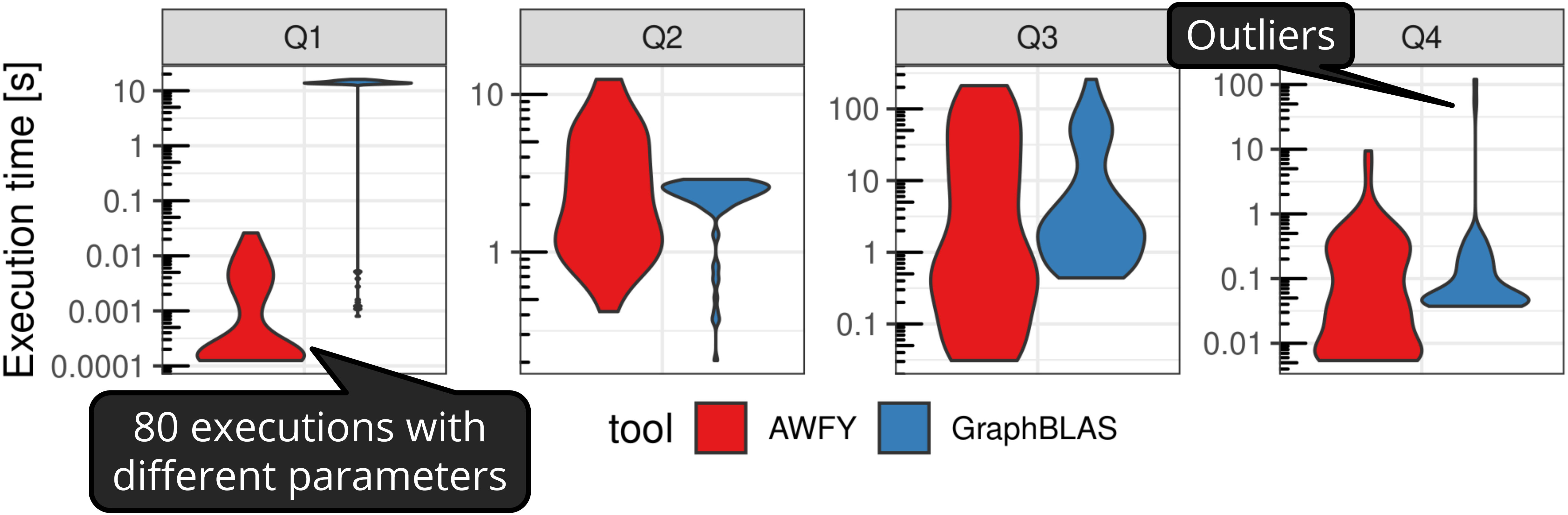
We found two 3-length paths:

- from ⑤ to ①
- from ⑤ to ⑥.

Benchmark results

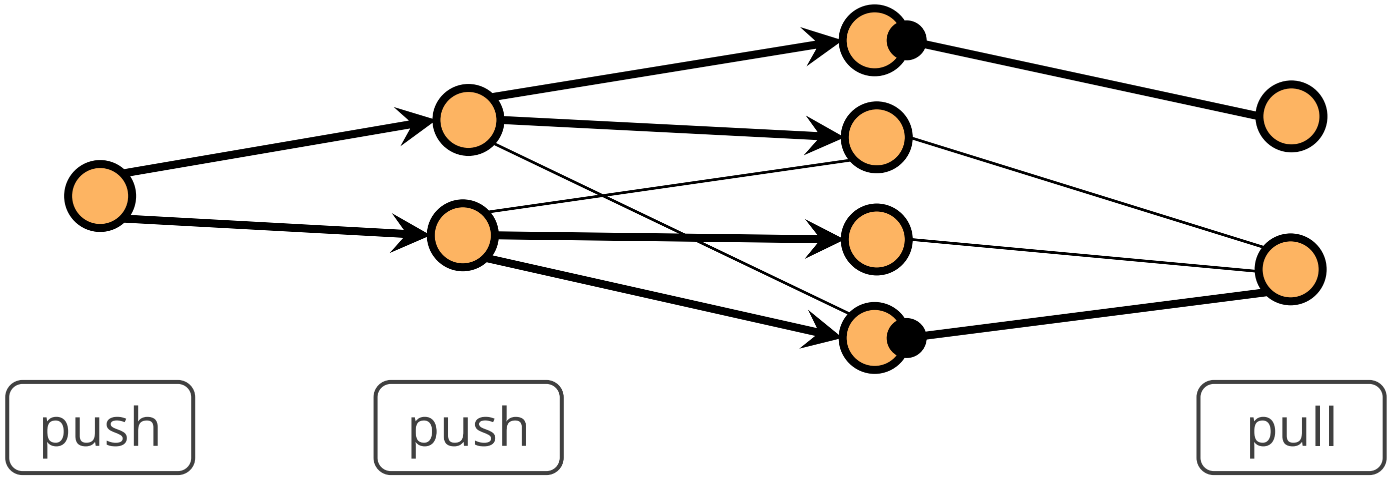
BENCHMARK RESULTS

- The top solution of AWFY vs. SuiteSparse:GraphBLAS v3.3.3
- AWFY's solution uses SIMD instructions → difficult to port
- GraphBLAS load times are slow (see details in paper)



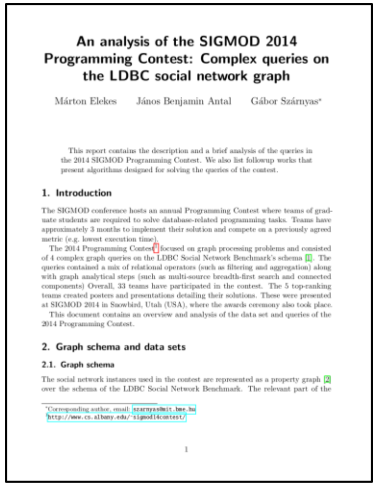
DIRECTION-OPTIMIZING TRAVERSAL

- This is optimization is subject to future work.
- For low diameter graphs, it is worth to use push/pull phases
- Push/pull is simple to express in GraphBLAS
  - See [Yang et al., ICPP'18]
- But deciding *when* to change is non-trivial



SUMMARY

- An interesting case study, see technical report



An analysis of the SIGMOD 2014 Programming Contest: Complex queries on the LDBC social network graph

Márton Elekes János Benjamin Antal Gábor Szárnyas*

This report contains the description and a brief analysis of the queries in the 2014 SIGMOD Programming Contest. We also list followup works that present algorithms designed for solving the queries of the contest.

1. Introduction

The SIGMOD conference hosts an annual Programming Contest where teams of graduate students are required to solve database-related programming tasks. Teams have approximately 3 months to implement their solution and compete on a previously agreed metric (e.g. lowest execution time).

The 2014 Programming Contest[1] focused on graph processing problems and consisted of 4 complex graph queries on the LDBC Social Network Benchmark's schema [1]. The queries contained a mix of relational operators (such as filtering and aggregation) along with graph analytical steps (such as multi-source breadth-first search and connected components) Overall, 33 teams have participated in the contest. The 5 top-ranking teams created posters and presentations detailing their solutions. These were presented at SIGMOD 2014 in Snowbird, Utah (USA), where the awards ceremony also took place.

This document contains an overview and analysis of the data set and queries of the 2014 Programming Contest.

2. Graph schema and data sets

2.1. Graph schema

The social network instances used in the contest are represented as a property graph [2] over the schema of the LDBC Social Network Benchmark. The relevant part of the

*Corresponding author, email: szarnyas@mit.bme.hu
[1]http://www.cs.albany.edu/~sigmod14contest/

1

- GraphBLAS can capture mixed workloads
  - Induced subgraph computations are simple to express
  - Algorithms are concise, bitwise optimizations can be adopted
  - Performance is *sometimes* on par with specialized solutions

- Future optimizations
  - Bitmap-based matrix/vector compression is WIP by Tim Davis
    → **≈5× speedup** without using bitwise operators in our code

GraphBLAS

Extended slide deck

sigmod2014-contest-graphblas

Ω