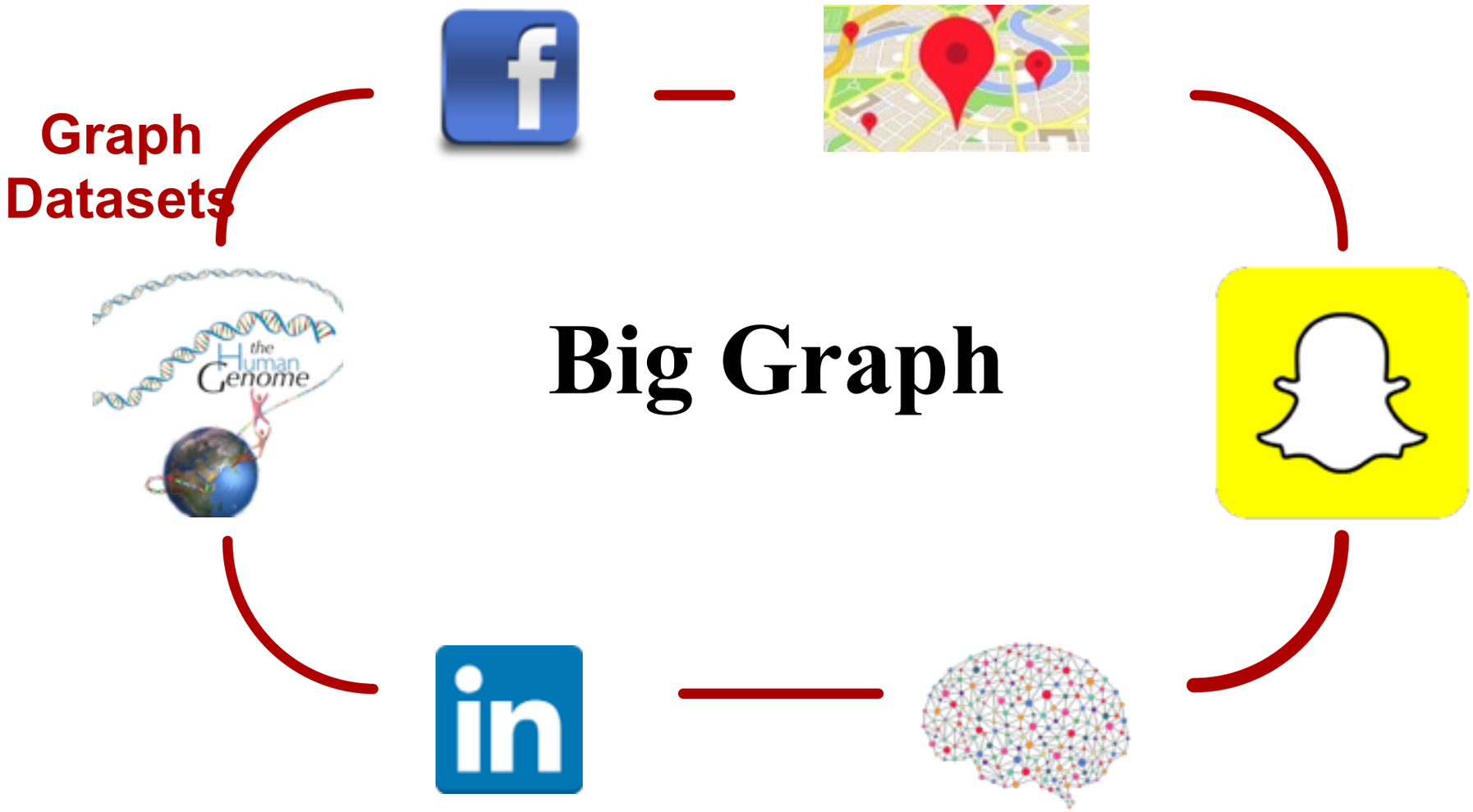
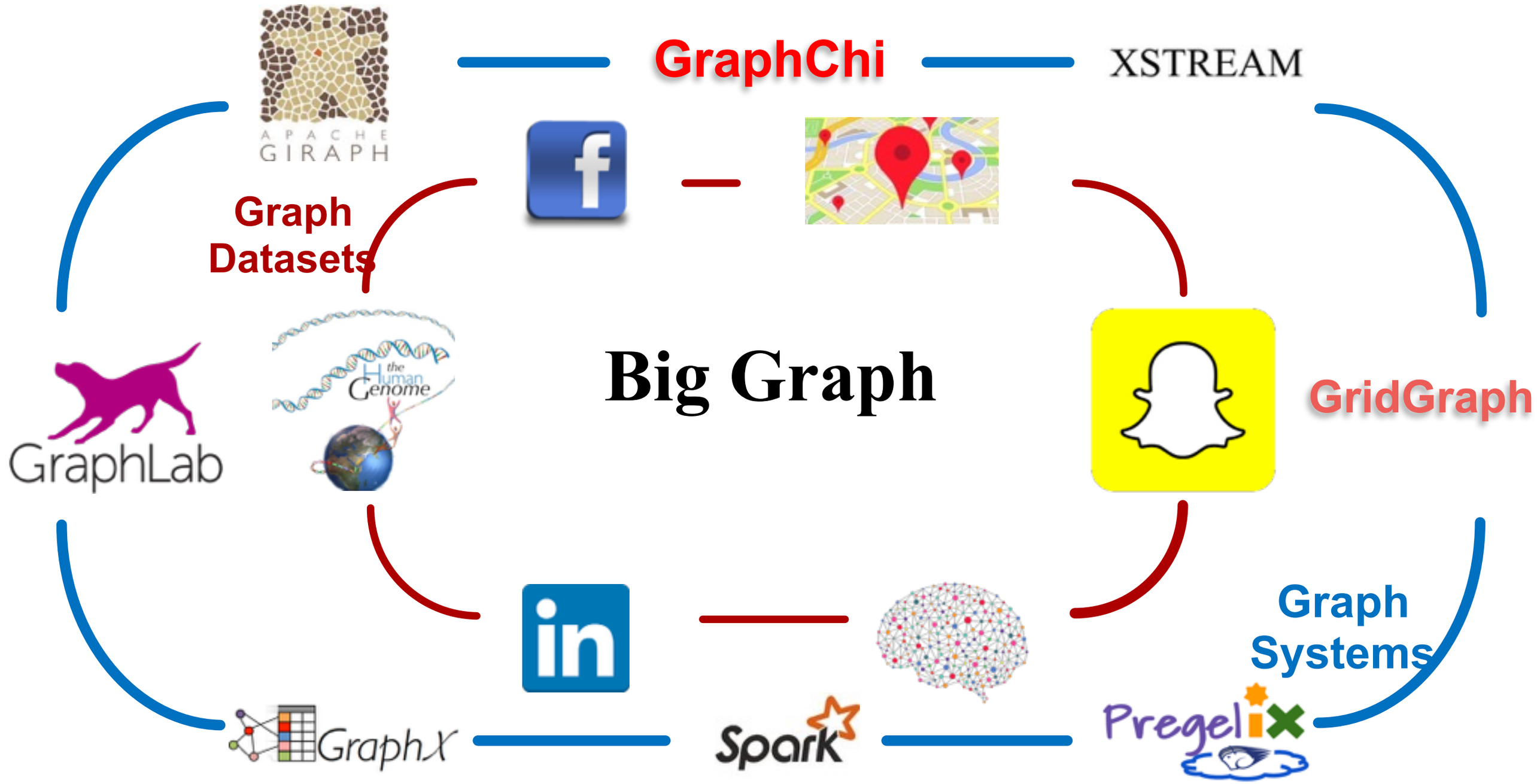
RStream:Marrying Relational Algebra with Streaming for Efficient Graph Mining on A Single Machine

Kai Wang,[1] Zhiqiang Zuo,[2] John Thorpe,[1] Tien Quang Nguyen,[3] Guoqing Harry Xu[1]

UCLA[1] Nanjing University[2] Facebook[3]

Big Graph


Graph Datasets
Big Graph
the Human Genome


APACHE GIRAPH
GraphChi
XSTREAM
Graph Datasets
the Human Genome
Big Graph
GridGraph
GraphLab
Graph Systems
GraphX
Spark
Pregelix

Graph Analytical Problems

Graph Analytical Problems

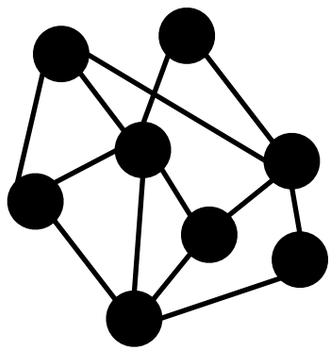
Graph Computation

Graph Analytical Problems

PageRank

Connected Component

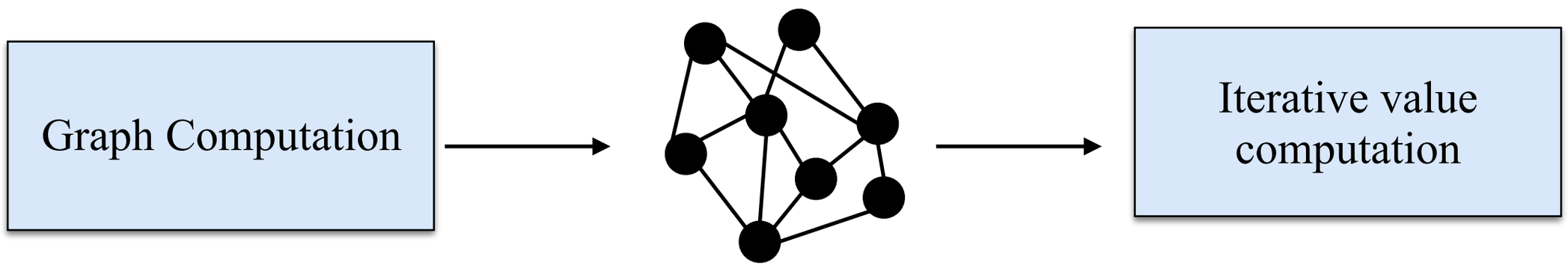
Graph Computation



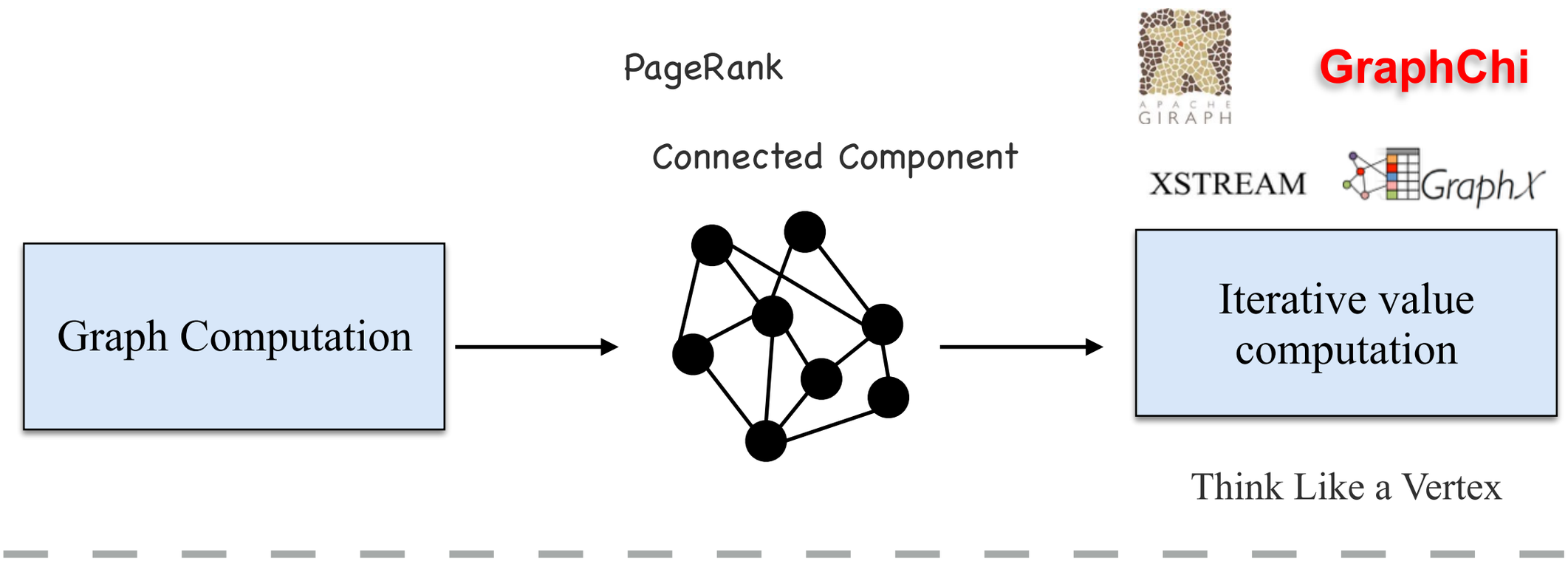
Graph Analytical Problems

PageRank

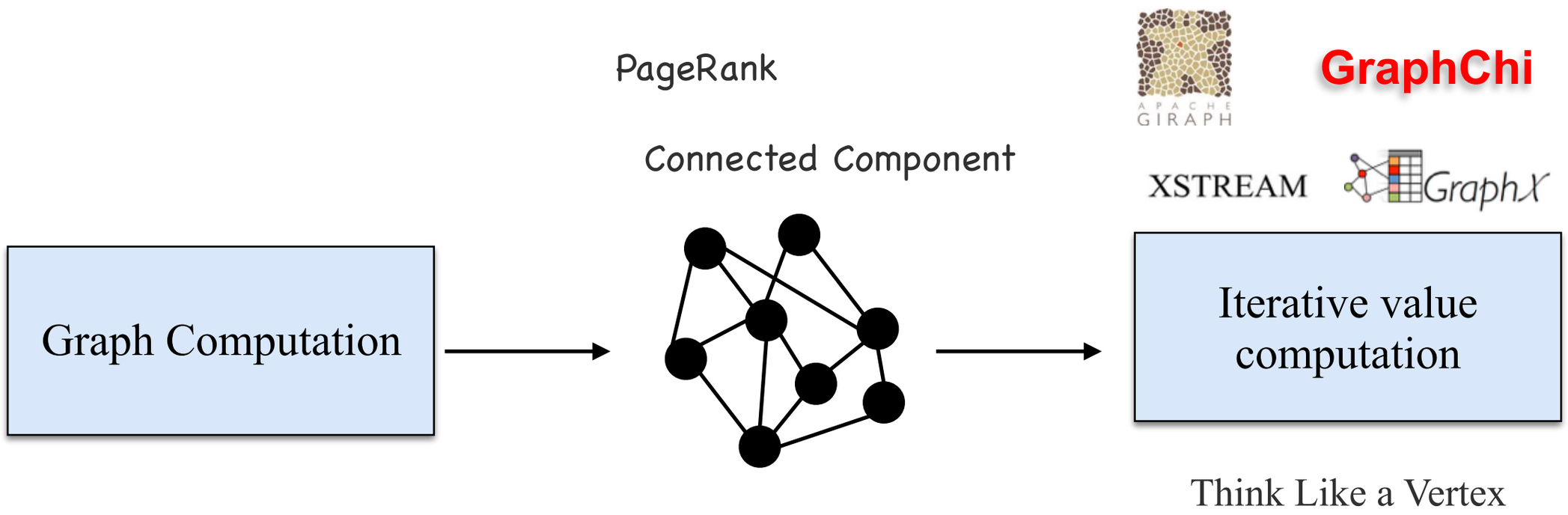
Connected Component



Graph Analytical Problems



Graph Analytical Problems

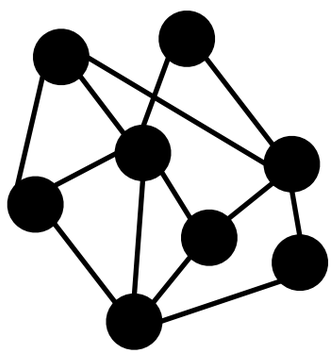


Graph Mining

Graph Analytical Problems

PageRank

Connected Component



Graph Computation

Iterative value computation

GraphChi

XSTREAM

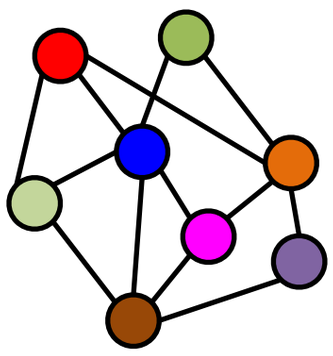
GraphX

Think Like a Vertex

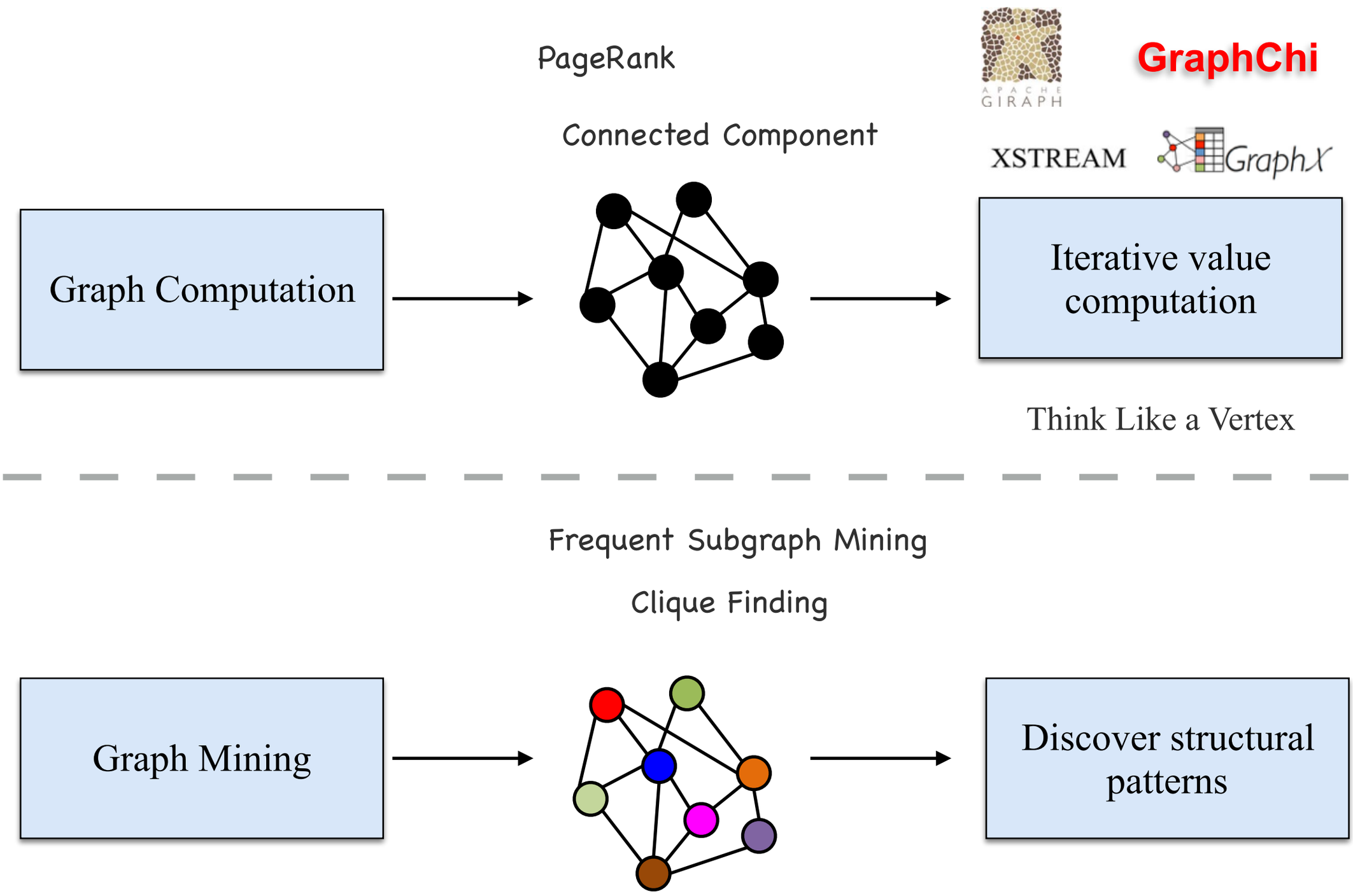
Frequent Subgraph Mining

Clique Finding

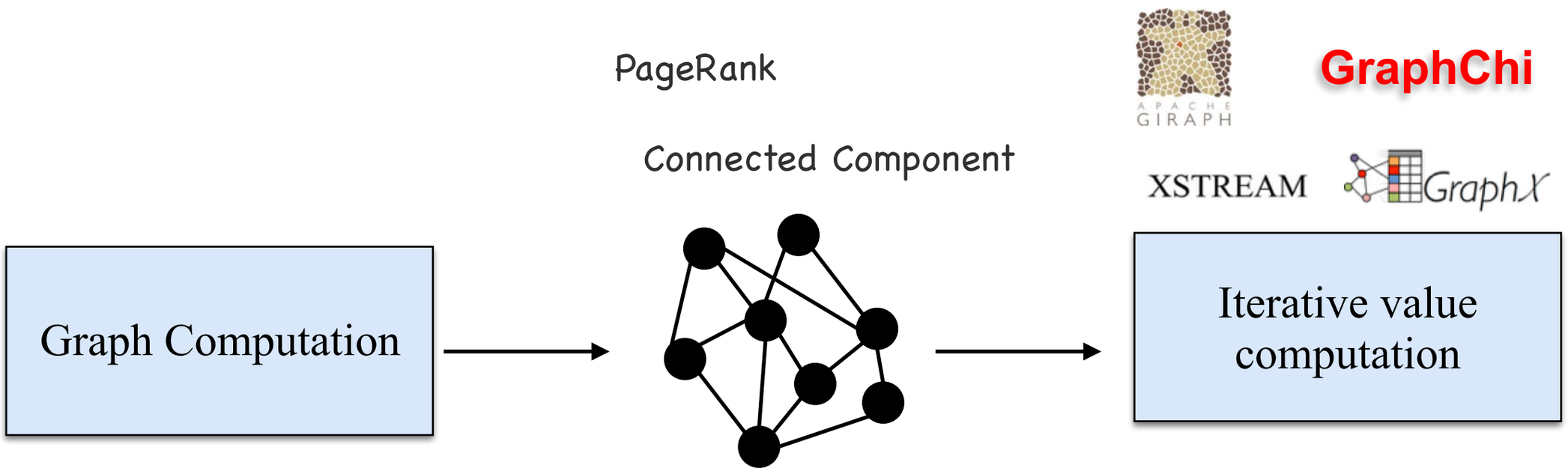
Graph Mining



Graph Analytical Problems



Graph Analytical Problems



Existing Mining Systems

- Enumerate all possible subgraphs
- For each subgraph, check if it matches the pattern
- Pattern is application-specific (Clique finding, motif counting, frequent subgraph mining)

Existing Datalog Systems

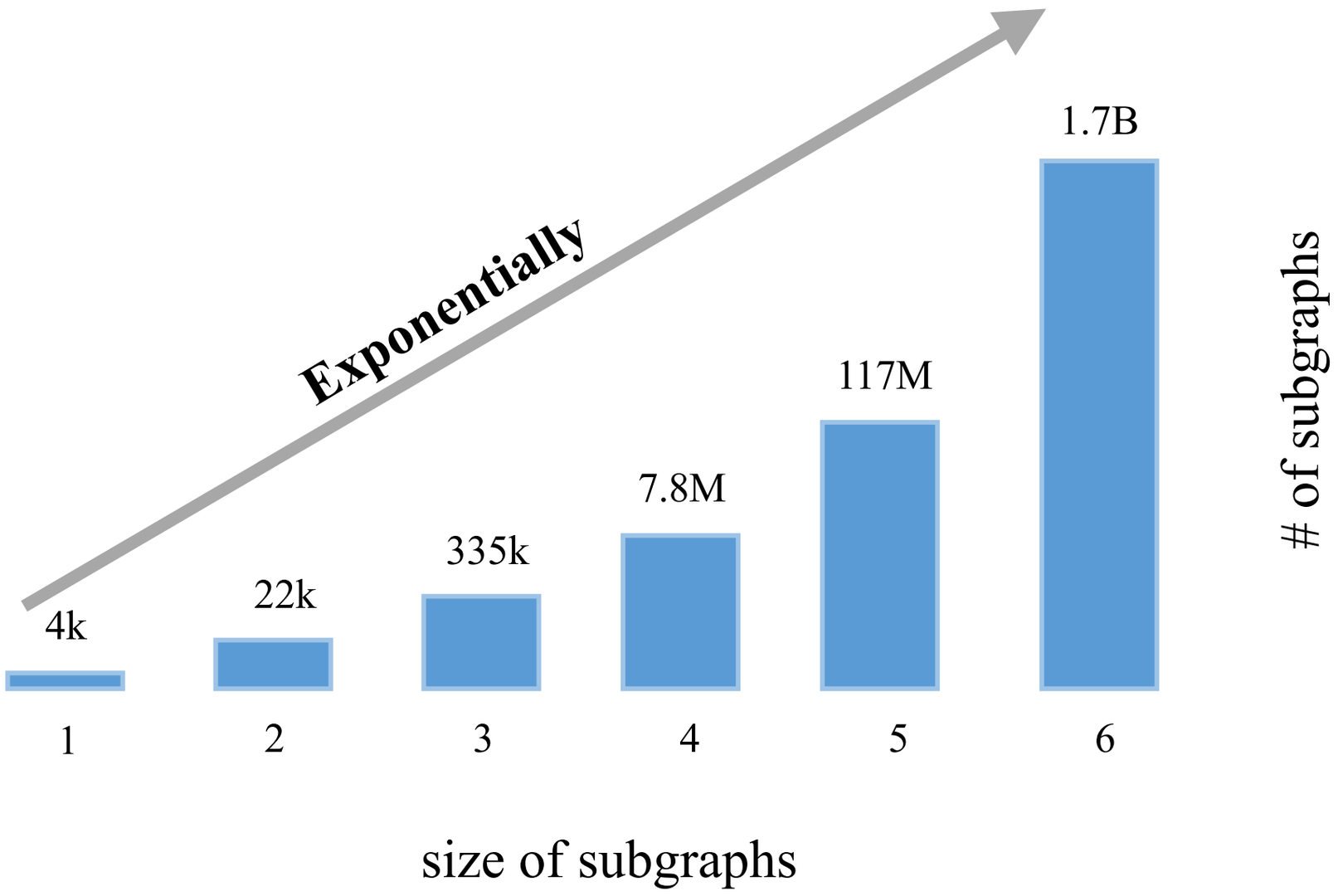
- Relational predicates
  - TC(a, b, c) ← R(a, b), a < b, R(b, c), b < c, R(c, a)
  - count TC(a, b, c)

- Relation algebra enables composition of small structures into big structures

Challenges in Graph Mining

- # of subgraphs grows *exponentially* with the size of subgraphs

Arabesque [CHC Teixeira et al. , SOSP'15]



Problems with Distributed Mining Systems

- Suffer from large startup and communication overhead
  - Arabesque on 10-node cluster, 35s startup, 3s execution
- Need enterprise clusters with large amounts of memory
  - DistGraph on 128-node cluster, 32,768GB memory
- Poor load balancing due to dynamic working sets
  - some nodes out of memory, other nodes with memory usage < 10%

Problems with Datalog Systems

- Programming model is not expressive enough for complex graph mining algorithms

Thoughts and Insight

- Distributed mining systems drawbacks: large startup, underutilized cpus, poor load balancing
- Not all users have access to enterprise cluster
- Many users are domain experts with limited background in hosting a cluster

Thoughts and Insight

- Distributed mining systems drawbacks: large startup, underutilized cpus, poor load balancing

Increasingly large SSDs

- Not all

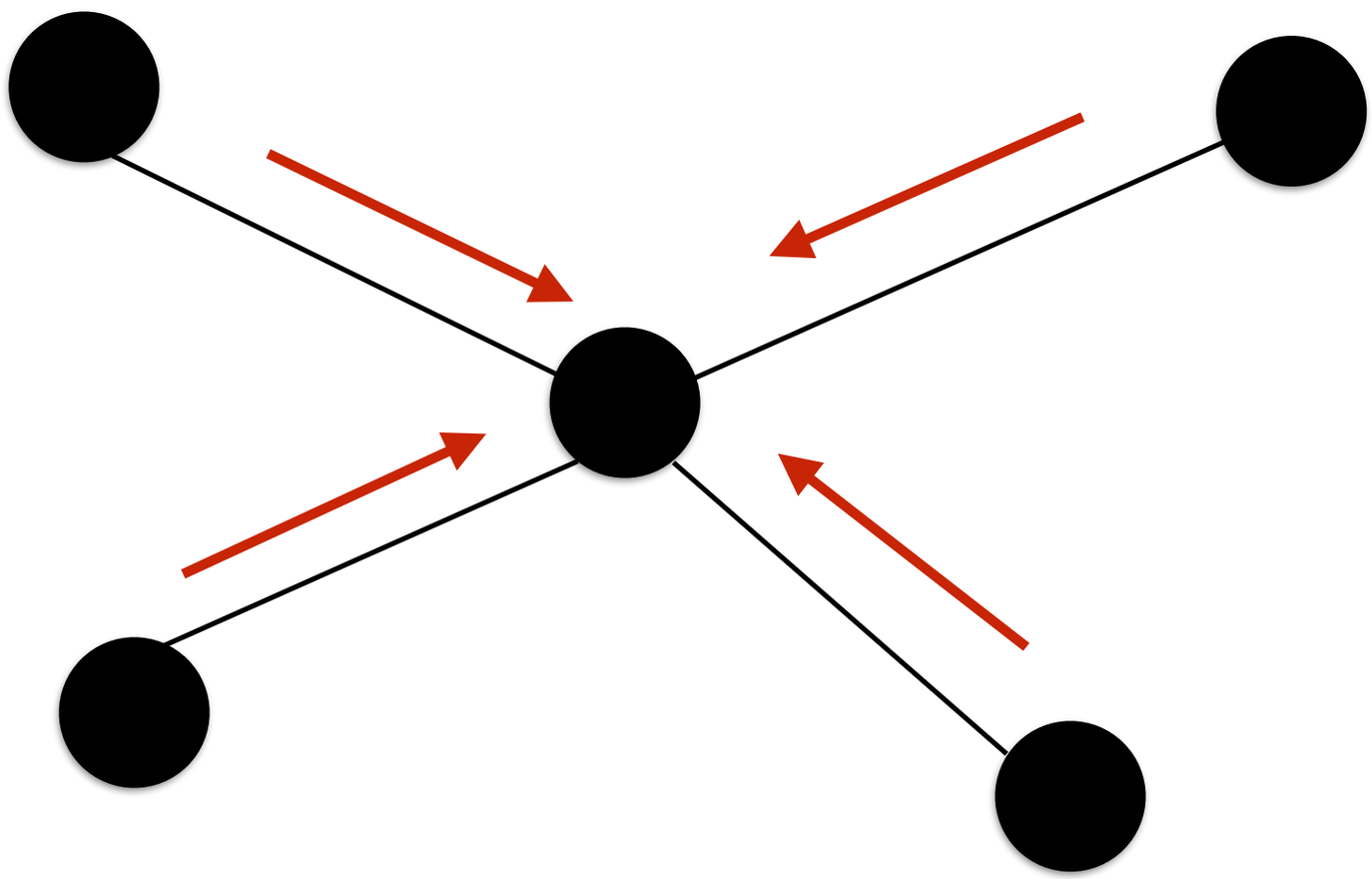
- Many users are domain experts with limited background in hosting a cluster

Our Proposal: RStream

A single machine, out-of-core graph mining system

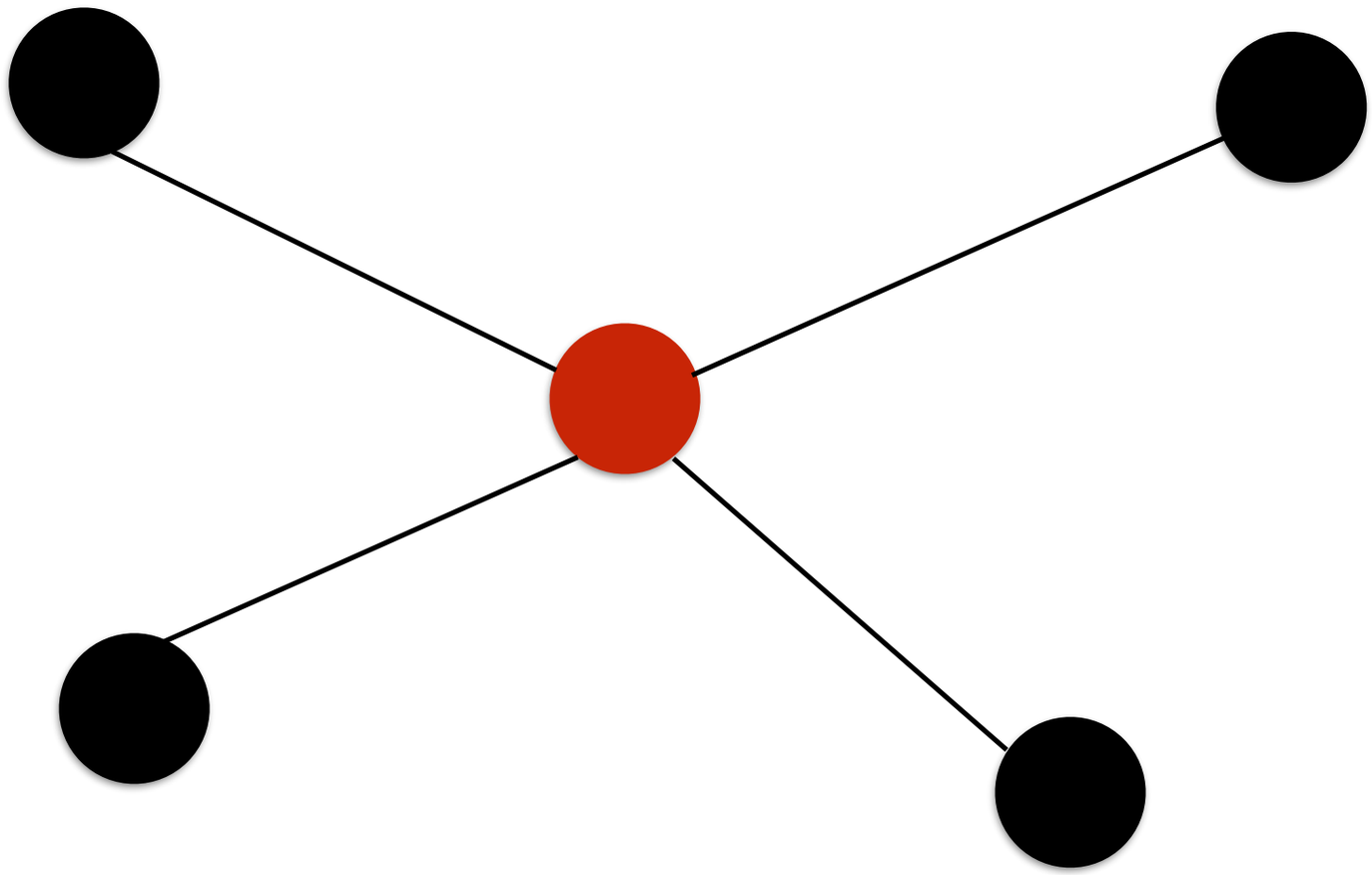
- A *simple* and *expressive* API
  - Gather-Apply-Scatter + Relational Algebra => GRAS
- An *efficient* runtime engine
  - implements relational algebra with streaming

GAS



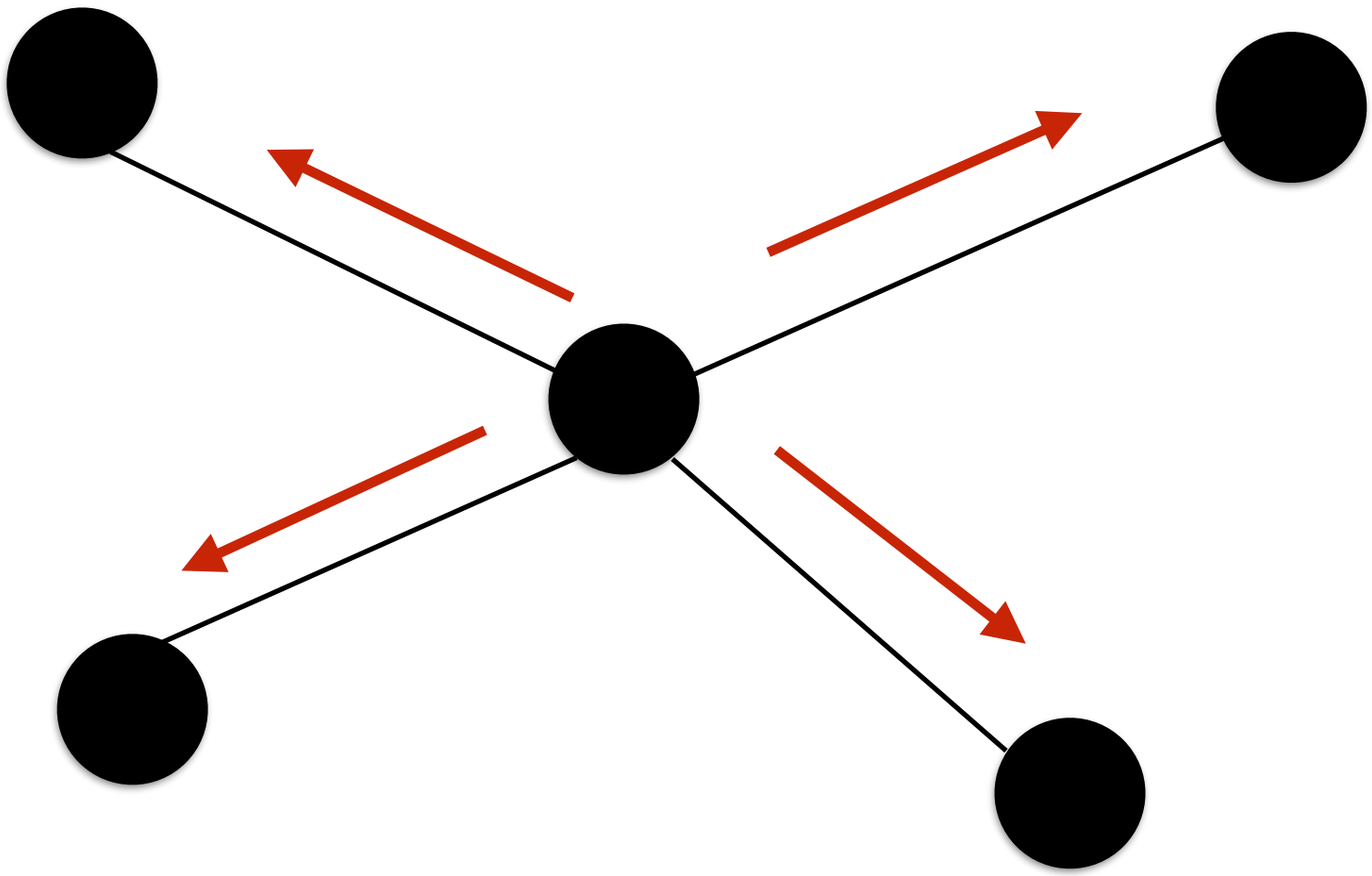
Gather information from neighbor vertices

GAS



Apply and update the vertex property

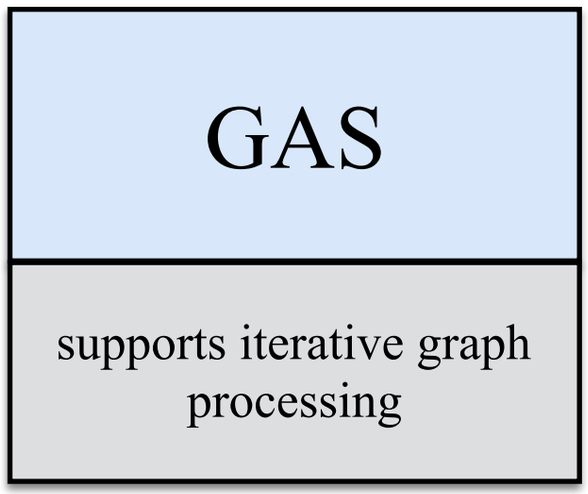
GAS



Scatter information to neighbor vertices

GRAS

GRAS



GRAS

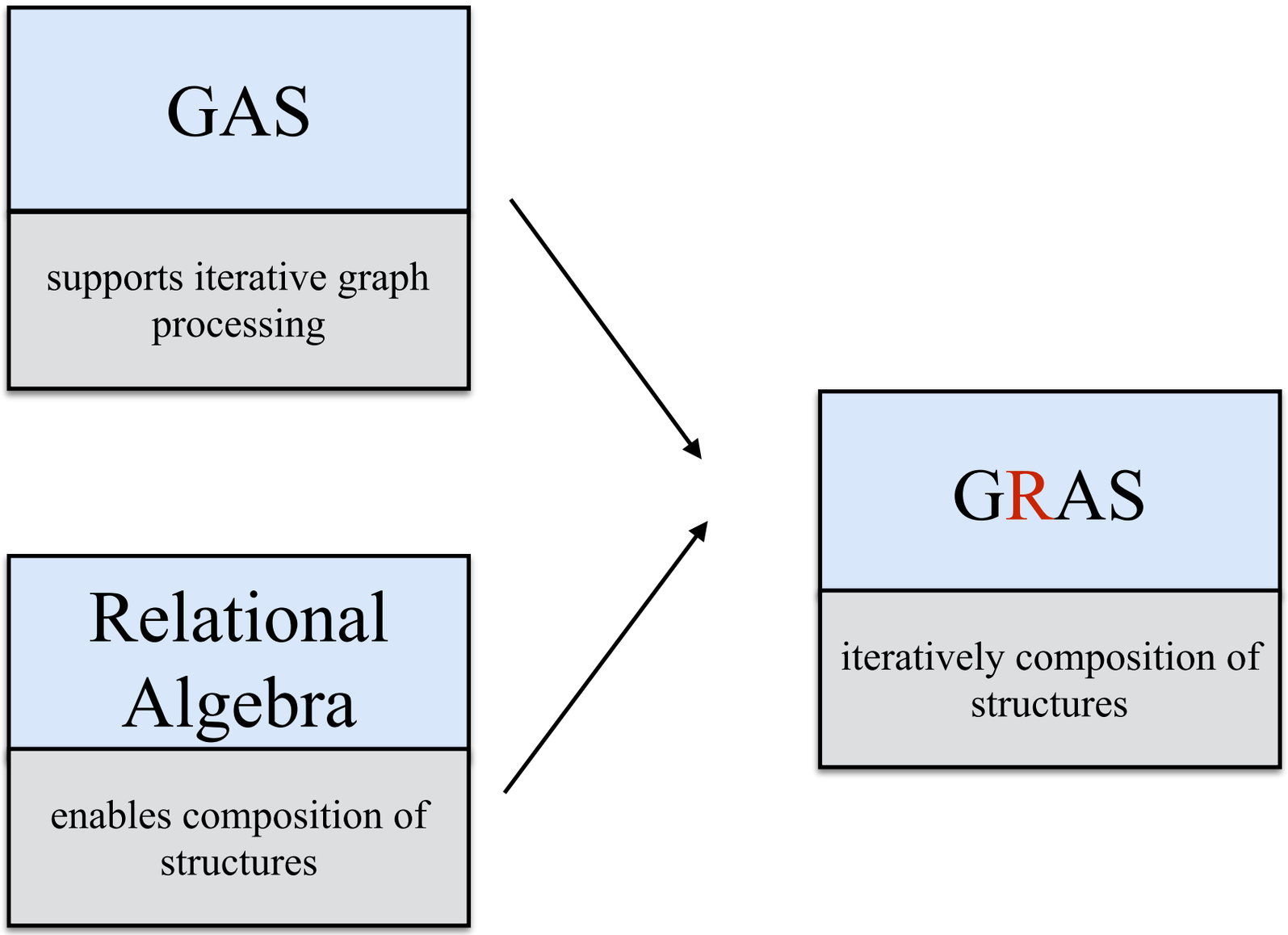
GAS

supports iterative graph processing

Relational Algebra

enables composition of structures

GRAS


GAS
supports iterative graph processing
Relational Algebra
enables composition of structures
GRAS
iteratively composition of structures

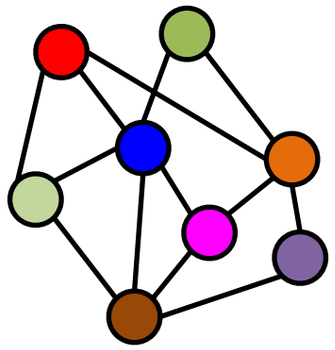
GRAS

GAS

supports iterative graph processing

Relational Algebra

enables composition of structures



GRAS

iteratively composition of structures

Edge Streaming

X-Stream [A Roy et al. , SOSP'13]

- Use streaming to reduce I/O costs

- Sequentially access (larger) datasets from disk, randomly access (smaller) datasets held in memory

Edge Streaming

A graph is partitioned into streaming partitions.
Each streaming partition contains

Vertex Table

VID	Value
1	1
2	2

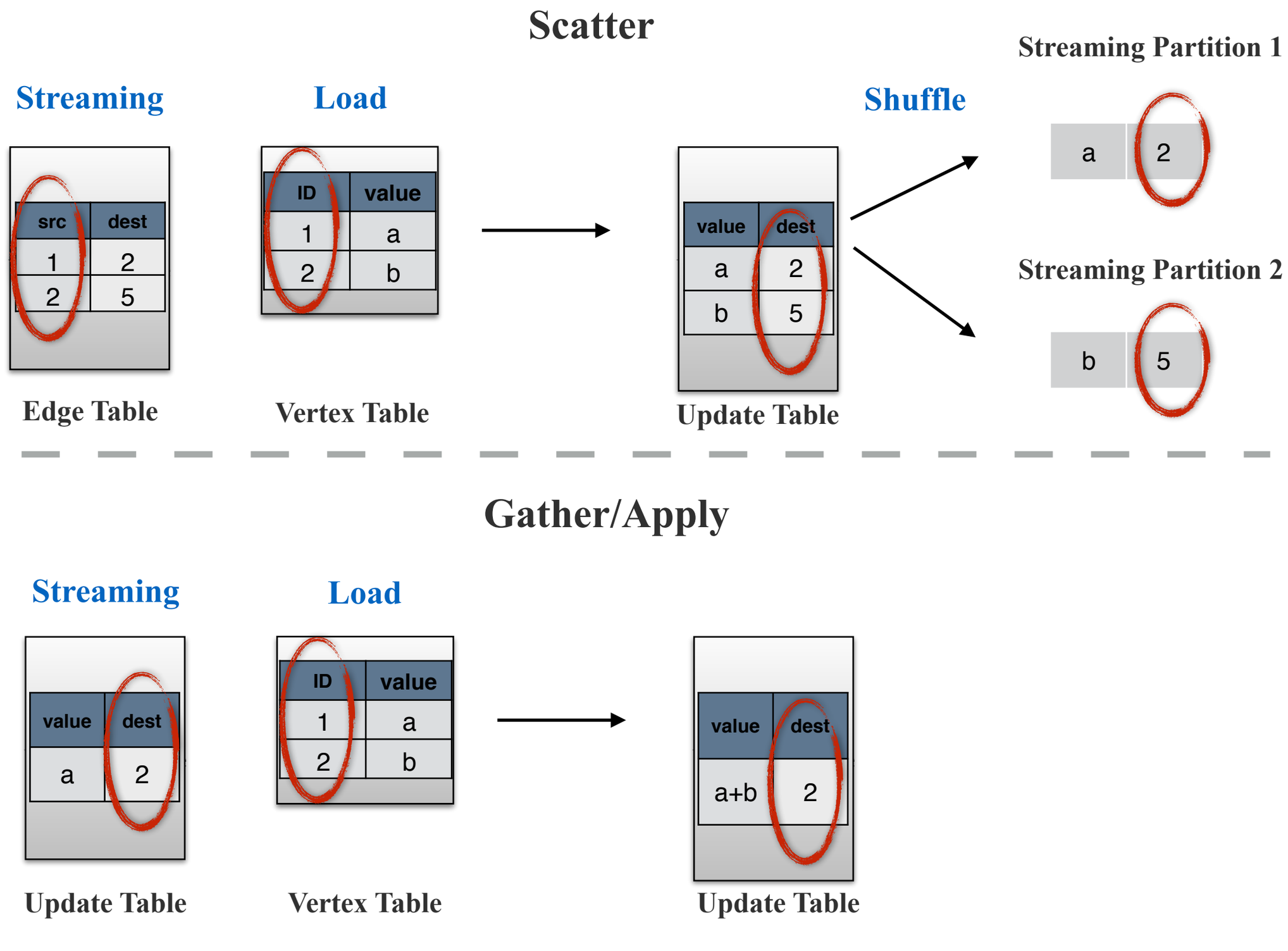
Edge Table

Src	Dest
1	4
2	5

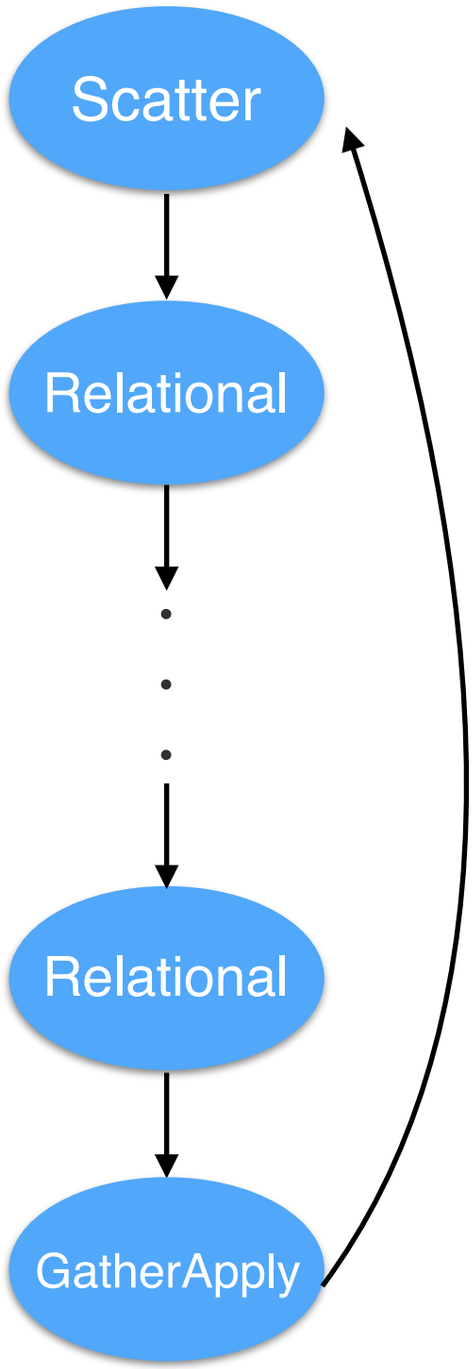
Update Table

Value	Dest
1	4
2	5

Streaming for Scatter/Gather



RStream API


Scatter
Relational
Relational
GatherApply


Scatter
GatherApply
Relational

Example:Triangle Counting

Example:Triangle Counting

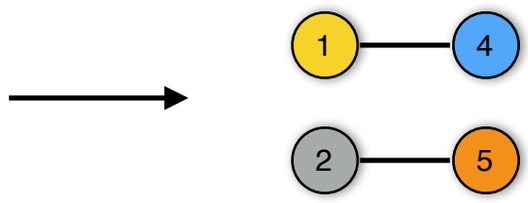
Scatter

src	dest
1	4
2	5
…	…

edge table

VID	value
1	4
2	5
…	…

vertex table



Example:Triangle Counting

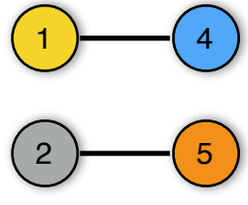
Scatter

src	dest
1	4
2	5
…	…

edge table

VID	value
1	4
2	5
…	…

vertex table



R1

(a, b) ⋈ (b, c)

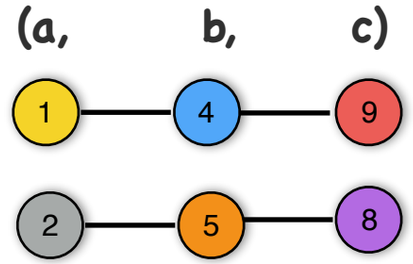
c1	c2
1	**4**
2	**5**
…	…

update table1

⋈

src	dest
4	9
5	8
…	…

edge table



Example:Triangle Counting

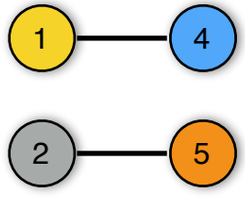
Scatter

src	dest
1	4
2	5
…	…

edge table

VID	value
1	4
2	5
…	…

vertex table



R1

(a, b) ⋈ (b, c)

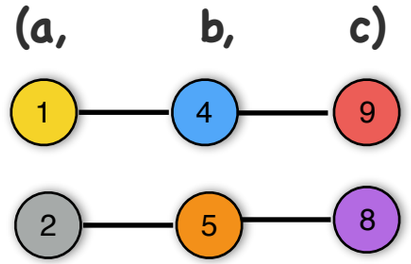
c1	c2
1	**4**
2	**5**
…	…

update table1

⋈

src	dest
4	9
5	8
…	…

edge table



R2

(a, b, c) ⋈ (c, a)

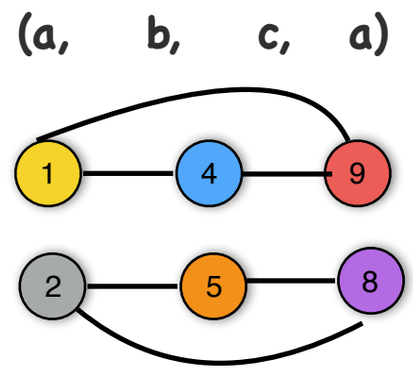
c1	c2	c3
1	4	**9**
2	5	**8**
…	…	…

update table2

⋈

src	dest
9	1
8	2
…	…

edge table

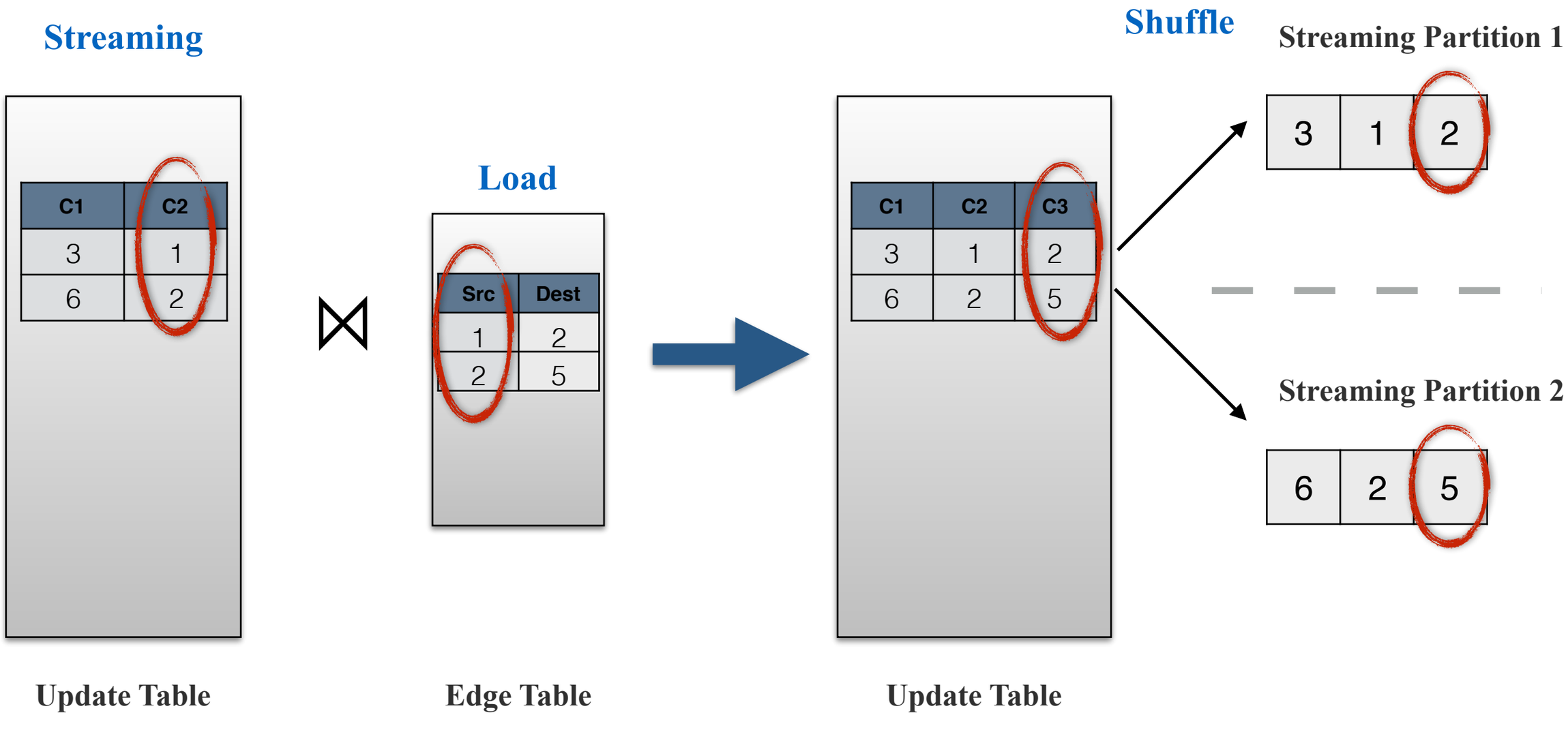


Outline

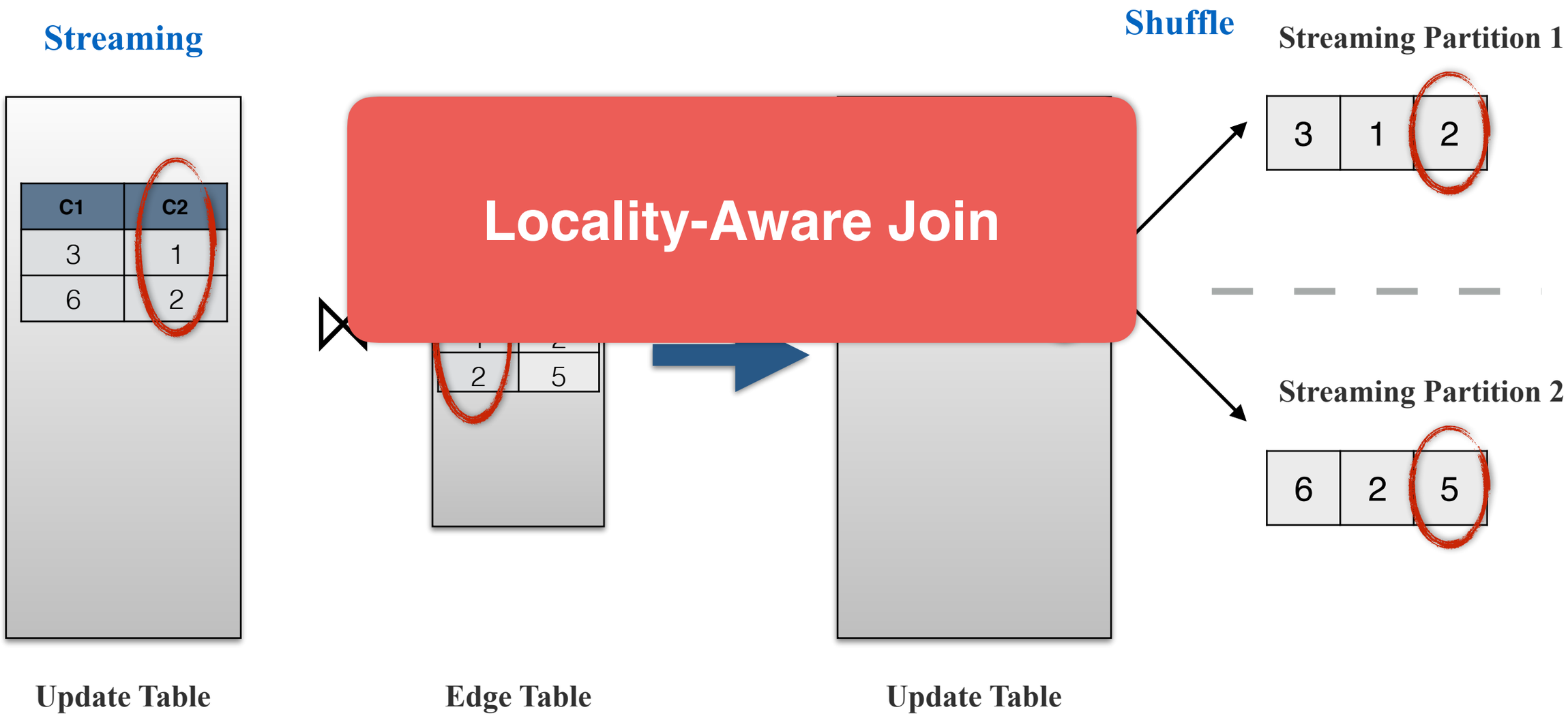
- How to provide a *general* programming interface for graph mining algorithms?

- How to implement relational operators *efficiently* for graphs?

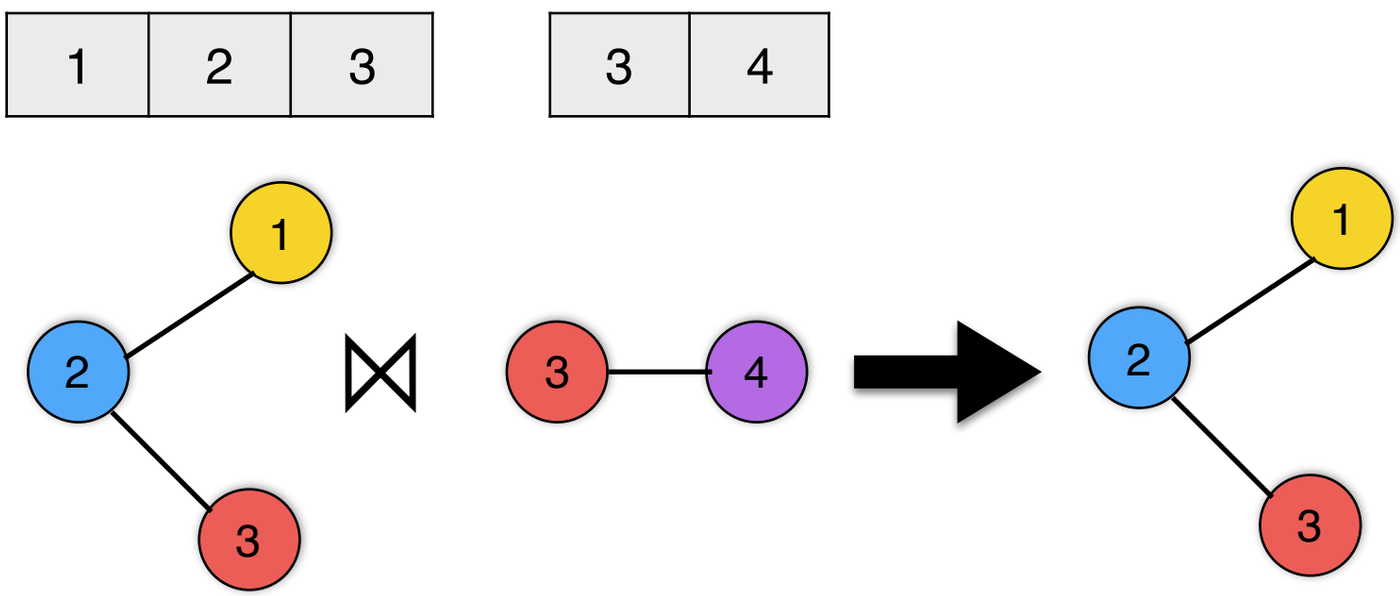
Streaming for Join Operator

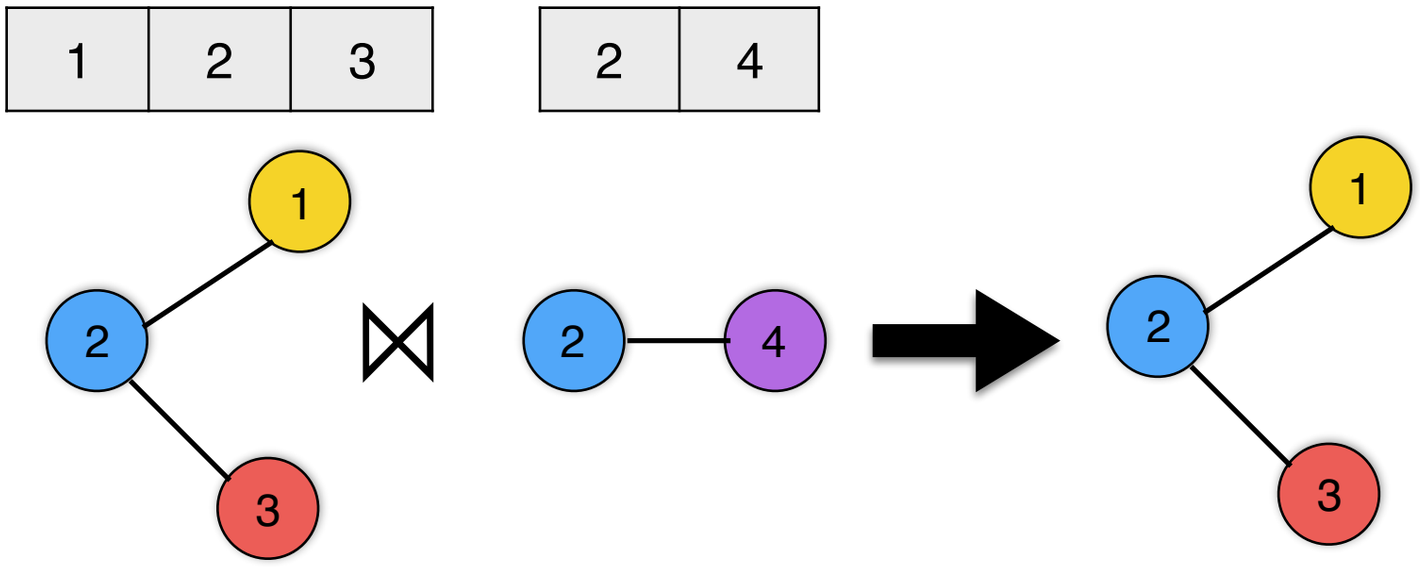


Streaming for Join Operator

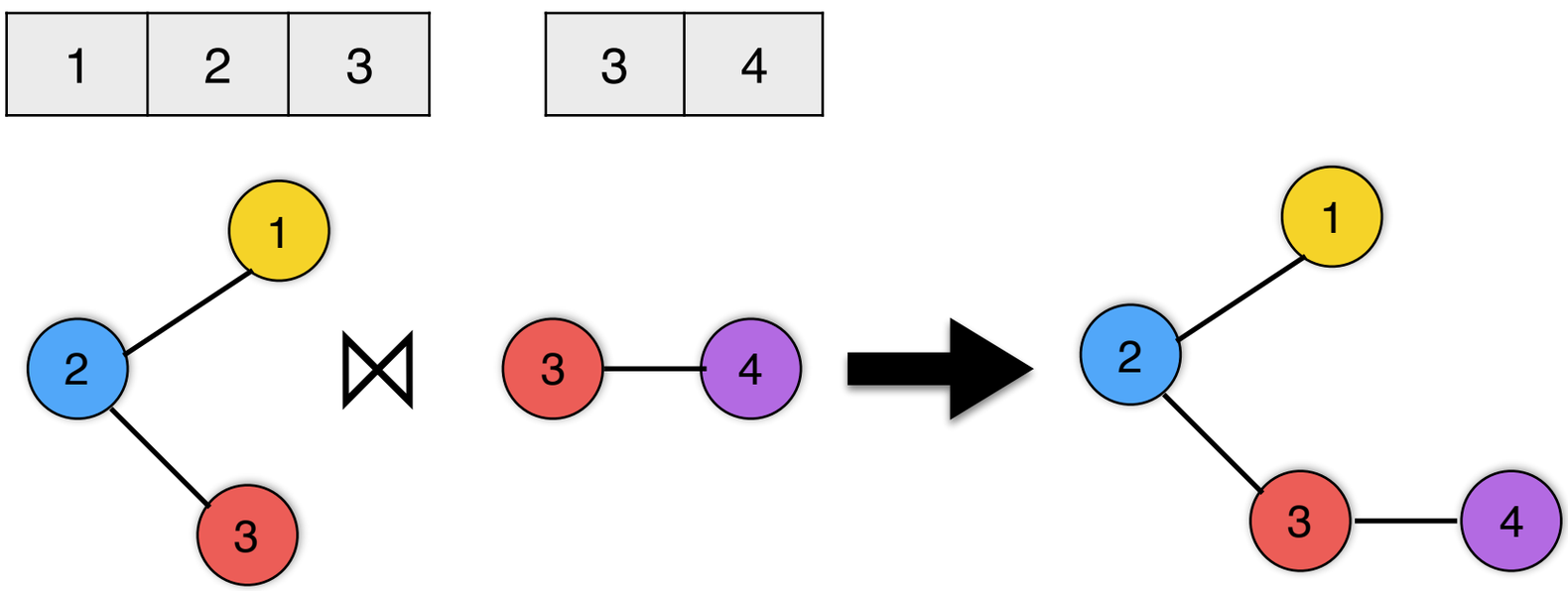


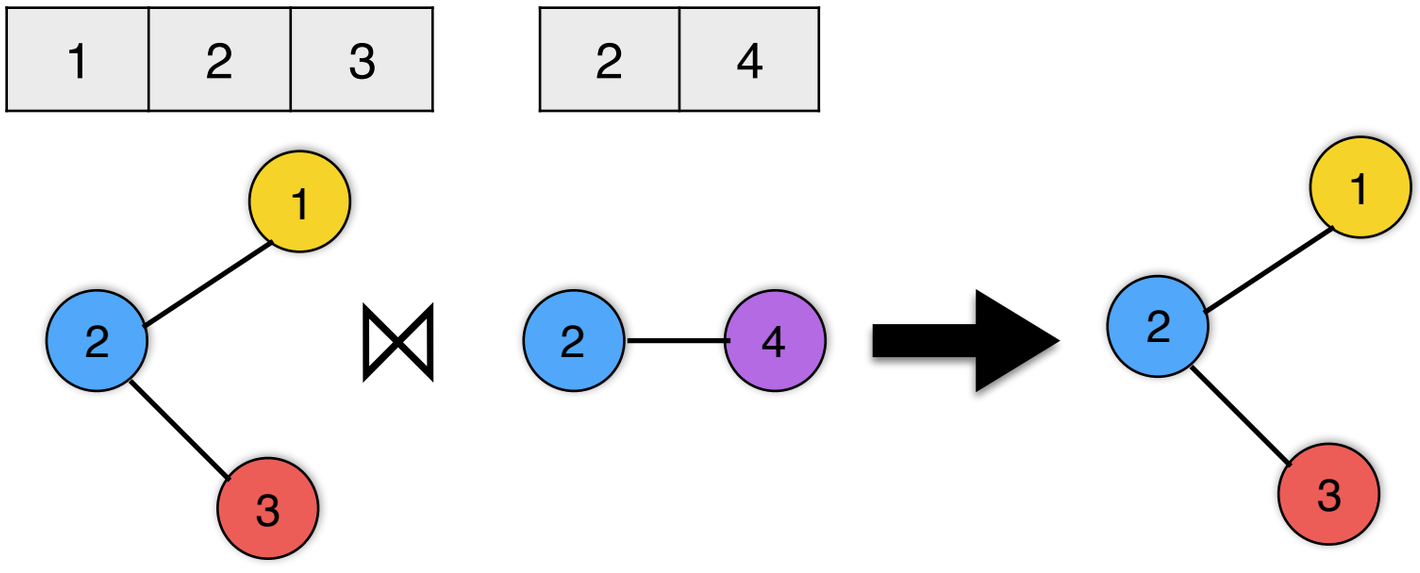
Structural Information



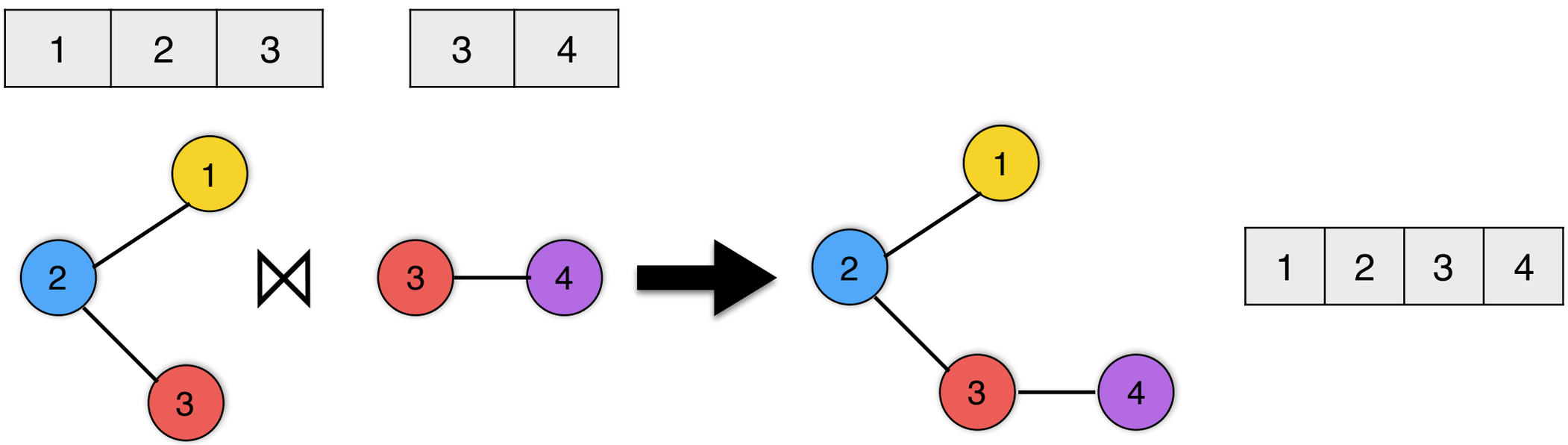


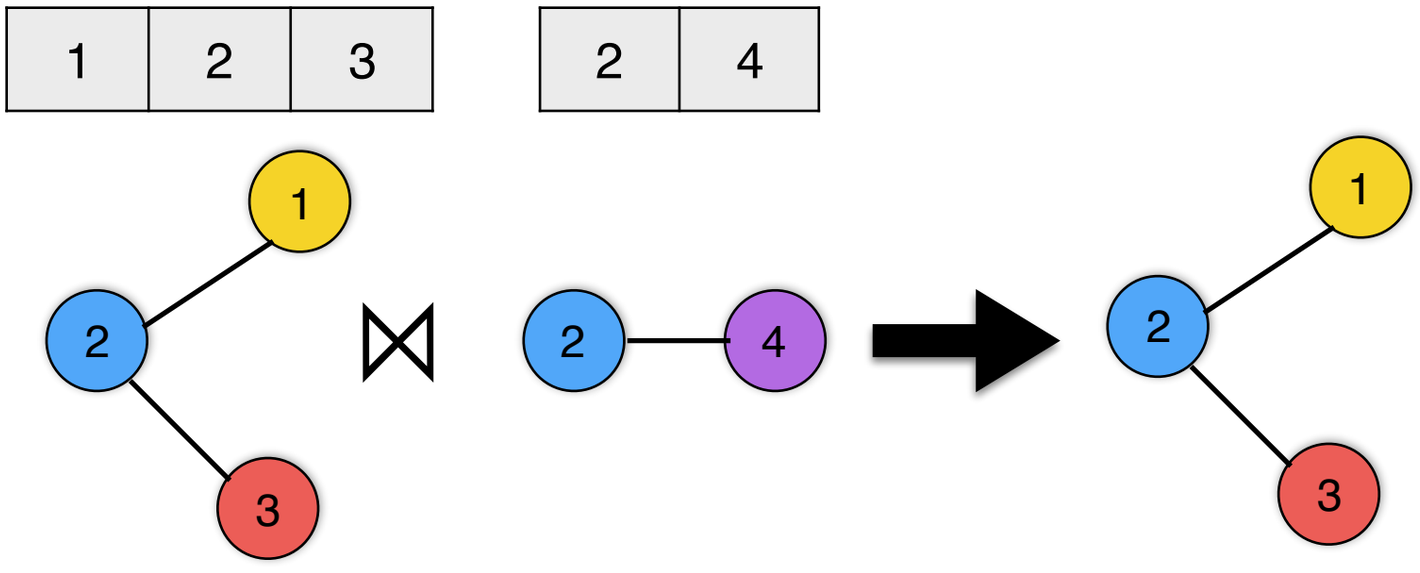
Structural Information



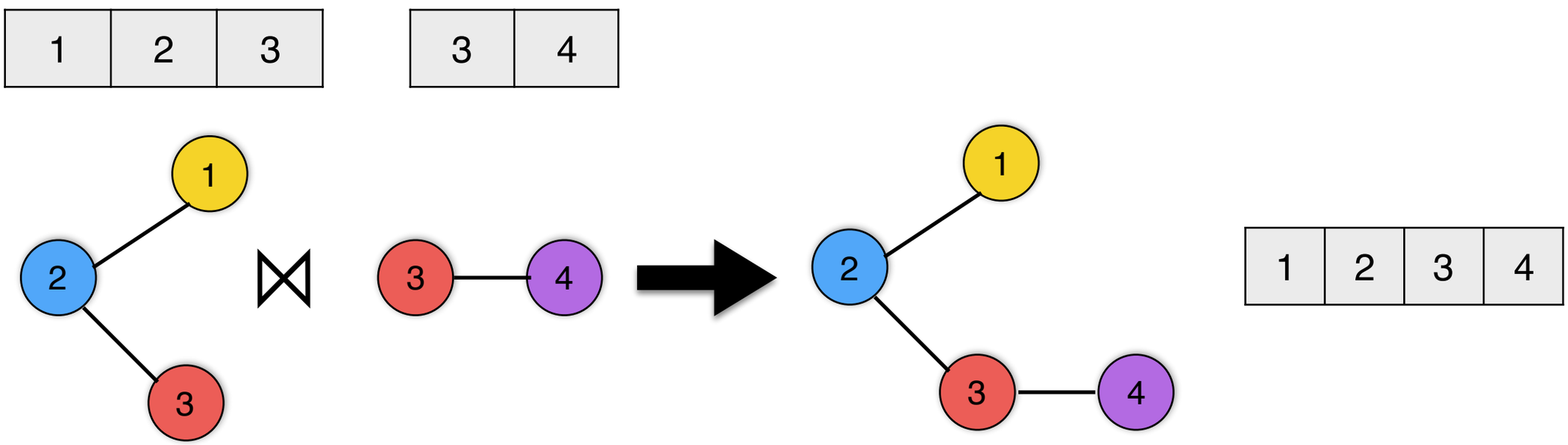


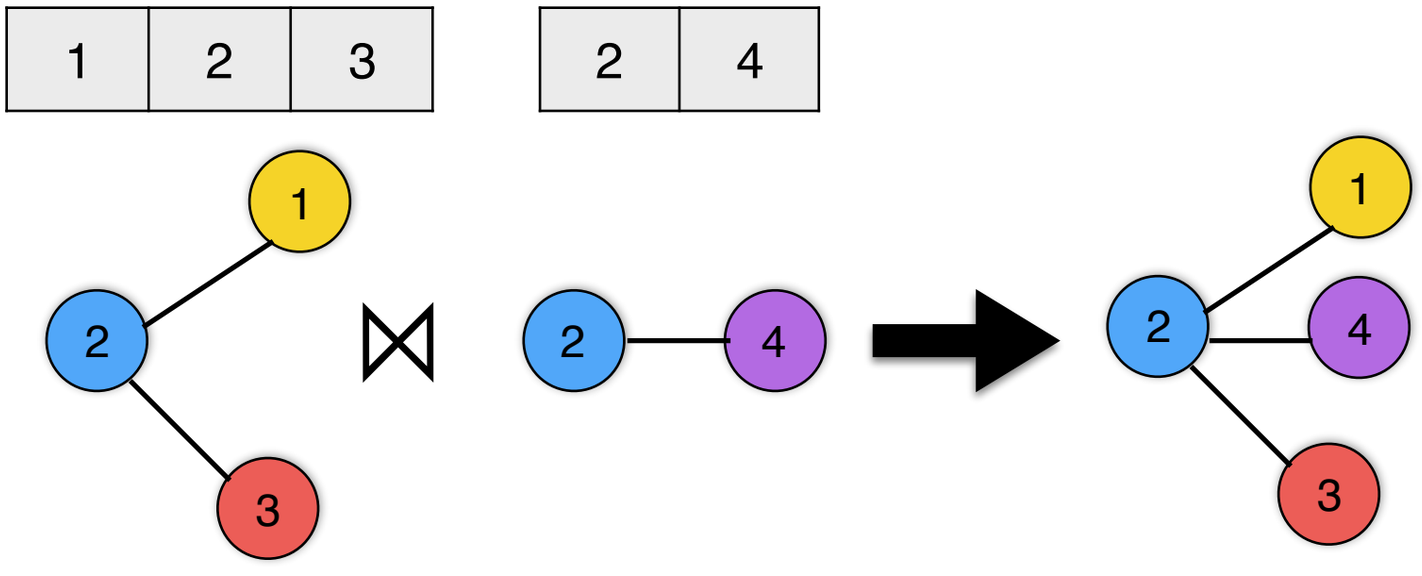
Structural Information



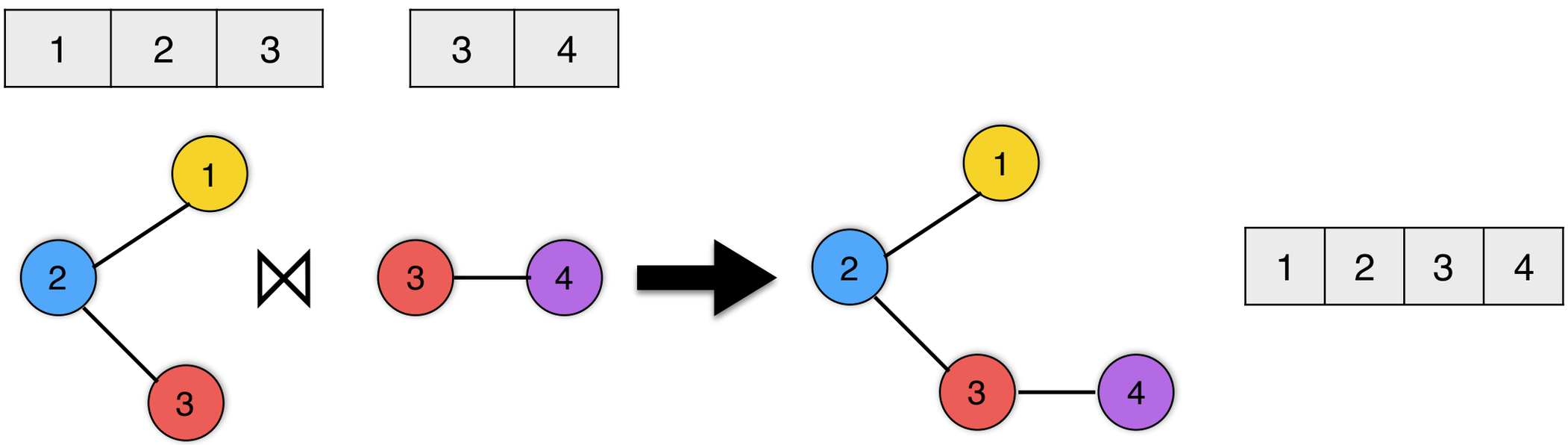


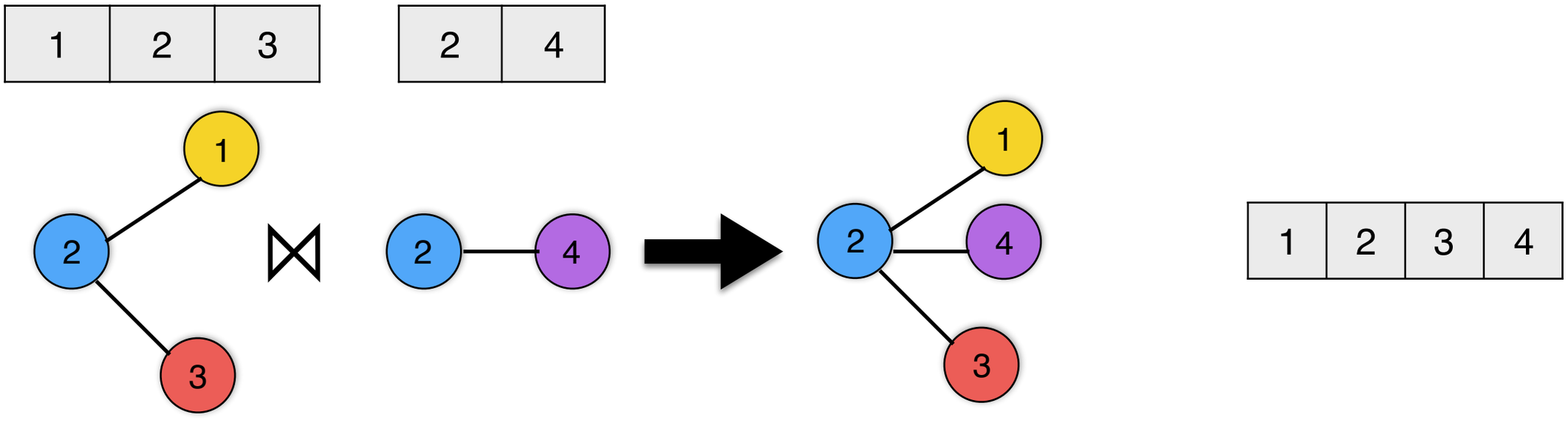
Structural Information



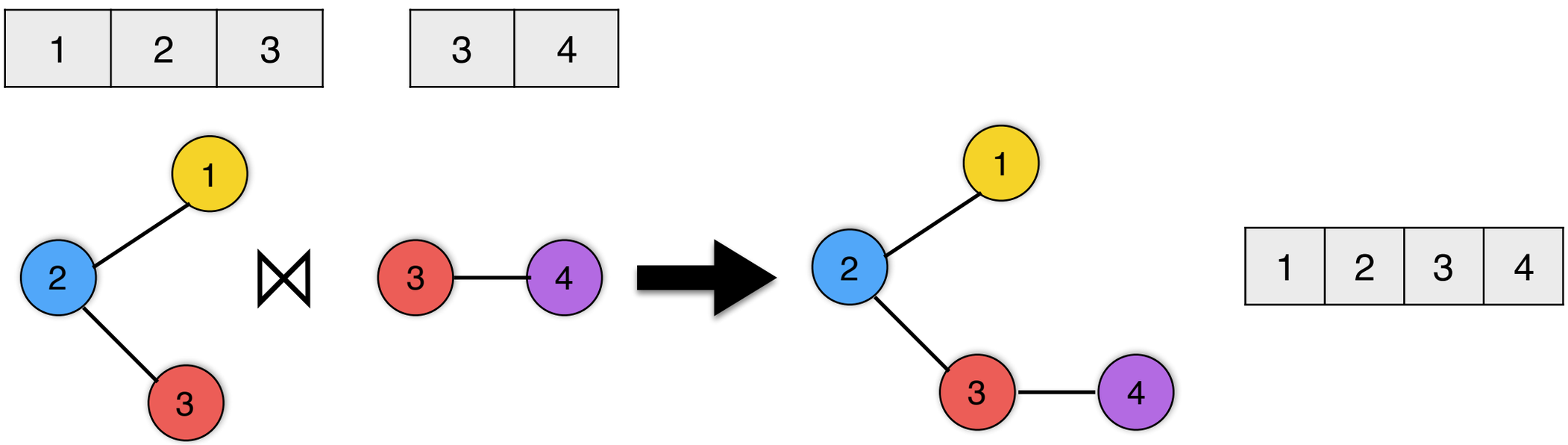


Structural Information



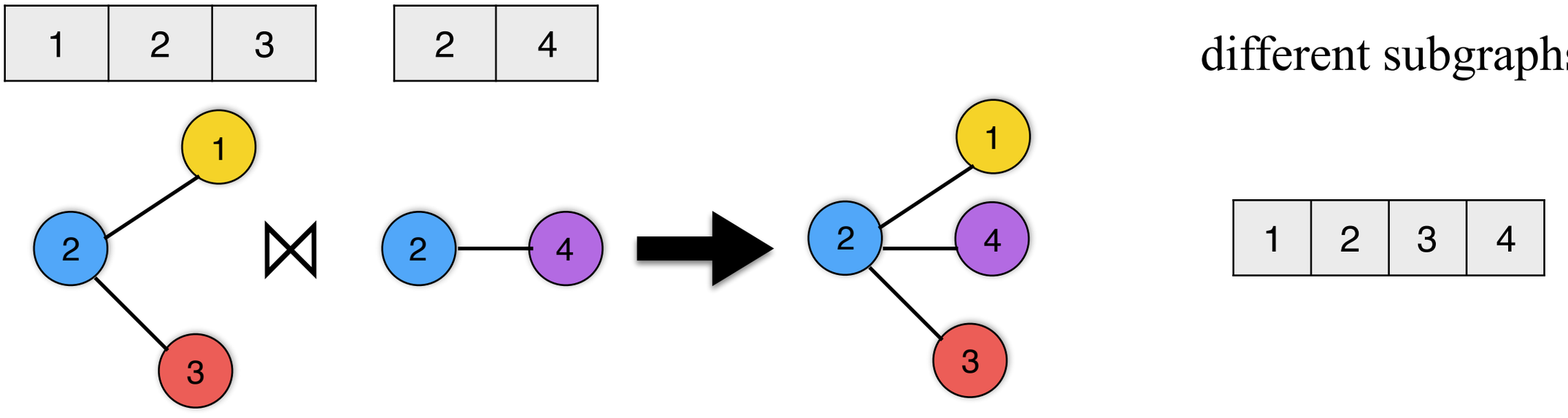


Structural Information

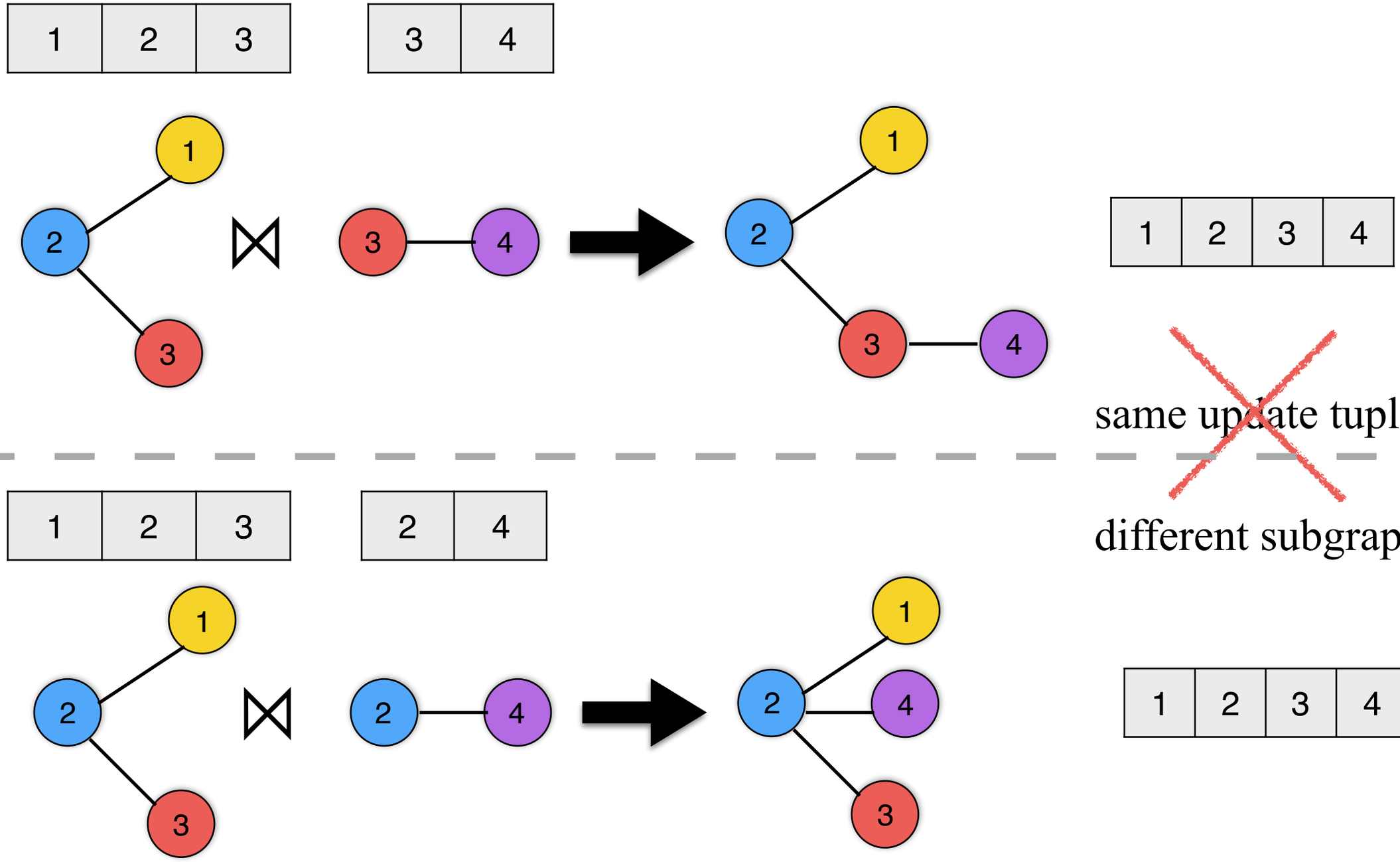


same update tuples

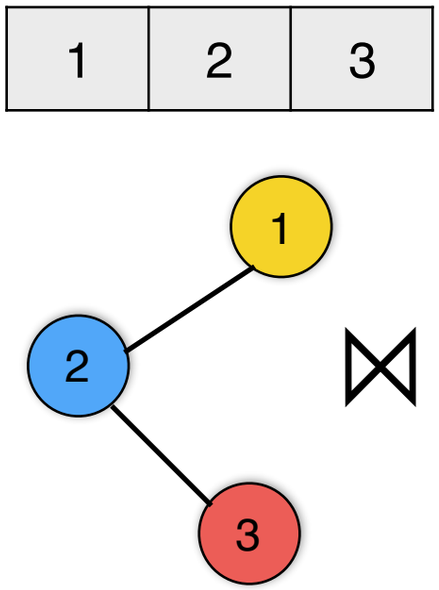
different subgraphs



Structural Information



Structural Information

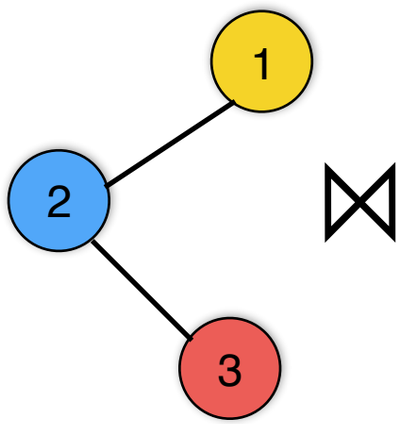


⋈

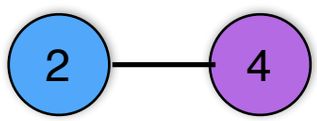
Structural info is missing!

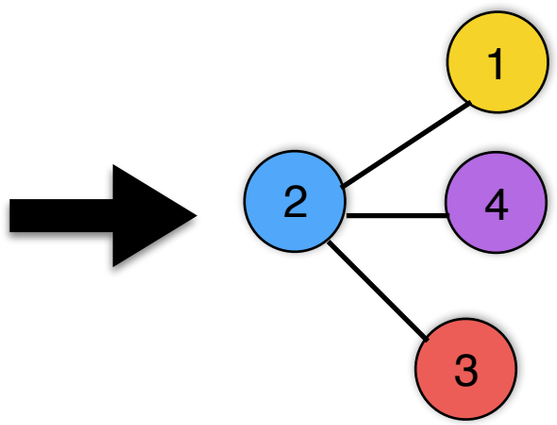
same update tuples

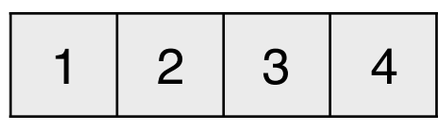
different subgraphs



⋈







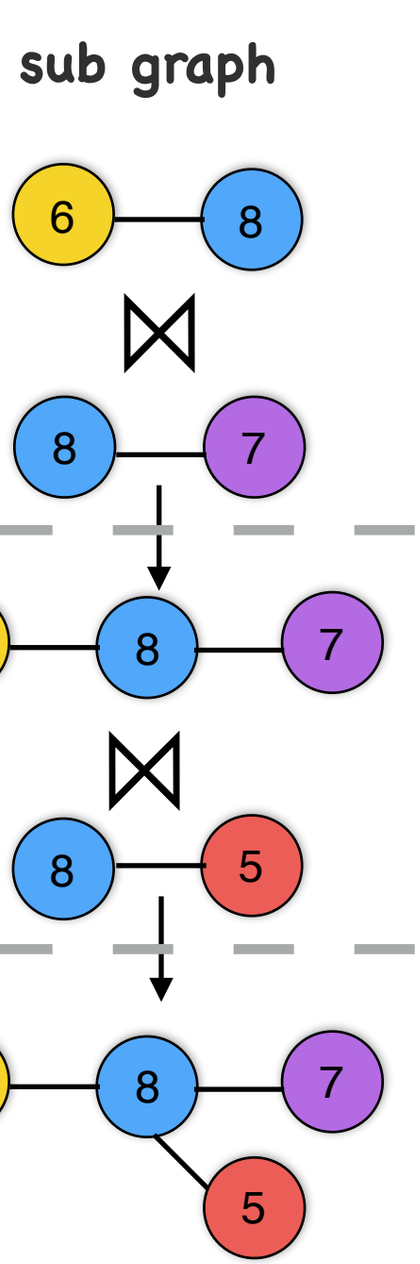
Missing Structural Information

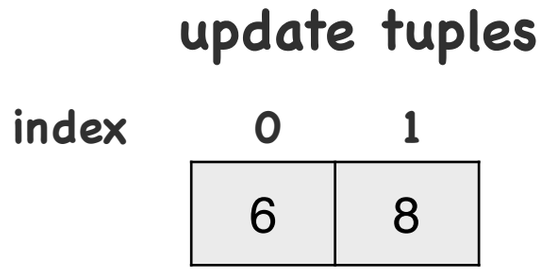
- Identical tuples may represent different structures

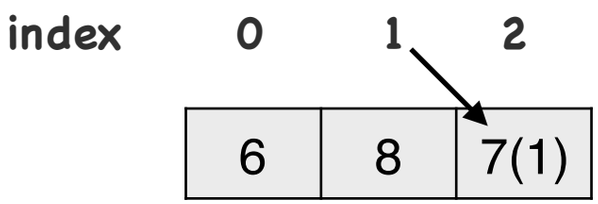
- Different tuples may represent identical structures

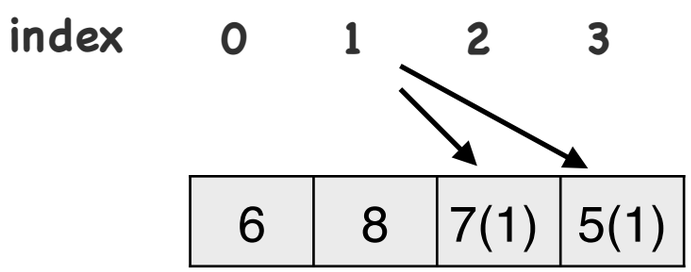
Adding Structural Info

- Encodes the *history* of joins in update tuples







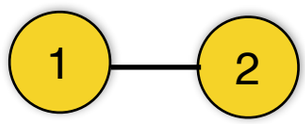


Is Join Enough?

- Join *grows* a subgraph from one of its vertices
- For Frequent Subgraph Mining, we need to explore all possibilities of existing subgraphs
- A different way of joining to *grow* a subgraph from *all* of its vertices

Join on All Columns

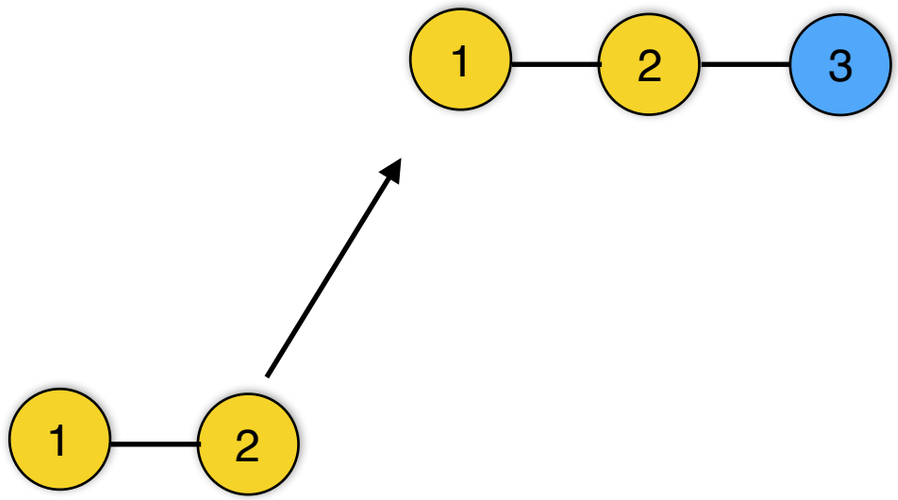
- Joins update table with edge table on *every* column



10

Join on All Columns

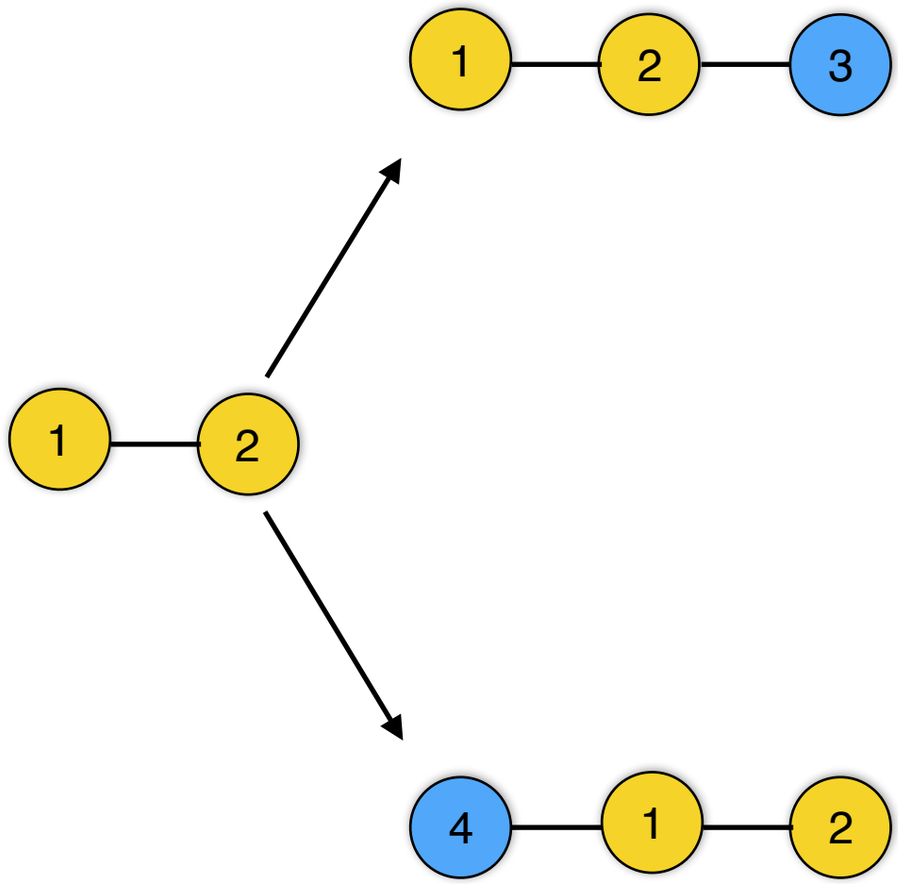
- Joins update table with edge table on *every* column



10

Join on All Columns

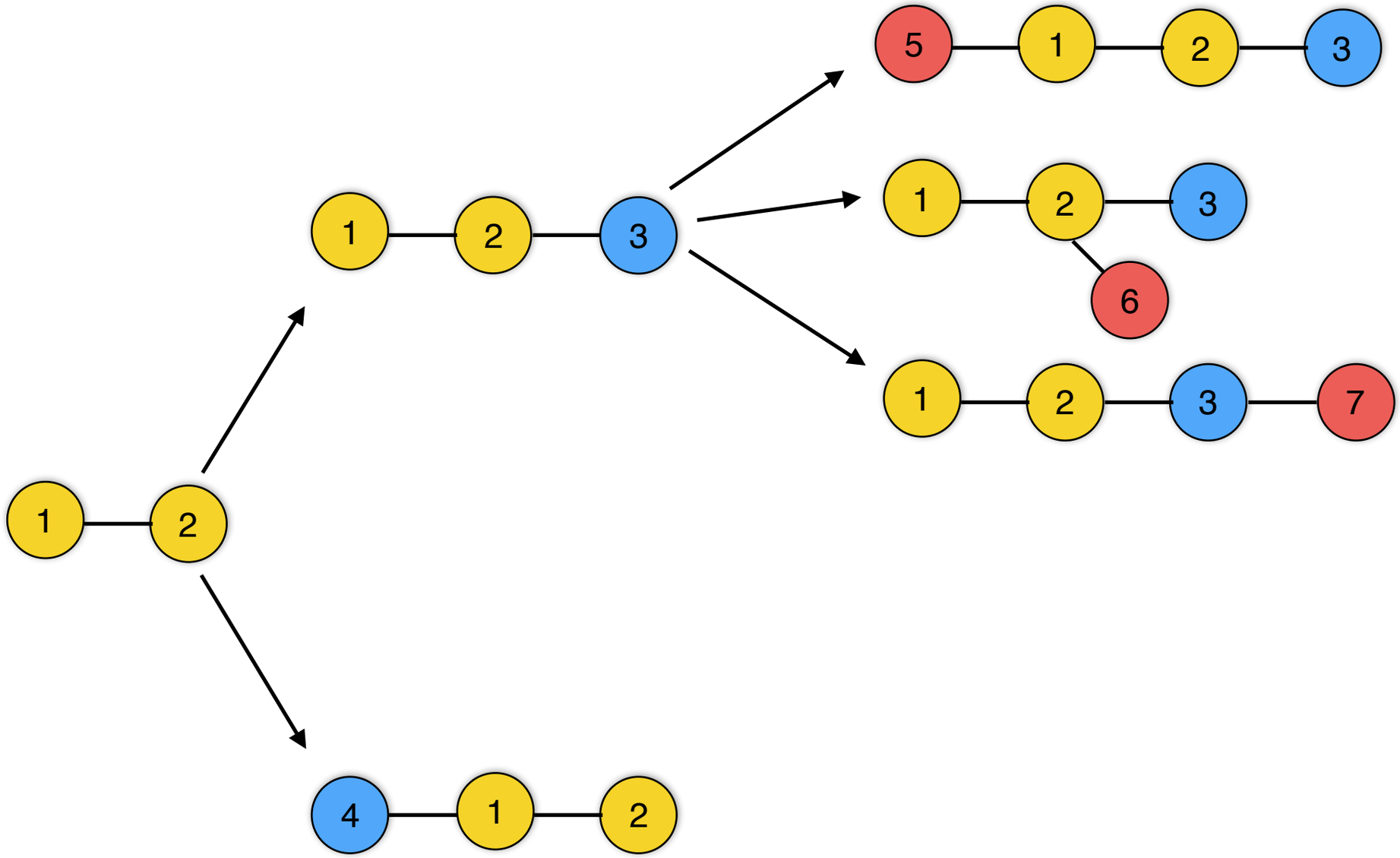
- Joins update table with edge table on *every* column



10

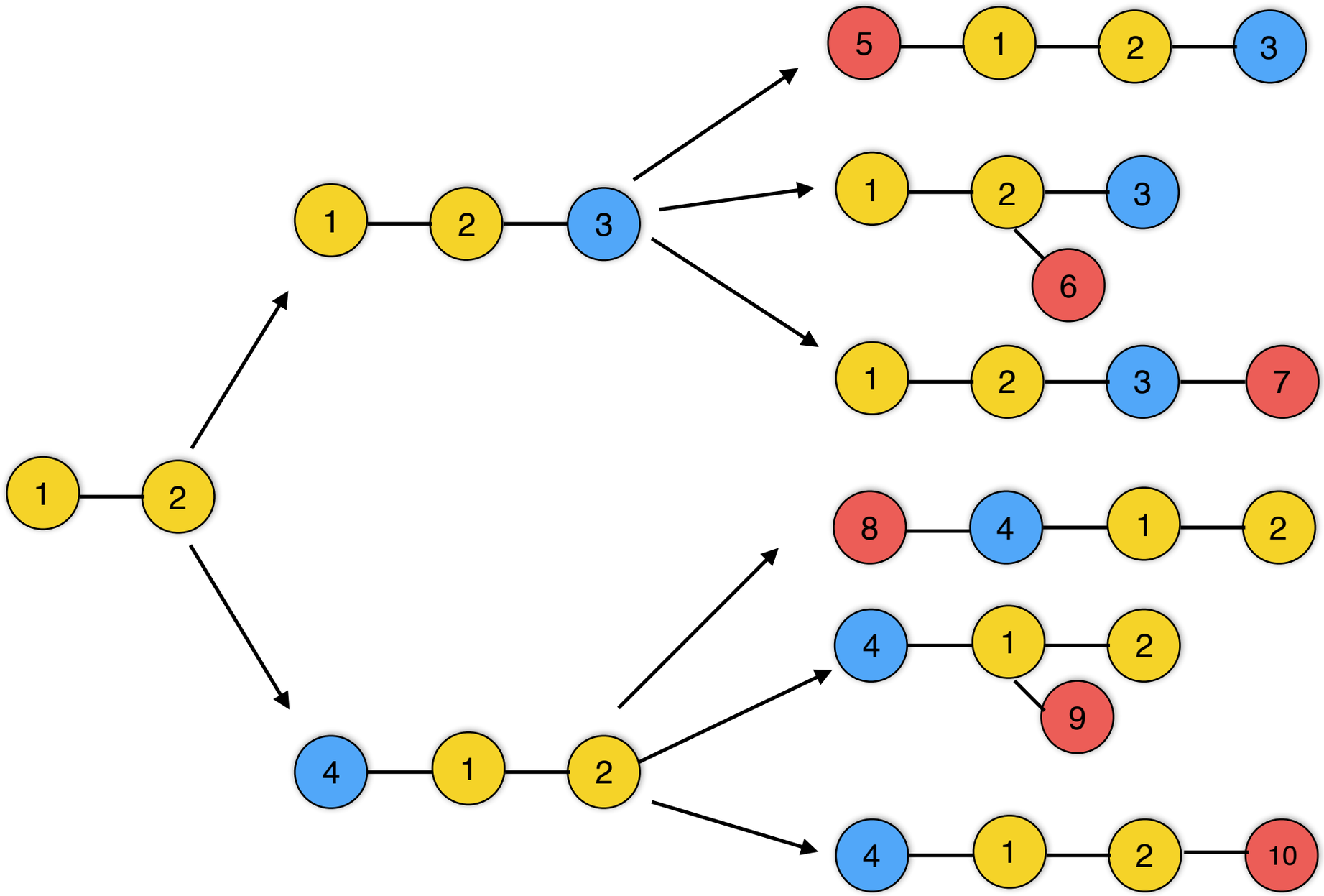
Join on All Columns

- Joins update table with edge table on *every* column



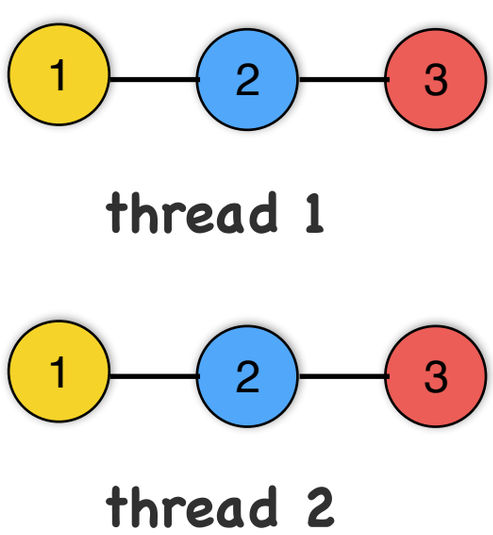
Join on All Columns

- Joins update table with edge table on *every* column

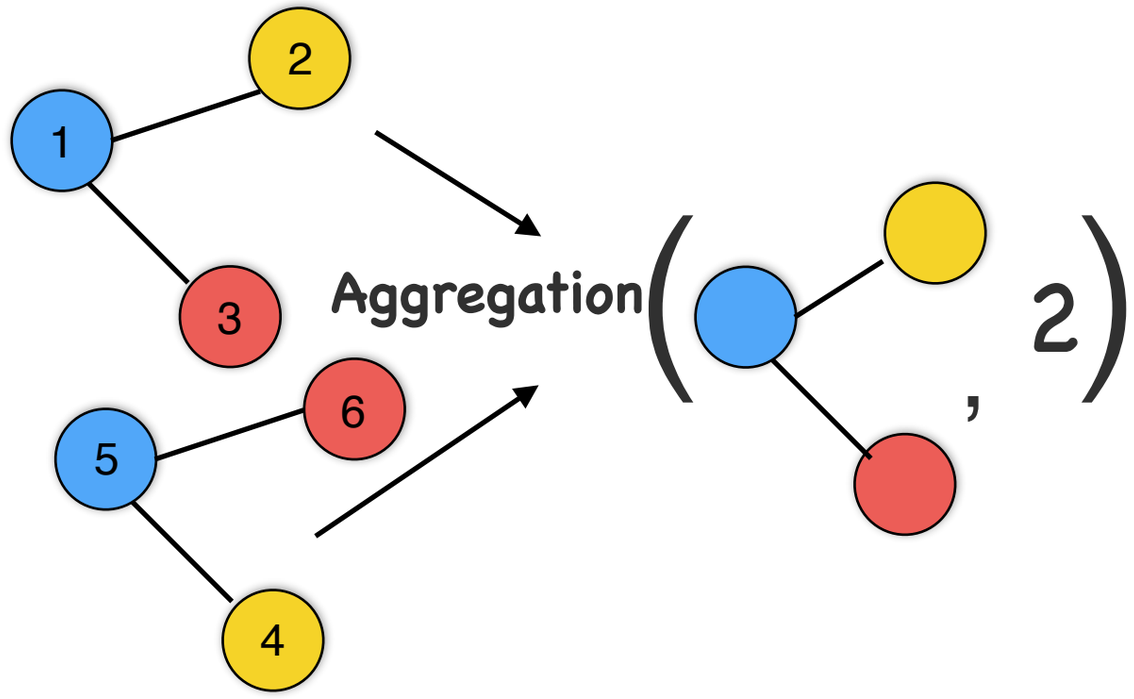


Automorphism and Isomorphism

Arabesque [CHC Teixeira et al. , SOSP'15]



- Different threads can generate identical(automorphic) update tuples
- Select and keep one, remove all the other duplicates



- Aggregate to count number of each distinct shape
- Different tuples may belong to same isomorphism class

Evaluation

- Platform
  - 10-node cluster, 5TB SSD
  - Each node: 2 Xeon(R) CPU E5-2640 v3 processors,32GB memory
- Application
  - Triangle Counting
  - Transitive Closure
  - N-Clique Finding
  - N-Motif Counting
  - Frequent Subgraph Mining
- Input graphs

Graphs	#Edges	#Vertices
Citeseer	4,732	3,312
Mico	1.1M	100K
Patents	14M	2.7M
LiverJournal	69M	4.8M
Orkut	117M	3M
UK-2005	936M	39.5M

Comparisons with Mining Systems

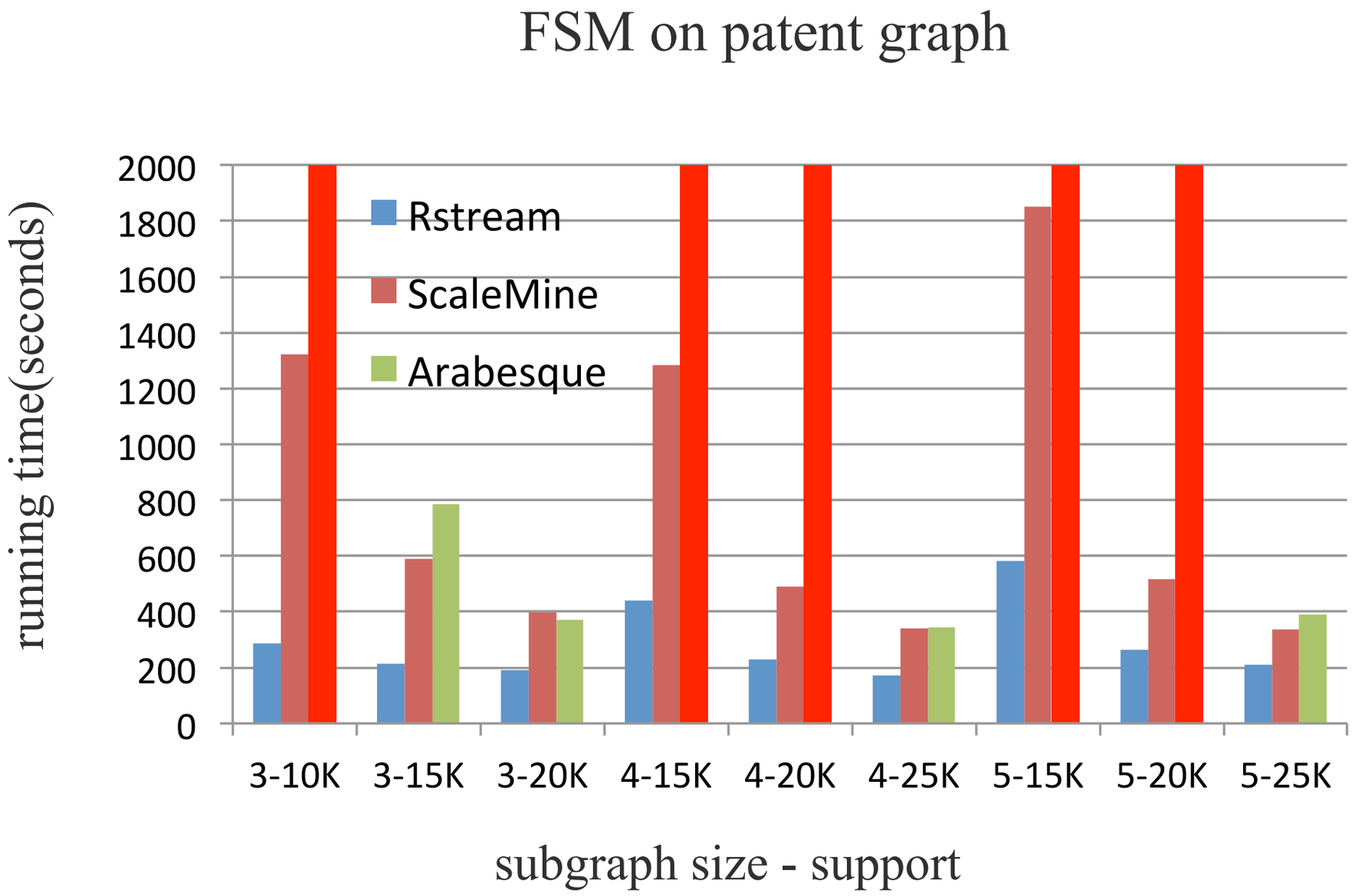
		Citese	Mico	Patent
Triangle Counting	**RStream**	**0.04**	**15.8**	**6.7**
	Arabesque-10	38.1	43.1	114.9
5-Clique	**RStream**	**0.01**	**115.1**	**35.3**
	Arabesque-10	42.8	132	174.5
3-FSM 1K	**RStream**	**0.06**	**351.7**	**383.7**
	Arabesque-10	35.6	5790.1	-
	ScaleMine-10	1.2	802.6	-
	DistGraph-10	0.4	-	-

RStream outperforms Arabesque by 60.9x

ScaleMine by 12.1x

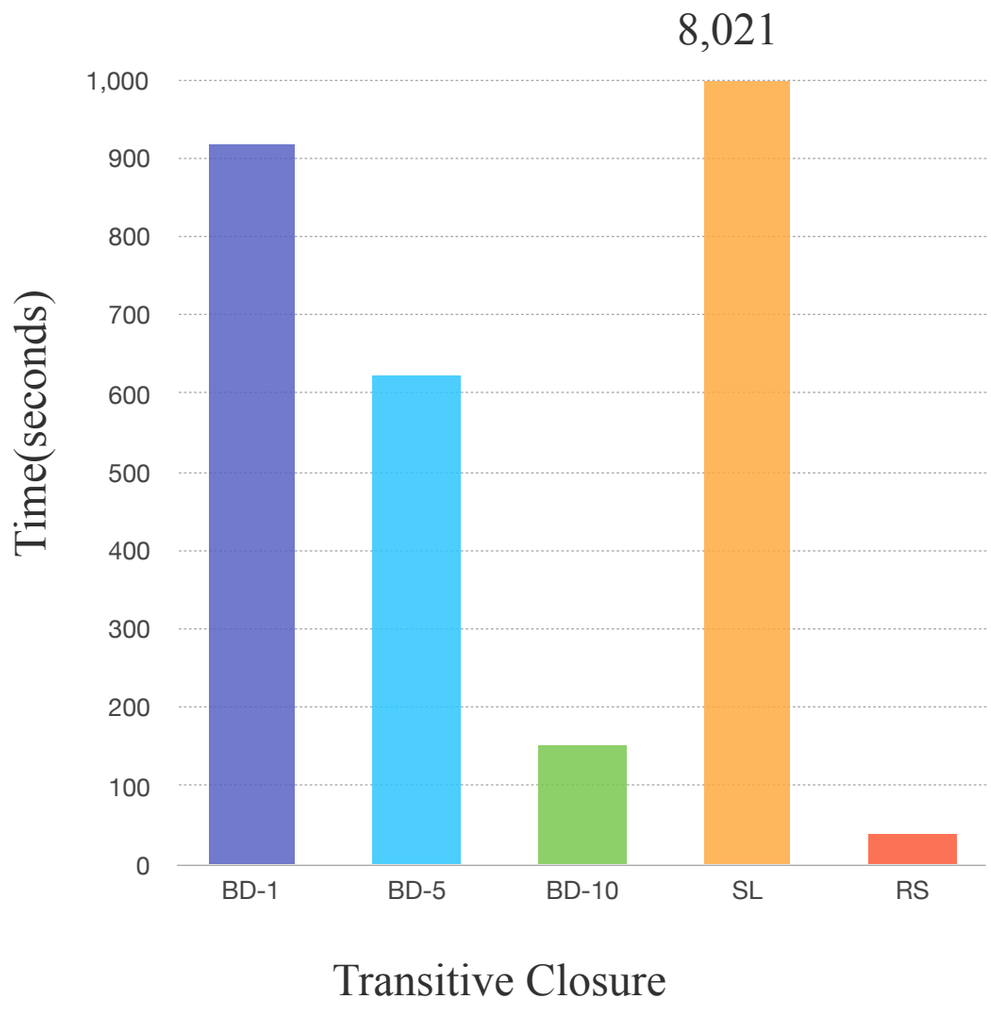
DistGraph by 7.2x

Comparisons with Mining Systems



Comparisons with Datalog Systems

		LiveJournal	Orkut
Triangle Counting	RStream	87	827.4
	BigDatlog-10	94.8	1205.3
	BigDatalog-5	109.6	1850.3
	BigDatalog-1	567.3	-
	SociaLite	896.1	-



Size of Intermediate Data

	Phase	#MB
4-Motif Counting Mico	0	16.5
	1	2086
	2	886378
	3	672194
	Total	1.49TB

Size of Intermediate Data

	Phase	#MB
4-Motif Counting Mico	0	16.5
	1	2086
	2	886378
	3	672194
	Total	1.49TB

13MB initial graph
68182 X

Conclusions

RStream: A single machine, out-of-core graph mining system

- A *simple* and *expressive* API
  - GAS + Relational Algebra => GRAS
- An *efficient* runtime engine
  - implements relational algebra with tuple streaming

https://github.com/rstream-system