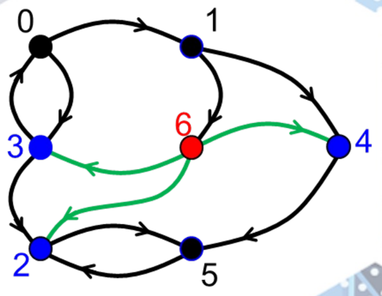


Graph Processing using GraphBLAS

Scott McMillan, *CMU Software Engineering Institute*

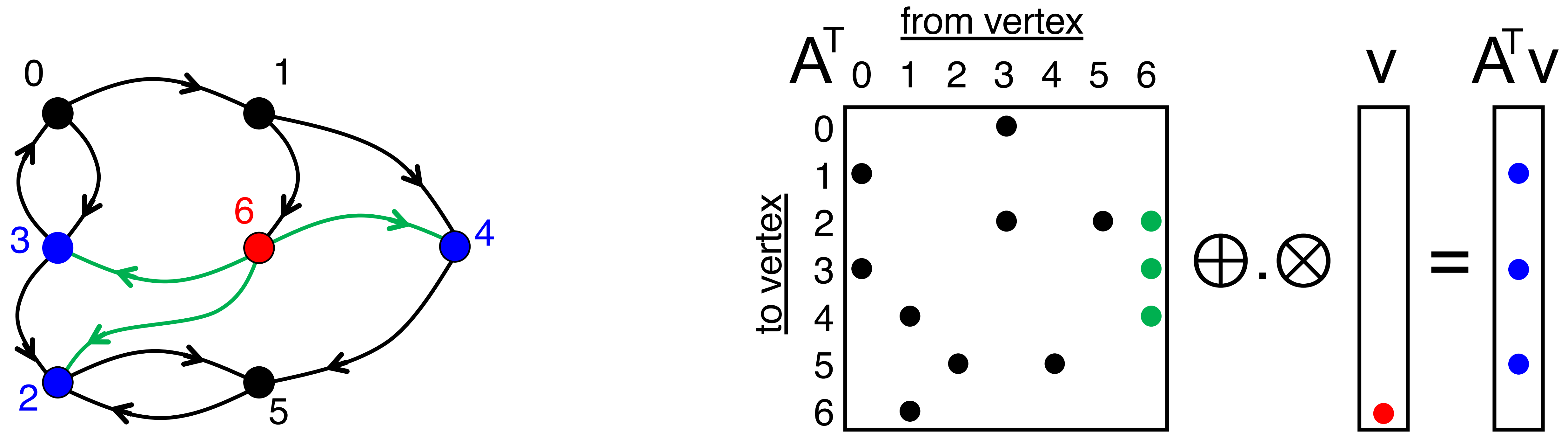
with collaborators Benjamin Brock, Tim Mattson,
Jose E. Moreira, Aydin Buluc, Tim Davis,
Gabor Szarnyas, Roi Lipman,
Jim Kitchen, Erik Welch,
and many more…

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

DM23-0646

Outline

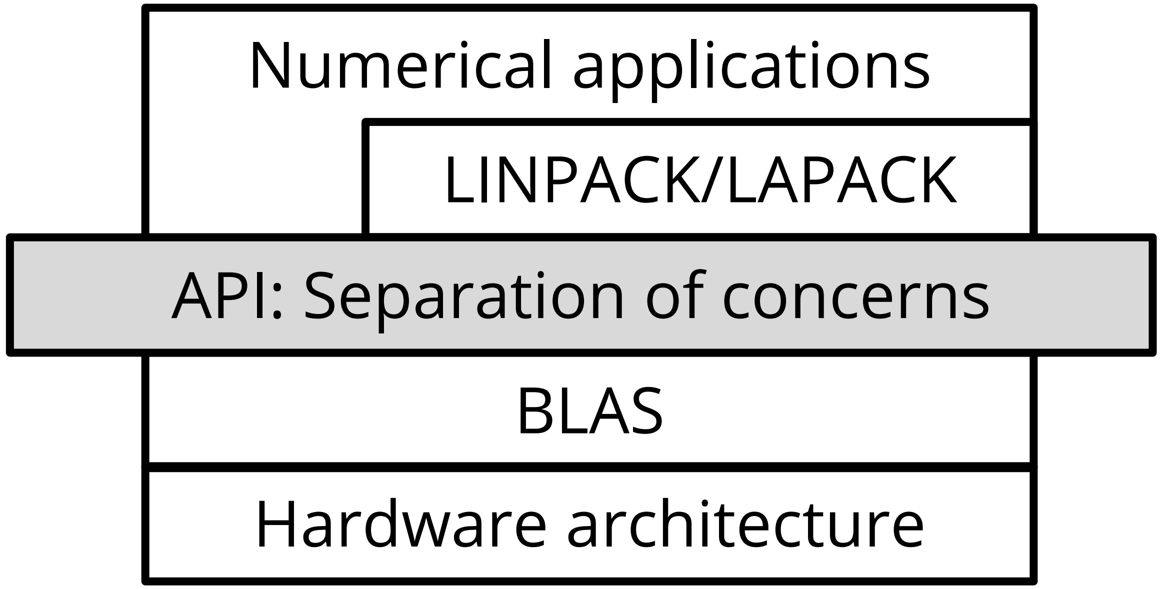
- Background & Motivation
- Graphs can be represented as **matrices**
- Basic graph **operations** can be performed with linear algebra
- These operations can be composed to implement useful **algorithms**



The GraphBLAS Application Programming Interface (API)

Goal: separate the concerns of hardware/library & application designers.

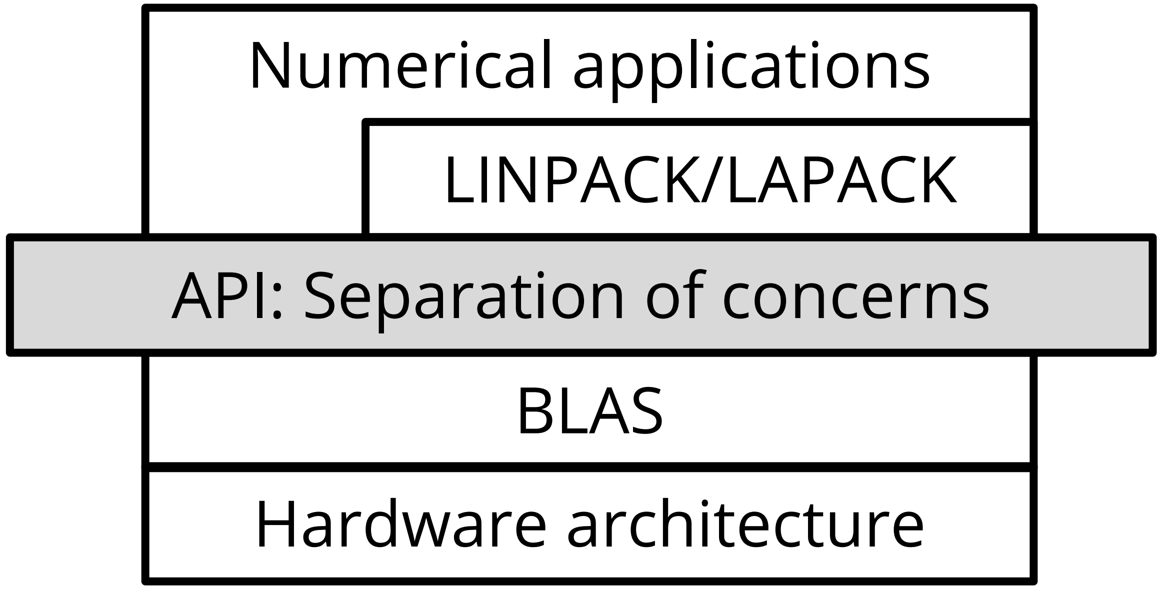
1979: BLAS Basic Linear Algebra Subprograms (BLAS 2 ’88, BLAS 3 ’90)



The GraphBLAS API

Goal: separate the concerns of hardware/library & application designers.

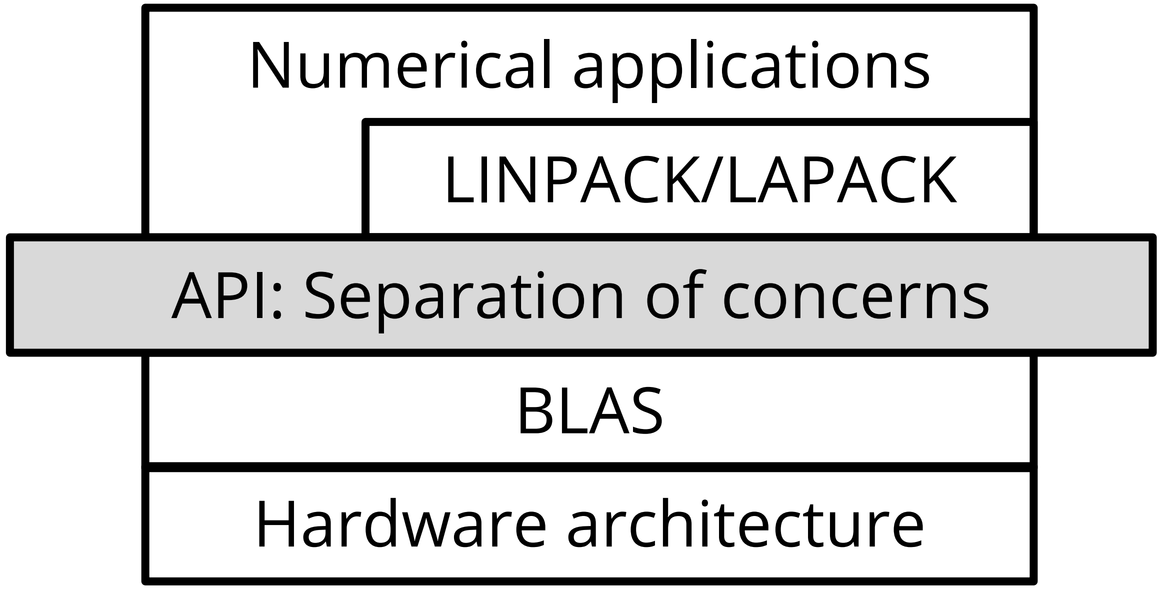
1979: BLAS	Basic Linear Algebra Subprograms (BLAS 2 ’88, BLAS 3 ’90)
2001: Sparse BLAS	an extension to BLAS (little uptake)

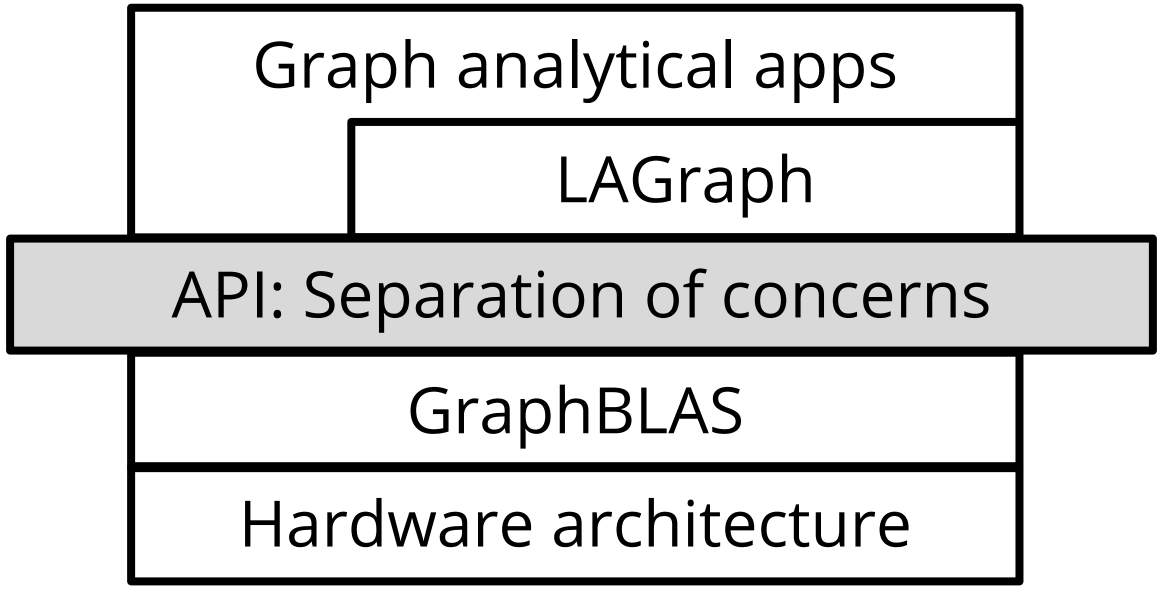


The GraphBLAS API

Goal: separate the concerns of hardware/library & application designers.

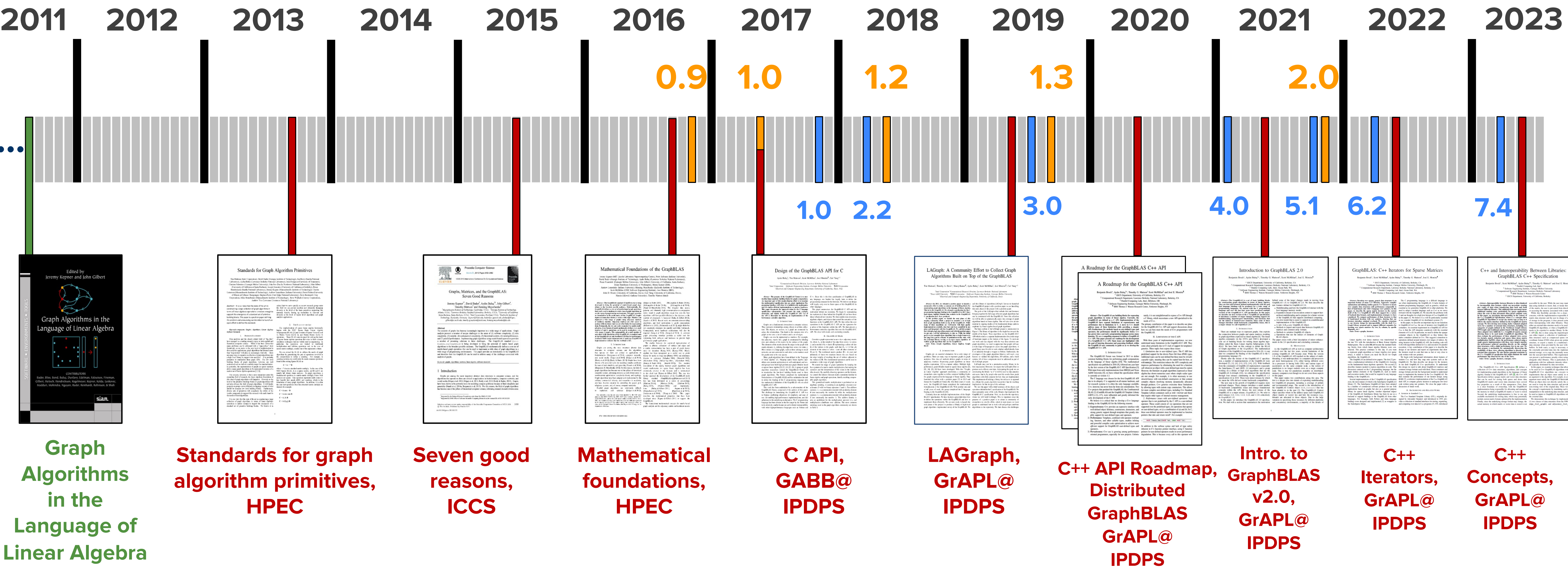
1979: BLAS	Basic Linear Algebra Subprograms (BLAS 2 ’88, BLAS 3 ’90)
2001: Sparse BLAS	an extension to BLAS (little uptake)
2013: GraphBLAS	an effort to define standard building blocks for graph algorithms in the language of linear algebra





GraphBLAS C/C++ Timeline

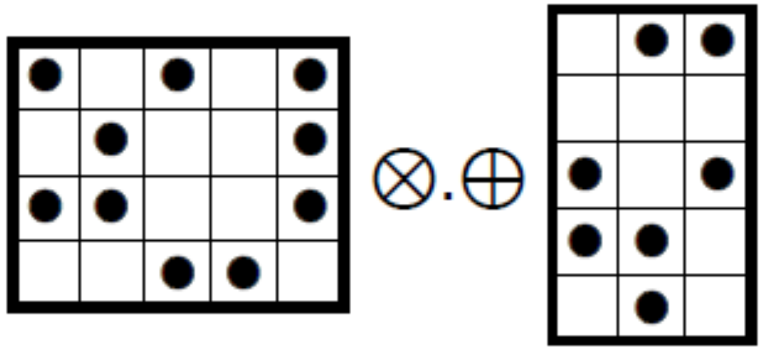
Book — Papers — GraphBLAS API version — SuiteSparse:GraphBLAS releases



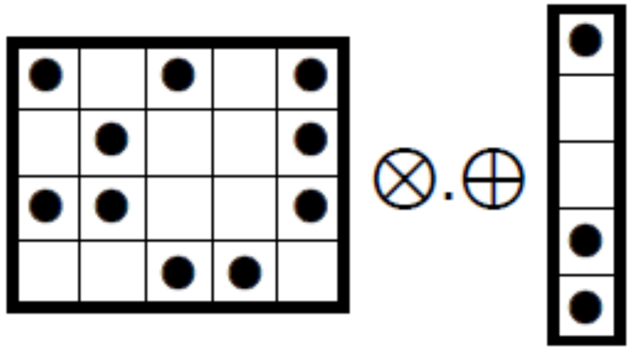
GraphBLAS Primitives

- Basic objects (opaque types)
  - Matrices (sparse or dense), vectors (sparse or dense), algebraic operators (semirings)
- Fundamental operations over these objects

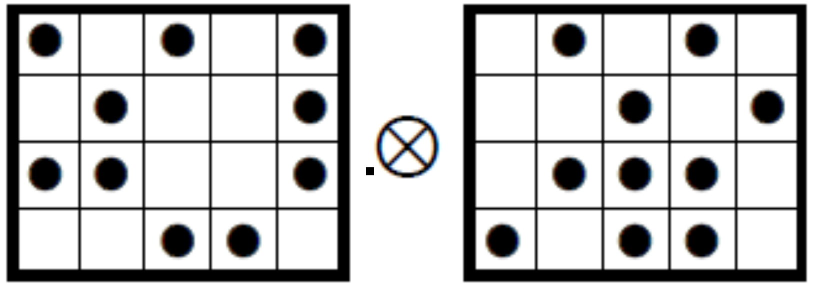
Sparse matrix times sparse matrix



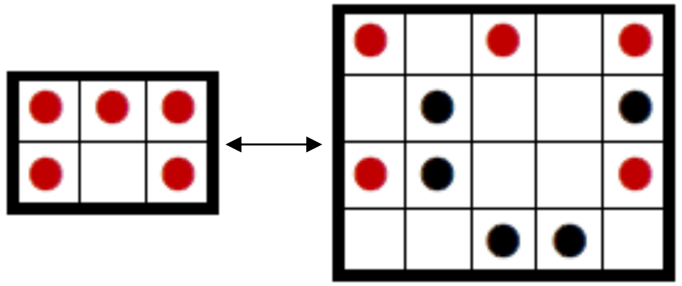
Sparse matrix times sparse vector



Element-wise multiplication (and addition)

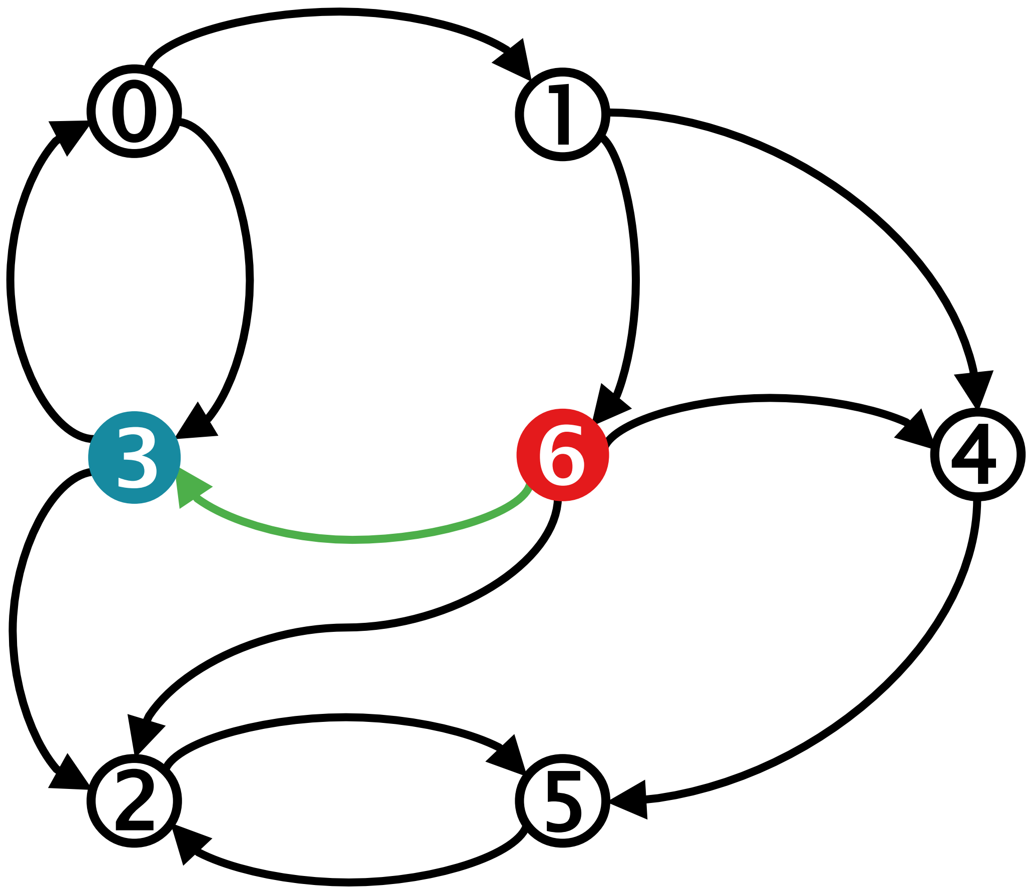


Sparse matrix extraction (and assignment)



…plus reduction, transpose, Kronecker product, filtering, transform, etc.

Graphs as Adjacency Matrices

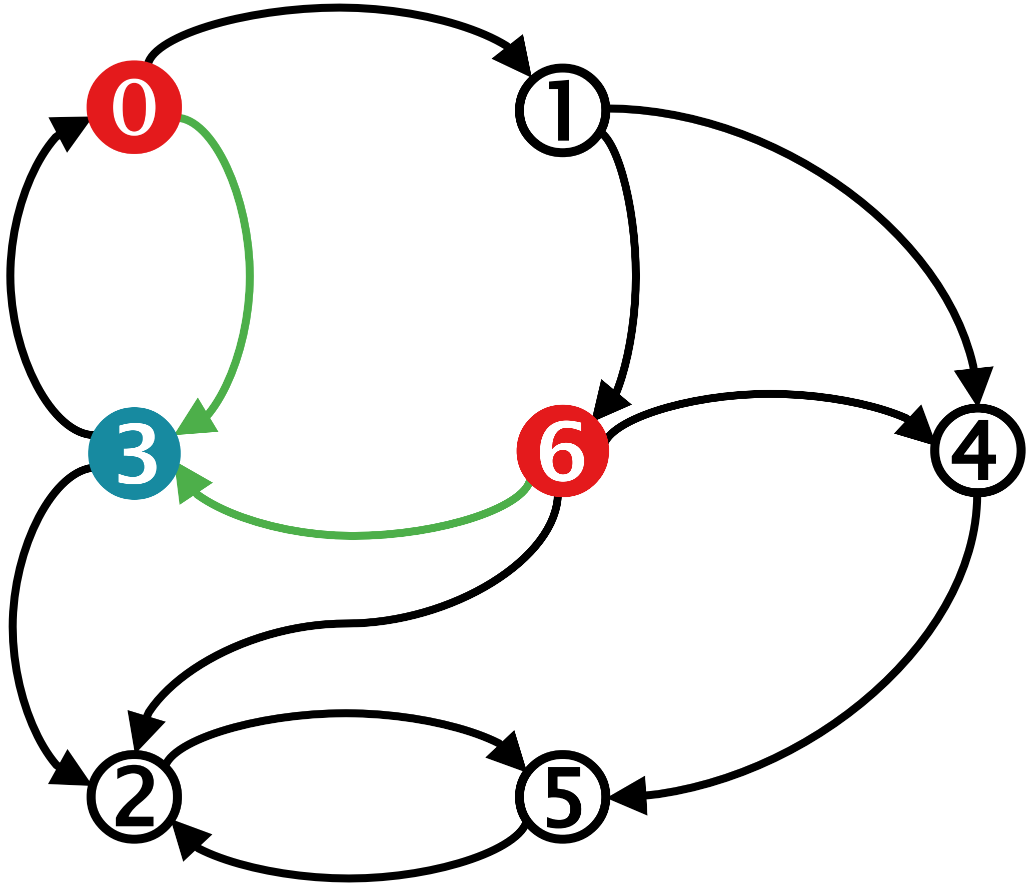


dest.

A	0	1	2	3	4	5	6
0		●		●			
1					●		●
2						●	
3	●		●				
4						●	
5			●				
source 6			●	●	●		

$$\mathbf{A}_{ij} = \begin{cases} \bullet & (v_i, v_j) \in E \\ \emptyset & (v_i, v_j) \notin E \end{cases}$$

Graphs as Adjacency Matrices

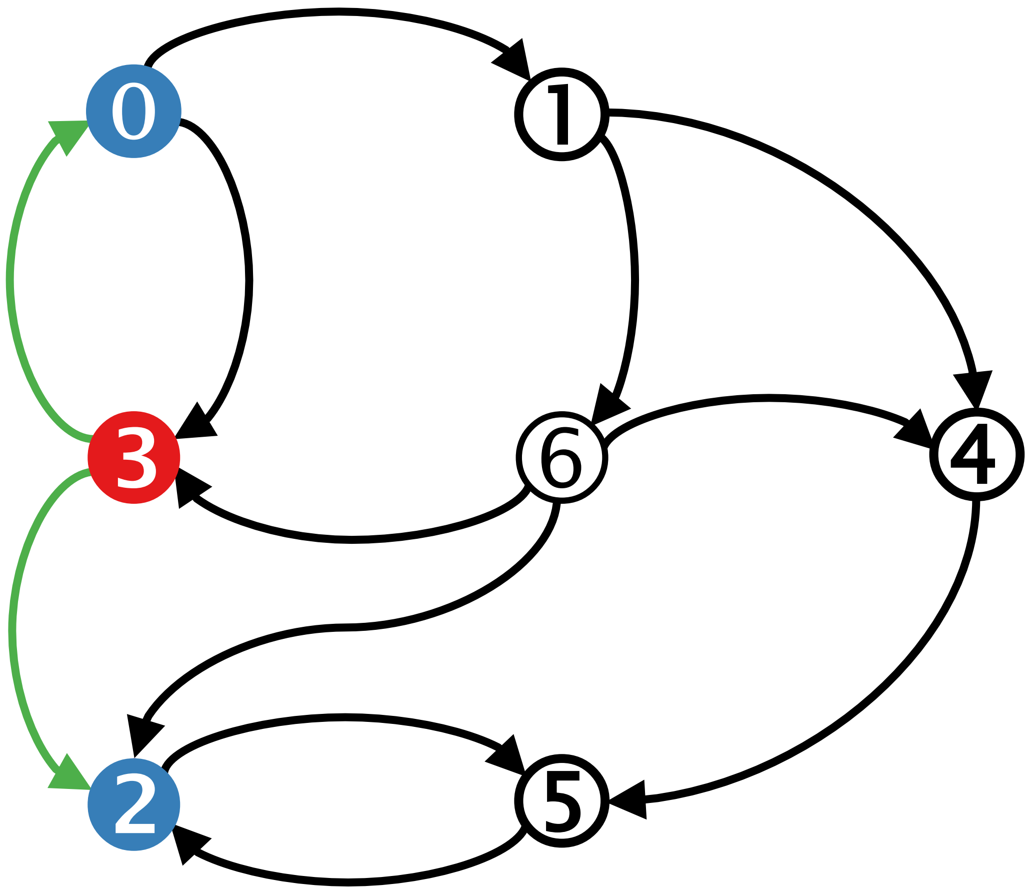


dest.

A	⓪	①	②	③	④	⑤	⑥
source ⓿		●		●			
①					●		●
②						●	
③	●		●				
④						●	
⑤			●				
source ❻			●	●	●		

$$\mathbf{A}_{ij} = \begin{cases} \bullet & (v_i, v_j) \in E \\ \emptyset & (v_i, v_j) \notin E \end{cases}$$

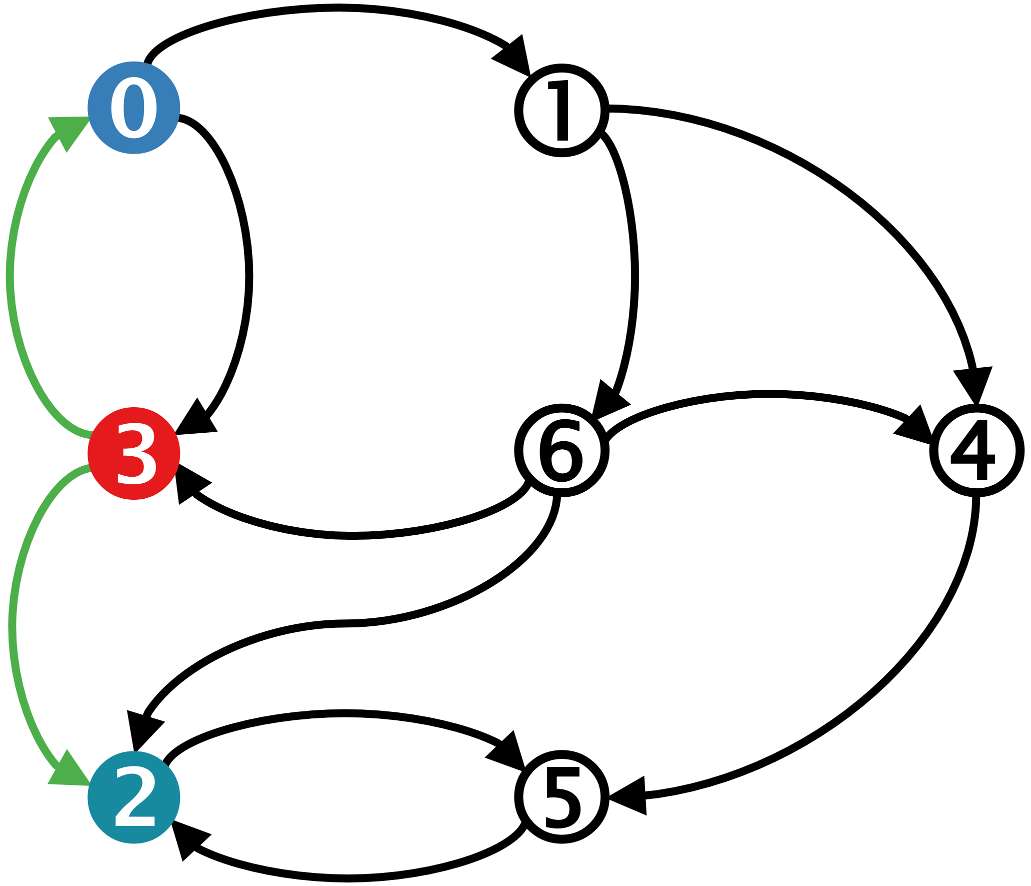
Graphs as Adjacency Matrices

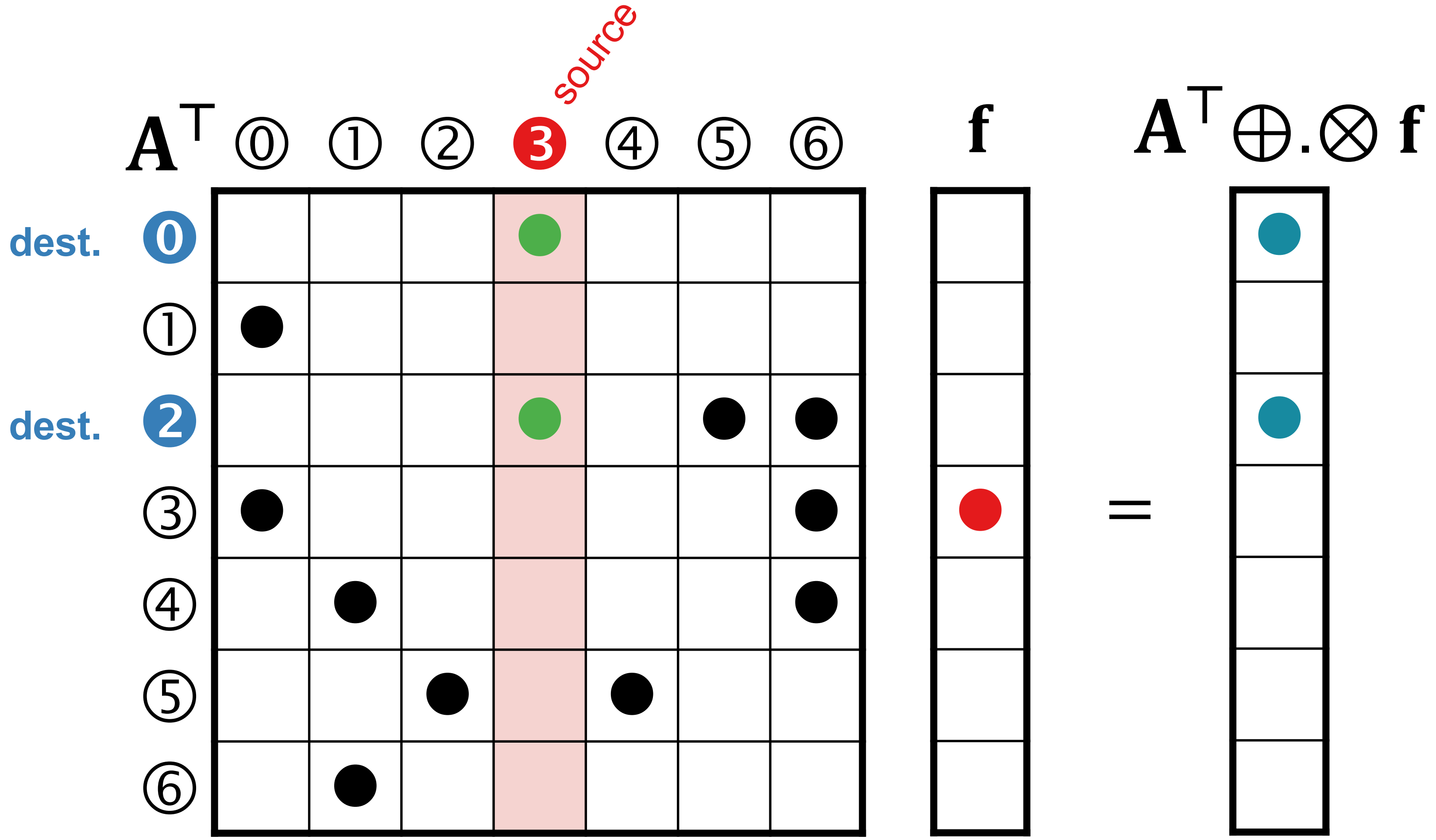


A	dest. 0	1	dest. 2	3	4	5	6
0		●		●			
1					●		●
2						●	
source 3	●		●				
4						●	
5			●				
6			●	●	●		

$$\mathbf{A}_{ij} = \begin{cases} \bullet & (v_i, v_j) \in E \\ \varnothing & (v_i, v_j) \notin E \end{cases}$$

Graph Operations as Matrix Operations

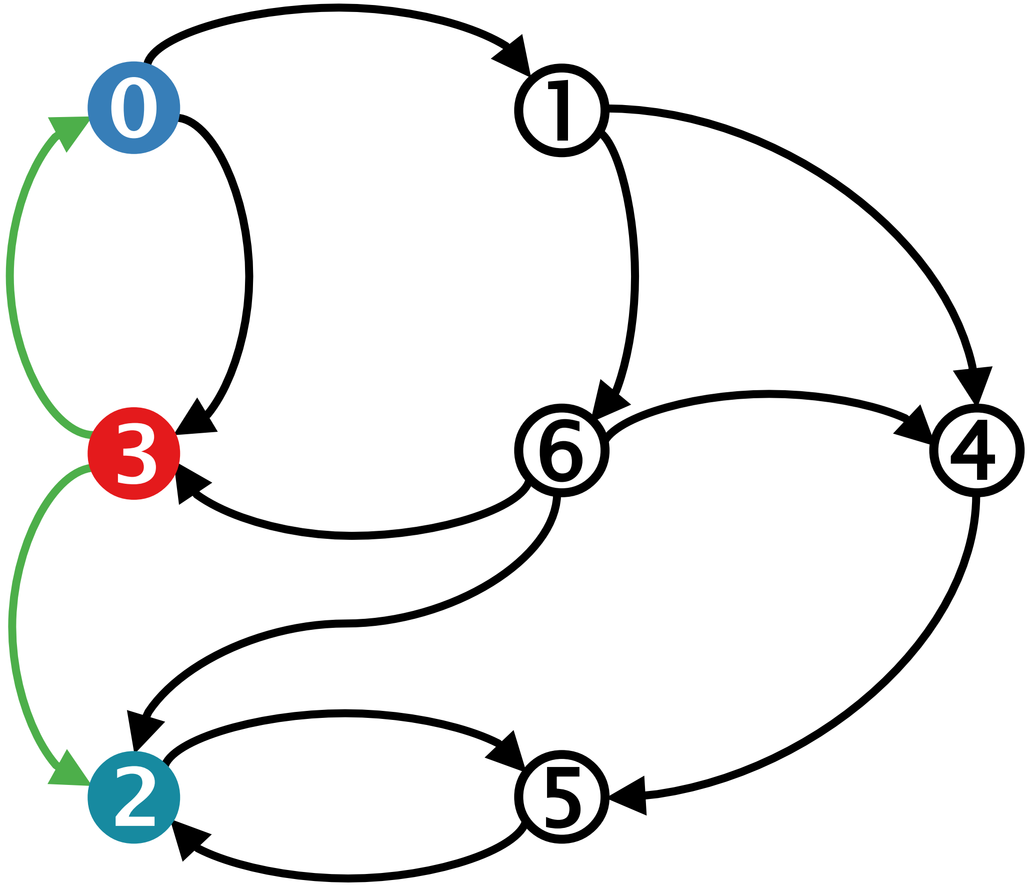


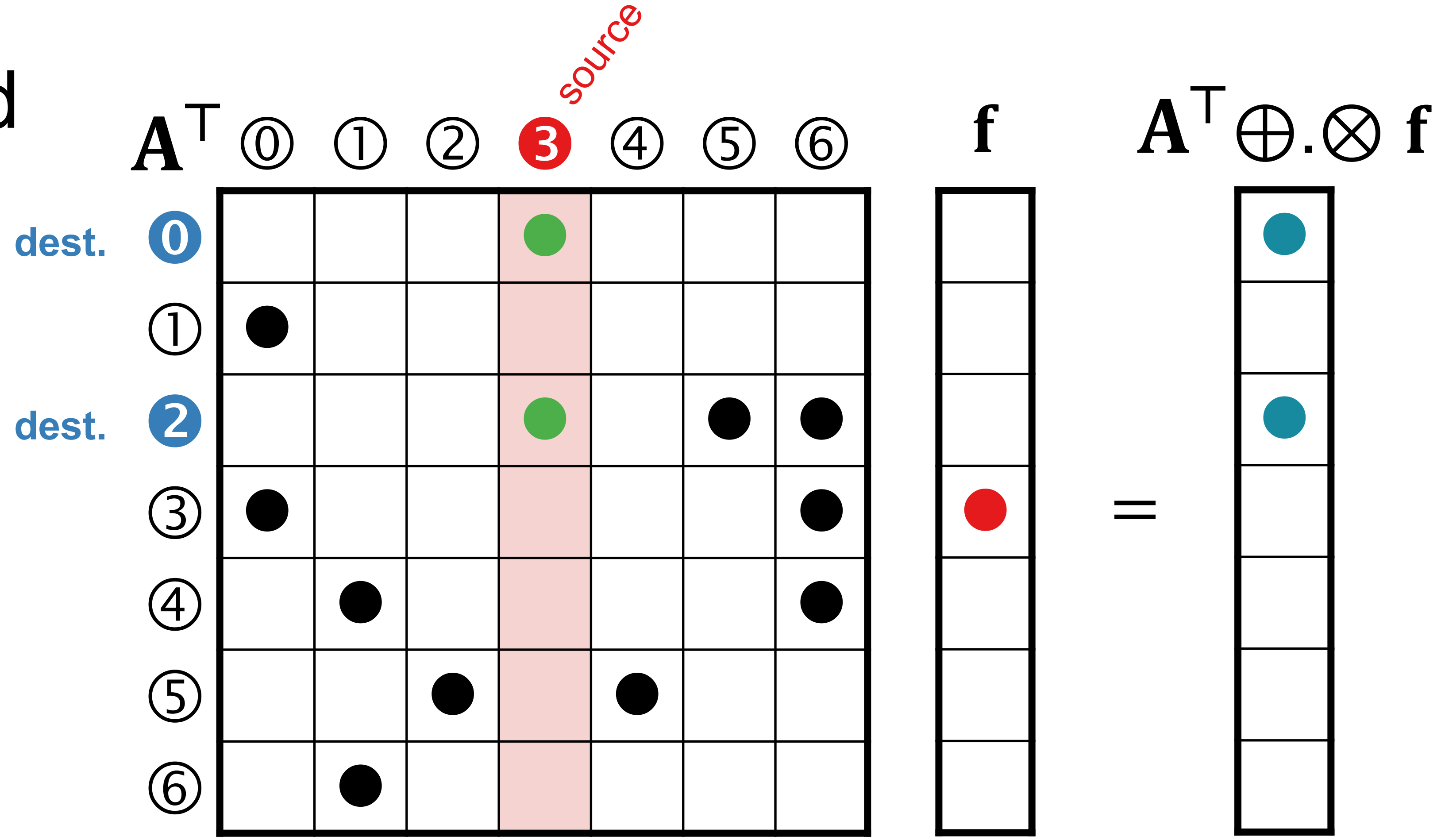


- Matrix-vector multiply → find neighbors
  - In-neighbors: use A
  - Out-neighbors: use A^T

Graph Operations as Matrix Operations

Finding out-neighbors is used many graph algorithms.

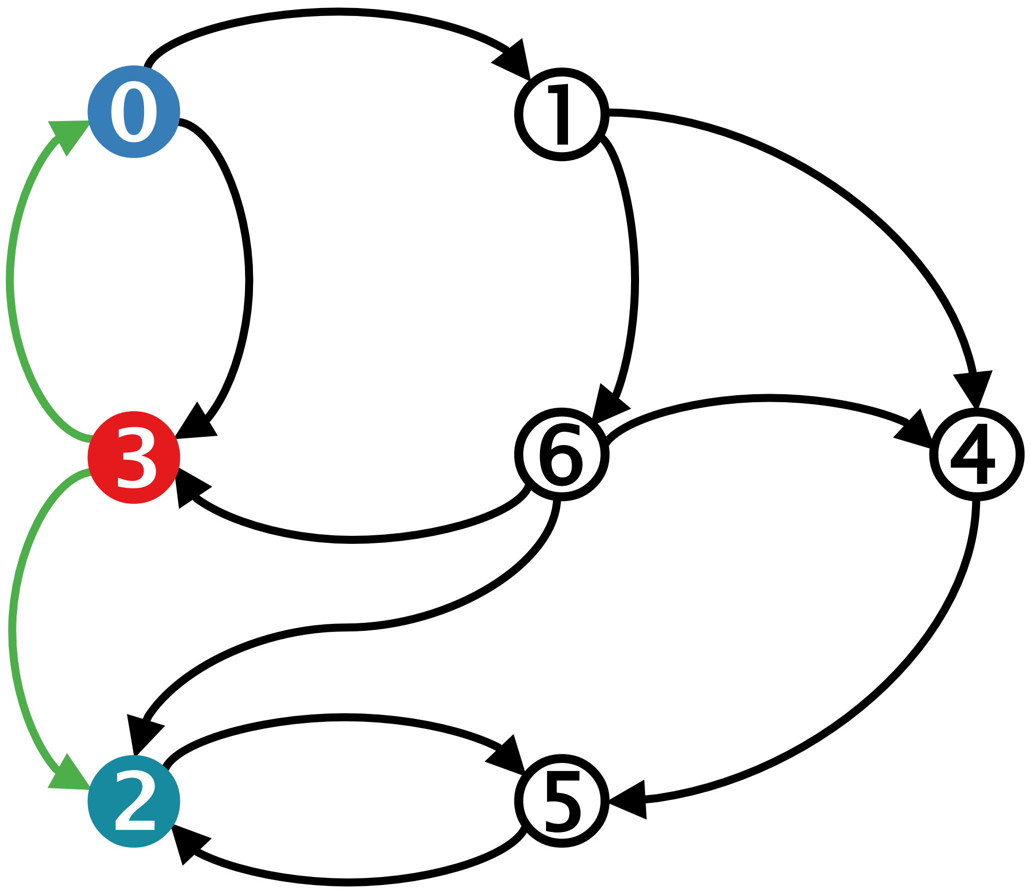


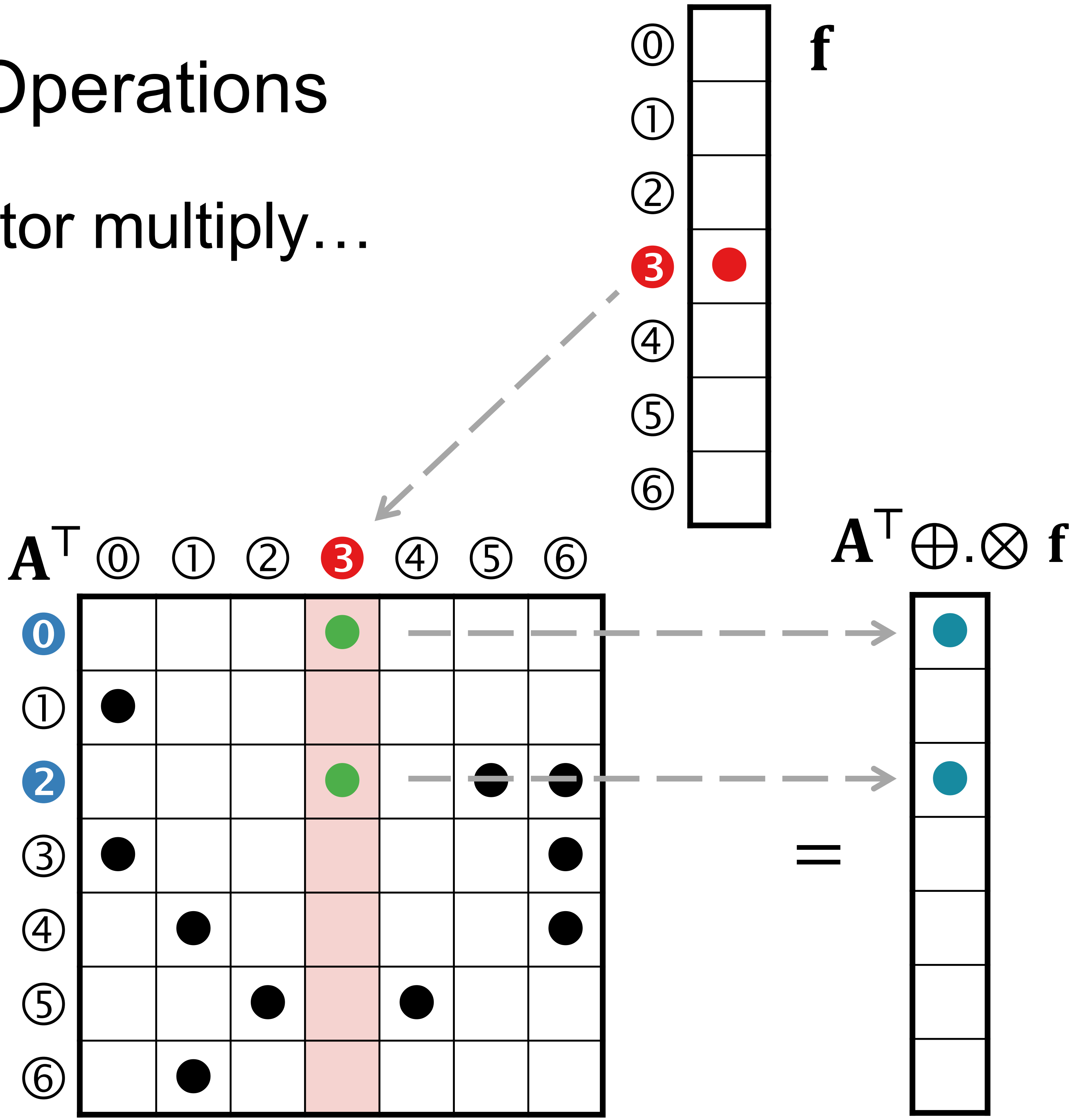


- Matrix-vector multiply → find neighbors
  - In-neighbors: use A
  - Out-neighbors: use A^T

Graph Operations as Matrix Operations

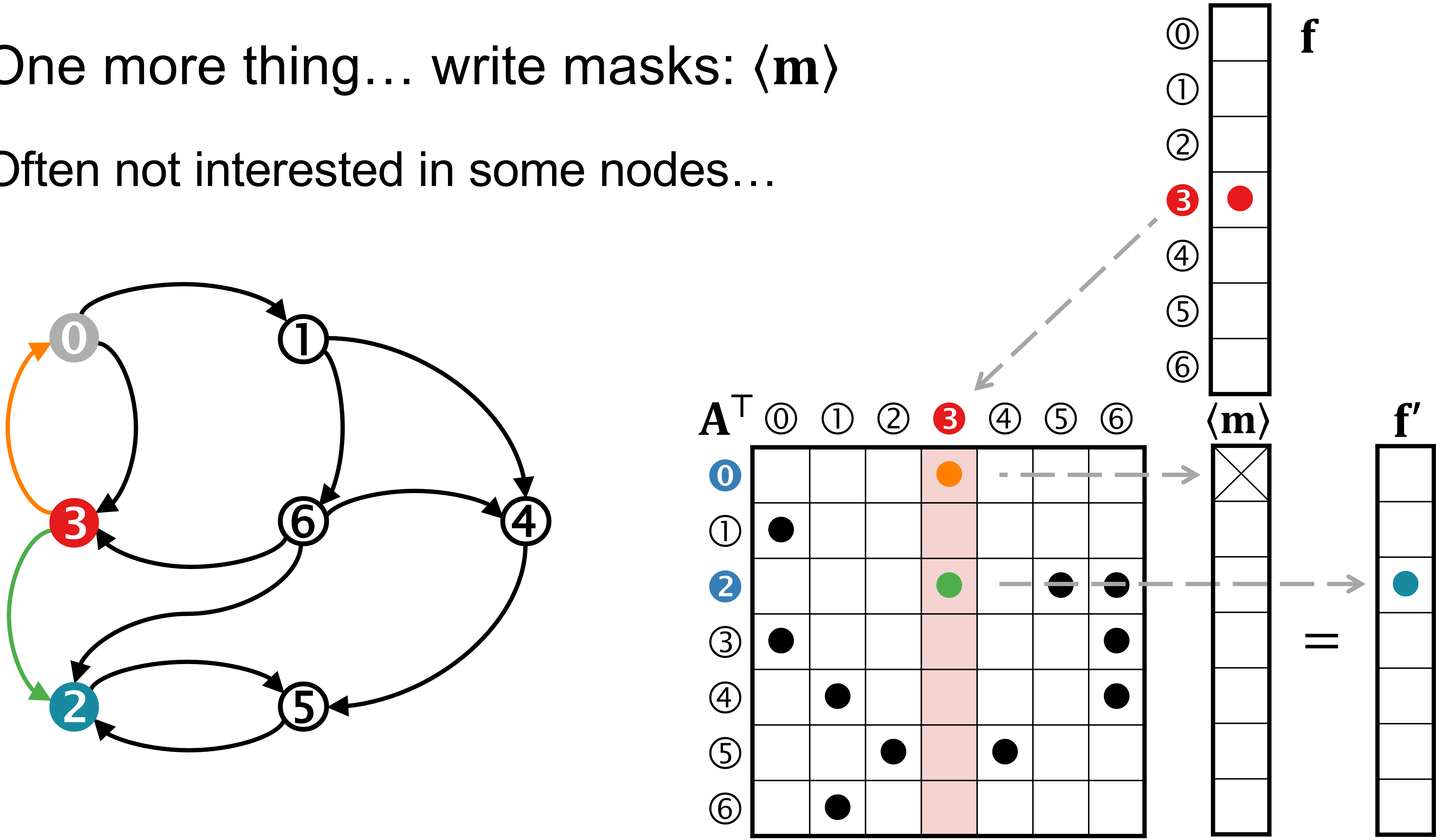
Another way to look at matrix-vector multiply…





One more thing… write masks: ⟨m⟩

Often not interested in some nodes…

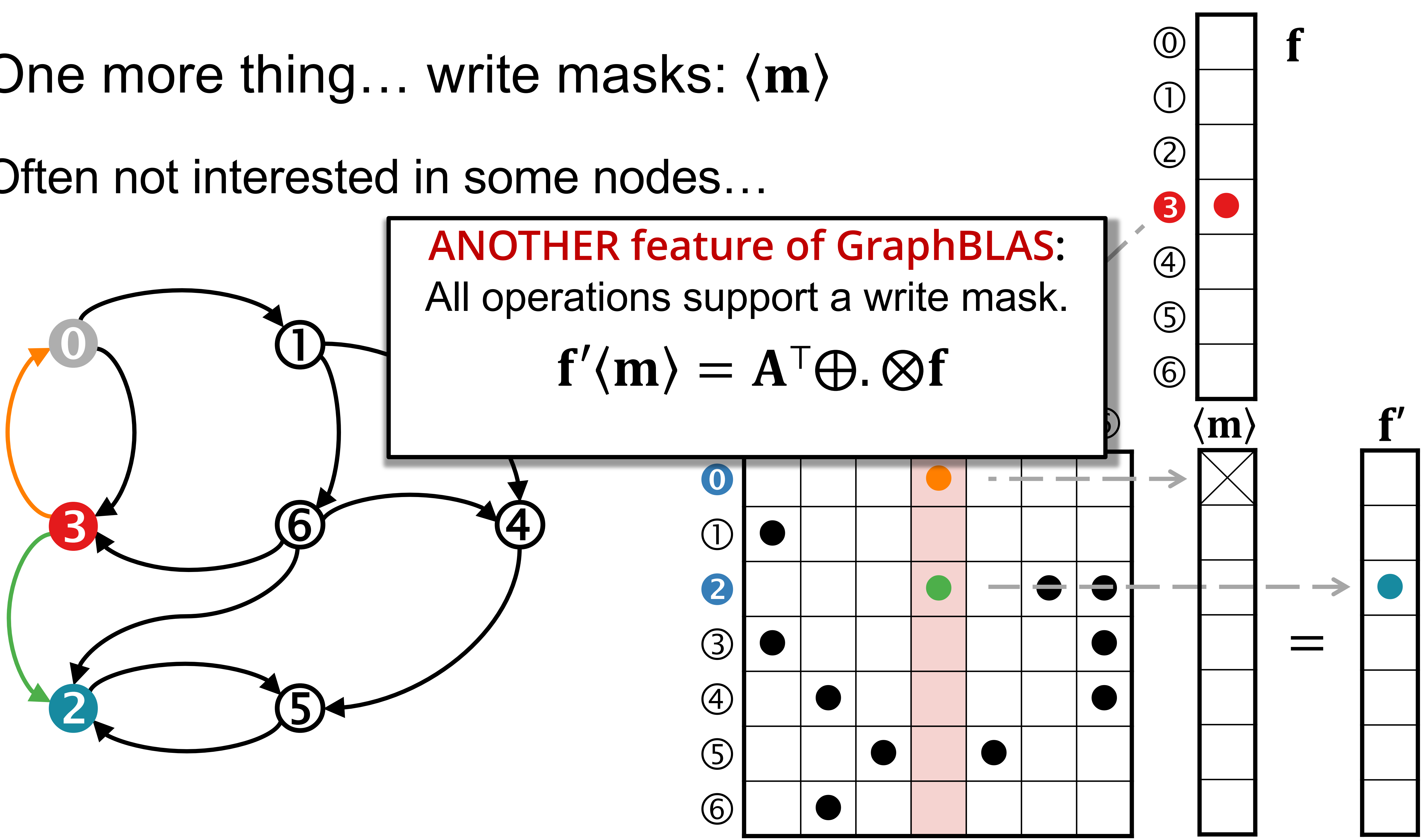


One more thing… write masks: ⟨m⟩

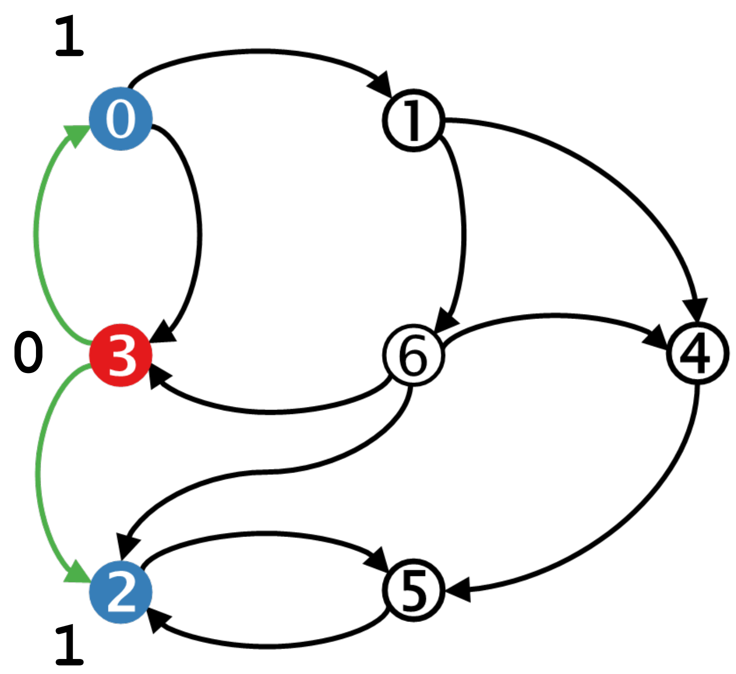
Often not interested in some nodes…

ANOTHER feature of GraphBLAS:
All operations support a write mask.

$$\mathbf{f}'\langle\mathbf{m}\rangle = \mathbf{A}^{\mathsf{T}} \oplus . \otimes \mathbf{f}$$



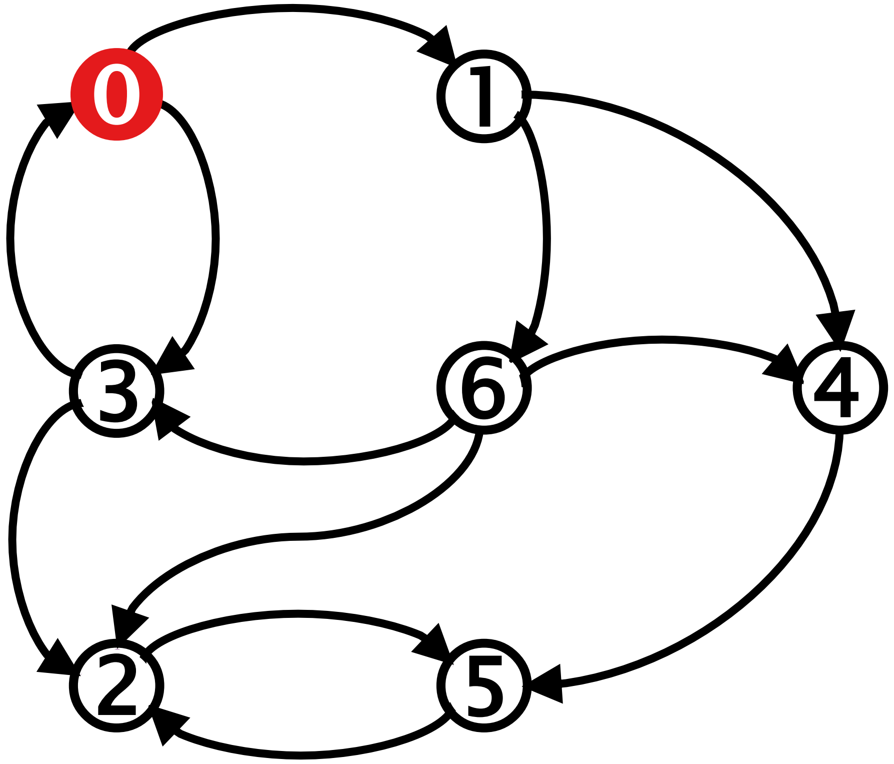
Algorithm: Breadth-First Search (BFS)



Example: Breadth-First Search (levels)

$\mathbf{f}(src) =$ ●

	$\mathbf{f}$	$\mathbf{v}$
0	●	
1		
2		
3		
4		
5		
6		

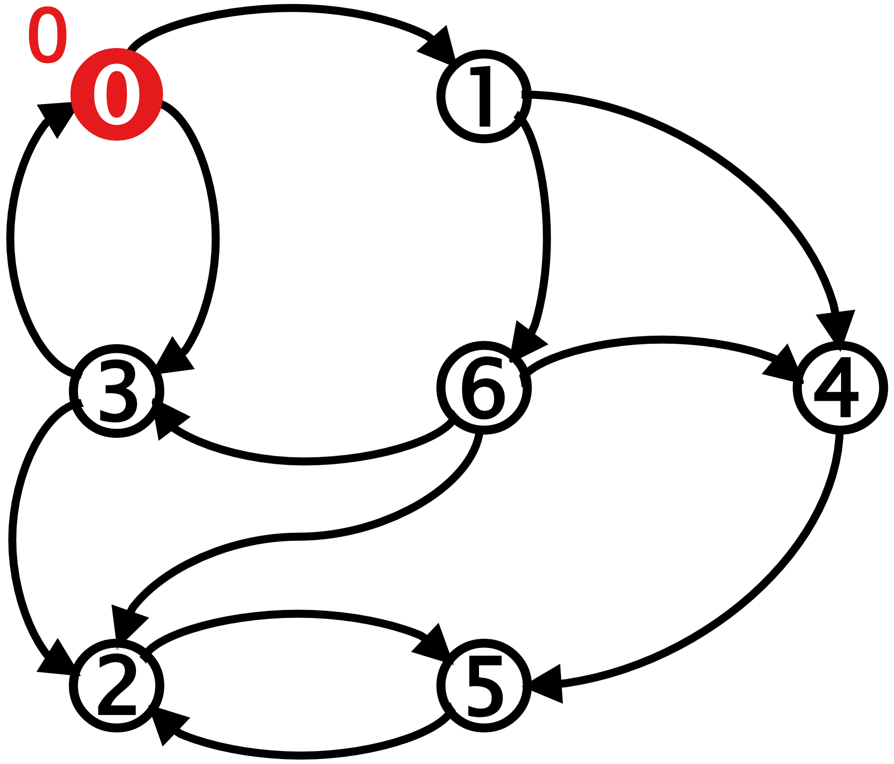


$\mathbf{A}^\mathsf{T}$	0	1	2	3	4	5	6
0				●			
1	●						
2				●		●	●
3	●						●
4		●					●
5			●		●		
6		●					

Example: Breadth-First Search (levels)

$$level = 0$$

$$\mathbf{v} \mathrel{+}= level * \mathbf{f}$$



$\mathbf{A}^\mathsf{T}$	0	1	2	3	4	5	6
0				●			
1	●						
2				●		●	●
3	●						●
4		●					●
5			●		●		
6		●					

	$\mathbf{f}$
0	●
1	
2	
3	
4	
5	
6	

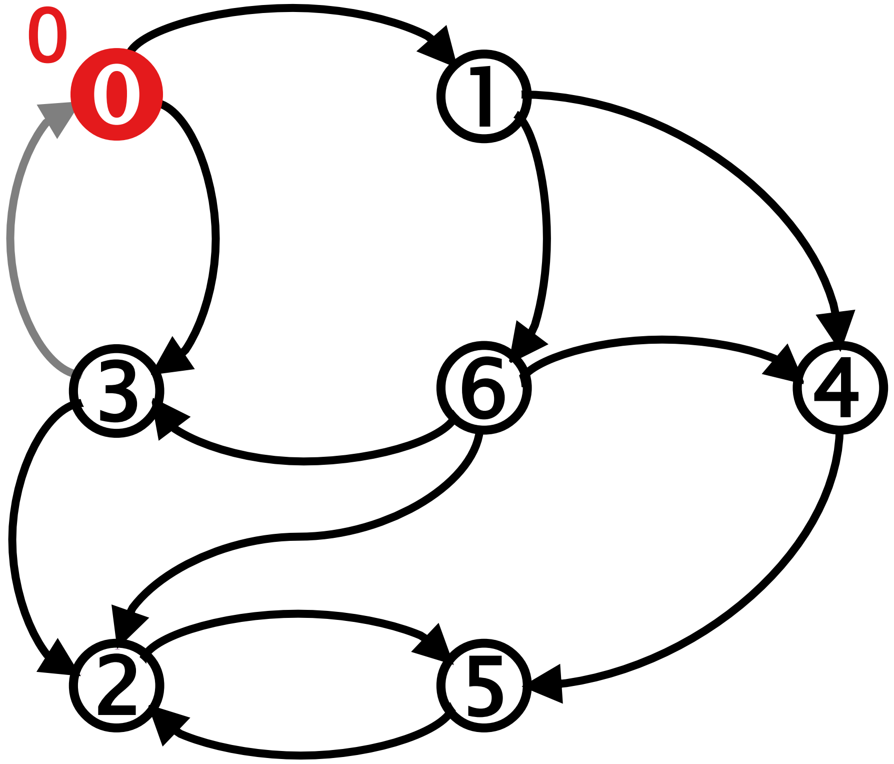
$level = 0$ ⇒

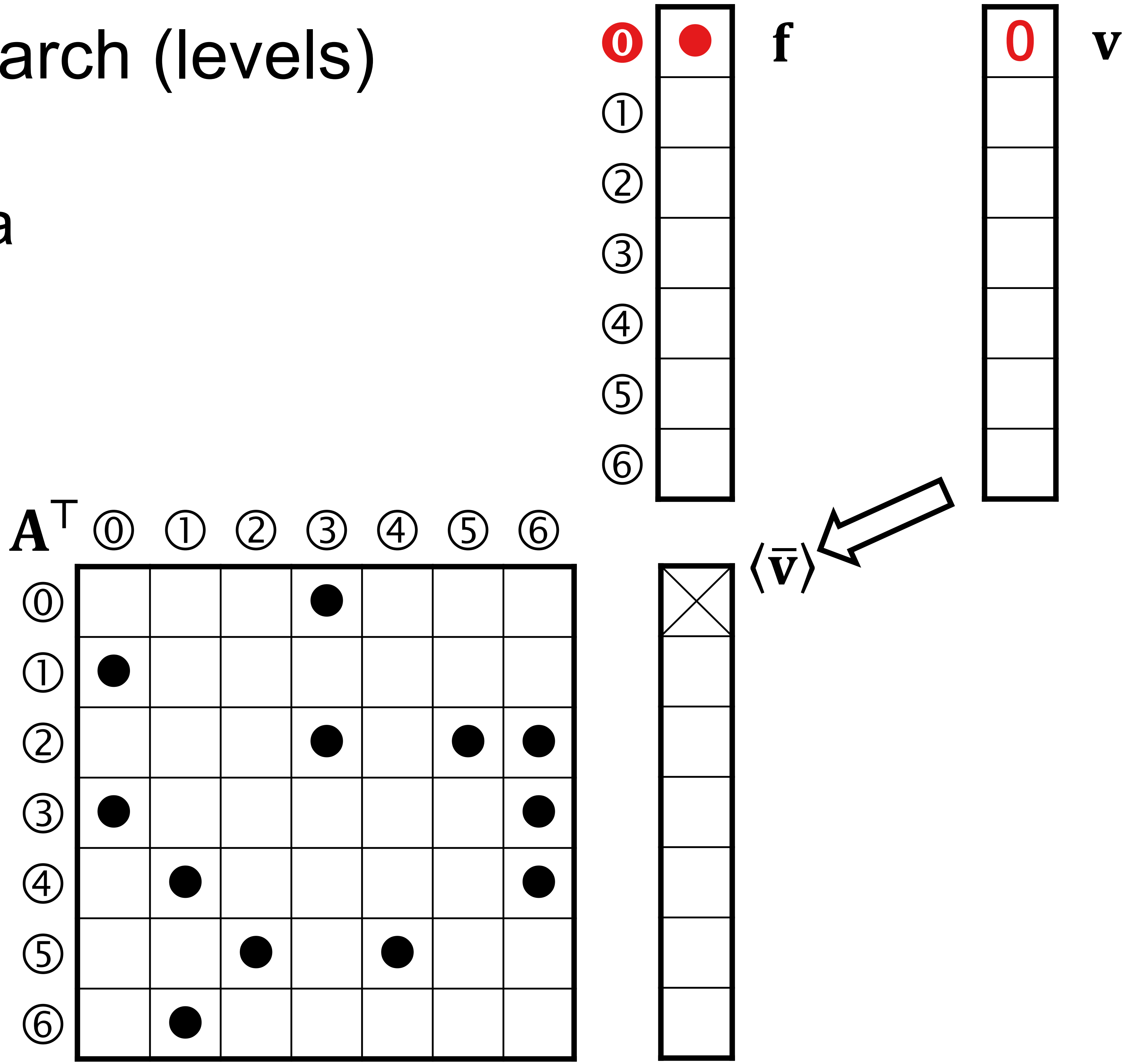
$\mathbf{v}$
0

Example: Breadth-First Search (levels)

$level = 0$

$\mathbf{v} \mathrel{+}= level * \mathbf{f}$ // Use $\mathbf{v}$ as a mask, $\langle\bar{\mathbf{v}}\rangle$.



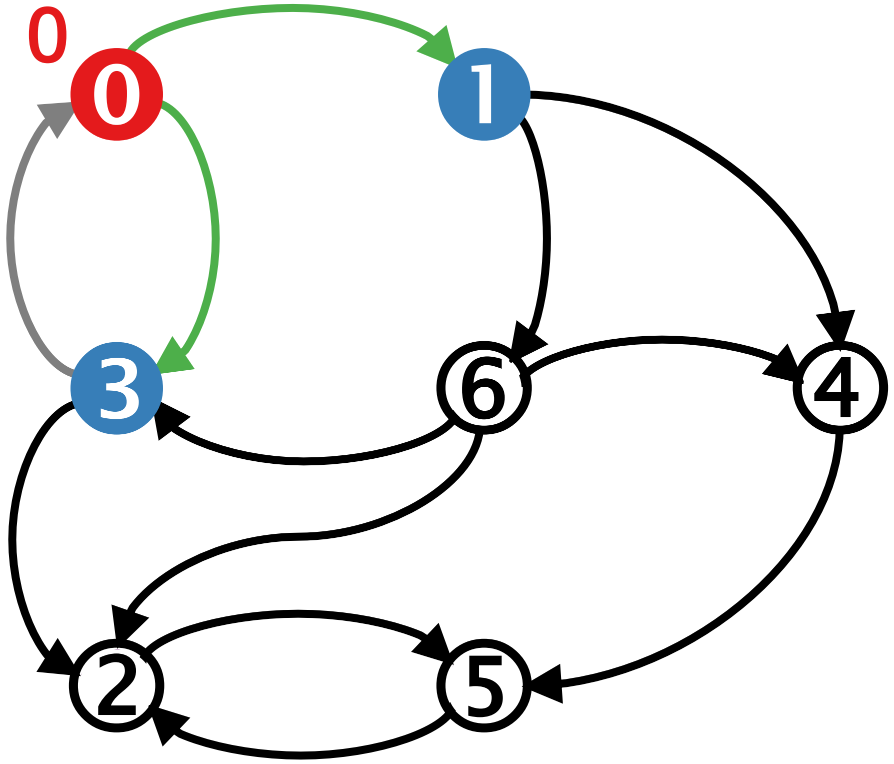


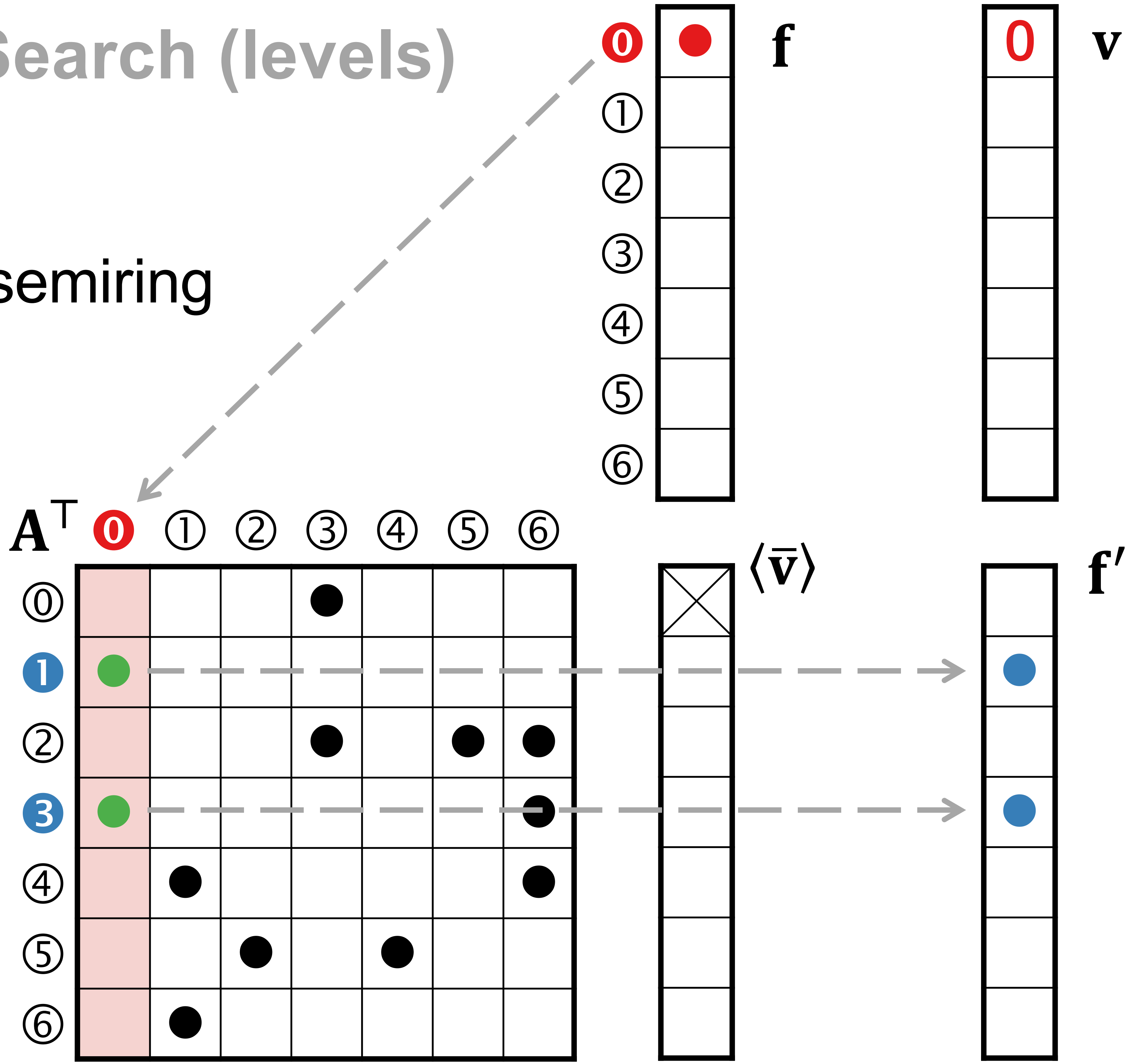
Example: Breadth-First Search (levels)

$$level = 0$$

$$\mathbf{v} \mathrel{+}= level * \mathbf{f}$$

$$\mathbf{f}'\langle\bar{\mathbf{v}}\rangle = \mathbf{A}^{\mathsf{T}} \oplus . \otimes \mathbf{f}$$ // Boolean semiring





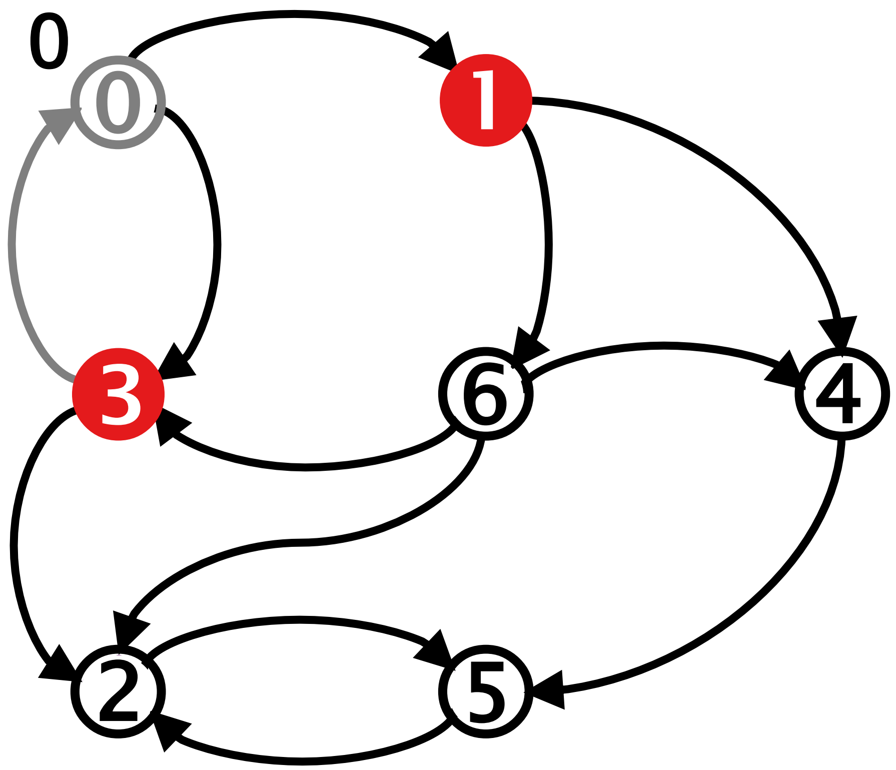
Example: Breadth-First Search (levels)

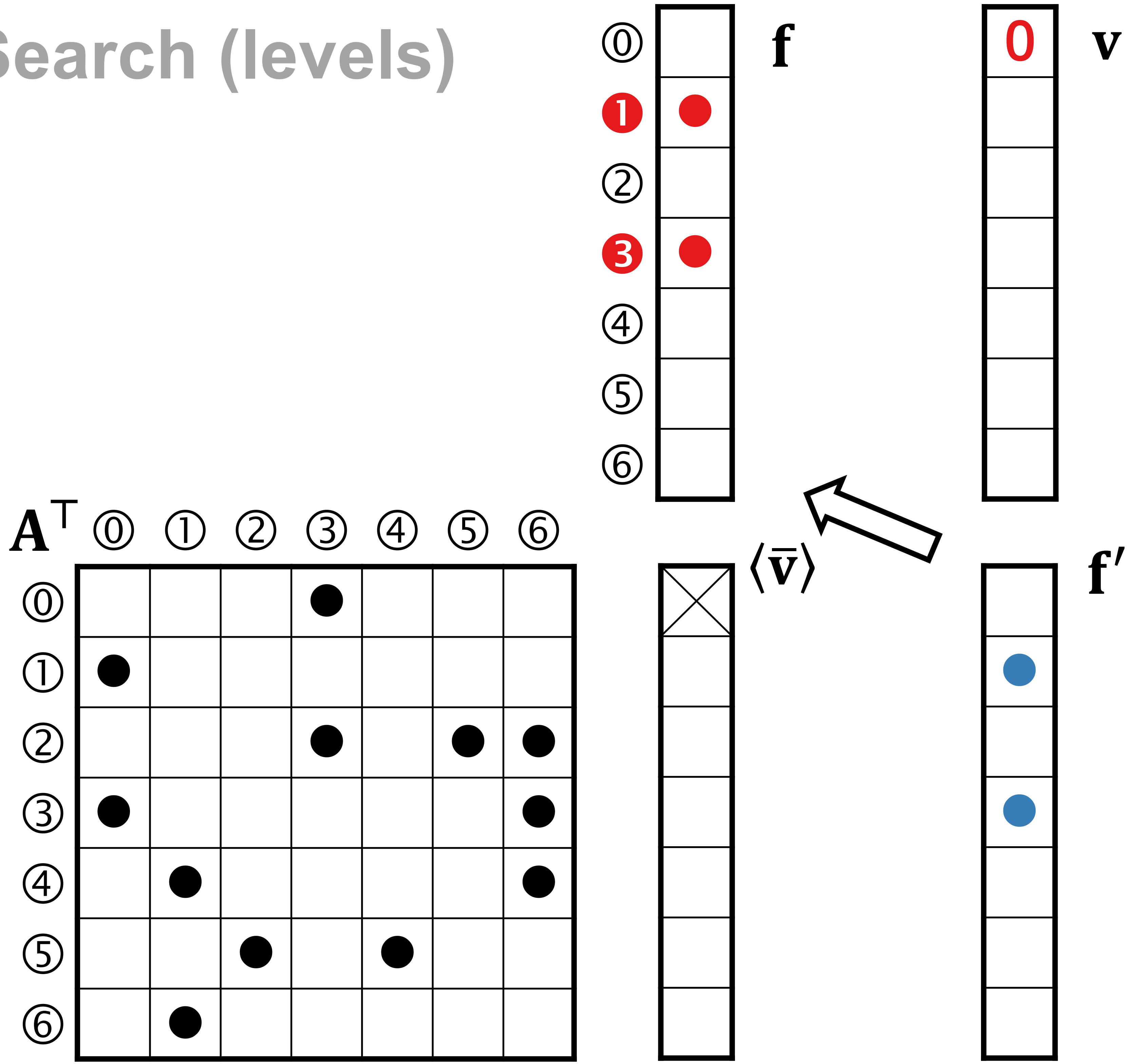
$$\mathit{level} = 0$$

$$\mathbf{v} \mathrel{+}= \mathit{level} * \mathbf{f}$$

$$\mathbf{f}'\langle\bar{\mathbf{v}}\rangle = \mathbf{A}^{\top} \oplus . \otimes \mathbf{f}$$

$$\mathbf{f} = \mathbf{f}'$$





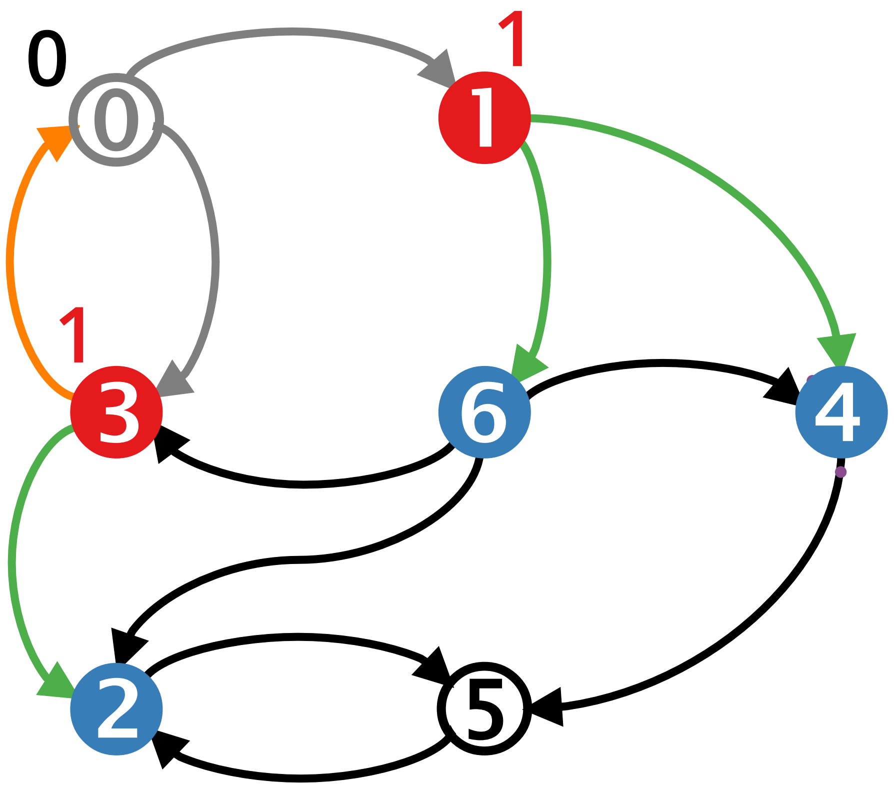
Example: Breadth-First Search (levels)

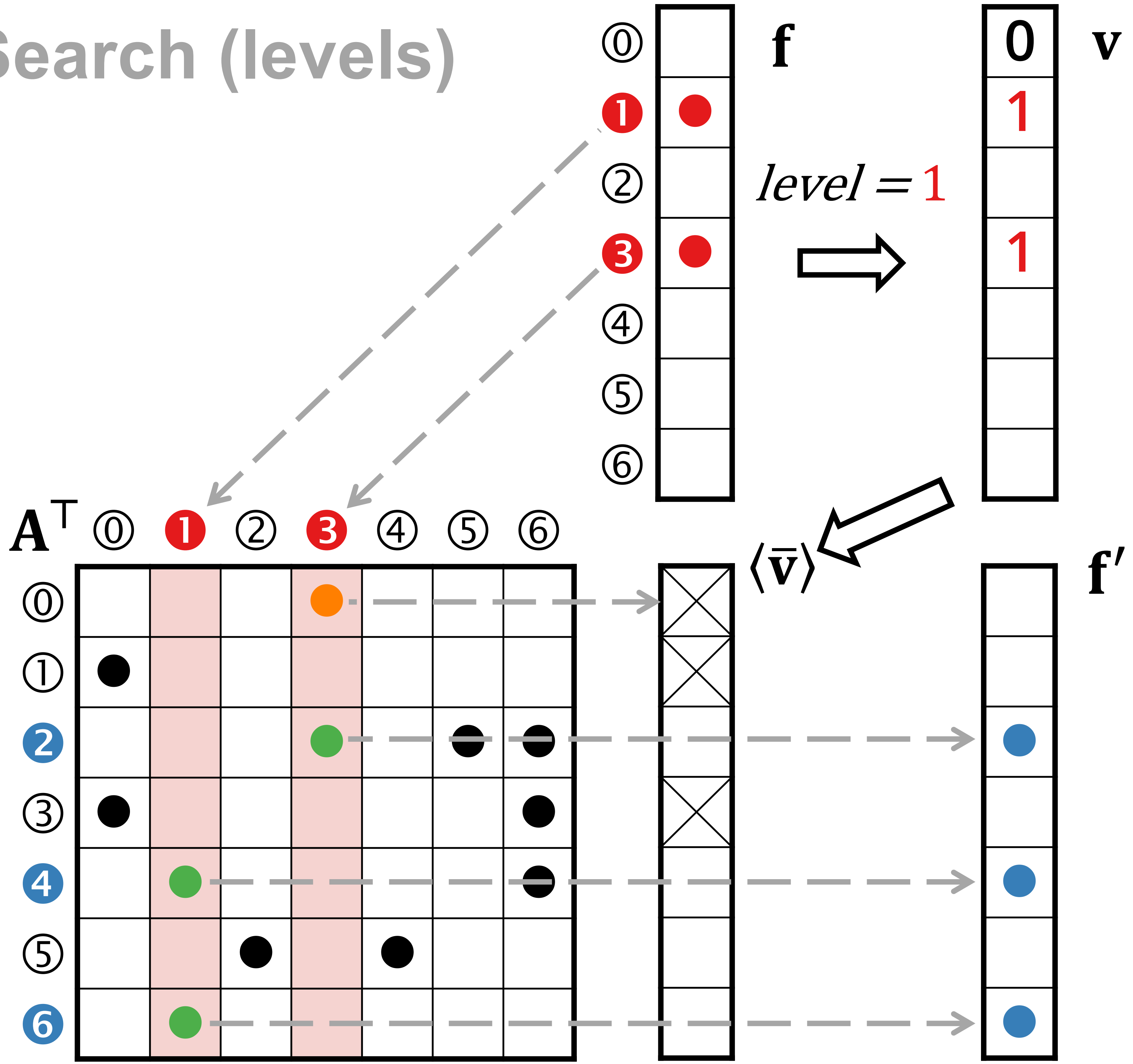
$$\mathit{level} = 1$$

$$\mathbf{v} \mathrel{+}= \mathit{level} * \mathbf{f}$$

$$\mathbf{f}'\langle\bar{\mathbf{v}}\rangle = \mathbf{A}^{\mathsf{T}} \oplus . \otimes \mathbf{f}$$

$$\mathbf{f} = \mathbf{f}'$$





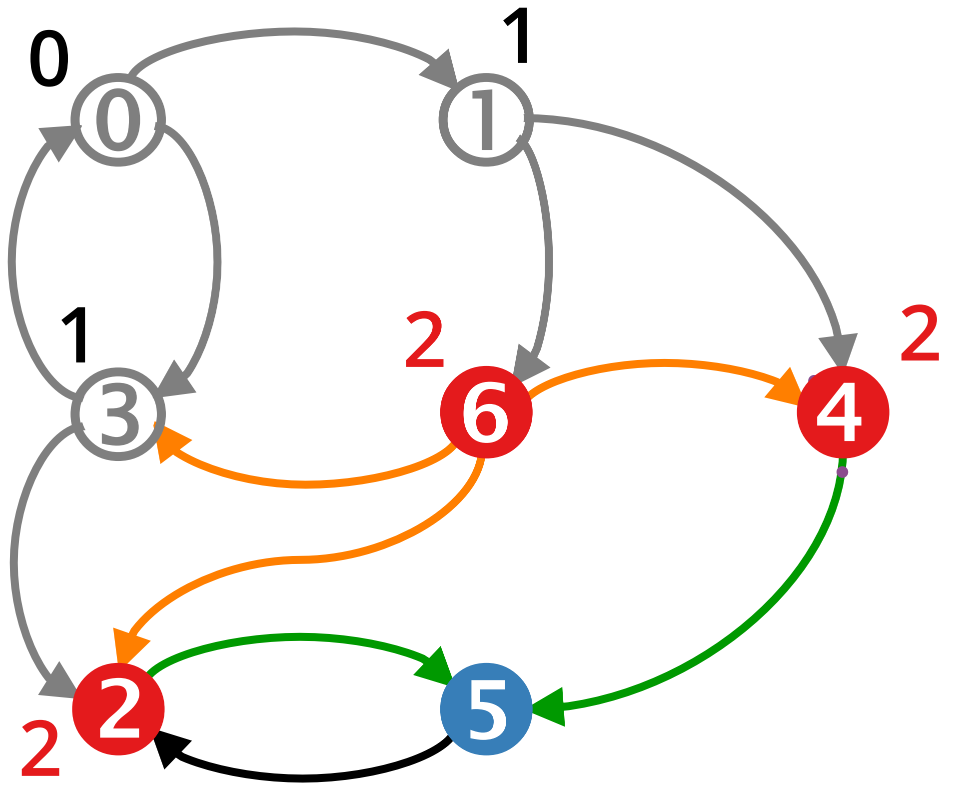
Example: Breadth-First Search (levels)

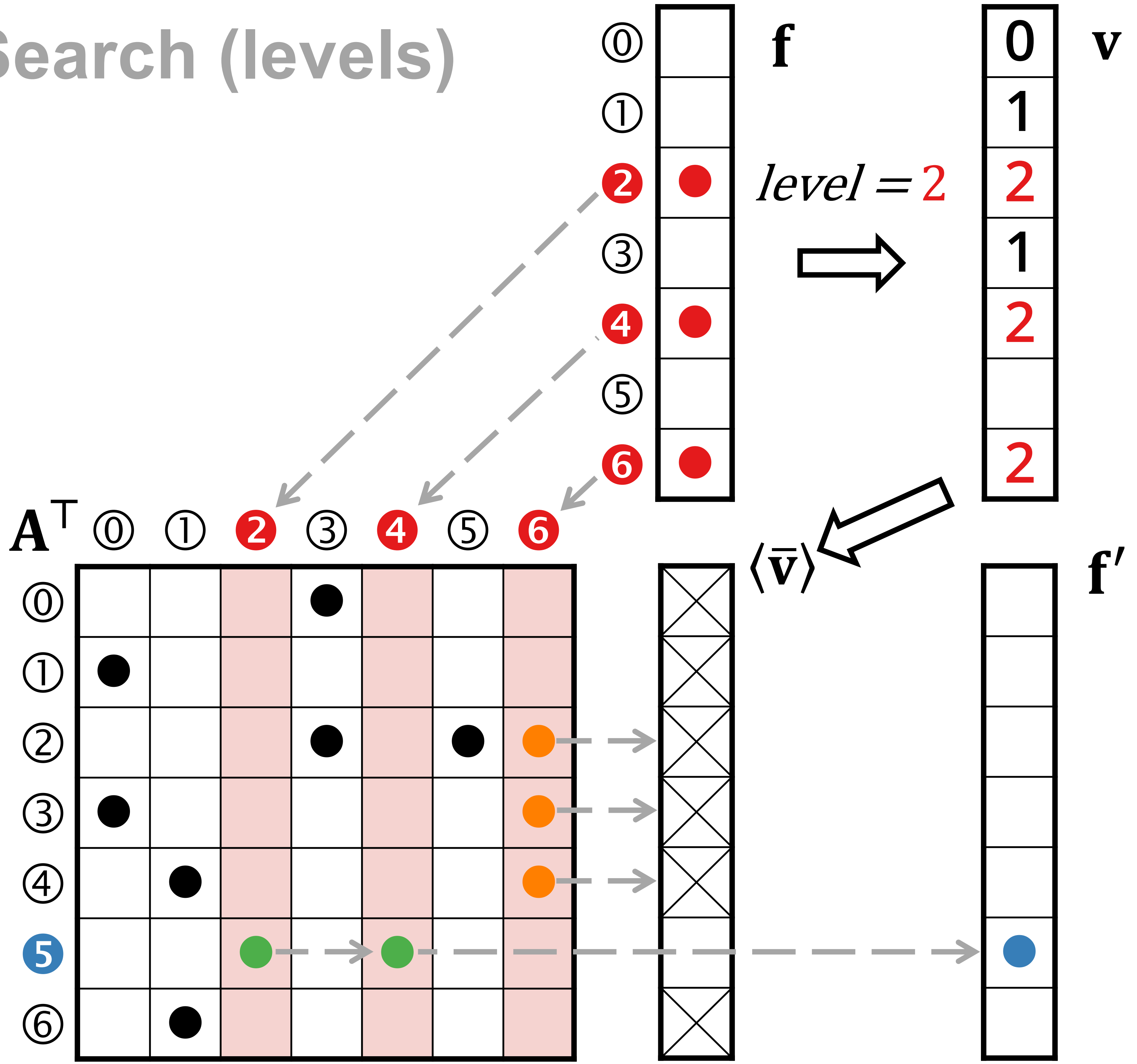
$$\mathit{level} = 2$$

$$\mathbf{v} \mathrel{+}= \mathit{level} * \mathbf{f}$$

$$\mathbf{f}'\langle\bar{\mathbf{v}}\rangle = \mathbf{A}^{\top} \oplus.\otimes \mathbf{f}$$

$$\mathbf{f} = \mathbf{f}'$$





Example: Breadth-First Search (levels)

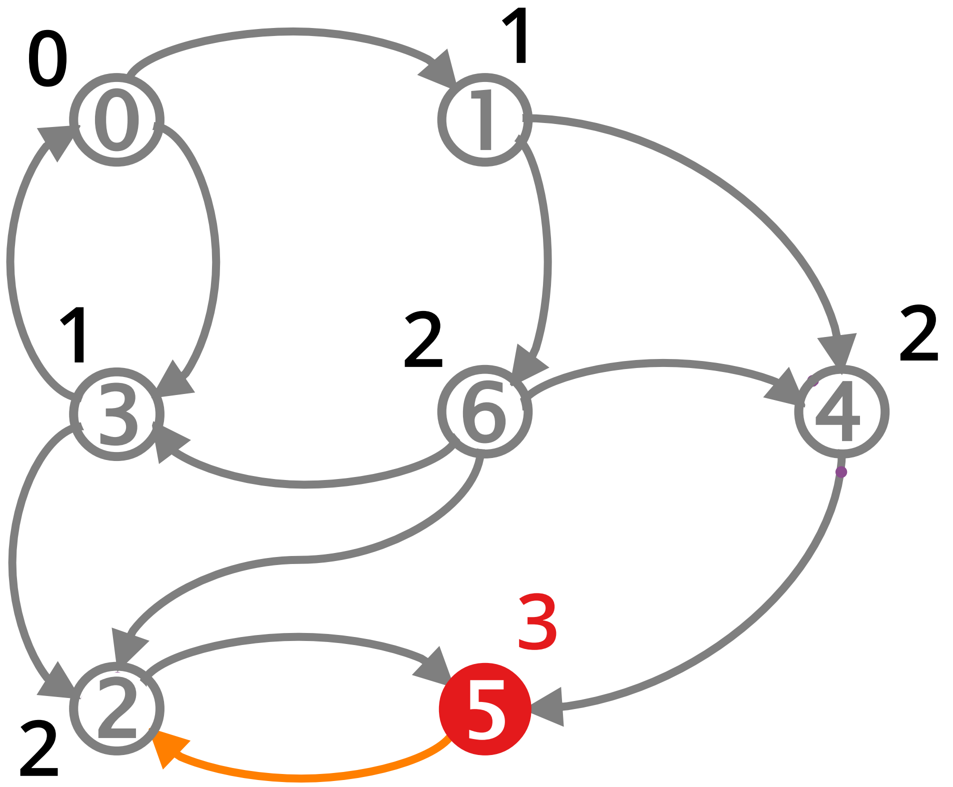
$$level = 3$$

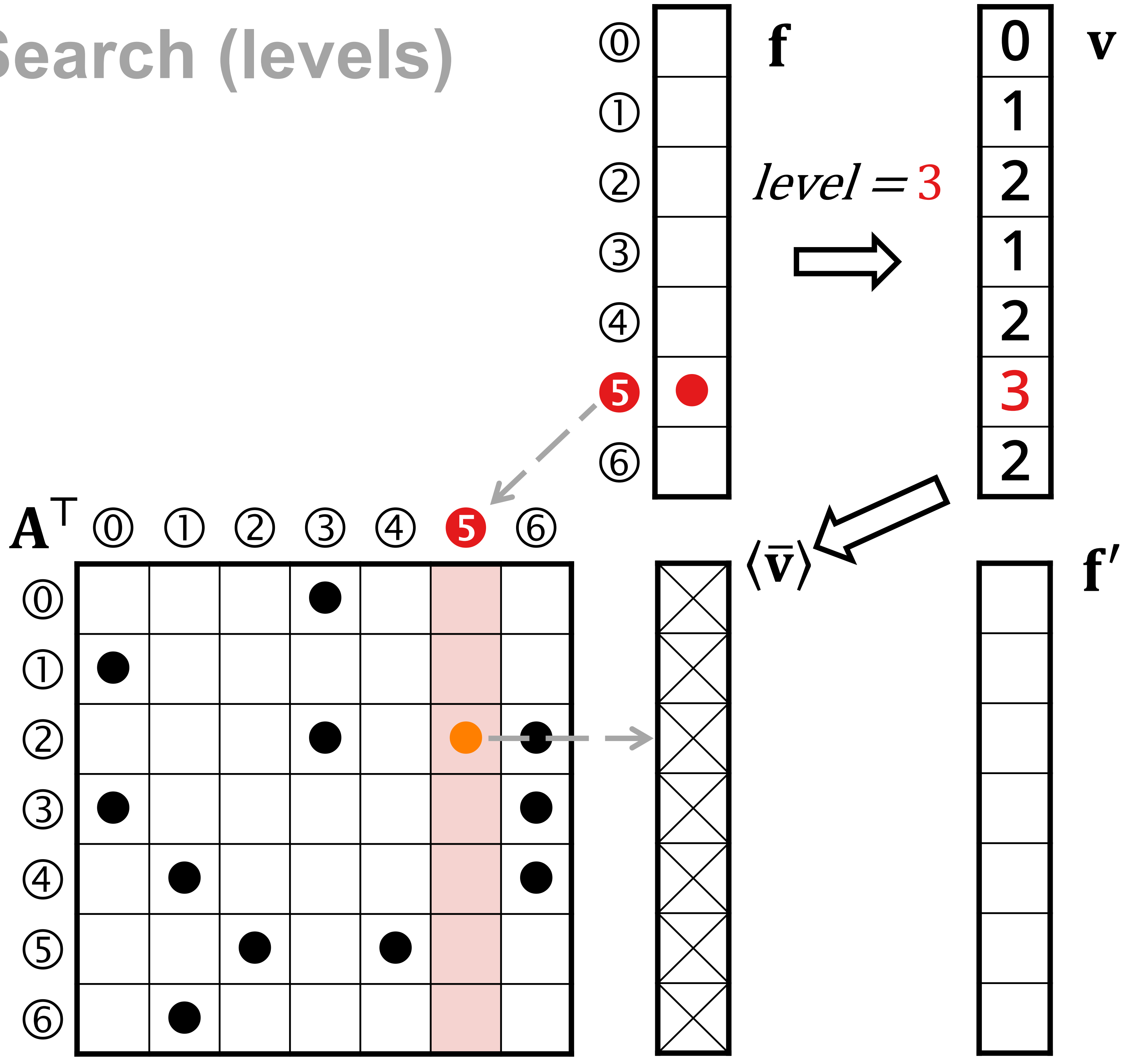
$$\mathbf{v} \mathrel{+}= level * \mathbf{f}$$

$$\mathbf{f}'\langle \bar{\mathbf{v}} \rangle = \mathbf{A}^{\top} \oplus . \otimes \mathbf{f}$$

$$\mathbf{f} = \mathbf{f}'$$

if f.empty() return v





Example: Breadth-First Search (levels)

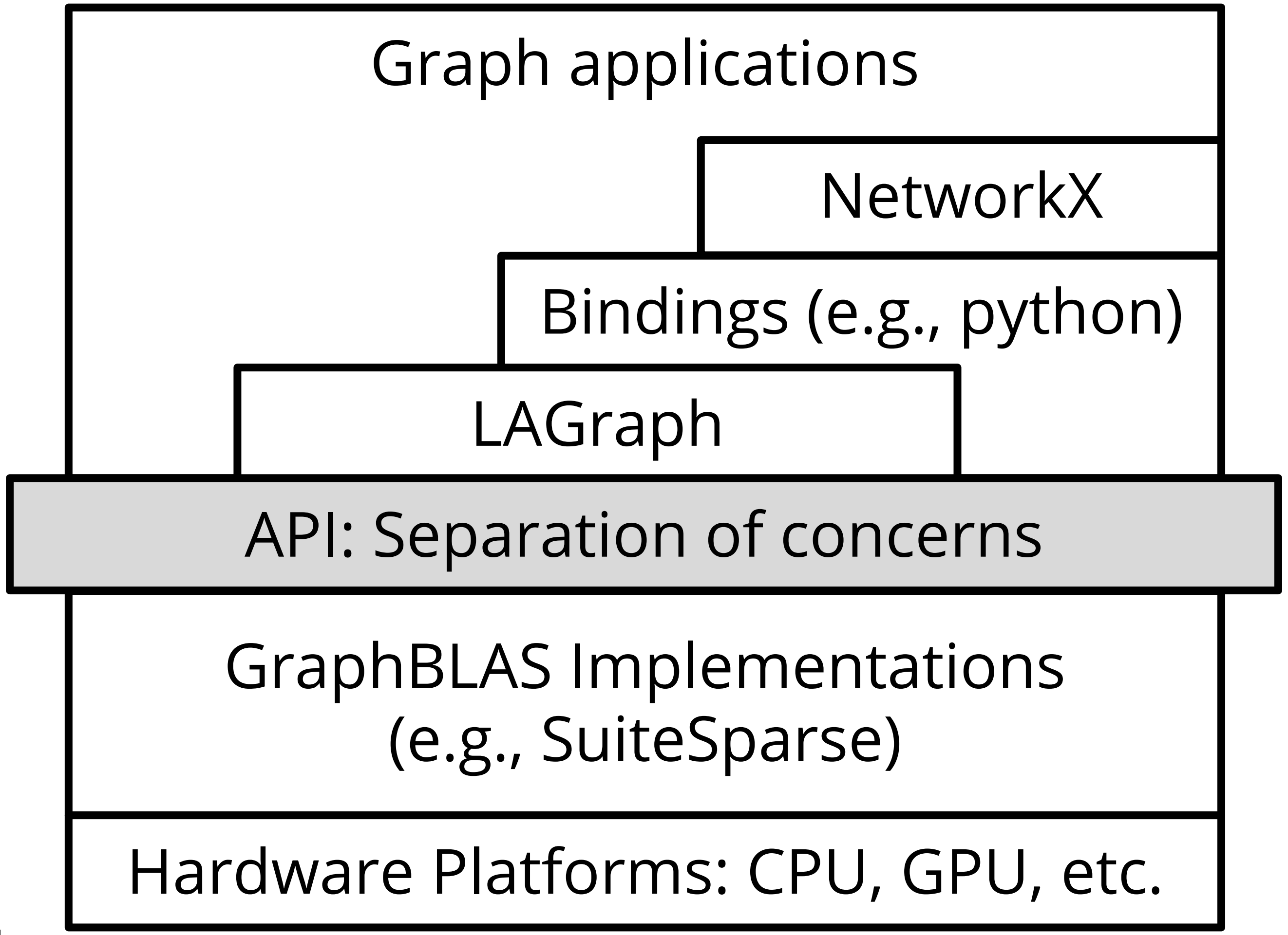
- **Input:** adjacency matrix $\mathbf{A}$ (Boolean), source vertex src (integer)
- **Output:** visited vertices vector, $\mathbf{v}$ (integer)
- **Workspace:** frontier vector $\mathbf{f}$ (Boolean)

1. $\mathbf{f}(src) = \text{true}$
2. $level = 0$
3. $\text{while } !\,\mathbf{f}.\text{empty}()$
4. $\quad \mathbf{v} \mathrel{+}= level * \mathbf{f}$
5. $\quad \mathbf{f}\langle\overline{\mathbf{v}}\rangle = \mathbf{A}^{\top} \oplus . \otimes \mathbf{f}$ `// using the Boolean semiring (OR.AND)`
6. $\quad {+}{+}level$

Resources/Activities (some covered in the next talk?)

- **C API Specification**
  - https://github.com/GraphBLAS/graphblas-api-c
- **C API Implementation: SuiteSparse:GraphBLAS**
  - https://github.com/DrTimothyAldenDavis/GraphBLAS
- **LAGraph Algorithms Repository**
  - https://github.com/GraphBLAS/LAGraph
- **Language Bindings: python, Julia, postgres, etc**
  - https://github.com/python-graphblas/python-graphblas
  - https://github.com/JuliaSparse/SuiteSparseGraphBLAS.jl
  - https://github.com/michelp/pggraphblas

- **IN PROGRESS: C++ API Specification and Reference Lib.**
  - https://github.com/GraphBLAS/graphblas-api-cpp
  - https://github.com/GraphBLAS/rgri



Questions?

Scott McMillan
Principal Research Engineer

Advanced Computing Lab
AI Division
Software Engineering Institute
Carnegie Mellon University

mailto: smcmillan@sei.cmu.edu

Website: http://graphblas.org

- Lists workshops and conferences
- Links to the latest API Specifications
- Teams developing implementations
- Other useful resources

Mailing list: Graphblas@lists.lbl.gov

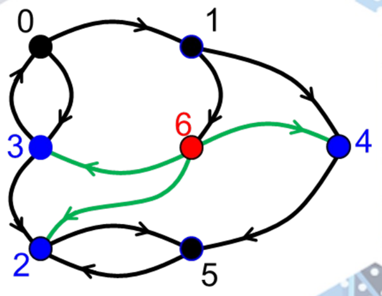
- Hosted by LBL (mailto:abuluc@lbl.gov)
- Join the Forum by joining the list

Monthly teleconference:

- Second Friday of every month, 12pm Eastern Time
- Send email (mailto:kepner@ll.mit.edu) to receive the calendar invite and Zoom ID.

Backups

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213



Graphs (GraphBLAS) and Storage (TileDB) as Sparse Linear Algebra

Timothy G. Mattson, Intel

Graph Algorithms and Linear Algebra

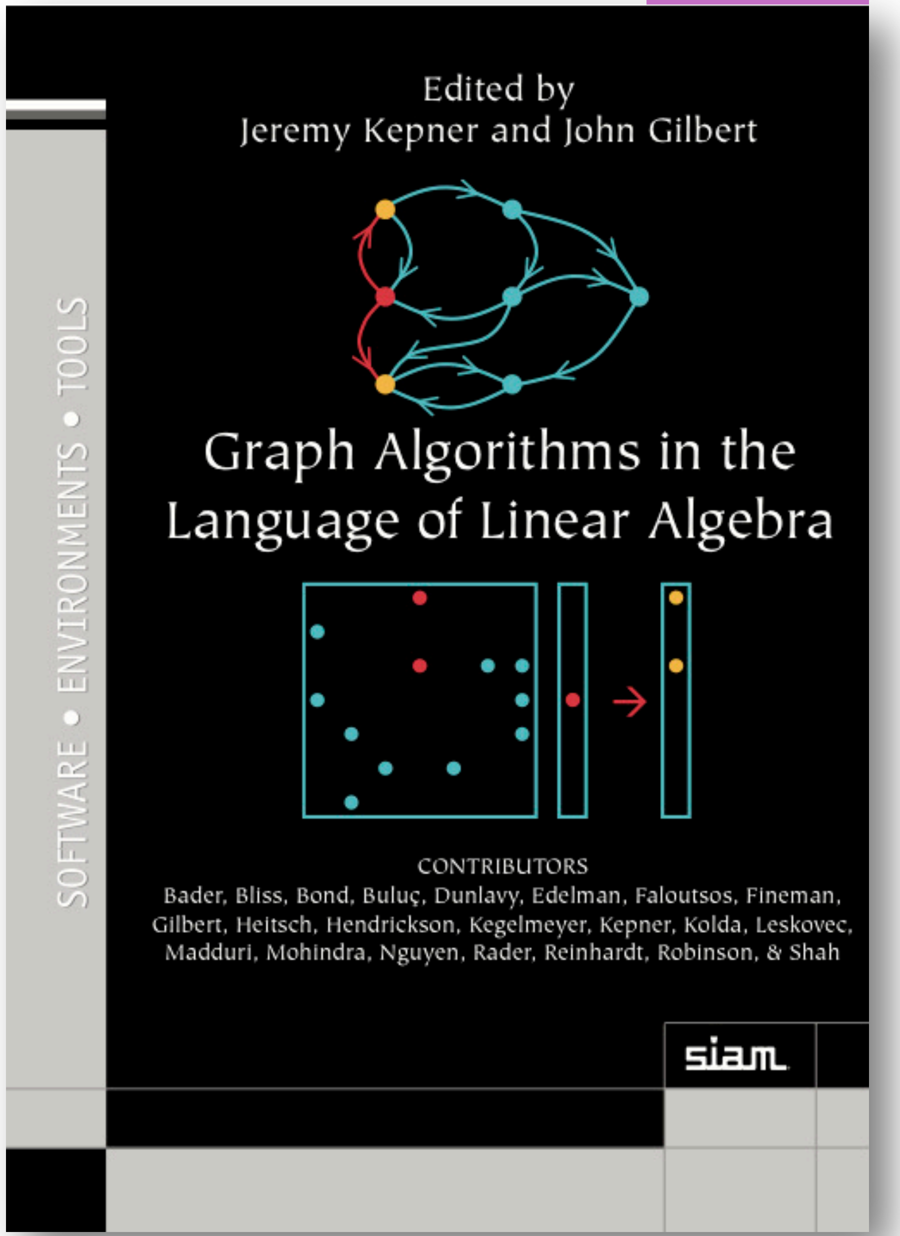
This is not a new idea

- At least since the 1950's
- There is even has a book.

Benefits of graphs as linear algebra

- Well suited to memory hierarchies of modern microprocessors
- Can utilize decades of experience in distributed/parallel computing from linear algebra in supercomputing.
- Easier to understand … for some people.

2011



SuiteSparse:GraphBLAS:
An Implementation of the C API

Tim Davis, *Texas A&M University*

SuiteSparse:GraphBLAS v7.4.x

- **Conforms to the v2.0 C API** (Nov 2021)
- New features:
  - faster hypersparse matrices (the “hyperhash”, avoids binary search), in v7.3.0beta
  - pack/unpack (O(1)-time move semantics)
  - named types and operators (for future JIT)
  - matrix and vector sort
  - eWiseUnion (like eWiseAdd but with 2 scalars; all entries in output go through the operator)
  - matrix and vector iterators
  - matrix reshape
- Performance:
  - GrB_mxm, particularly with sparse-times-dense or dense-times-sparse. AVX2 and AVX512 exploit
  - faster MATLAB interface
- Port to Octave 7
- Supported by Intel, NVIDIA, Redis, MIT Lincoln Lab, MathWorks, Julia Computing

SuiteSparse versus the Intel MKL sparse library

computation	format	MKL method	MKL time (sec) 1st	MKL time (sec) 2nd	SuiteSparse time (sec)	speedup 1st	speedup 2nd
y+=S*x	S by row	mkl_sparse_d_mv	2.54	1.27	1.21	2.10	1.05
y+=S*x	S by col	mkl_sparse_d_mv	7.22	7.22	1.98	3.65	3.65
C+=S*F	S by row, F by row	mkl_sparse_d_mm	2.95	1.90	1.98	1.49	**.96**
C+=S*F	S by row, F by col	mkl_sparse_d_mm	6.12	4.99	1.48	4.13	3.37
C+=S*F	S by col, F by row	mkl_sparse_d_mm	28.82	28.82	13.78	2.09	2.09
C+=S*F	S by col, F by col	mkl_sparse_d_mm	78.82	5.17	9.38	8.40	**.55**
C=S+B	S by row	mkl_sparse_d_add	30.77	30.77	1.44	21.37	21.37
C=S'+B	S by row	mkl_sparse_d_add	102.09	27.30	16.29	6.26	1.67
C=S'	S by row	mkl_sparse_convert_csr	77.27	77.27	14.80	5.22	5.22

Table 4. SuiteSparse vs MKL 2022 with the GAP-Twitter matrix

Work in progress and future work

- **CUDA acceleration** (with J. Eaton and C. Nolet, NVIDIA)**: 3x to 9x speedup in GrB_mxm**
- **Julia integration (just announced v0.7), replacing Julia SparseArrays**
- more MATLAB integration
- further Python integration
- JIT for faster user-defined types and operations
- aggressive non-blocking mode, kernel fusion
- x=A\b over a field
- more built-in types (FP16, complex integers, …)
- faster kernels (GrB_mxm for sampled dense-dense matrix multiply)
- matrices with shallow components

https://github.com/DrTimothyAldenDavis/GraphBLAS

LAGraph: graph algorithms library

Tim Davis, Scott McMillan, Gabor Szarnyas,
Tim Mattson, Jim Kitchen, Eric Welch,
David Bader, Roi Lipman, and contributors.

LAGraph: graph algorithm library

https://github.com/GraphBLAS/LAGraph

Version 1.0 released in September 2022

6 polished, stable algorithms (the GAP benchmark):

- Breadth-first search
- Betweenness-centrality
- PageRank
- Connected Components
- Single-source Shortest-Path
- Triangle Counting

Stable utilities

- malloc/calloc/realloc/free wrappers
- create/destroy the LAGraph_Graph
- compute properties: degree, A', # diag entries
- delete properties
- display graph
- Matrix Market file I/O (very slow)
- Sorting
- thread control
- timing
- type management

Graphalytics algorithms in next Release

Many experimental algorithms to be curated

- K-truss, All K-truss
- Bellman-Ford single-source shortest path
- Maximal independent set
- Triangle Centrality
- Community detection w/ label propagation
- Deep Neural Network Inference
- Strongly Connected Components
- Minimum Spanning Forest
- Local Clustering Coefficient
- K-core
- Counting all size-4 graphlets
- Triangle polling
- Fiedler vector

Experimental utilities

- random matrix, vector generators
- Binary matrix file I/O (very fast), serialize/deserialize, parallel LZ4 comp.

LAGraph: graph algorithm library

https://github.com/GraphBLAS/LAGraph

Version 1.0 released in September 2022

6 polished, stable algorithms (the GAP benchmark):

- Breadth-first search
- Betweenness-centrality
- PageRank
- Connected Components
- Single-source Shortest-Path
- Triangle Counting

Stable utilities

- malloc/calloc/realloc/free wrappers
- create/destroy the LAGraph_Graph
- compute properties: degree, A', # dia
- delete properties
- display graph
- Matrix Market file I/O (very slow)
- Sorting
- thread control
- timing
- type management

Graphalytics algorithms in next Release

Many experimental algorithms to be curated

- K-truss, All K-truss
- Bellman-Ford single-source shortest path
- Maximal independent set
- Triangle Centrality
- Community detection w/ label propagation
- Deep Neural Network Inference
- Strongly Connected Components
- Minimum Spanning Forest
- Local Clustering Coefficient
- K-core
- Counting all size-4 graphlets
- Triangle polling
- Fiedler vector

Experimental utilities

- random matrix, vector generators
- Binary matrix file I/O (very fast), serialize/deserialize, parallel LZ4 comp.

python-graphblas + NetworkX

Jim Kitchen, *Anaconda*,

Eric Welch, *NVIDIA*,

and contributors.

Python package for accelerated GraphBLAS

- python-graphblas
  - package that dispatches to SuiteSparse:GraphBLAS for computation
  - Stays in sync with advances in SuiteSparse:GraphBLAS
- graphblas-algorithms
  - Like LAGraph, a set of graphblas algorithms
  - Built on top of python graphblas

Dispatching Example with graphblas-algorithms

```python
import networkx as nx

G = nx.erdos_renyi_graph(8000, 0.02)

k = nx.k_truss(G, 5)
```

8000 nodes, ~ 640_000 edges

This takes 10.7 seconds

The k-truss is the maximal induced subgraph of G with each edge belonging to at least k-2 triangles.

```python
import networkx as nx
import graphblas_algorithms as ga

G = nx.erdos_renyi_graph(8000, 0.02)
G2 = ga.Graph.from_networkx(G)

k = nx.k_truss(G2, 5)
```

```
conda install -c conda-forge graphblas-algorithms
```

-or-

```
pip install graphblas-algorithms (Linux Only)
```

This takes 0.5 seconds

This takes 0.28 seconds

* Notice that dispatching is opt-in

Benchmarks: GraphBLAS vs NetworkX

Hardware: NVIDIA DGX-1
CPU: Dual 20 Core Intel Xeon E5-2698 v4 2.2GHz
RAM: 512 GB 2133 MHz DDR4 RDIMM

Speed-up

	amazon	google	pokec	enron	preferentialAttachment	caidaRouterLevel	dblp	citationCiteseer	coAuthorsDBLP	as-Skitter	coPapersCiteseer	coPapersDBLP	NetworkX run times
# of vertices	262,111	916,428	1,632,804	36,692	100,000	192,244	326,186	268,495	299,067	1,696,415	434,102	5,404,486	
# of edges	1,234,877	5,105,039	30,622,564	367,662	999,970	1,218,132	1,615,400	2,313,294	1,955,352	22,190,596	32,071,440	30,491,458	
degree centrality	32	48	31	29	60	140	65	180	200	530	190	220	0.25-1 s
reciprocity	290	370	470	230	600	840	1600	1000	1400	1700	2200	2200	3-5 min
generalized degree	N/A (Requires Undirected Graph)			140	160	190	150	220	150	1700	500	360	10-30 min
k-truss(k=5)				53	800	140	130	150	170	350	2000	1100	30-100 min
pagerank	130	340	930	50	240	250	390	580	810	1800	3900	4200	1 min
eigenvector centrality	53	120	150	61	650	740	1300	1100	1300	2000	5200	5300	30-100 min
katz centrality	420	530	830	300	1100	1400	1700	2100	2300	3400	7500	7600	hours-days
clustering	160	900	620	370	370	290	280	540	380	11000	2600	2100	10-30 min
transitivity	180	270	440	830	970	900	730	1600	970	20000	6600	5000	10-30 min
square clustering	N/A			1200	950	1400	1800	1100	1300	DNF	DNF	21000	days-weeks?
pagerank (scipy)	3.4	14	23	2.1	3.3	3.8	6.3	9.8	11	20	23	27	0.25-1 s

How to Try It Out

Dispatching is a feature in NetworkX 3.0

- Note: This is an experimental feature, and the API may change. Do not rely on this for production applications.

Install graphblas-algorithms and optional dependencies

- `conda install -c conda-forge graphblas-algorithms`
- `conda install pandas scipy` # needed for display and converting to NetworkX

Try the Dispatch Example

- https://github.com/python-graphblas/graphblas-algorithms