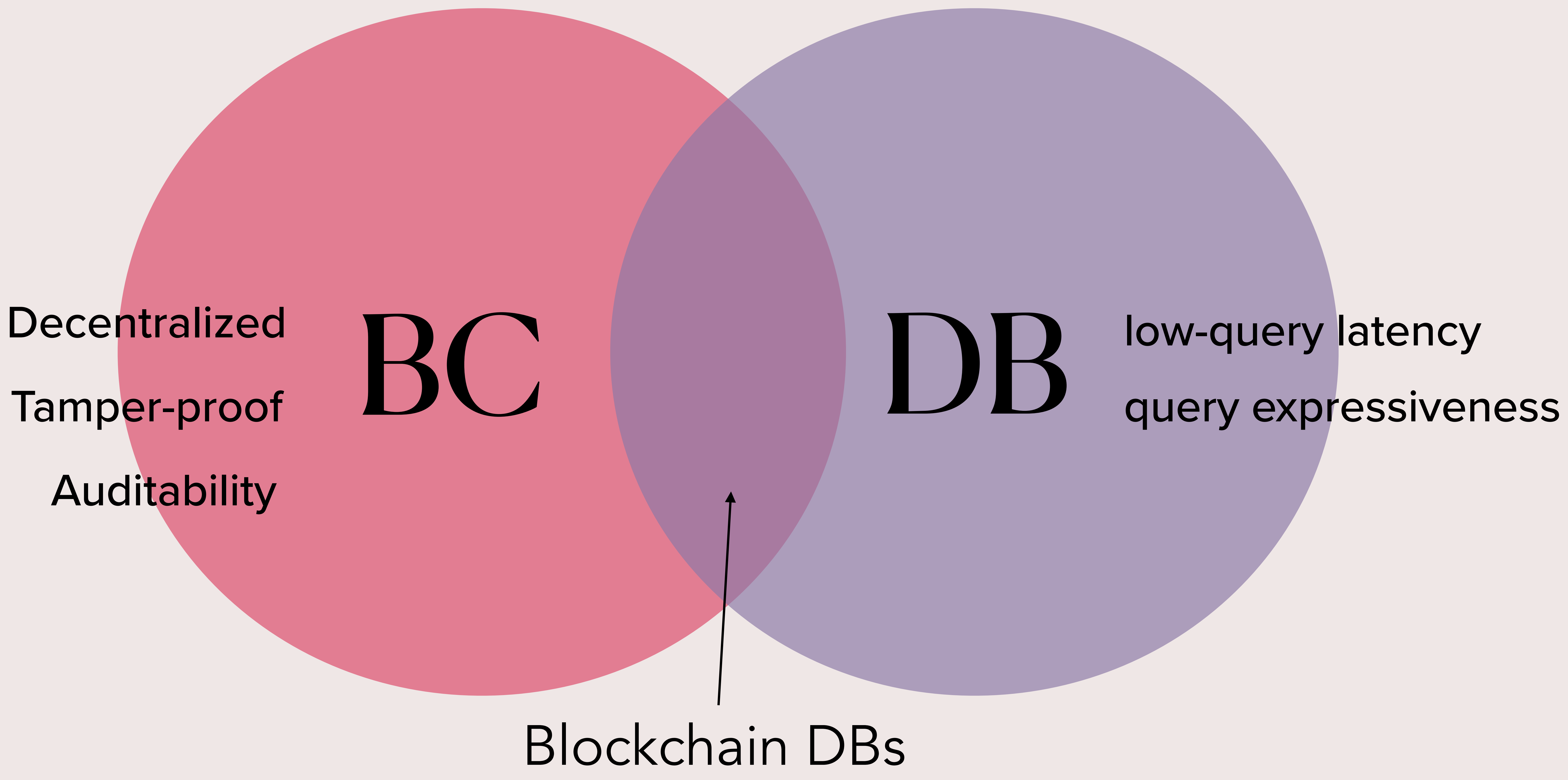
Encrypted Blockchain Databases

Daniel Adkins, Archita Agarwal, Seny Kamara, Tarik Moataz


Decentralized
Tamper-proof
Auditability
BC
DB
low-query latency
query expressiveness
Blockchain DBs

Blockchain DBs

Blockchain DBs

Blockchain DBs

BIGCHAIN DB

bluzelle

- electronic healthcare records
- financial records
- legal documents
- customer data

Blockchain DBs

BIGCHAIN DB

bluzelle

CONFIDENTIALITY ?

- electronic healthcare records
- financial records
- legal documents
- customer data

Blockchain DBs

BIGCHAIN DB

bluzelle

CONFIDENTIALITY ?

- electronic healthcare records
- financial records
- legal documents
- customer data

In this work …

How to store Encrypted Multi-Maps on BCs

In this work ...

How to store Encrypted Multi-Maps on BCs

a kind of NoSQL Databases, like Key-Value Stores

What are Multimaps?

Set of label/value tuples

ℓ_1	(v_1, v_2)
ℓ_2	(v_1, v_2, v_3)
ℓ_3	(v_3)
ℓ_4	(v_4, v_5)
ℓ_5	(v_2, v_5, v_7, v_8)

Varying lengths

MM

What are

Encrypted

What are Multimaps?

Set of label/value tuples

ℓ_1	(v_1, v_2)
ℓ_2	(v_1, v_2, v_3)
ℓ_3	(v_3)
ℓ_4	(v_4, v_5)
ℓ_5	(v_2, v_5, v_7, v_8)

EMM

What are Encrypted Multimaps?

Set of label/value tuples

Encrypted

ℓ_1	(v_1, v_2)
ℓ_2	(v_1, v_2, v_3)
ℓ_3	(v_3)
ℓ_4	(v_4, v_5)
ℓ_5	(v_2, v_5, v_7, v_8)

EMM

What are Encrypted Multimaps?

Set of label/value tuples

Encrypted

t ← Query(ℓ)

ℓ_1	(v_1, v_2)
ℓ_2	(v_1, v_2, v_3)
ℓ_3	(v_3)
ℓ_4	(v_4, v_5)
ℓ_5	(v_2, v_5, v_7, v_8)

EMM

What are Encrypted Multimaps?

Set of label/value tuples

Encrypted

$t \leftarrow \mathsf{Query}(\ell)$

$\mathsf{EMM}' \leftarrow \mathsf{Add}(\ell, t')$

ℓ_1	(v_1, v_2)
ℓ_2	(v_1, v_2, v_3)
ℓ_3	(v_3)
ℓ_4	(v_4, v_5)
ℓ_5	(v_2, v_5, v_7, v_8)

EMM

What are Encrypted Multimaps?

Set of label/value tuples

Encrypted

$\mathbf{t} \leftarrow \text{Query}(\ell)$

$\text{EMM}' \leftarrow \text{Add}(\ell, \mathbf{t}')$

ℓ_1	(v_1, v_2)
ℓ_2	(v_1, v_2, v_3)
ℓ_3	(v_3) (v_4)
ℓ_4	(v_4, v_5)
ℓ_5	(v_2, v_5, v_7, v_8)

EMM

What are Encrypted Multimaps?

Set of label/value tuples

Encrypted

$\mathbf{t} \leftarrow$ Query(ℓ)

EMM$'$ $\leftarrow$ Add($\ell, \mathbf{t}'$)

EMM$'$ $\leftarrow$ Delete($\ell, \mathbf{t}'$)

ℓ_1	(v_1, v_2)
ℓ_2	(v_1, v_2, v_3)
ℓ_3	(v_3) (v_4)
ℓ_4	(v_4, v_5)
ℓ_5	(v_2, v_5, v_7, v_8)

EMM

In our work

LSX

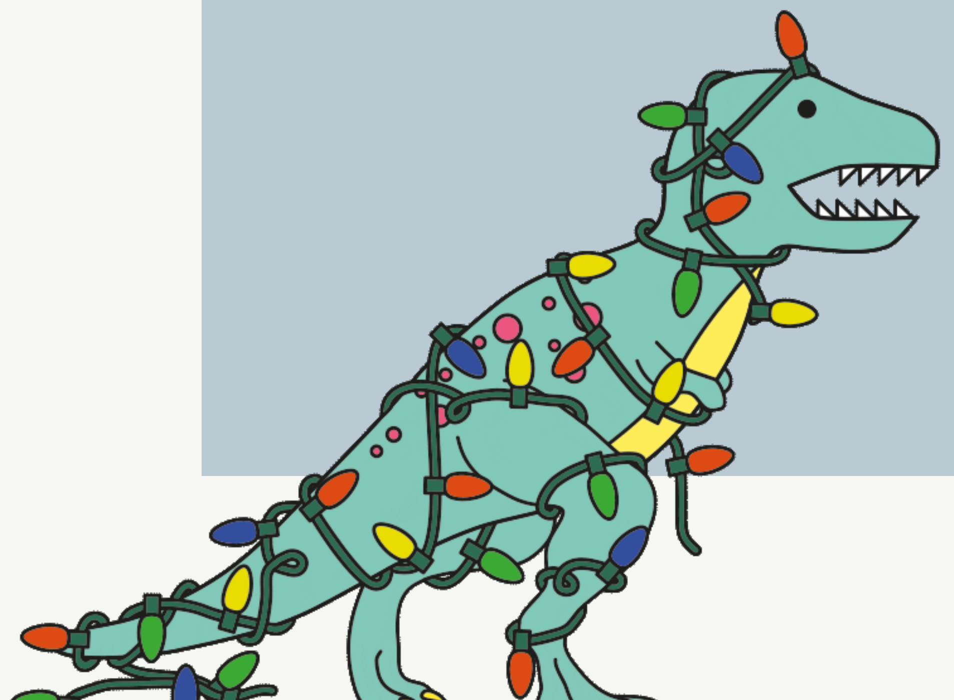
List-Based

TRX

Tree-Based

PAX

Patch-Based



In our work

LSX

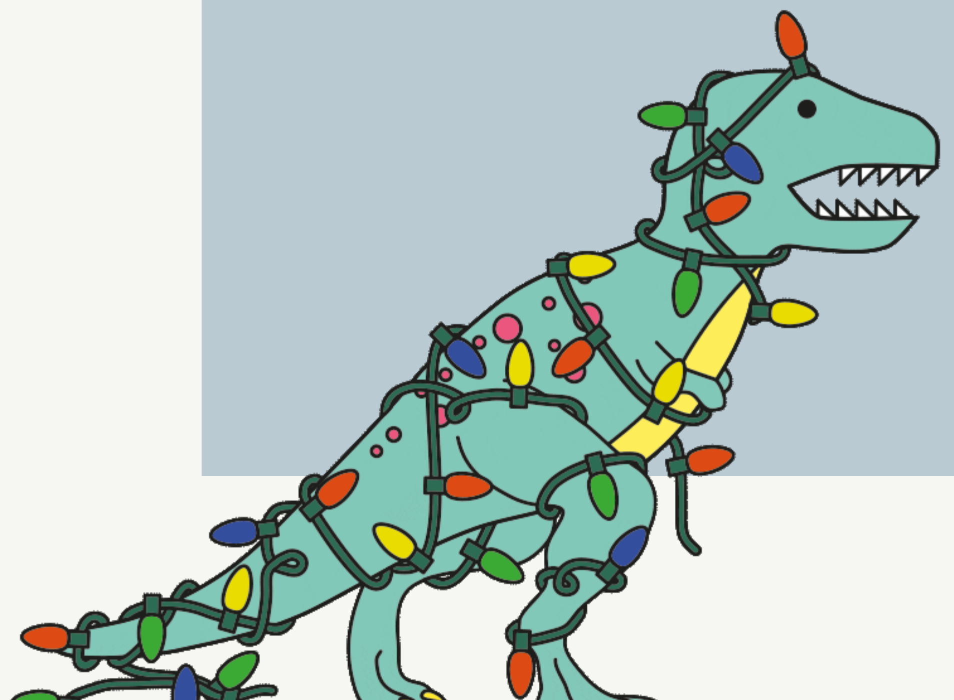
List-Based

TRX

Tree-Based

PAX

Patch-Based



In our work

LSX

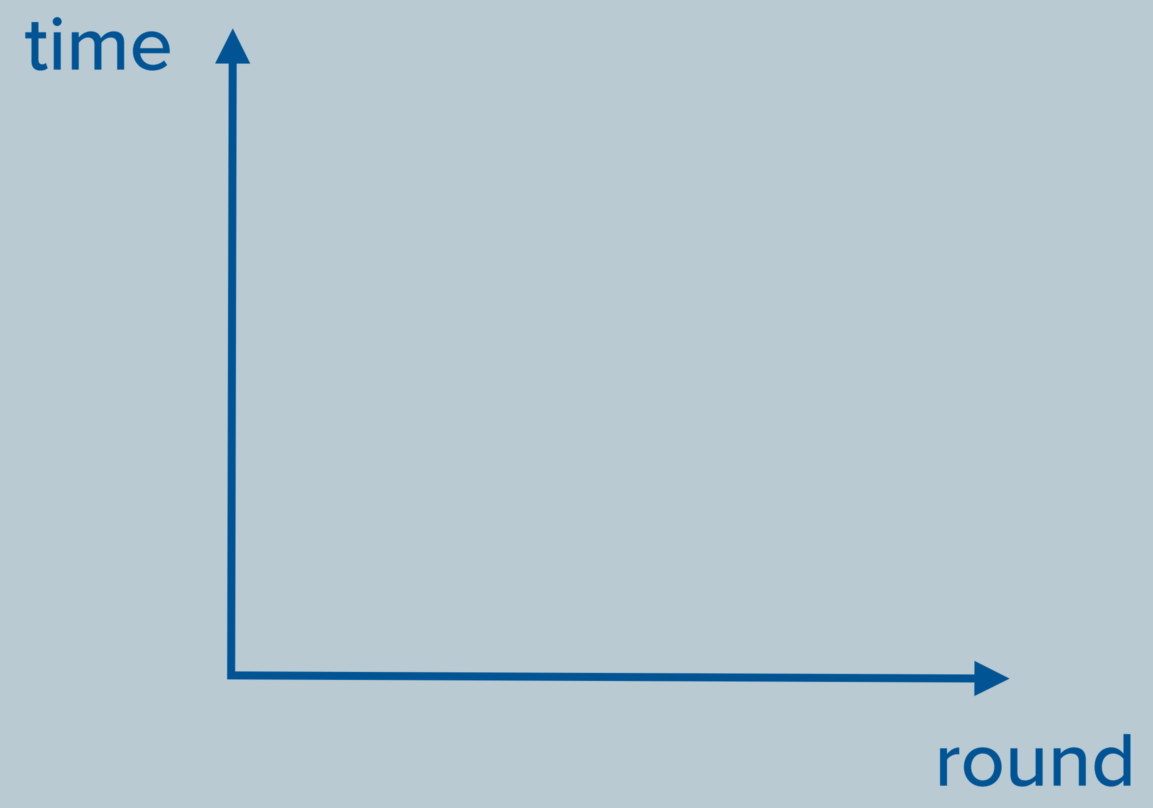
List-Based

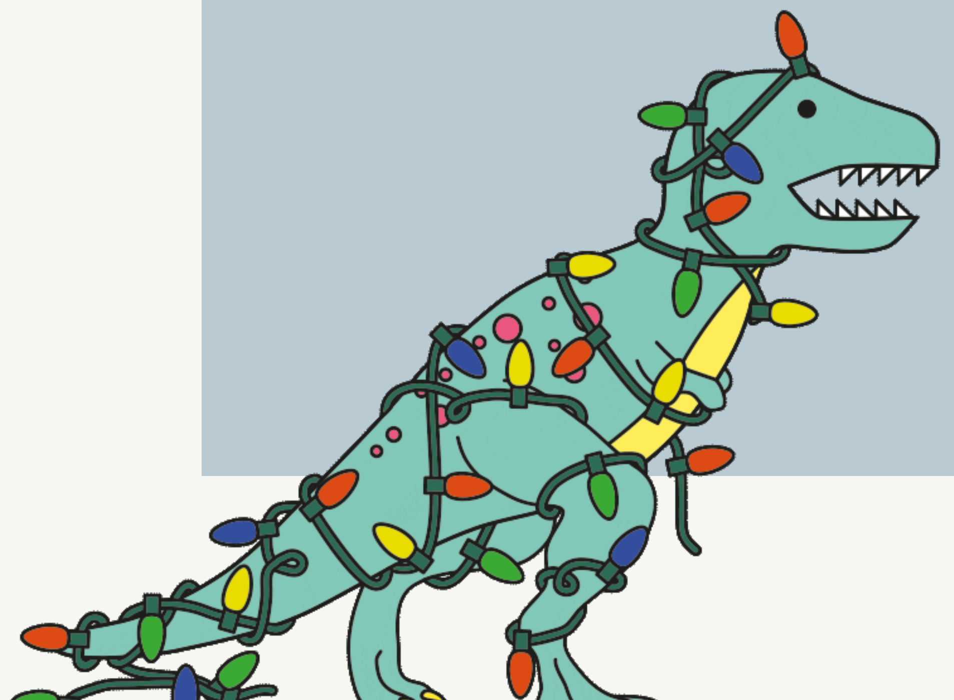
TRX

Tree-Based

PAX

Patch-Based





In our work

LSX

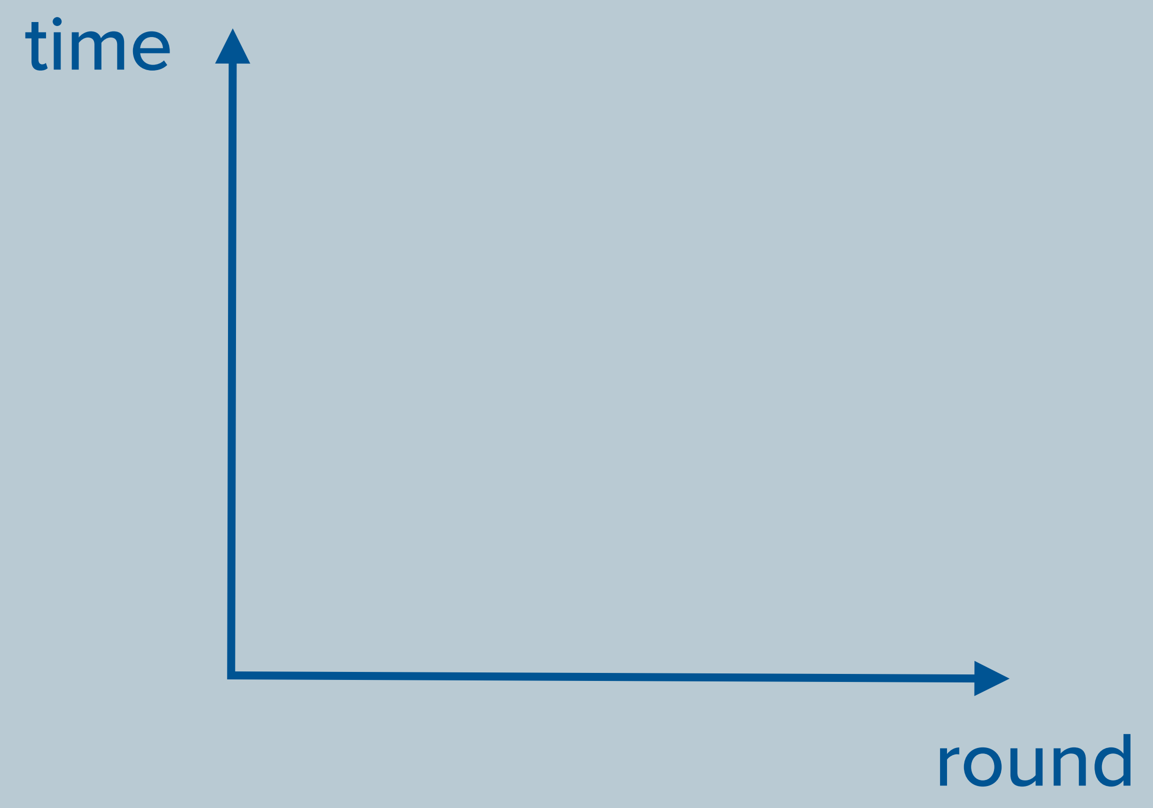
List-Based

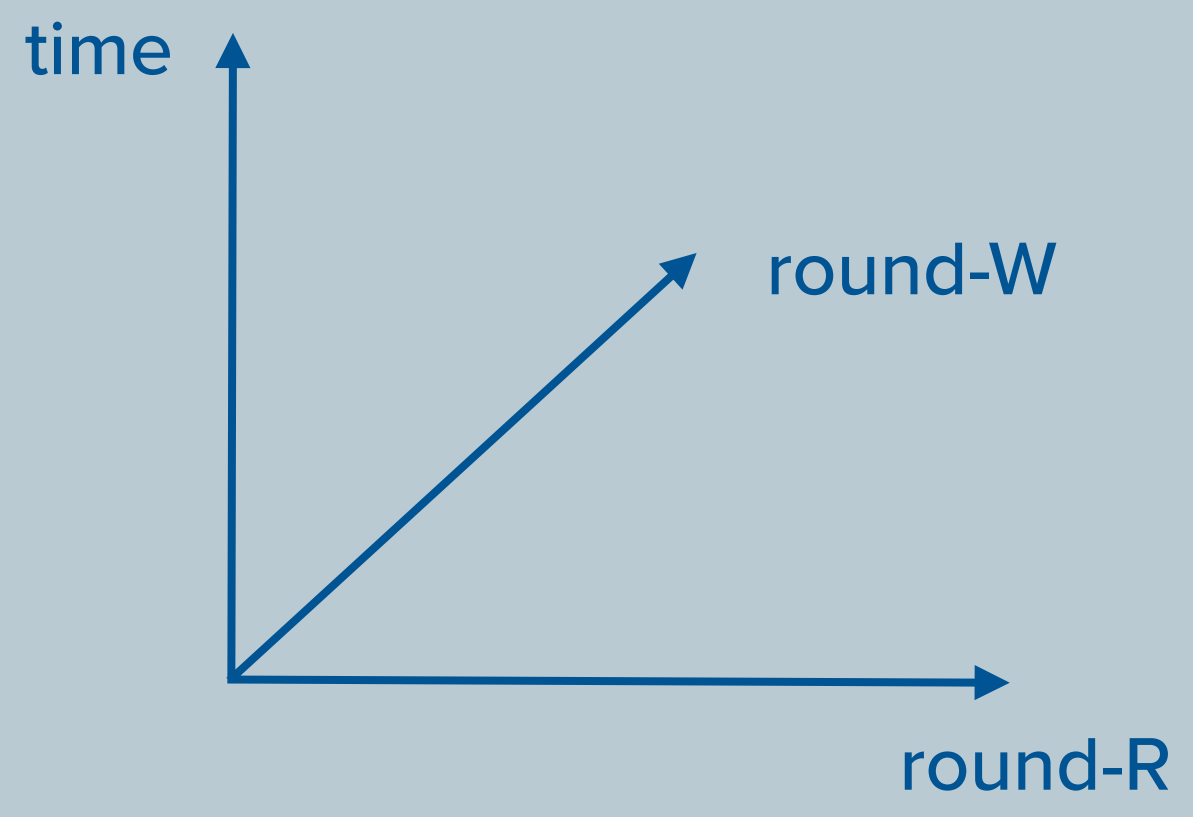
TRX

Tree-Based

PAX

Patch-Based





In our work

LSX

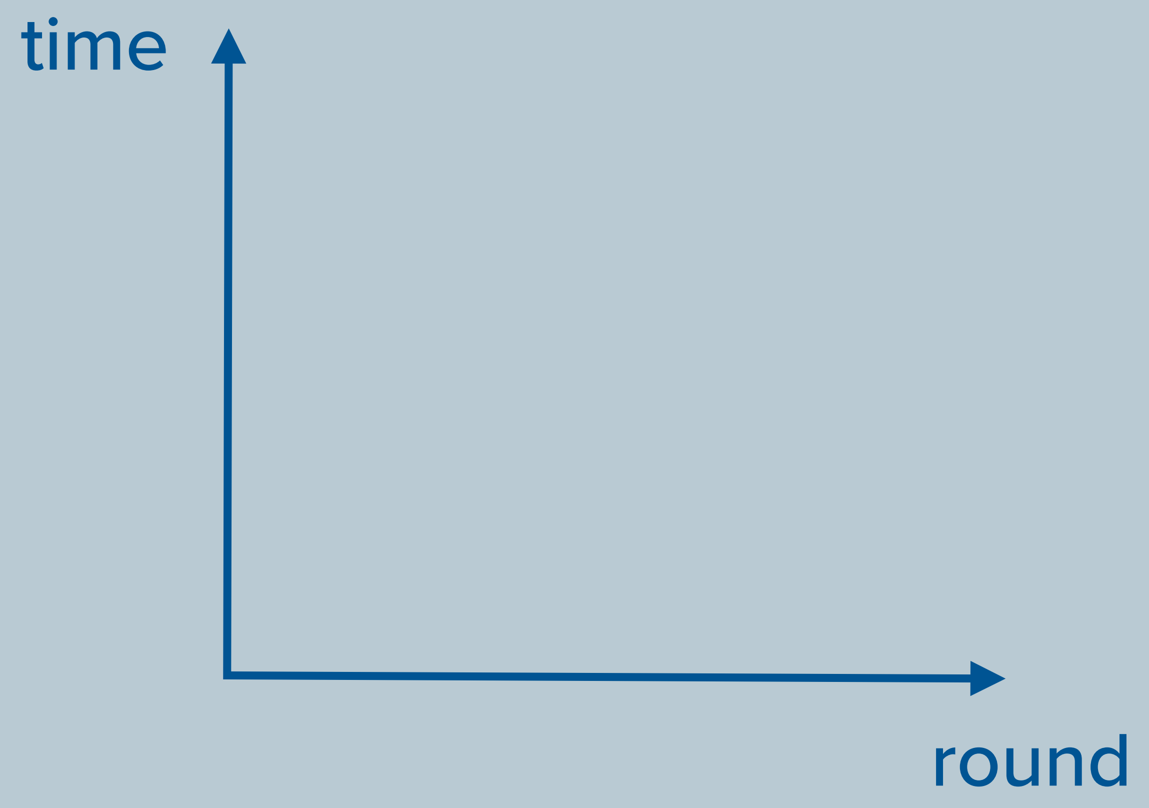
List-Based

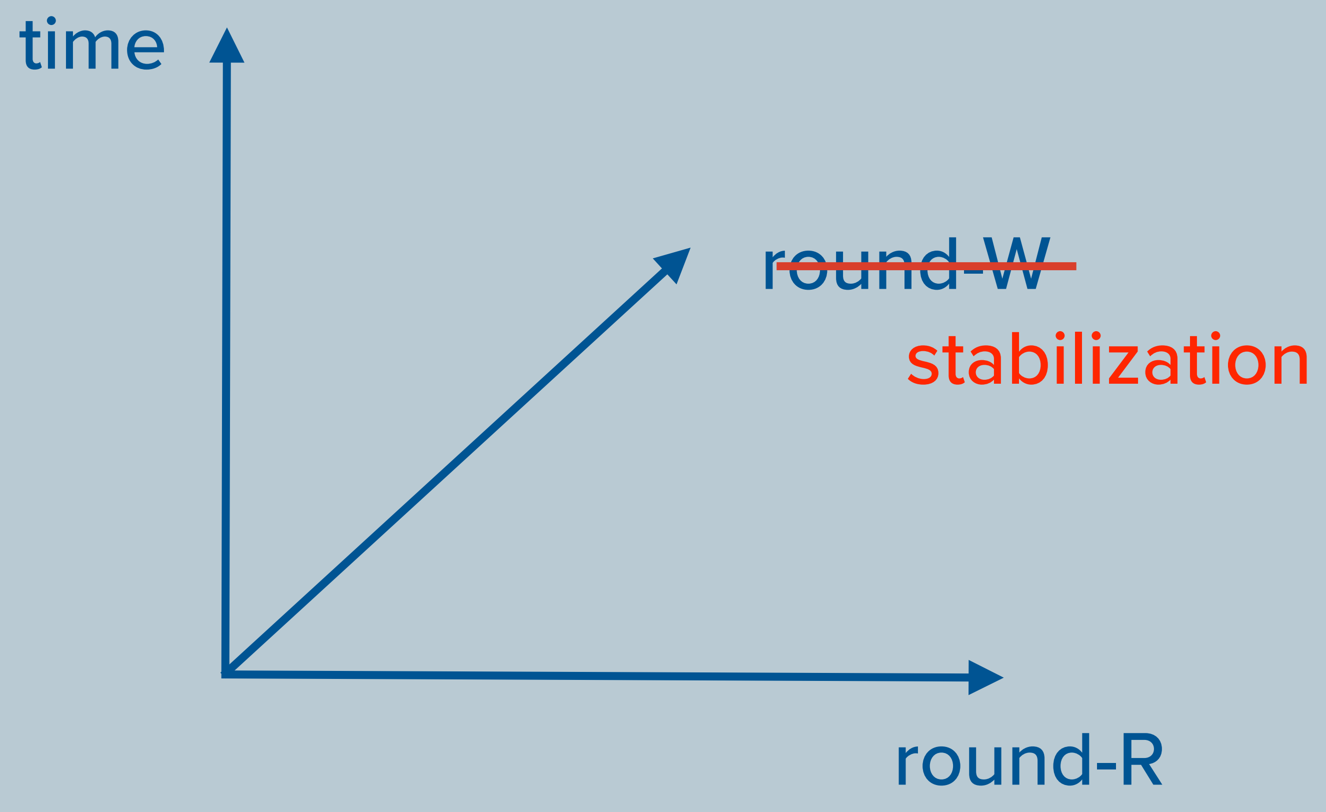
TRX

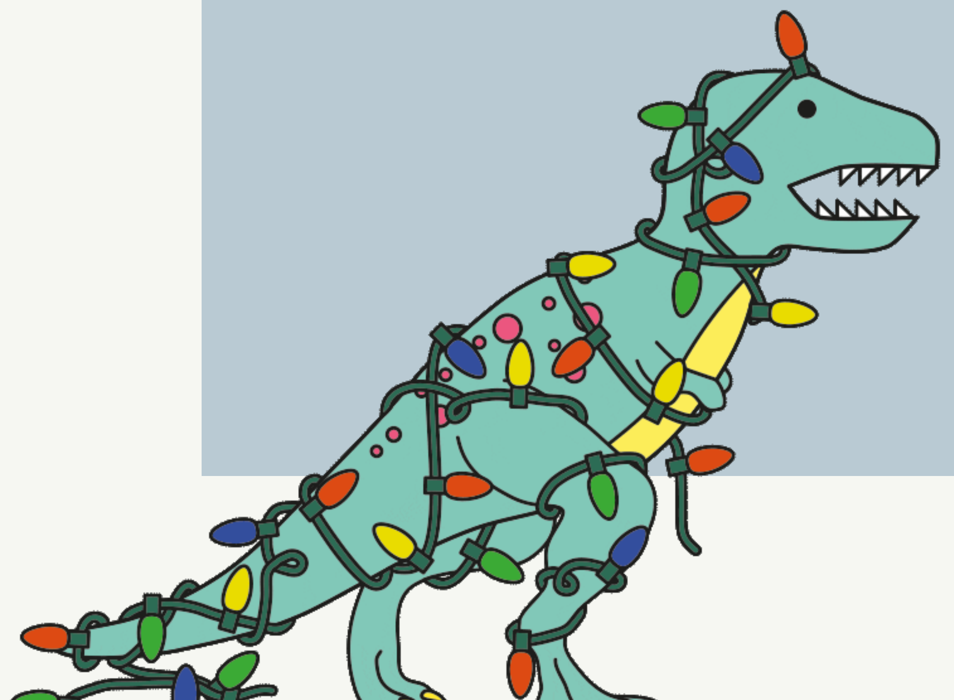
Tree-Based

PAX

Patch-Based

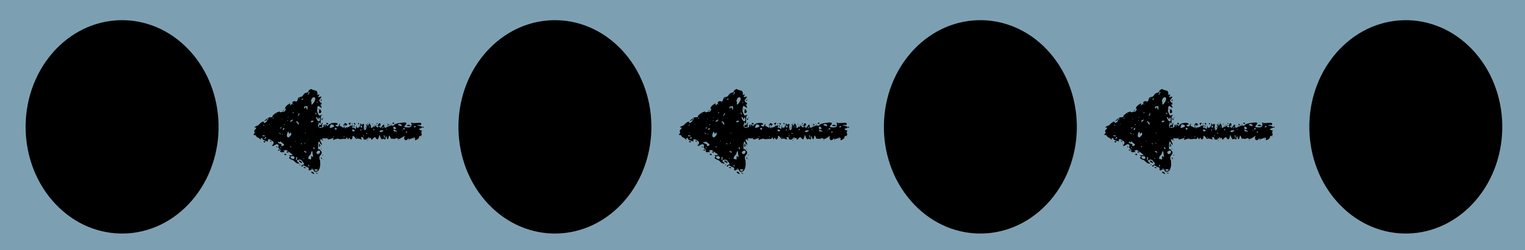




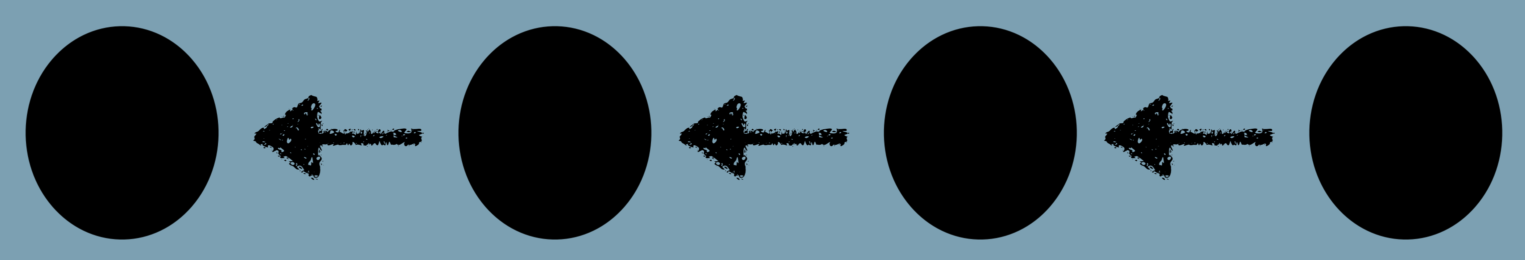


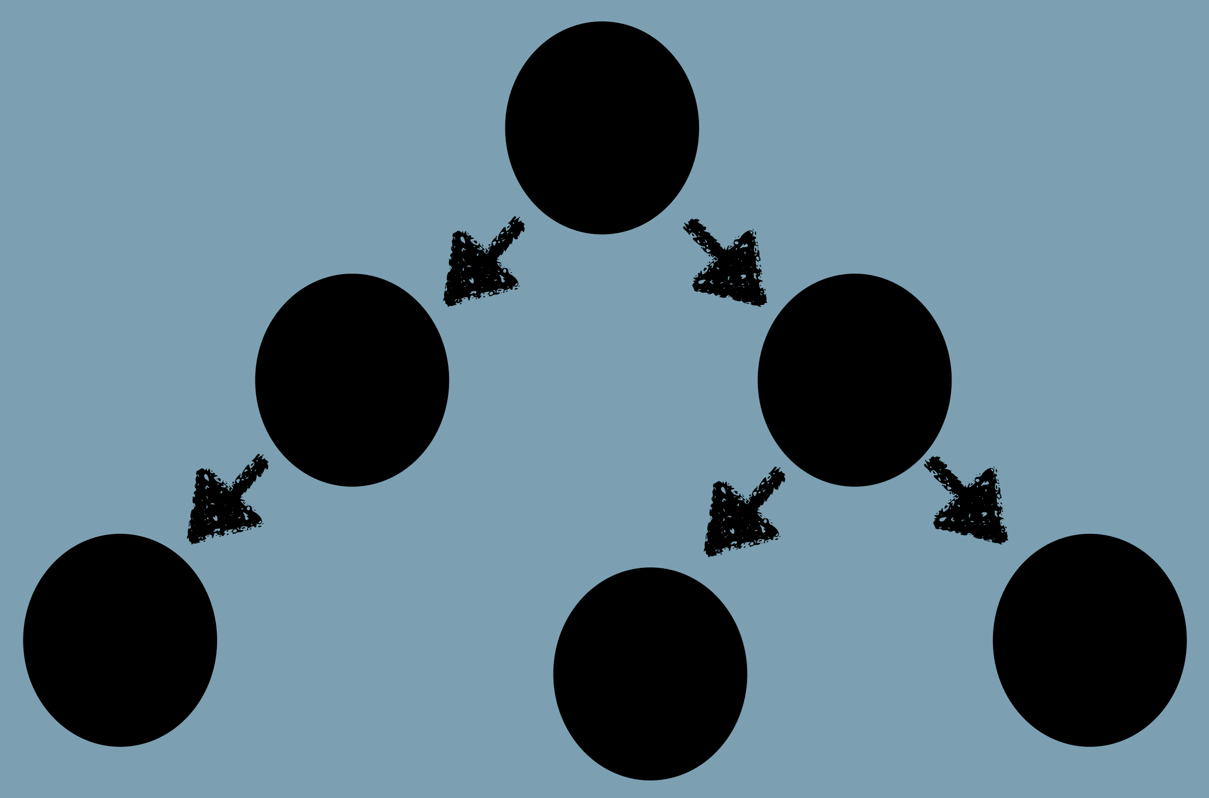
Storing data structures on BCs

Storing data structures on BCs



Storing data structures on BCs

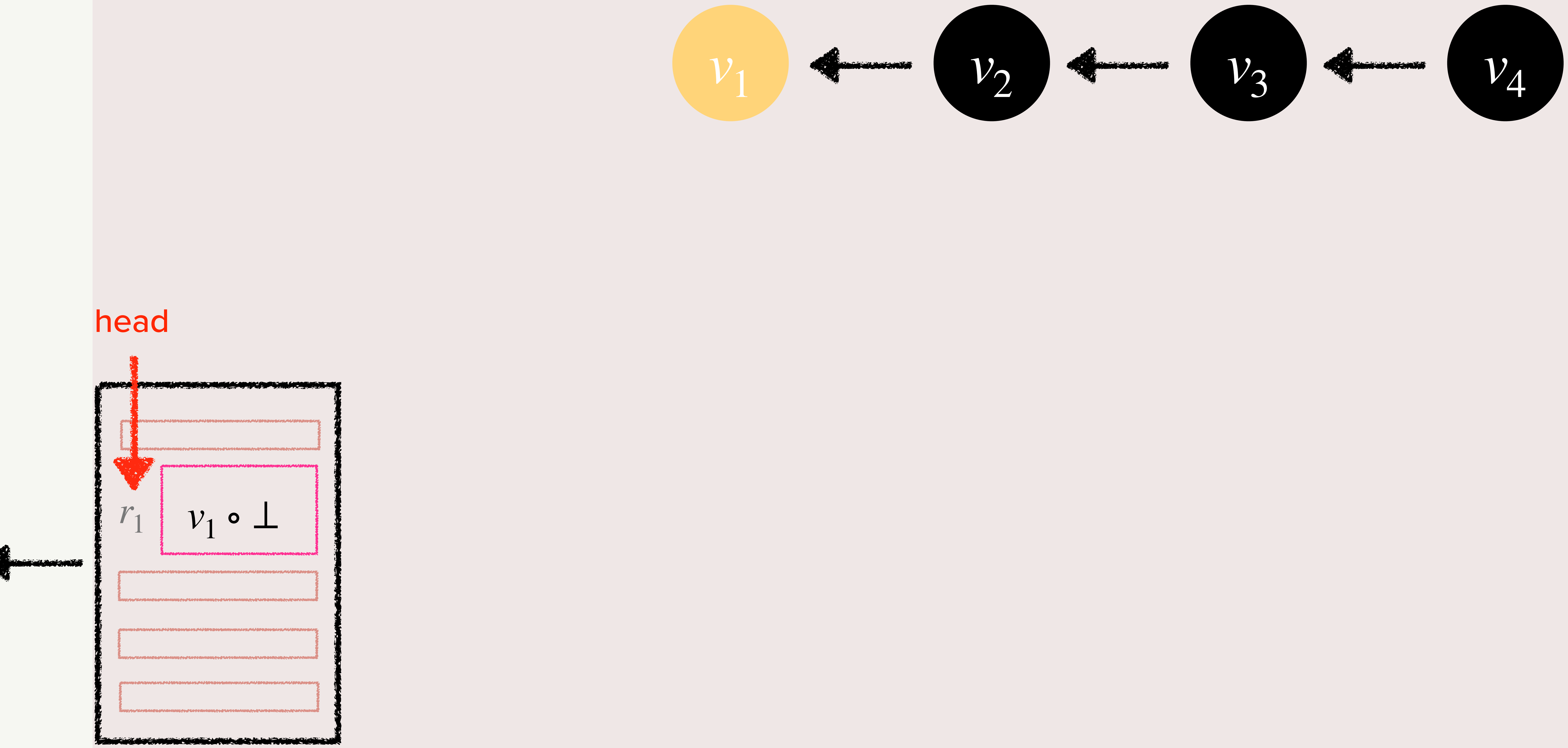




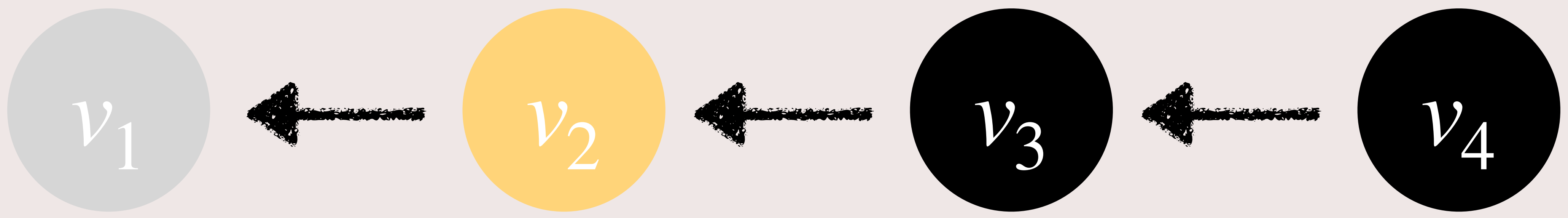
Storing linked-list on BCs

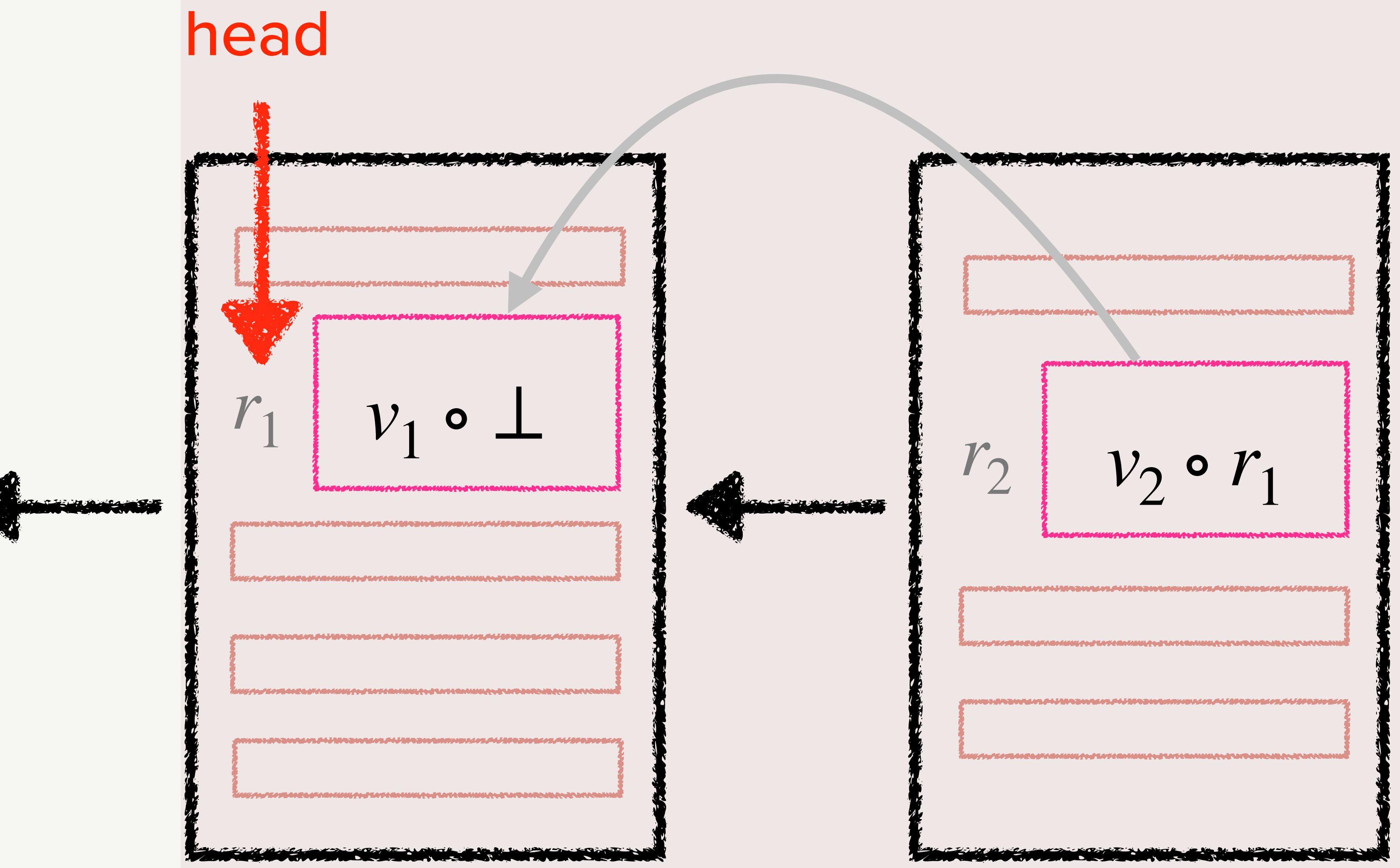
$$v_1 \leftarrow v_2 \leftarrow v_3 \leftarrow v_4$$

Storing linked-list on BCs

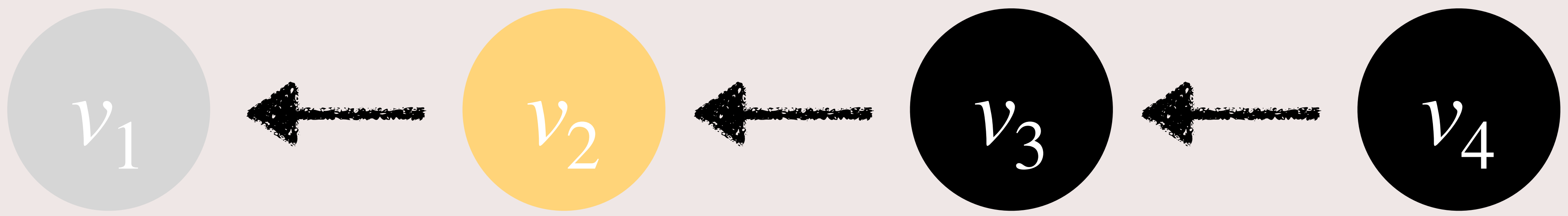


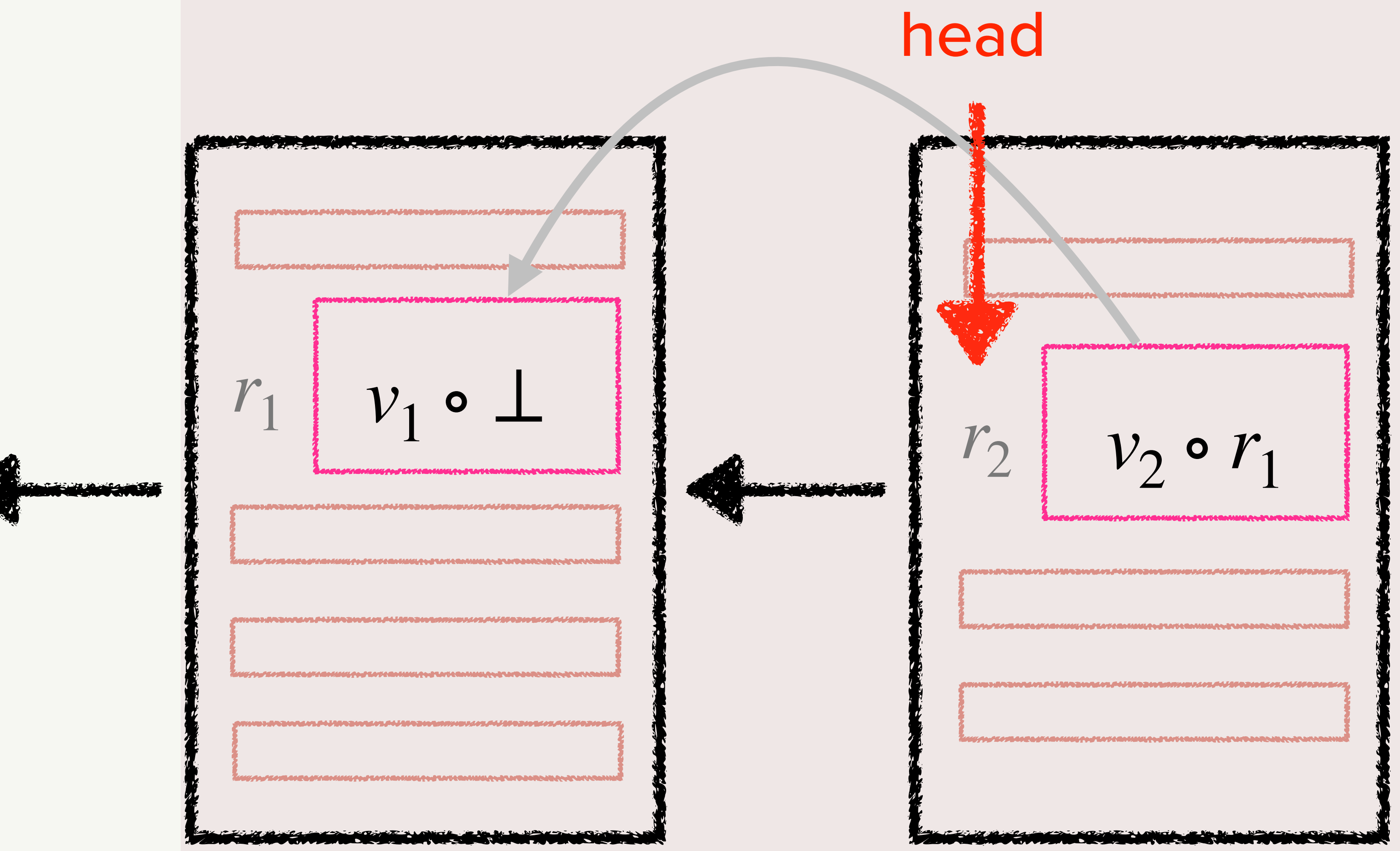
Storing linked-list on BCs



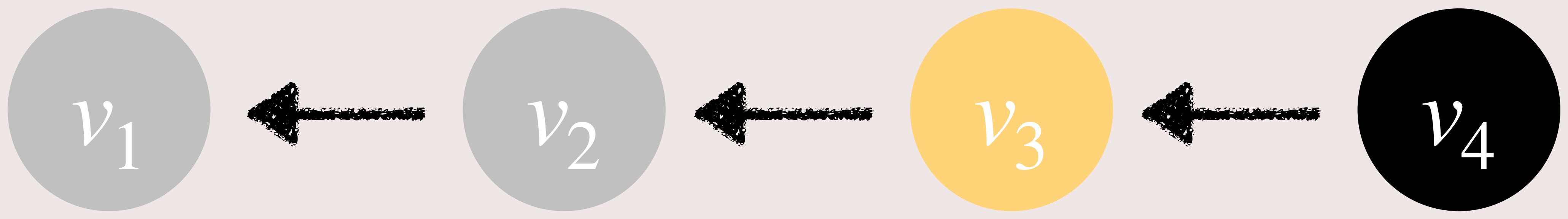


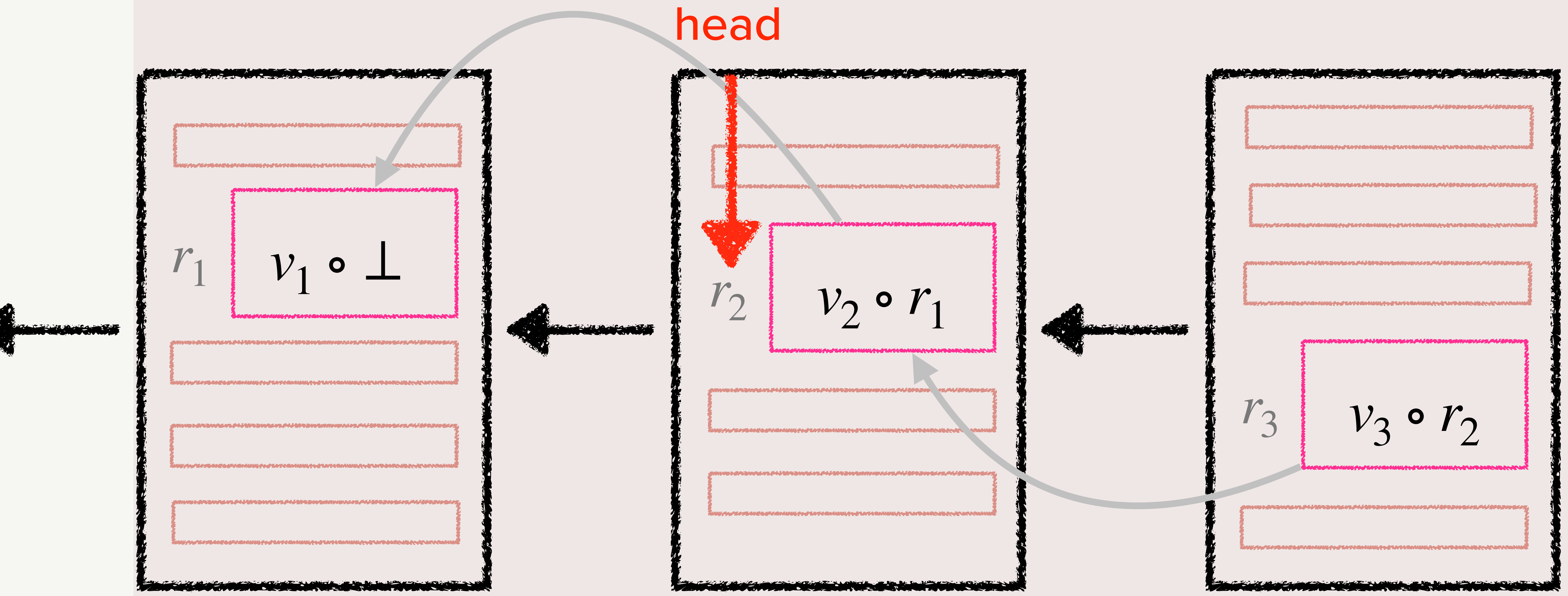
Storing linked-list on BCs



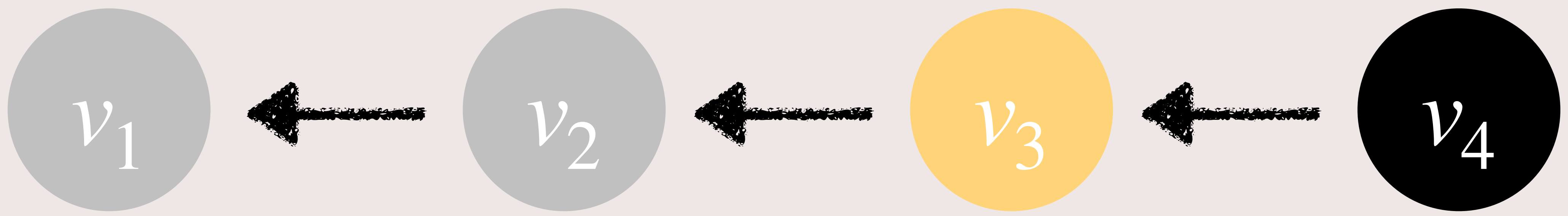


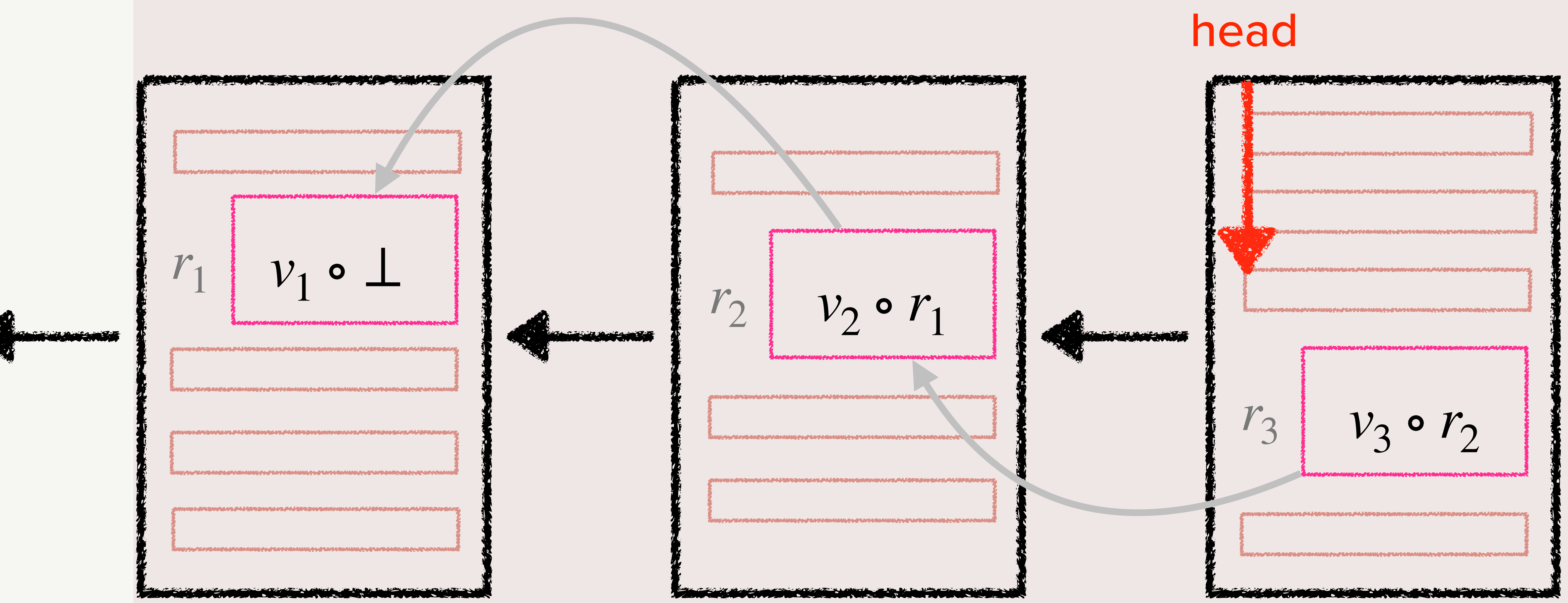
Storing linked-list on BCs



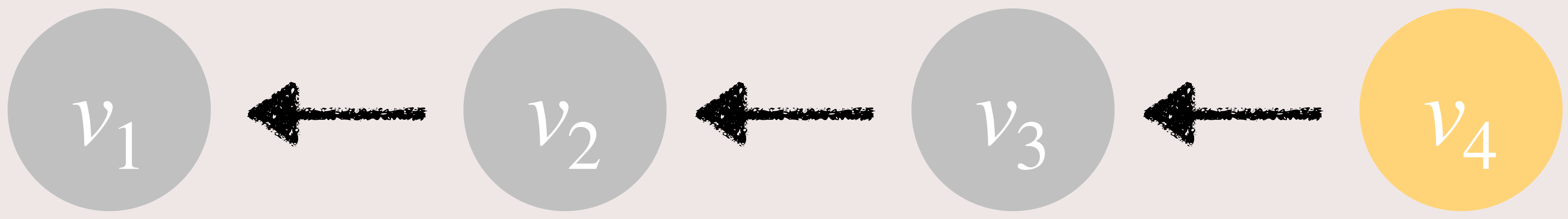


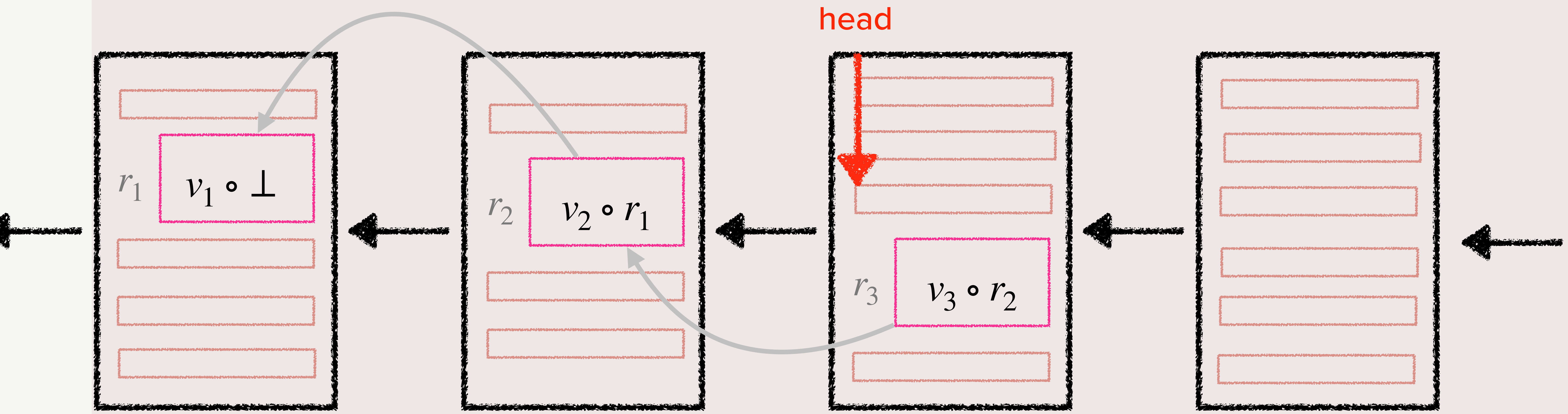
Storing linked-list on BCs



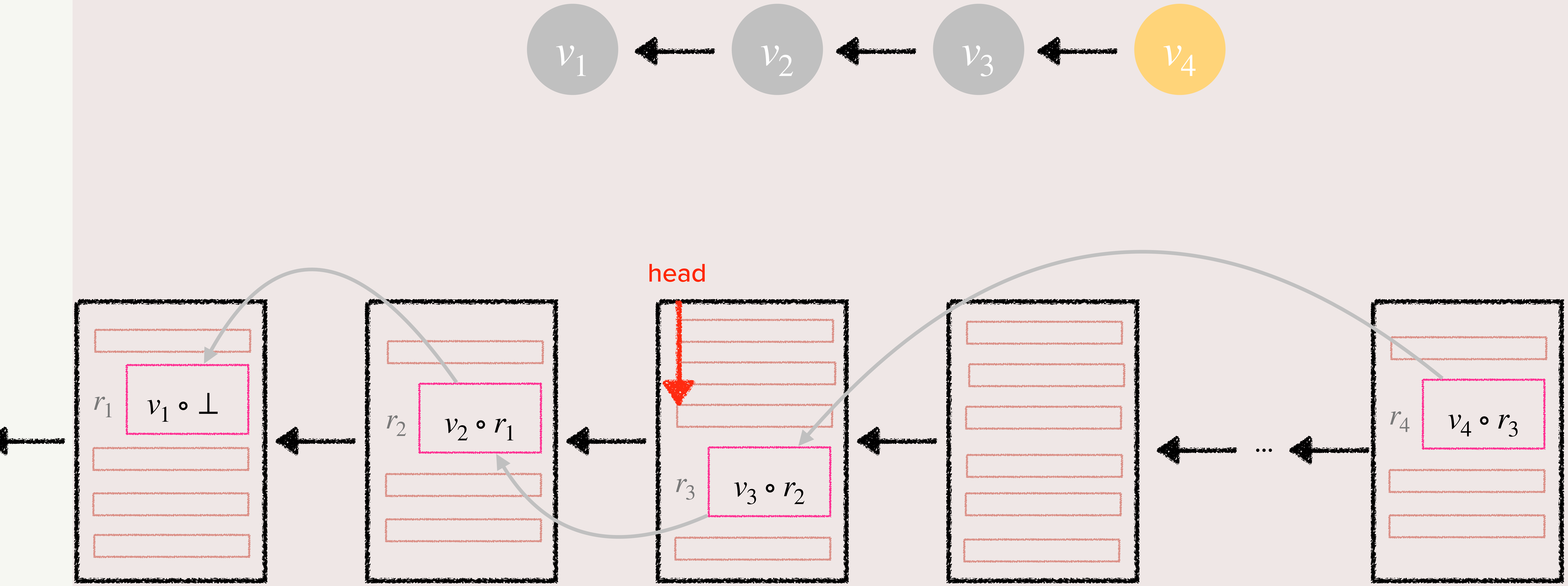


Storing linked-list on BCs

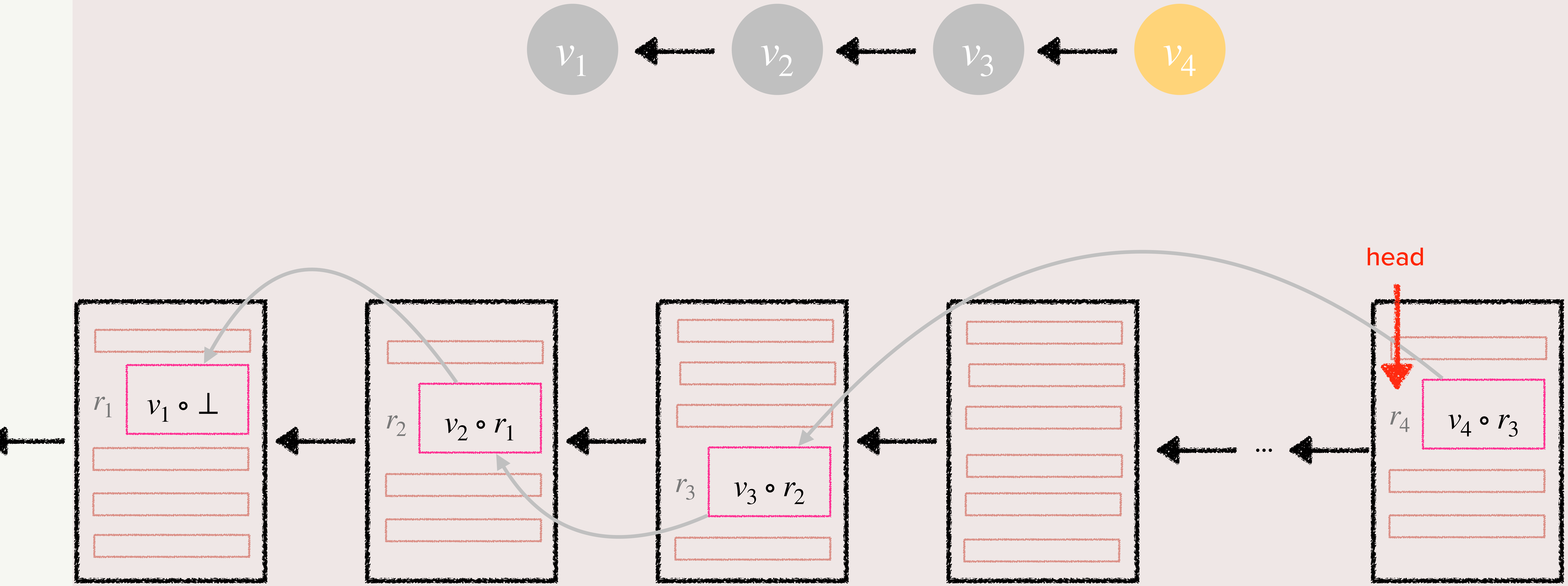




Storing linked-list on BCs



Storing linked-list on BCs



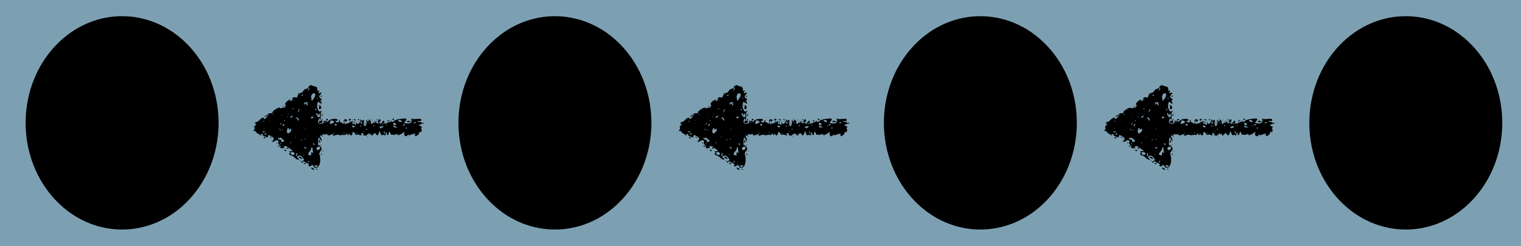
Storing linked-list on BCs

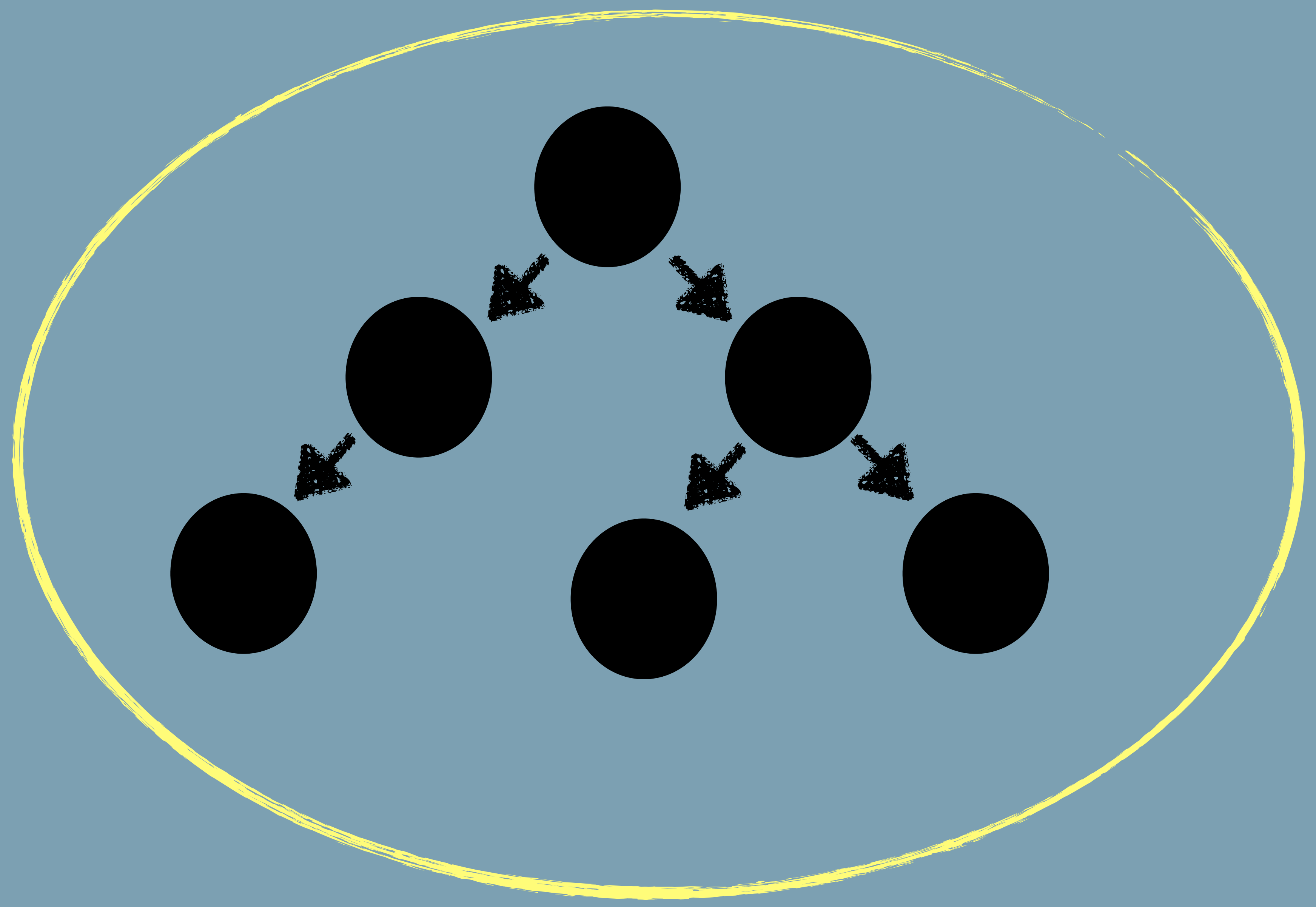
Stabilization Complexity

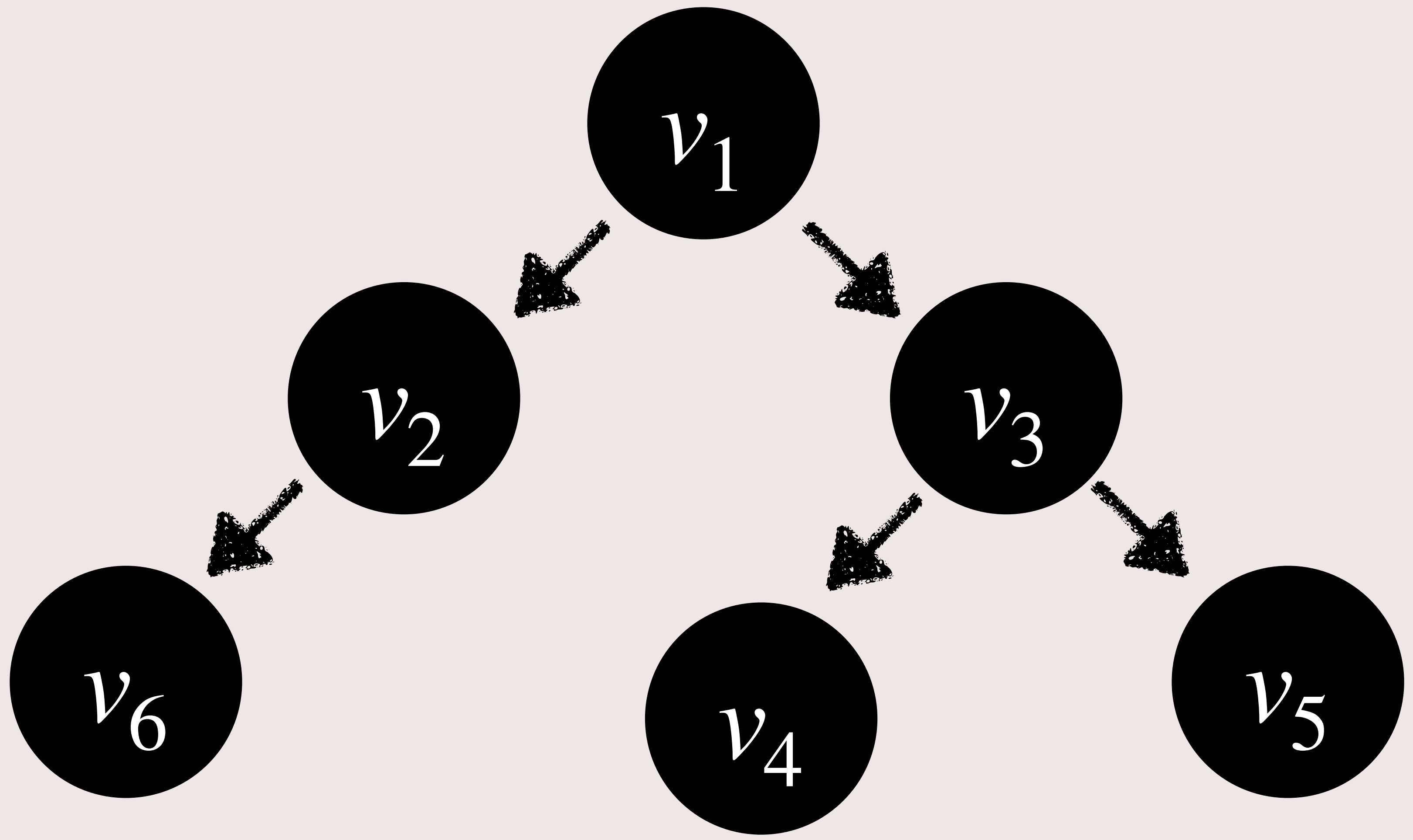
(aka number of write rounds)

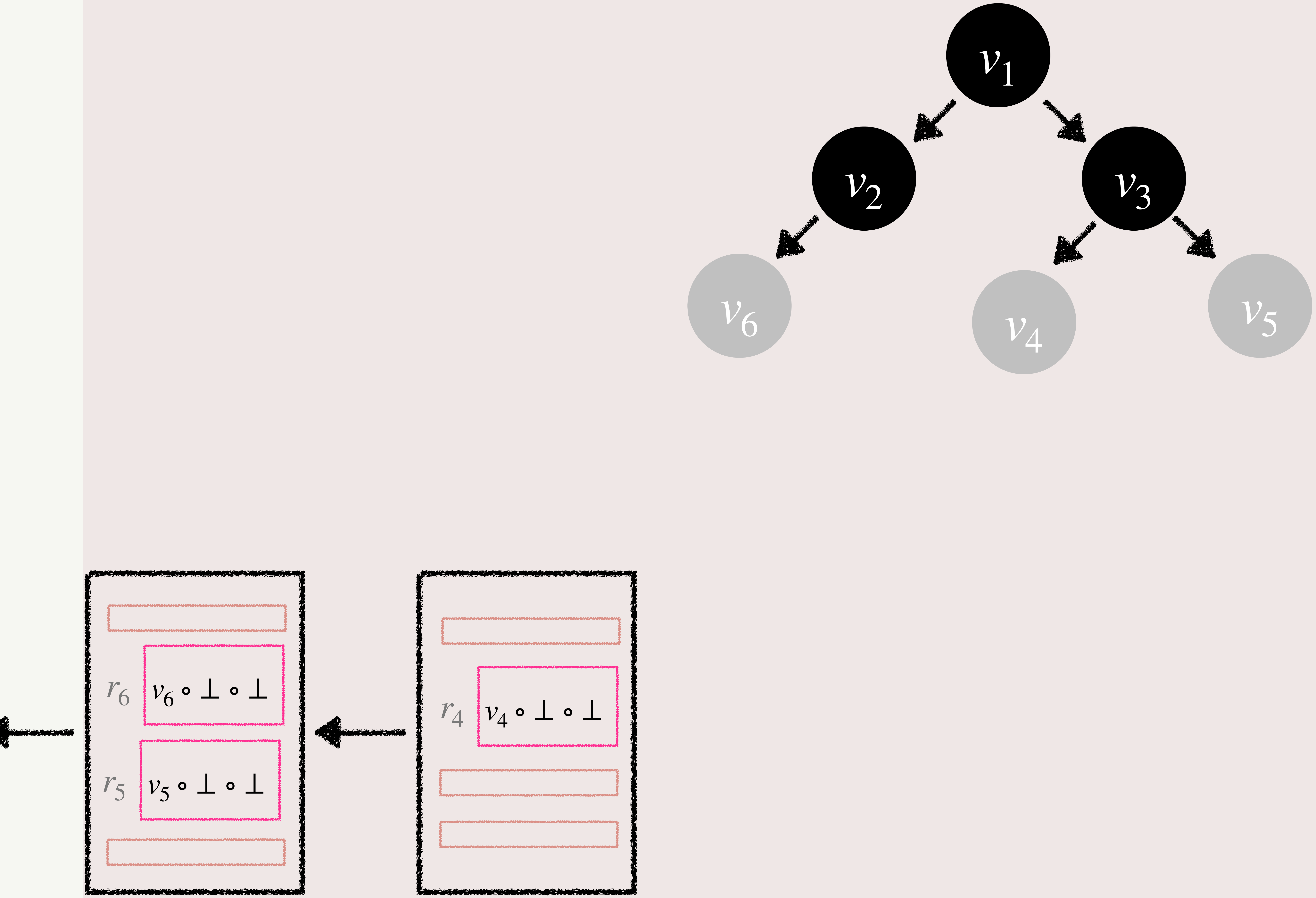
number of values in linked-list

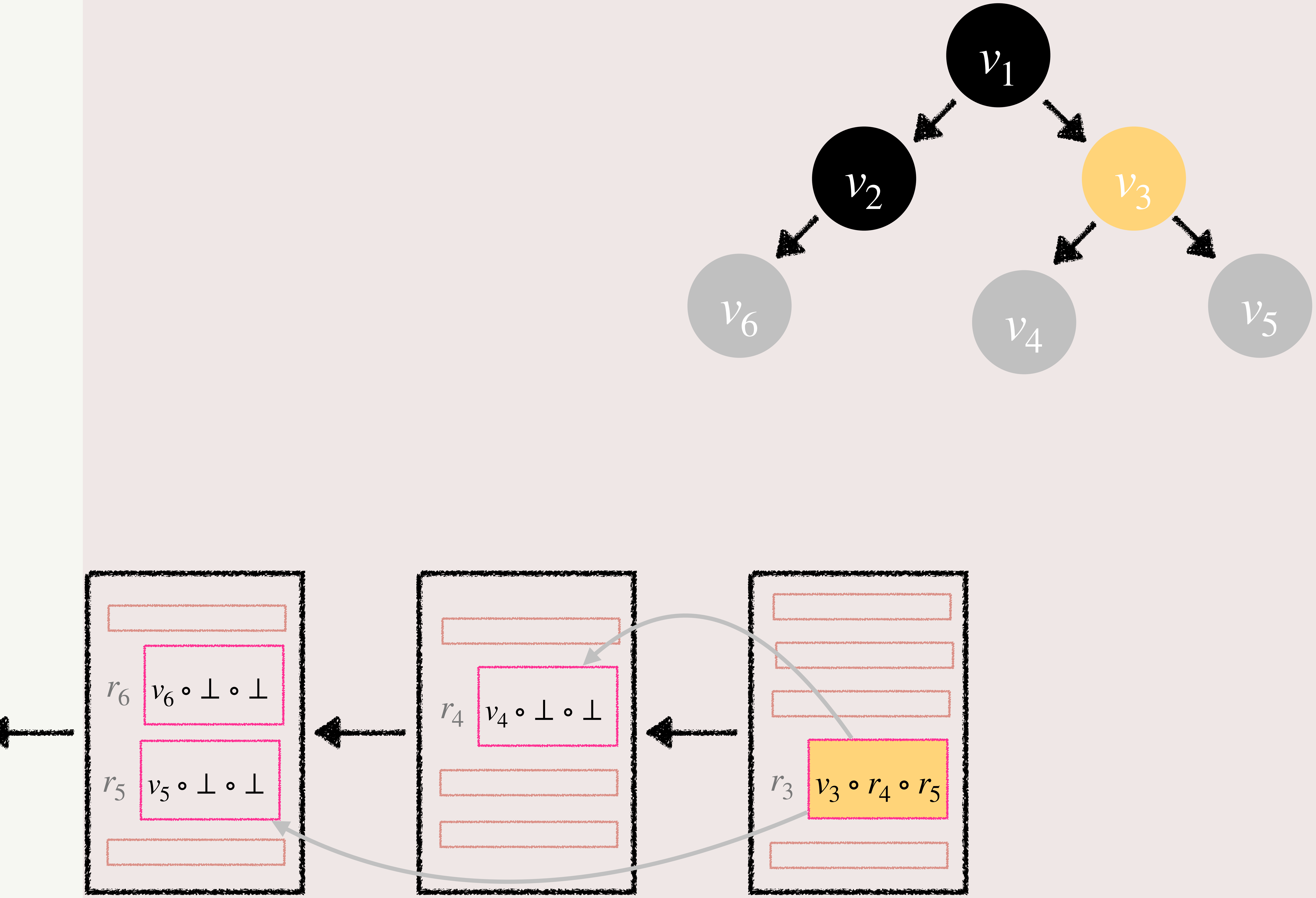
Storing data structures on BCs






v_1
v_2
v_3
v_6
v_4
v_5


v_1
v_2
v_3
v_6
v_4
v_5
r_6
$v_6 \circ \perp \circ \perp$
r_5
$v_5 \circ \perp \circ \perp$
r_4
$v_4 \circ \perp \circ \perp$

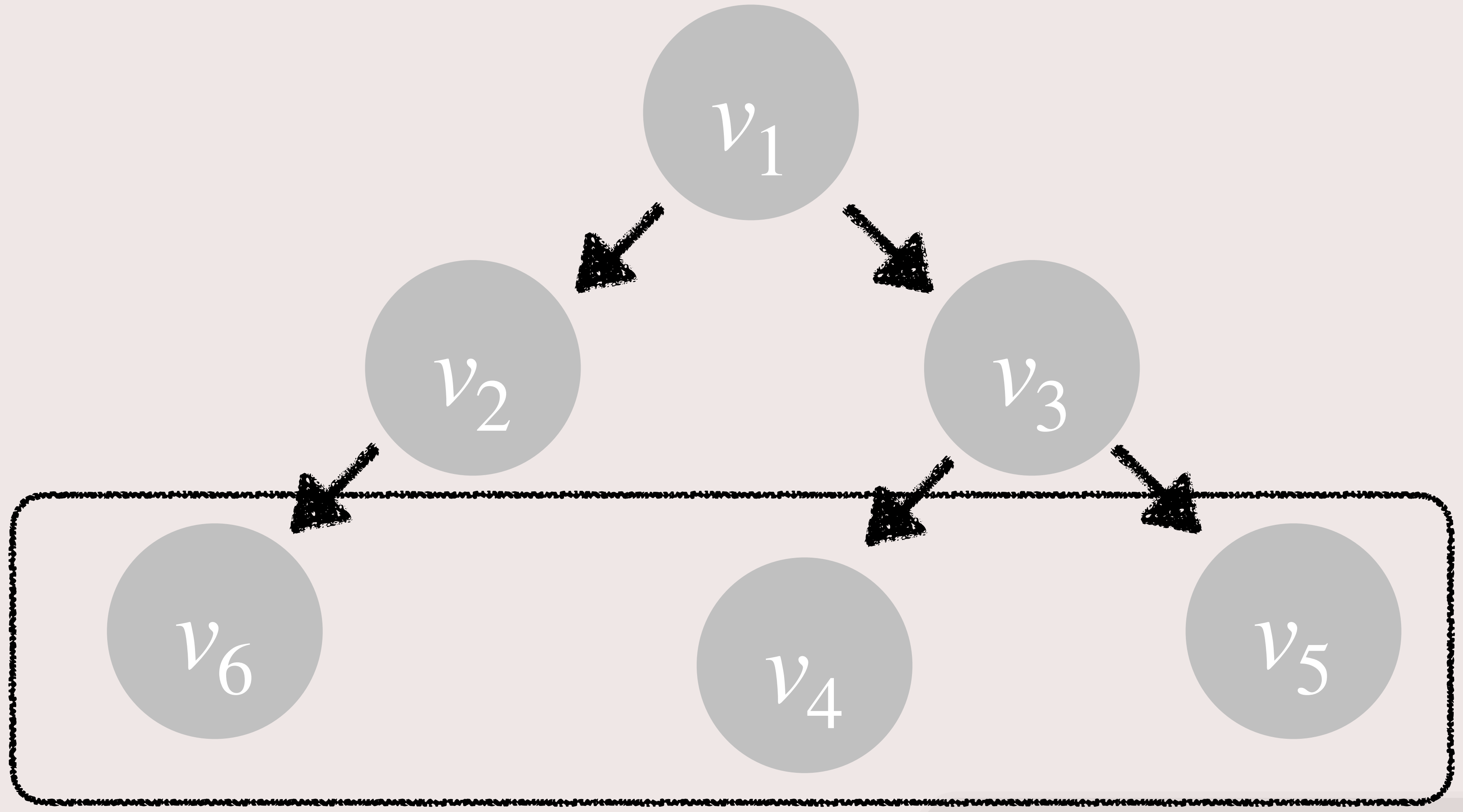


v_1
v_2
v_3
v_6
v_4
v_5
r_6
$v_6 \circ \perp \circ \perp$
r_5
$v_5 \circ \perp \circ \perp$
r_4
$v_4 \circ \perp \circ \perp$
r_3
$v_3 \circ r_4 \circ r_5$

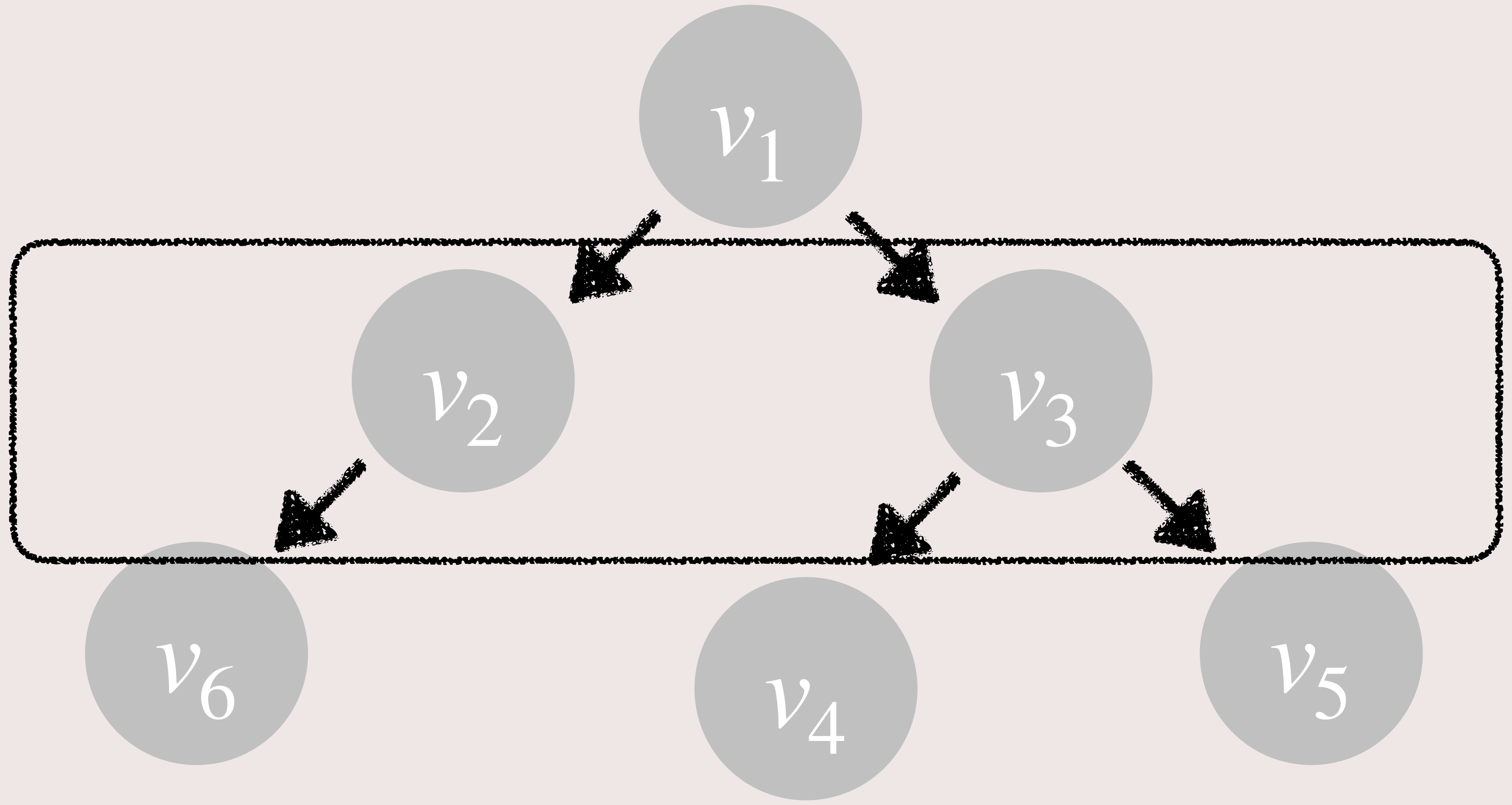
v_1
v_2
v_3
v_6
v_4
v_5
root
r_6
$v_6 \circ \perp \circ \perp$
r_5
$v_5 \circ \perp \circ \perp$
r_4
$v_4 \circ \perp \circ \perp$
r_3
$v_3 \circ r_4 \circ r_5$
r_2
$v_2 \circ r_6 \circ \perp$
...
r_1
$v_1 \circ r_2 \circ r_3$


v_1
v_2
v_3
v_6
v_4
v_5
v_2 just dependent on v_6
root
r_6 $v_6 \circ \perp \circ \perp$
r_5 $v_5 \circ \perp \circ \perp$
r_4 $v_4 \circ \perp \circ \perp$
r_3 $v_3 \circ r_4 \circ r_5$
r_2 $v_2 \circ r_6 \circ \perp$
...
r_1 $v_1 \circ r_2 \circ r_3$

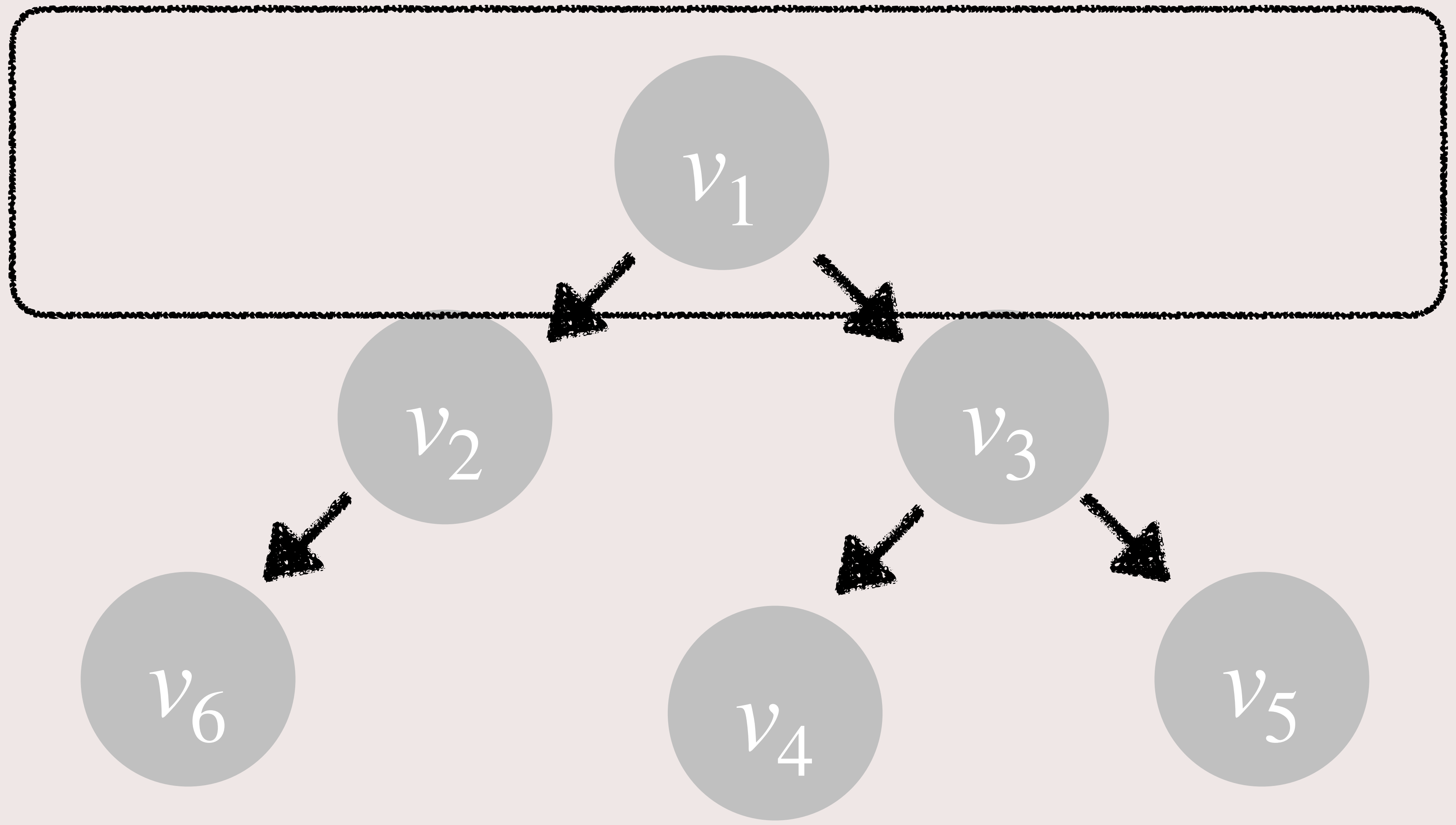
Written in parallel

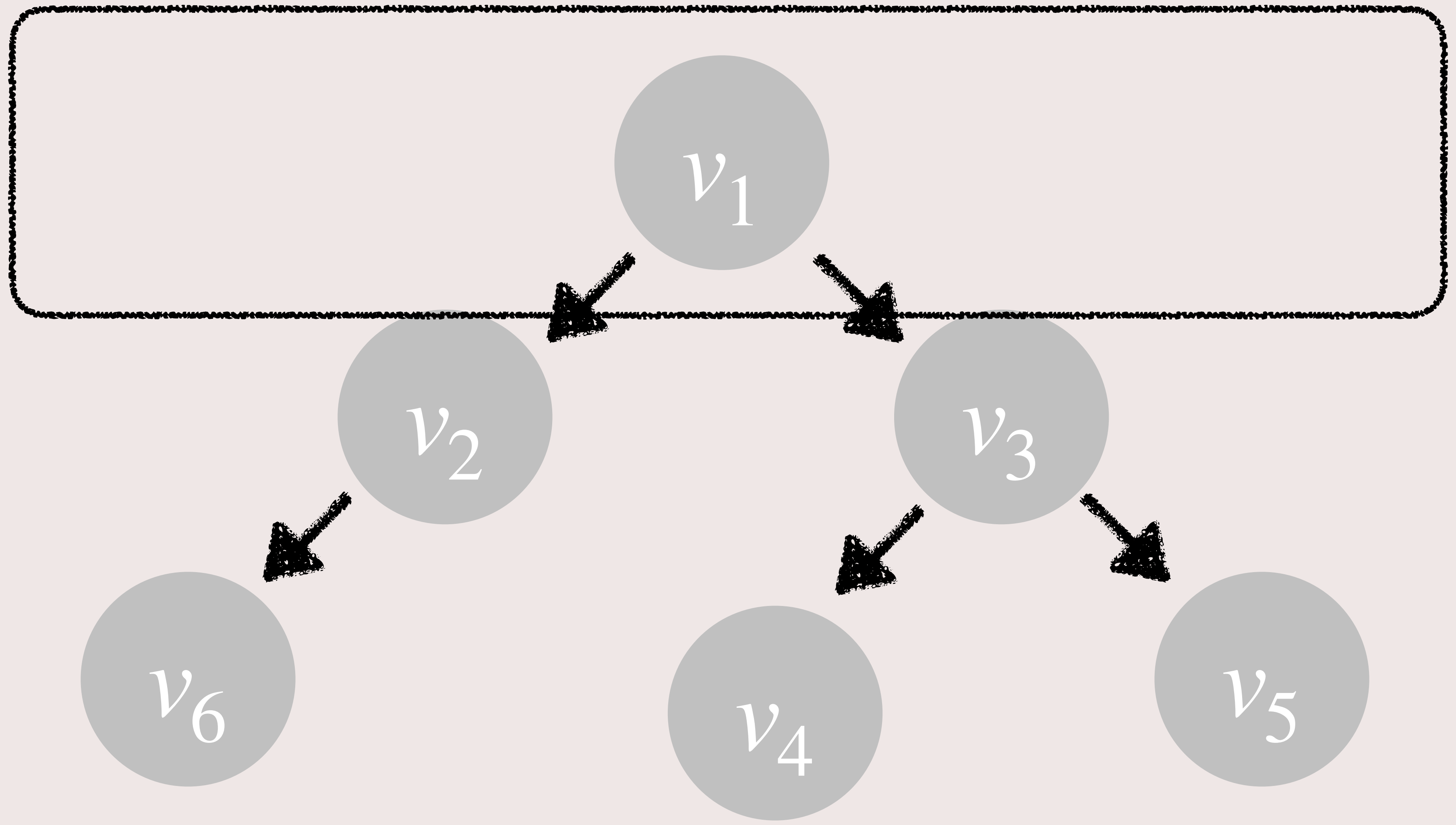


Written in parallel



Written in parallel





Written in parallel

Stabilization Complexity

logarithmic in number of values

- What are Multi-Maps (MMs)
- What are Encrypted Multi-Maps (EMMs)
- **How to store EMMs on Blockchains**
  - LSX
  - TRX
  - PAX
- Real World Deployment

OUTLINE

LSX

A list-based EMM

LSX

ℓ_1	(v_1, v_2)
ℓ_2	(v_1, v_2, v_3)

LSX

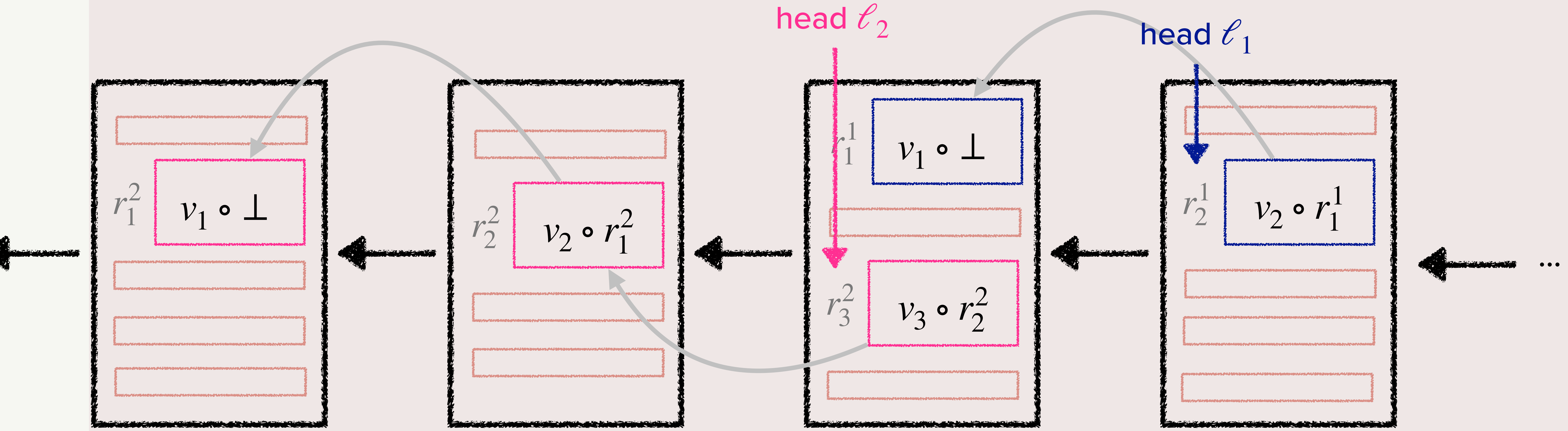
- Represent each label as separate list on BC

ℓ_1	(v_1, v_2)
ℓ_2	(v_1, v_2, v_3)

LSX

- Represent each label as separate list on BC

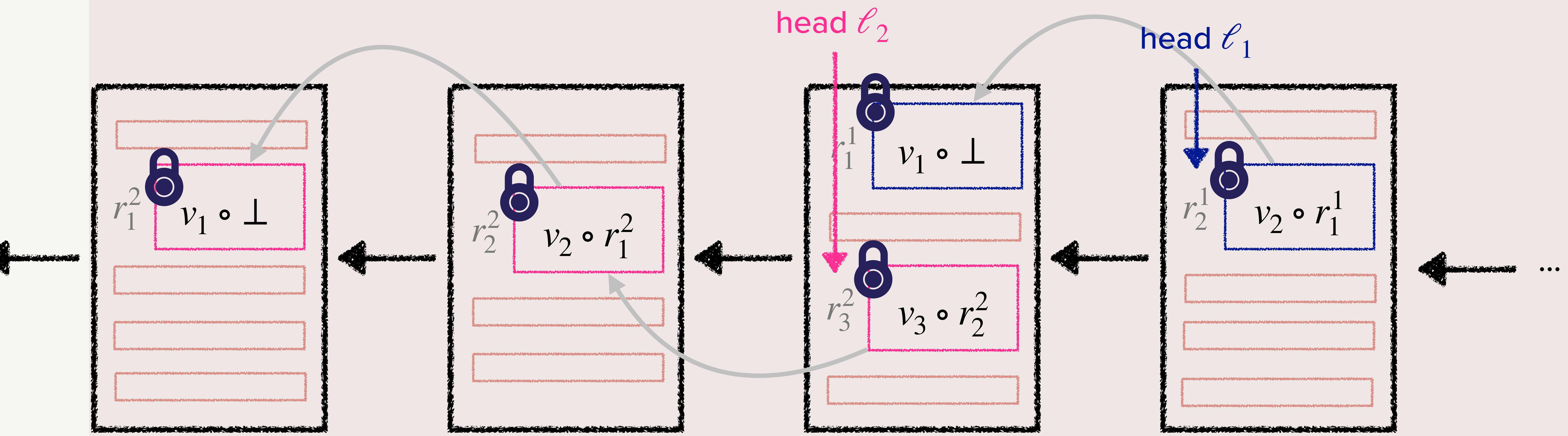
ℓ_1	(v_1, v_2)
ℓ_2	(v_1, v_2, v_3)



LSX

- Represent each label as separate list on BC
- Encrypt each value before storing on BC

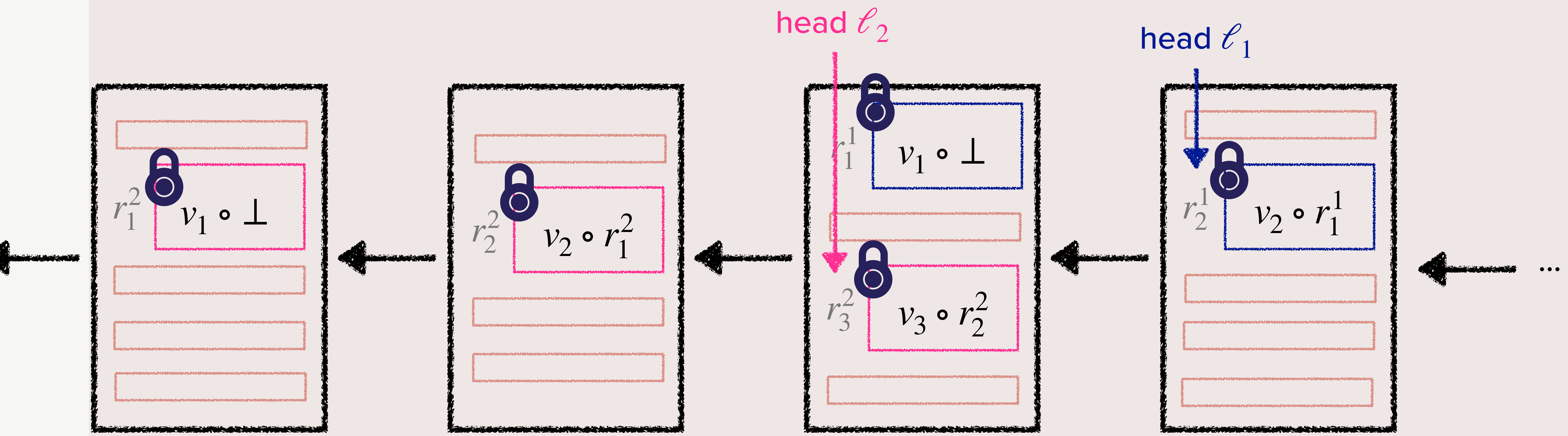
ℓ_1	(v_1, v_2)
ℓ_2	(v_1, v_2, v_3)



LSX

- Represent each label as separate list on BC
- Encrypt each value before storing on BC

ℓ_1	(v_1, v_2)
ℓ_2	(v_1, v_2, v_3)

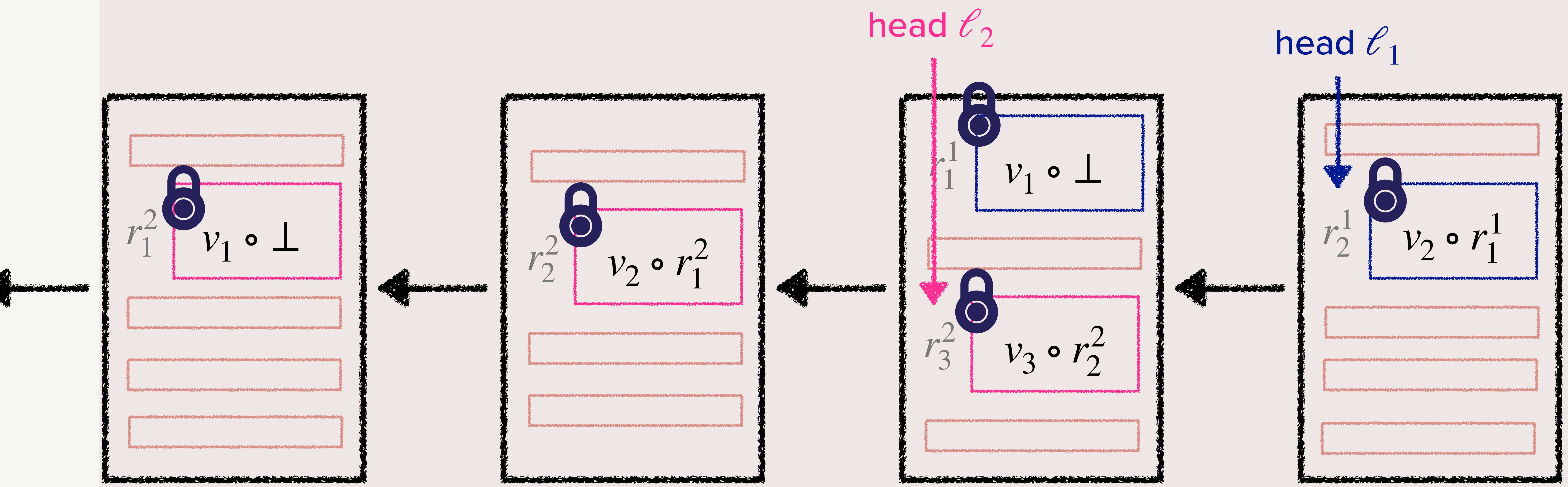


LSX

$\text{Add}(\ell_2, v_4)$

$\text{Delete}(\ell_1, v_1)$

$\text{Query}(\ell_1)$



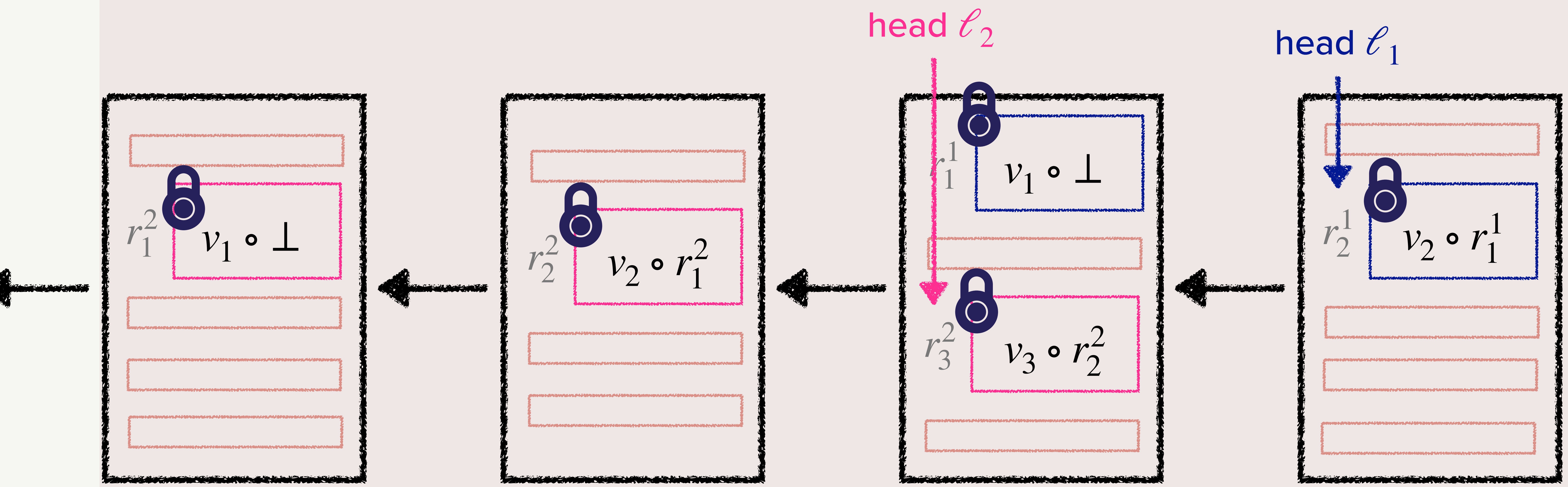
LSX

$\mathsf{Add}(\ell_2, v_4)$

- Append $\mathsf{Enc}(v_4, r_3^2)$ at head of ℓ_2
- Update the head

$\mathsf{Delete}(\ell_1, v_1)$

$\mathsf{Query}(\ell_1)$



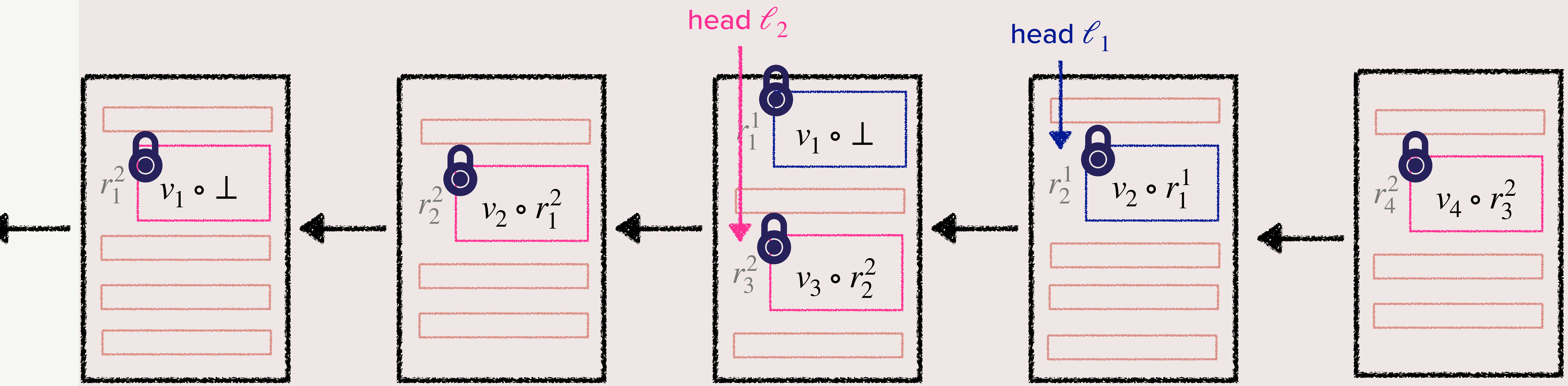
LSX

$\mathsf{Add}(\ell_2, v_4)$

$\mathsf{Delete}(\ell_1, v_1)$

$\mathsf{Query}(\ell_1)$

- Append $\mathsf{Enc}(v_4, r_3^2)$ at head of ℓ_2
- Update the head



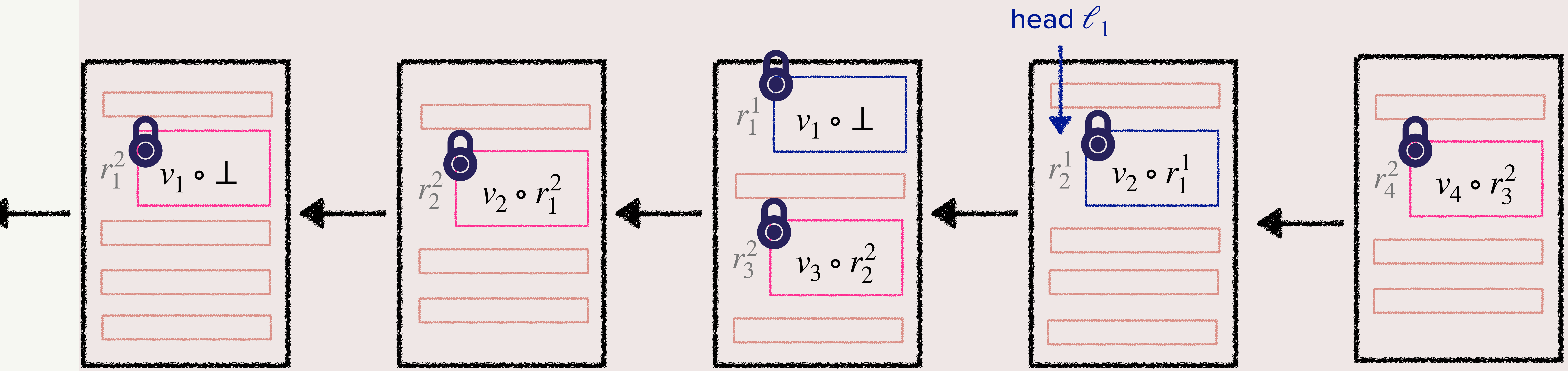
LSX

$\mathsf{Add}(\ell_2, v_4)$

- Append $\mathsf{Enc}(v_4, r_3^2)$ at head of ℓ_2
- Update the head

$\mathsf{Delete}(\ell_1, v_1)$

$\mathsf{Query}(\ell_1)$



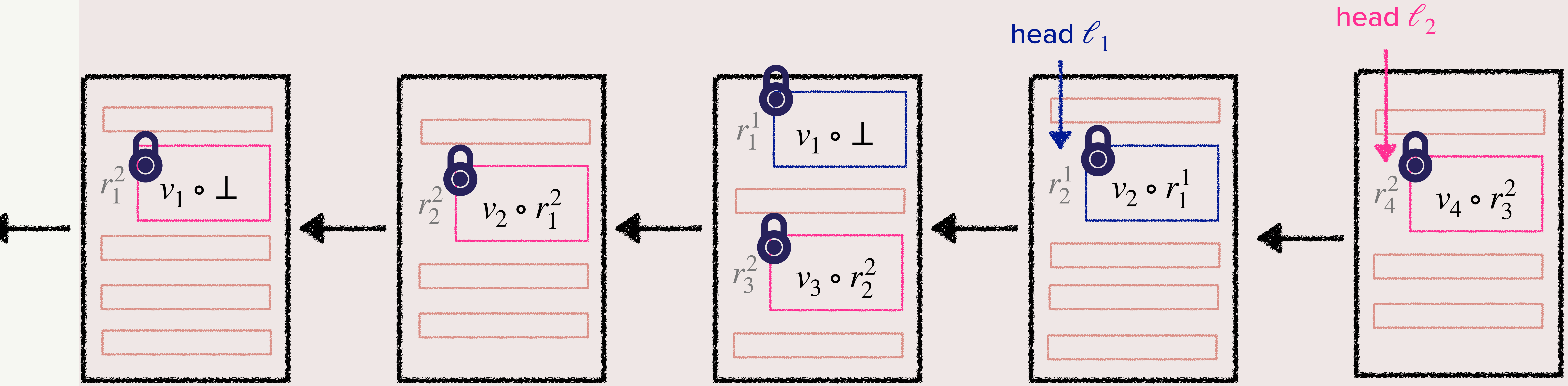
LSX

$\mathsf{Add}(\ell_2, v_4)$

$\mathsf{Delete}(\ell_1, v_1)$

$\mathsf{Query}(\ell_1)$

- Append $\mathsf{Enc}(v_4, r_3^2)$ at head of ℓ_2
- Update the head



LSX

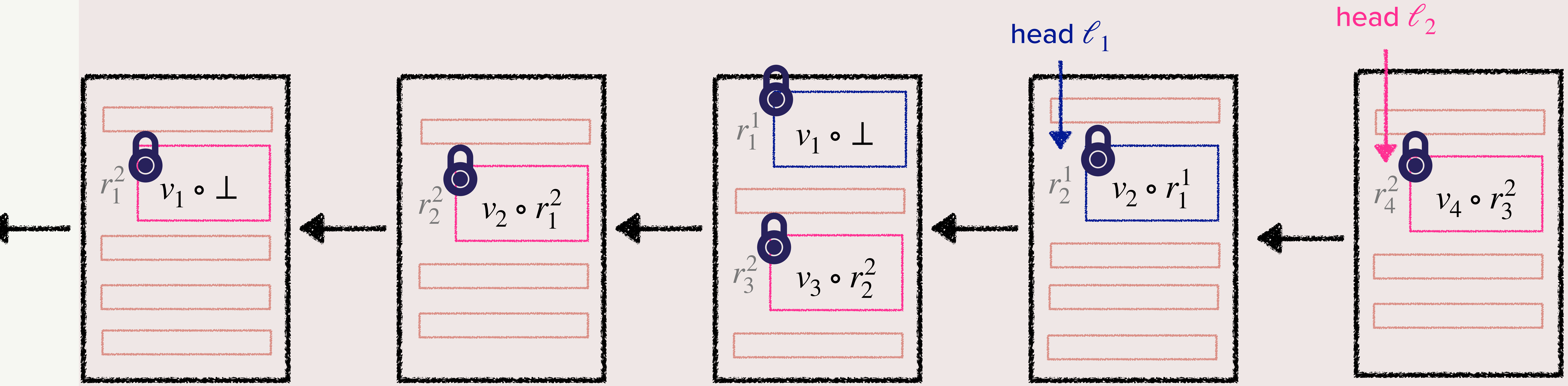
$\mathsf{Add}(\ell_2, v_4)$

- Append $\mathsf{Enc}(v_4, r_3^2)$ at head of ℓ_2
- Update the head

$\mathsf{Delete}(\ell_1, v_1)$

- Same as Add
- Append a `DEL` flag to the value

$\mathsf{Query}(\ell_1)$



LSX

$\mathsf{Add}(\ell_2, v_4)$

- Append $\mathsf{Enc}(v_4, r_3^2)$ at head of ℓ_2
- Update the head

$\mathsf{Delete}(\ell_1, v_1)$

- Same as Add
- Append a `DEL` flag to the value

$\mathsf{Query}(\ell_1)$

LSX

$\text{Add}(\ell_2, v_4)$

- Append $\text{Enc}(v_4, r_3^2)$ at head of ℓ_2
- Update the head

$\text{Delete}(\ell_1, v_1)$

- Same as Add
- Append a `DEL` flag to the value

$\text{Query}(\ell_1)$

LSX

Add(ℓ_2, v_4)

- Append Enc(v_4, r_3^2) at head of ℓ_2
- Update the head

Delete(ℓ_1, v_1)

- Same as Add
- Append a `DEL` flag to the value

Query(ℓ_1)

LSX

$\mathsf{Add}(\ell_2, v_4)$

- Append $\mathsf{Enc}(v_4, r_3^2)$ at head of ℓ_2
- Update the head

$\mathsf{Delete}(\ell_1, v_1)$

- Same as Add
- Append a `DEL` flag to the value

$\mathsf{Query}(\ell_1)$

- Read all values from the list
- Output un-deleted values

LSX

Add(ℓ_2, v_4)

- Append Enc(v_4, r_3^2) at head of ℓ_2
- Update the head

time `Optimal:` $\mathbf{O(|t|)}$

Delete(ℓ_1, v_1)

- Same as Add
- Append a `DEL` flag to the value

time `Optimal:` $\mathbf{O(|t|)}$

Query(ℓ_1)

- Read all values from the list
- Output un-deleted values

LSX

$\mathsf{Add}(\ell_2, v_4)$

- Append $\mathsf{Enc}(v_4, r_3^2)$ at head of ℓ_2
- Update the head

time

`Optimal:` $\mathbf{O(|t|)}$

$\mathsf{Delete}(\ell_1, v_1)$

- Same as Add
- Append a `DEL` flag to the value

time

`Optimal:` $\mathbf{O(|t|)}$

$\mathsf{Query}(\ell_1)$

- Read all values from the list
- Output un-deleted values

time

`# values ever added/ deleted`

LSX

Add(ℓ_2, v_4)

- Append Enc(v_4, r_3^2) at head of ℓ_2
- Update the head

time `Optimal:` $\mathbf{O(|t|)}$

stabilization $\mathbf{O(|t|)}$

Delete(ℓ_1, v_1)

- Same as Add
- Append a `DEL` flag to the value

time `Optimal:` $\mathbf{O(|t|)}$

stabilization $\mathbf{O(|t|)}$

Query(ℓ_1)

- Read all values from the list
- Output un-deleted values

time `# values ever added/ deleted`

LSX

Add(ℓ_2, v_4)

- Append Enc(v_4, r_3^2) at head of ℓ_2
- Update the head

time Optimal: $\mathbf{O(|t|)}$

stabilization $\mathbf{O(|t|)}$

Delete(ℓ_1, v_1)

- Same as Add
- Append a `DEL` flag to the value

time Optimal: $\mathbf{O(|t|)}$

stabilization $\mathbf{O(|t|)}$

Query(ℓ_1)

- Read all values from the list
- Output un-deleted values

time # values ever added/ deleted

TRX

A tree-based EMM

time `Optimal:` $O(|t|)$

stabilization $\boxed{\cancel{O(|t|)}}$

$O(\log|t|)$

time `Optimal:` $O(|t|)$

stabilization $\boxed{\cancel{O(|t|)}}$

$O(\log|t|)$

time $\boxed{\texttt{\# values ever added/ deleted}}$

TRX

A tree-based EMM

time Optimal: **O(|t|)**

stabilization ~~**O(|t|)**~~ **O(log|t|)**

time Optimal: **O(|t|)**

stabilization ~~**O(|t|)**~~ **O(log|t|)**

time # values ever added/ deleted

PAX

A patched EMM

Add(ℓ_2, v_4)

time

Optimal: **O(|t|)**

stabilization

O(|t|)

Delete(ℓ_1, v_1)

time

Optimal: **O(|t|)**

stabilization

O(|t|)

Query(ℓ_1)

time

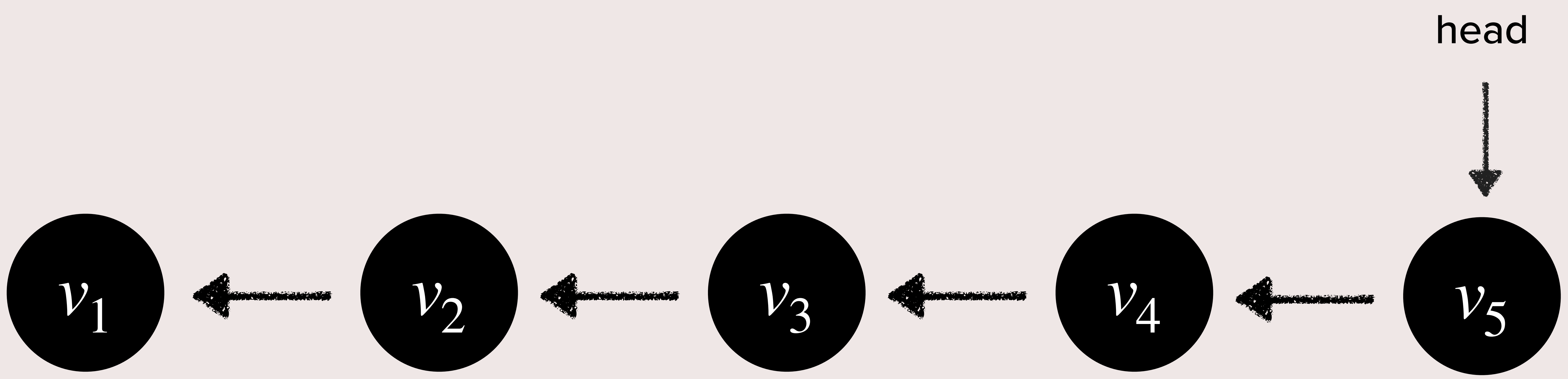
values ever added/ deleted

PAX

Query(ℓ_1)

time

values ever added/ deleted

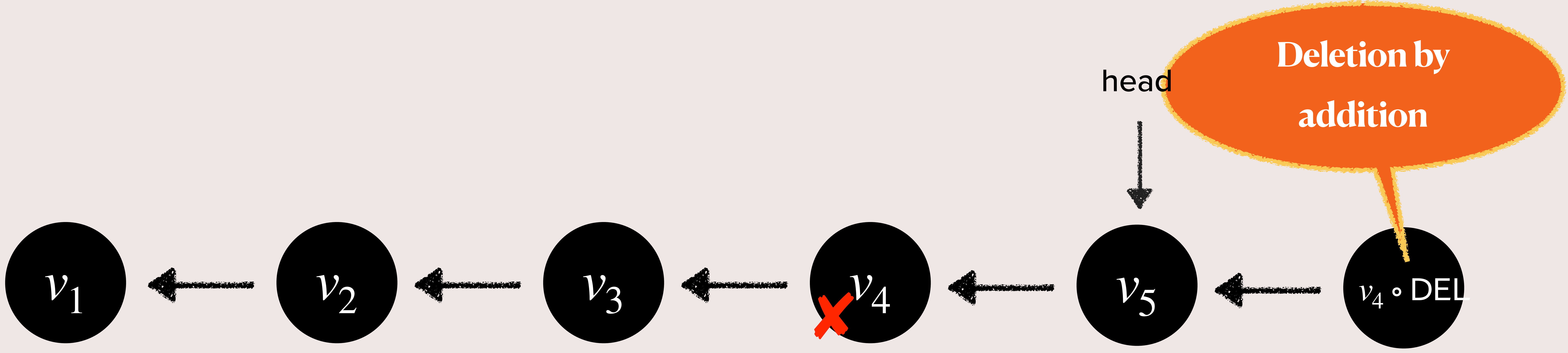


PAX

Query(ℓ_1)

time

values ever added/ deleted

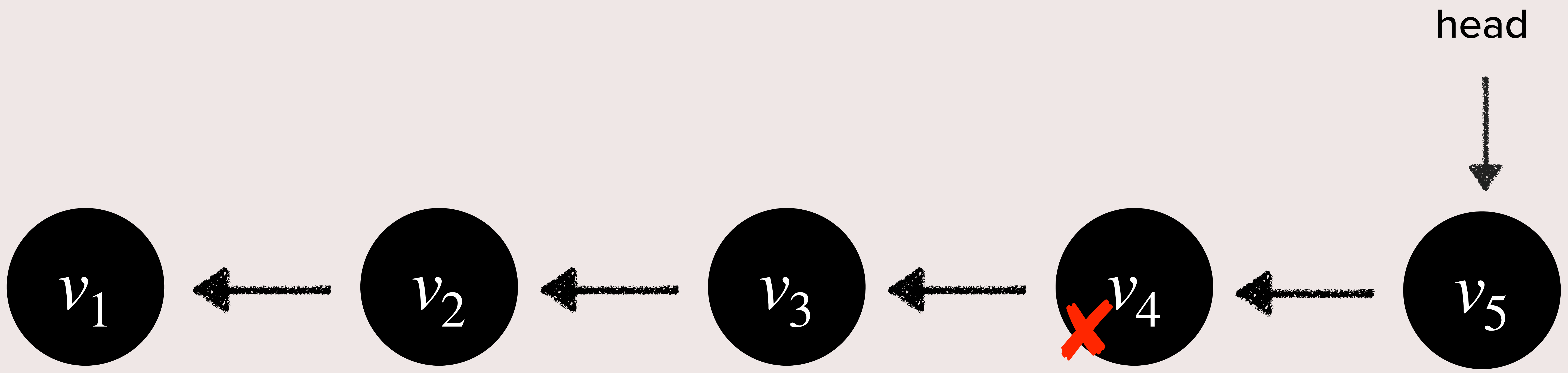


PAX

Query(ℓ_1)

time

values ever added/ deleted

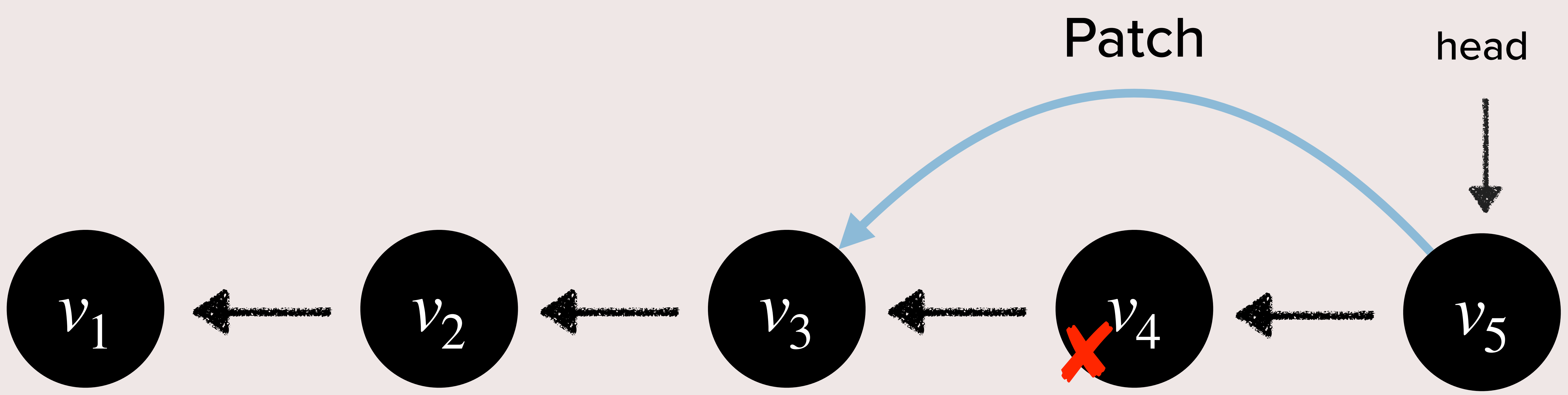


PAX

Query(ℓ_1)

time

`# values ever added/ deleted`

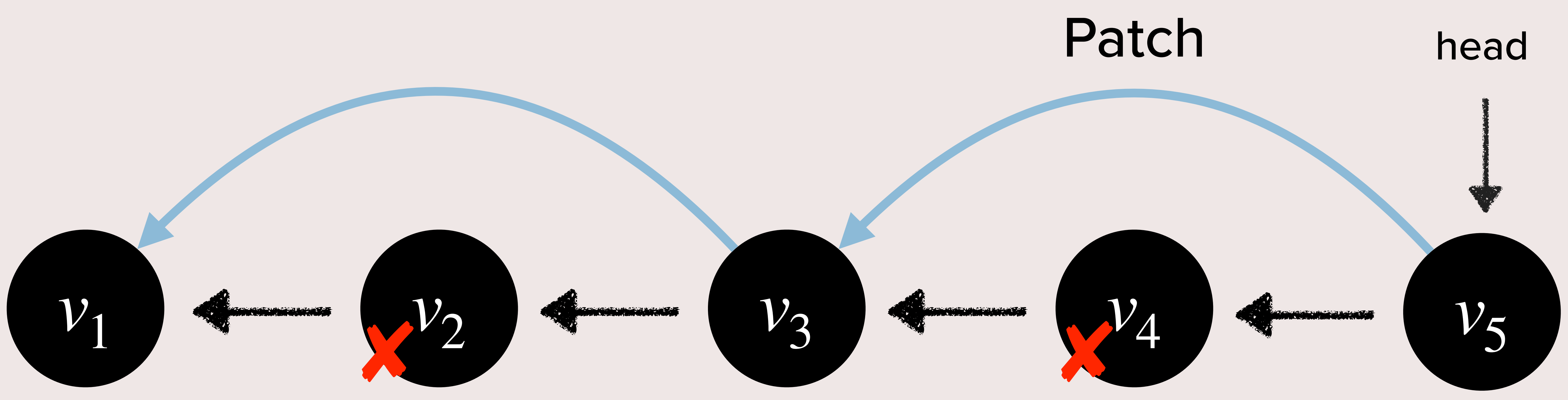


PAX

Query(ℓ_1)

time

values ever added/ deleted

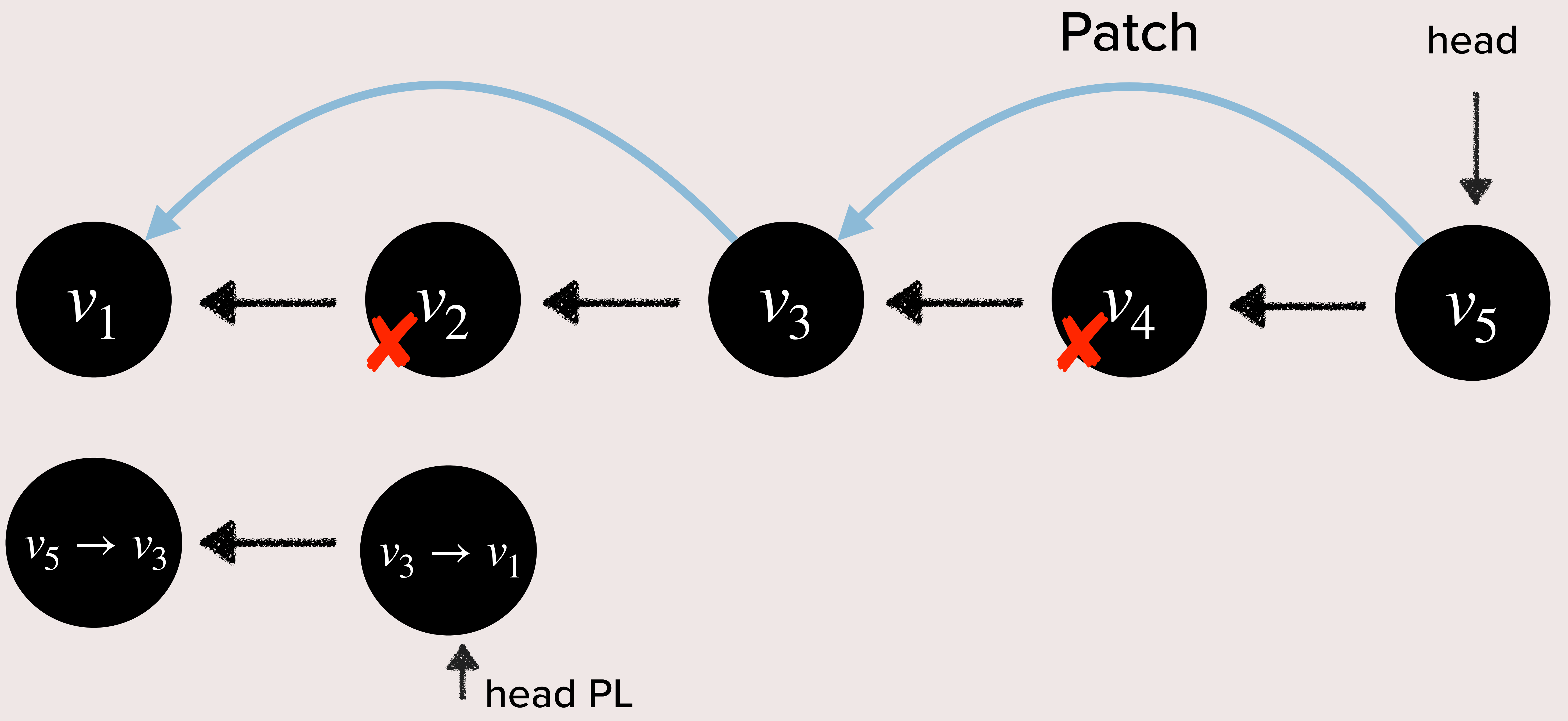


PAX

$\text{Query}(\ell_1)$

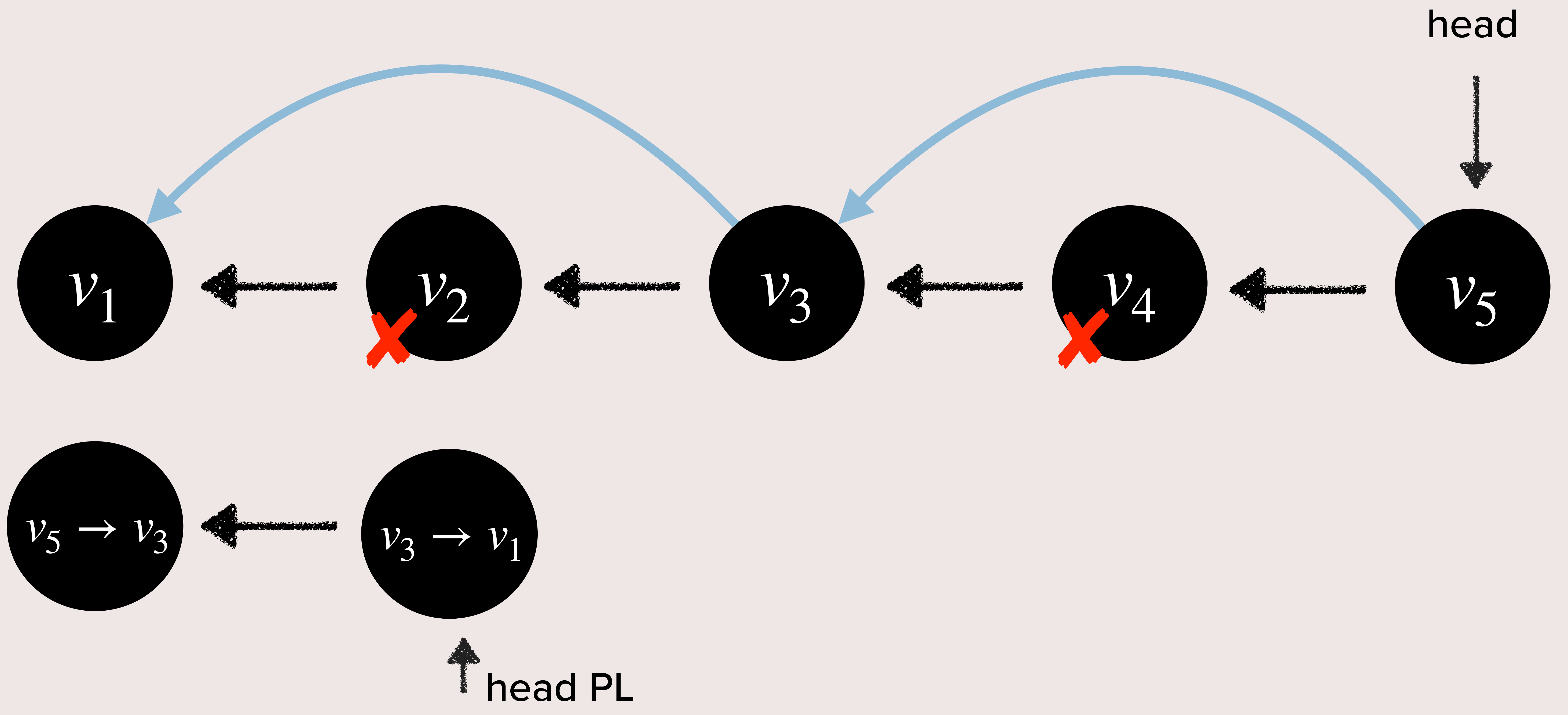
time

values ever added/ deleted



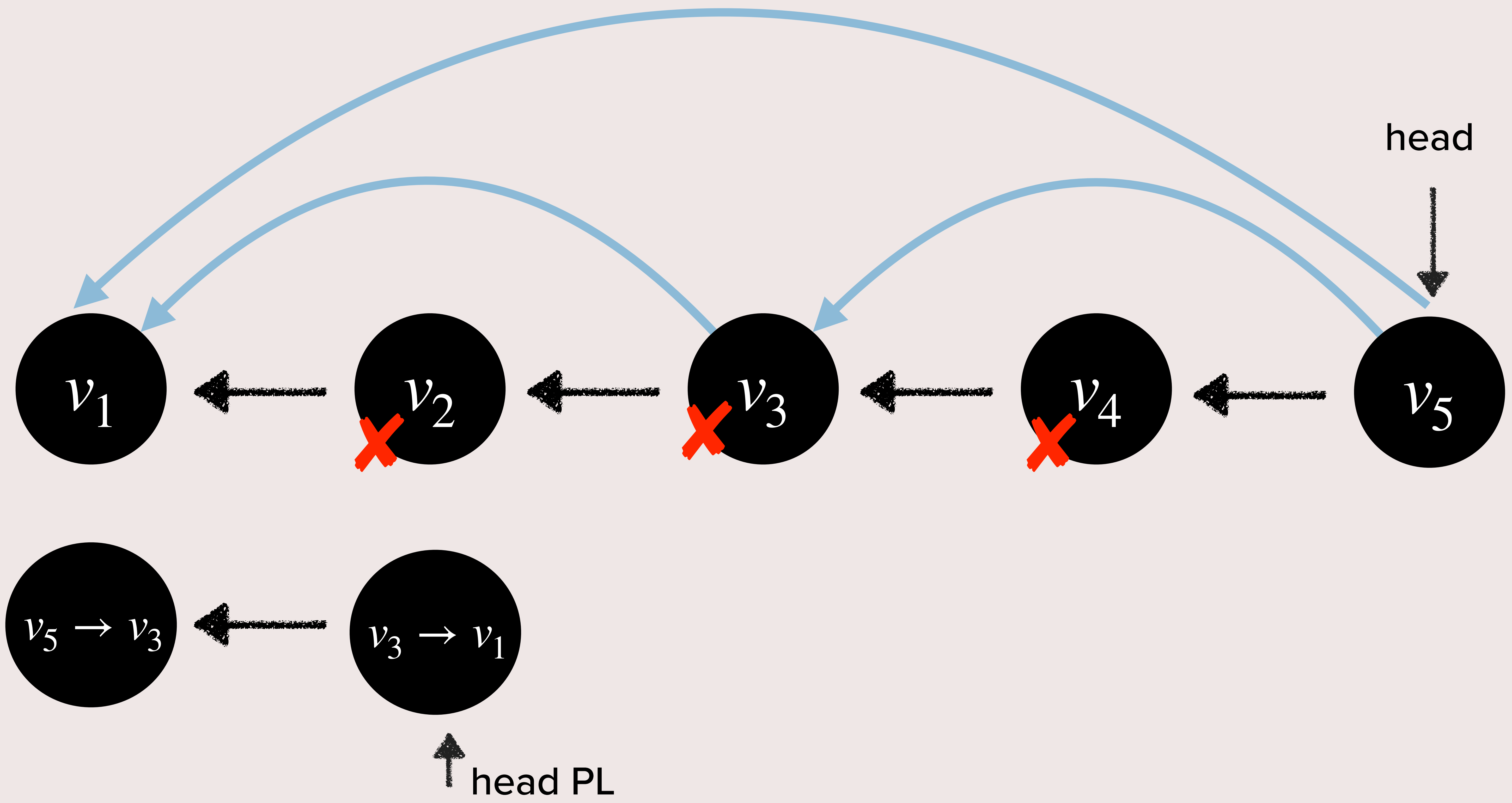
PAX

Let's delete v_3



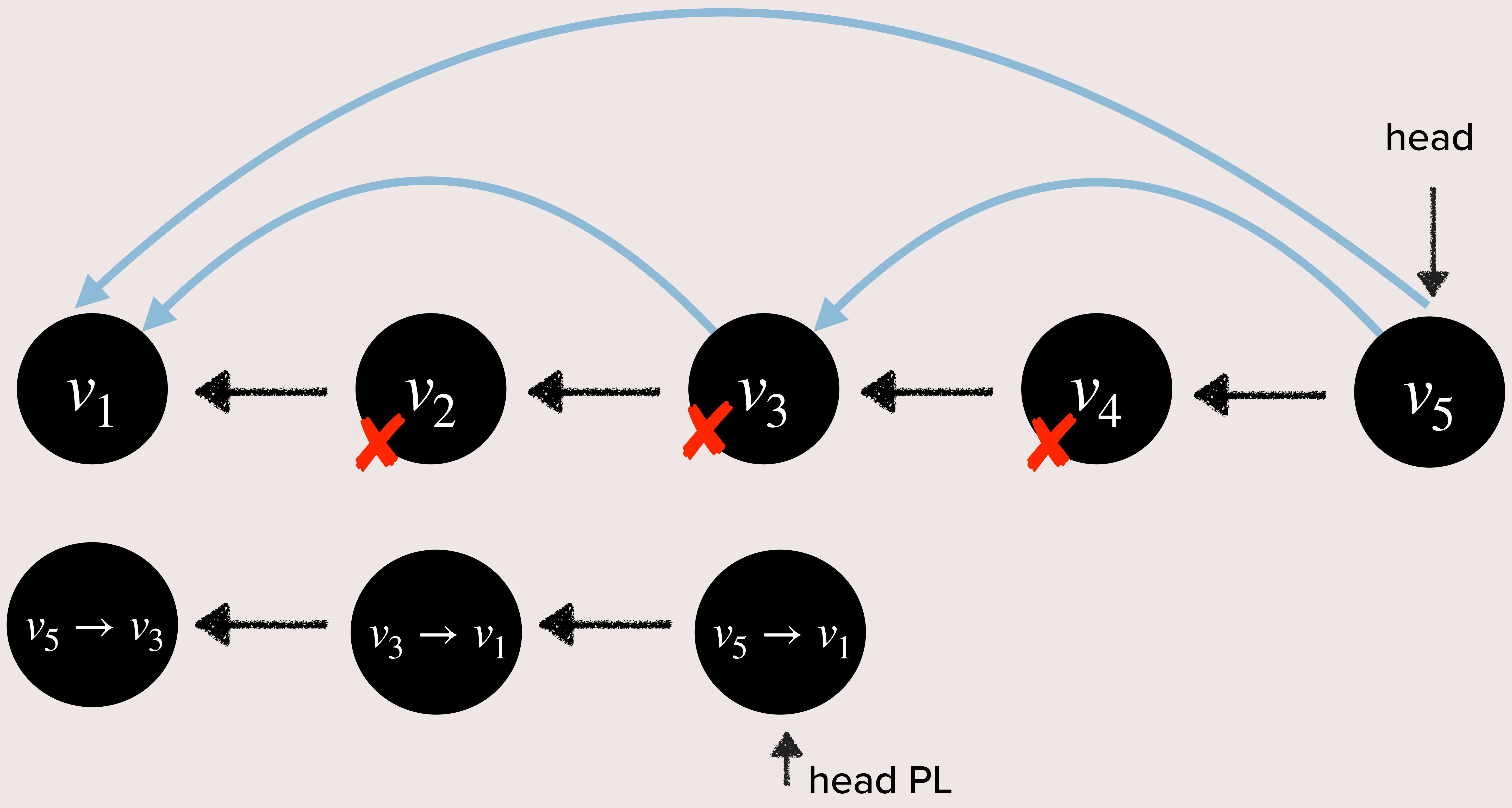
PAX

Let's delete v_3



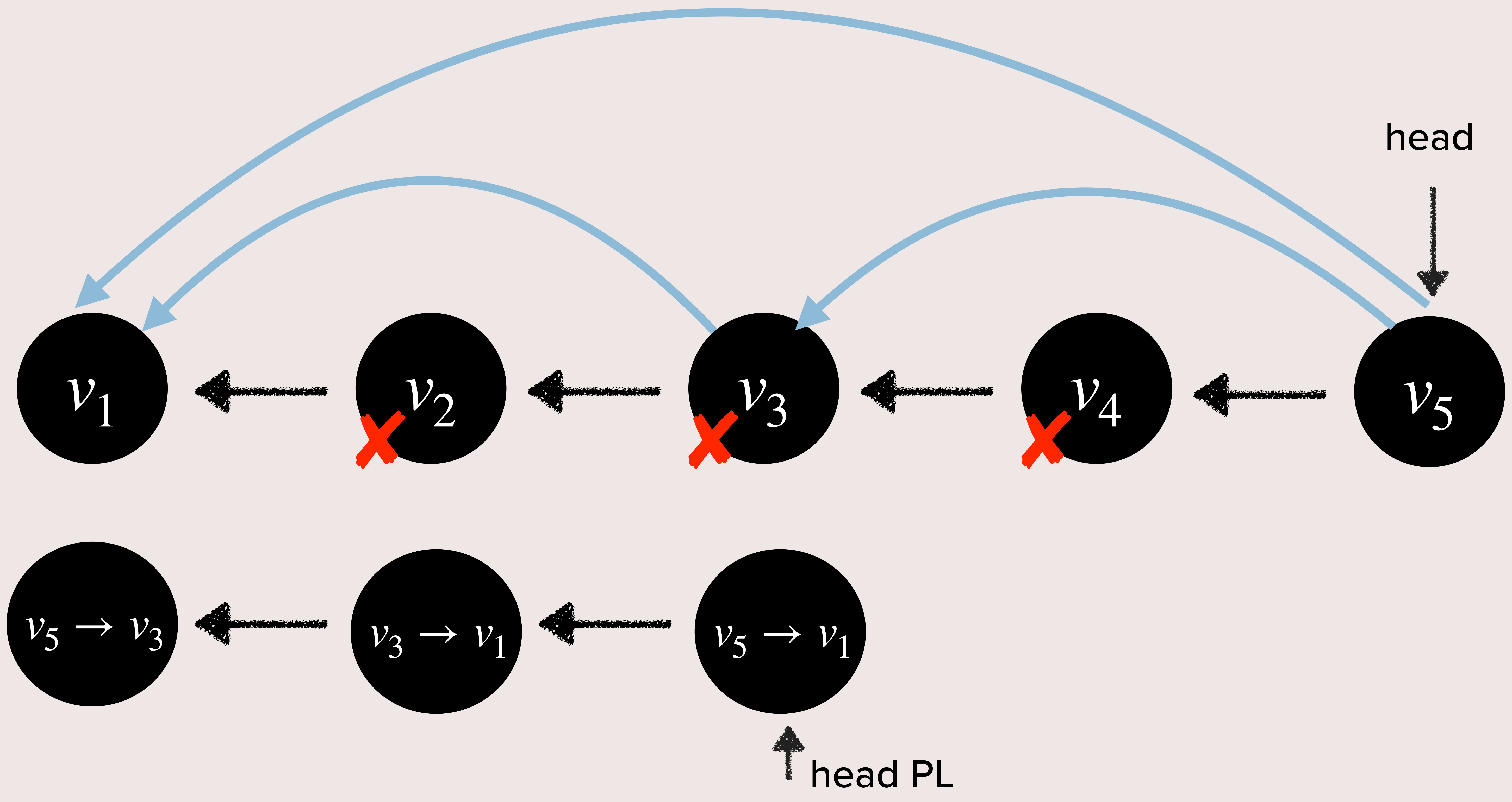
PAX

Let's delete v_3



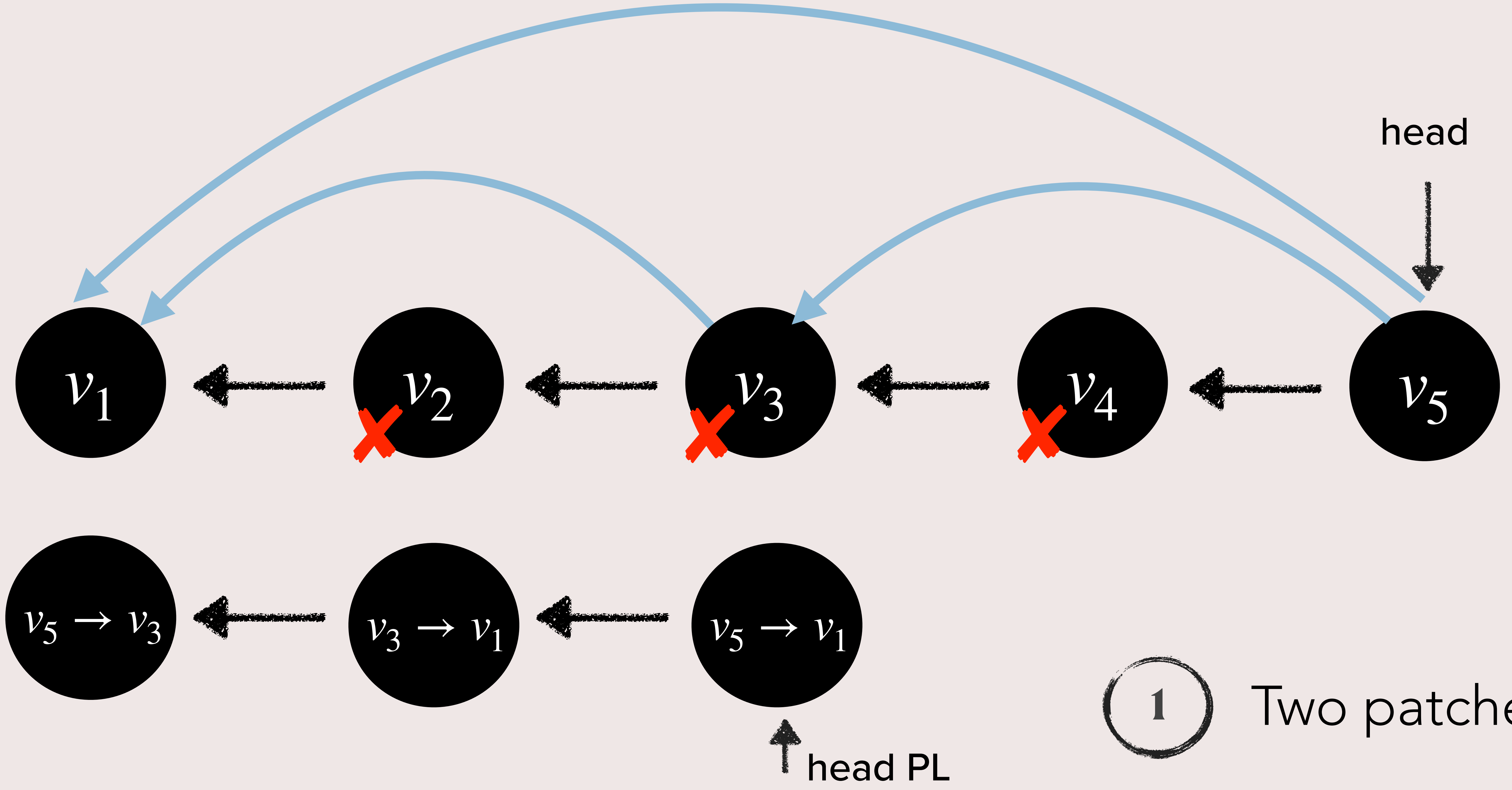
PAX

Let's delete v_3



PAX

Let's delete v_3

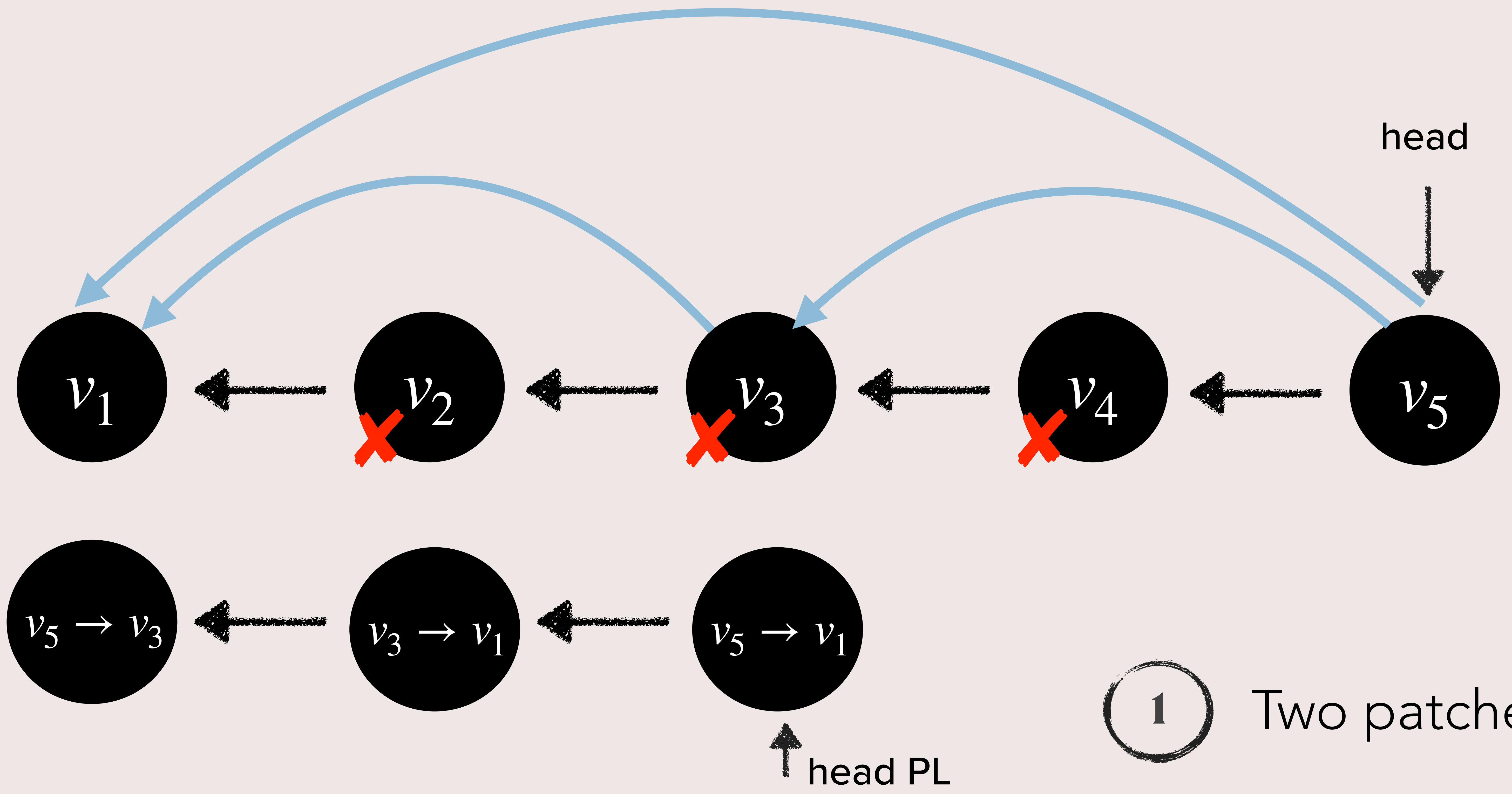


1 Two patches from v_5

Bad for correctness

PAX

Let's delete v_3

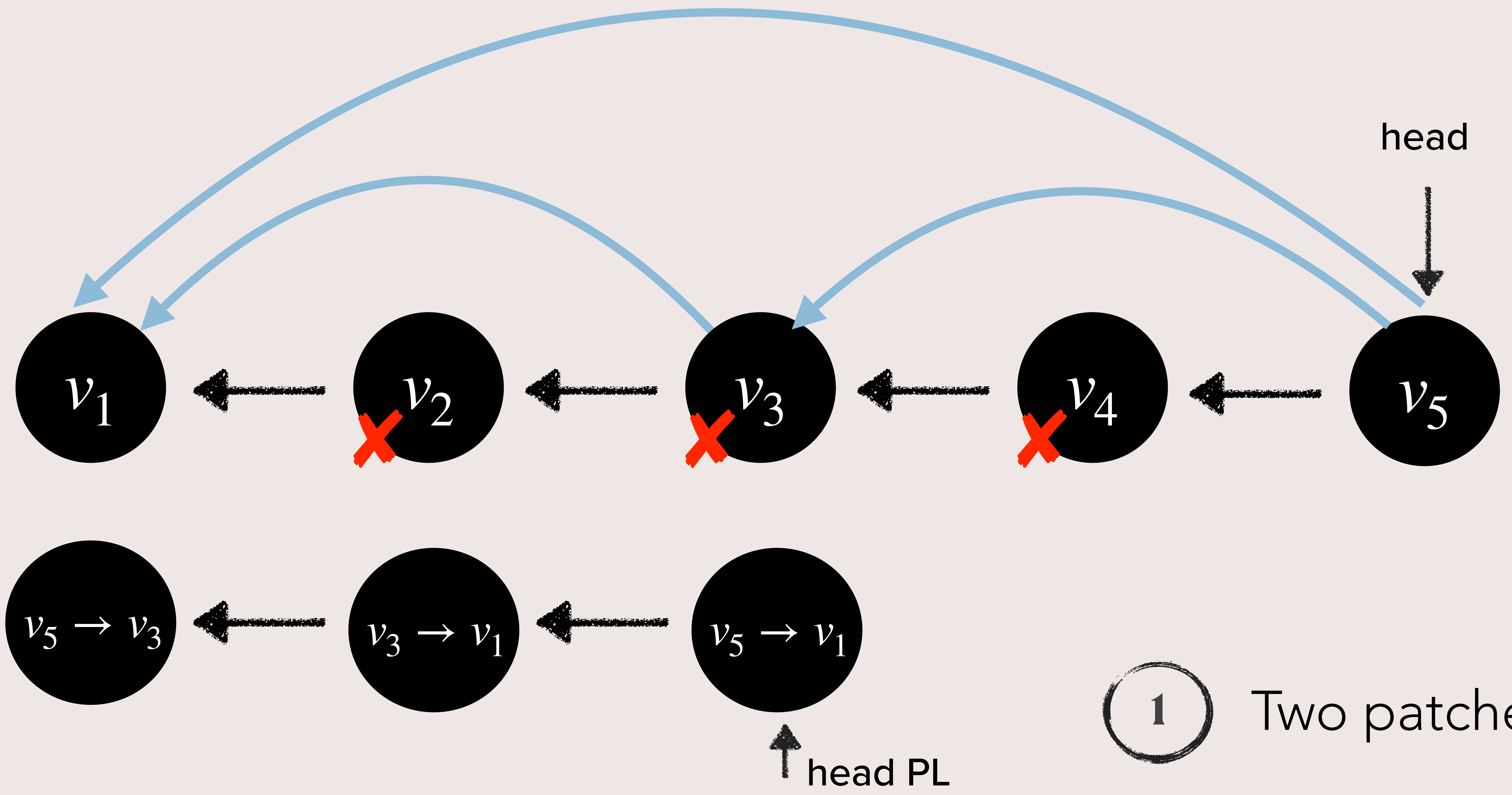


1 Two patches from v_5

Bad for correctness

PAX

Let's delete v_3



1. Two patches from v_5

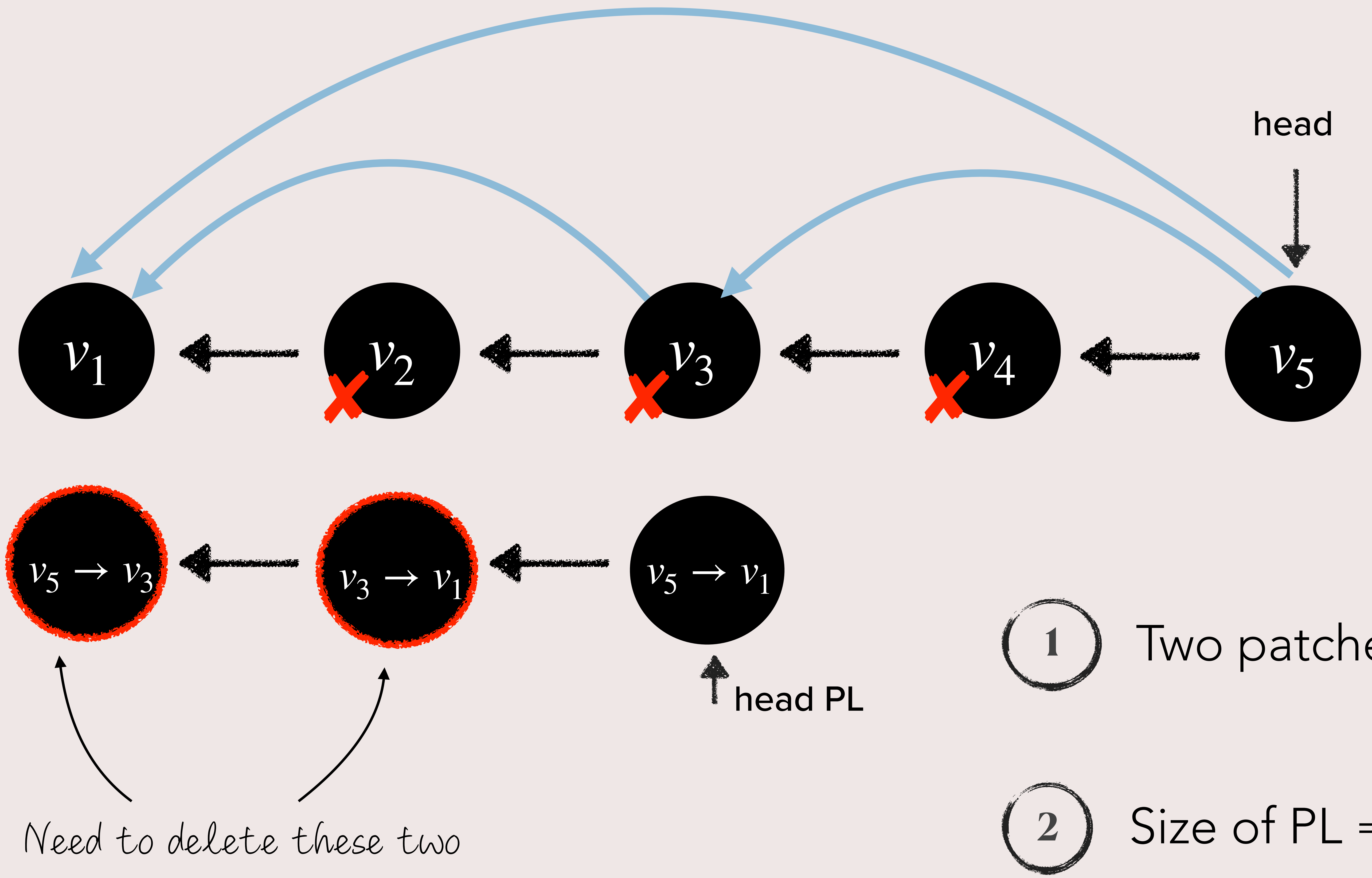
   Bad for correctness

2. Size of PL = # values deleted

   Bad for efficiency

PAX

Let's delete v_3



1. Two patches from v_5

Bad for correctness

2. Size of PL = # values deleted

Bad for efficiency

PAX

Let's delete v_3

1. Two patches from v_5

Bad for correctness

2. Size of PL = # v … s … ted

Bad for efficiency

PAX

Let's delete v_3

1. Two patches from v_5

Bad for correctness

2. Size of PL = # v[illegible]s [illegible]ted

Bad for efficiency

PAX

Copy On Write

Let's delete v_3

head

$v_5 \rightarrow v_3$ $v_3 \rightarrow v_1$ $v_5 \rightarrow v_1$

head PL

Need to delete these two

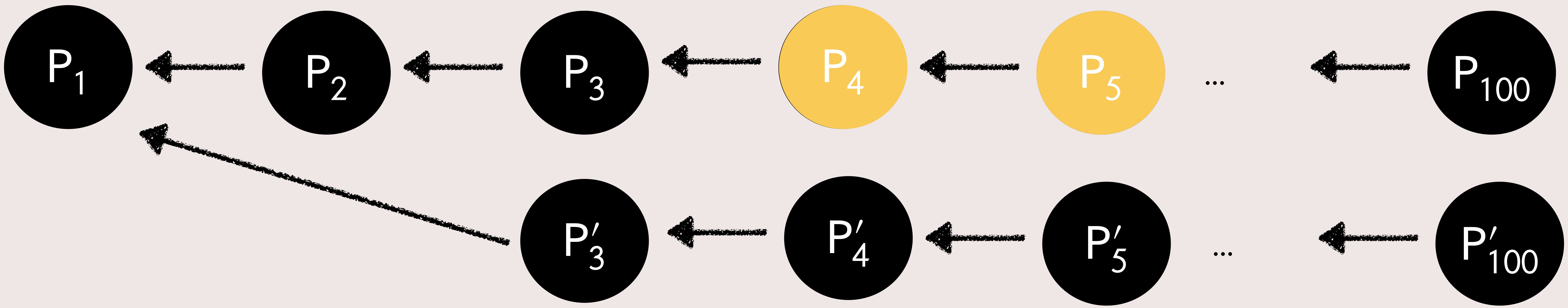
1. Two patches from v_5

Bad for correctness

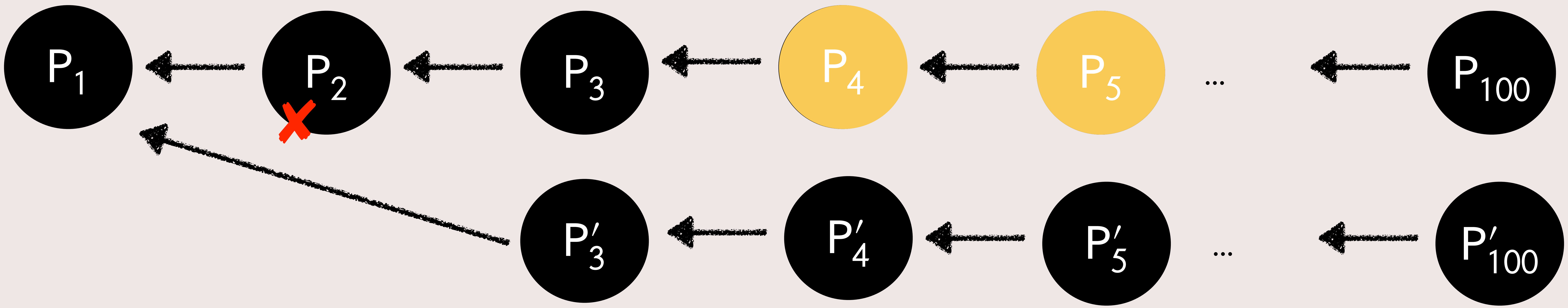
2. Size of PL = # v... ...ted

Bad for efficiency

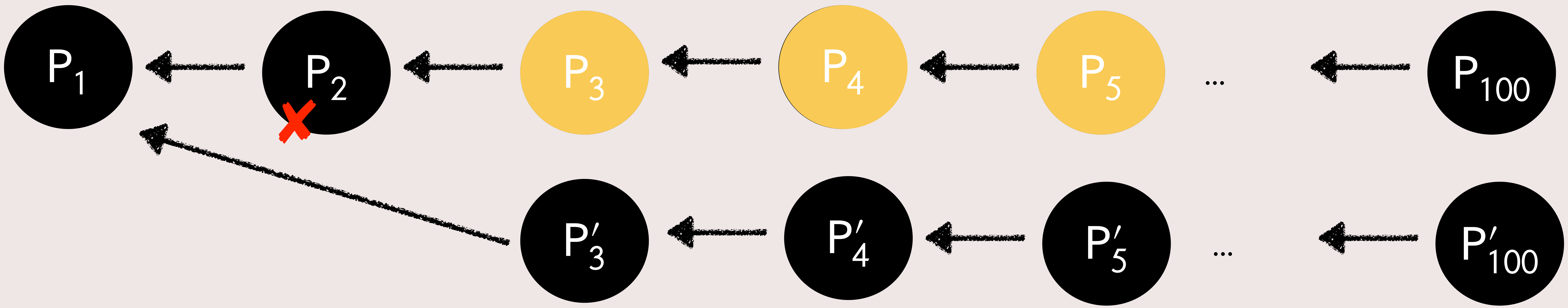
PAX



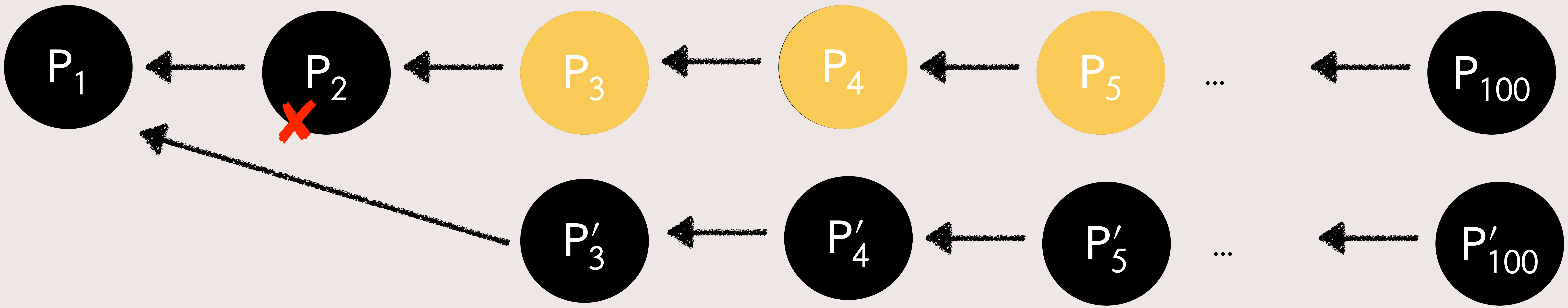
PAX



PAX

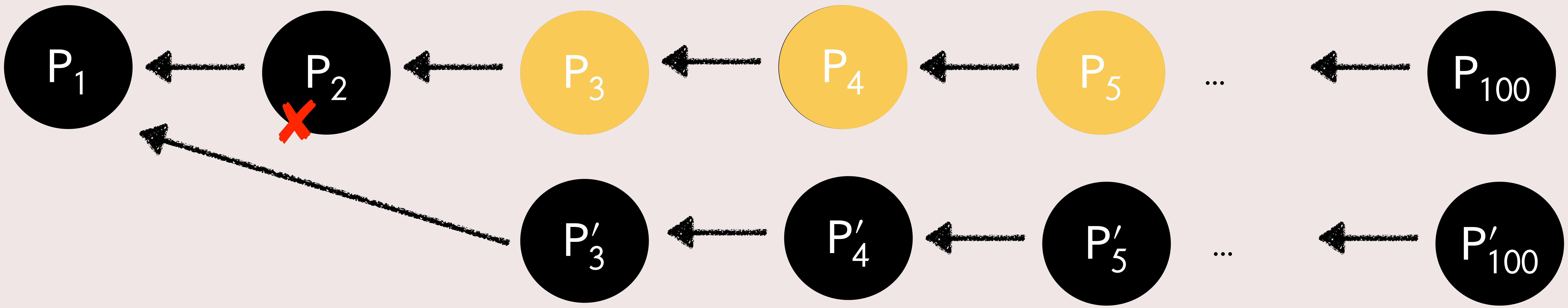


PAX

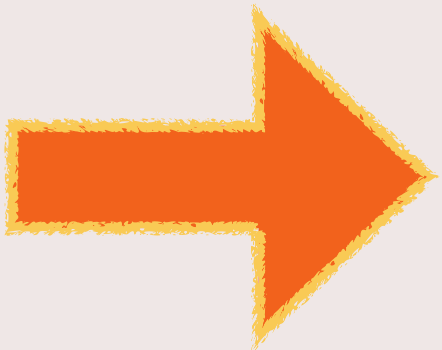


Expensive!

PAX

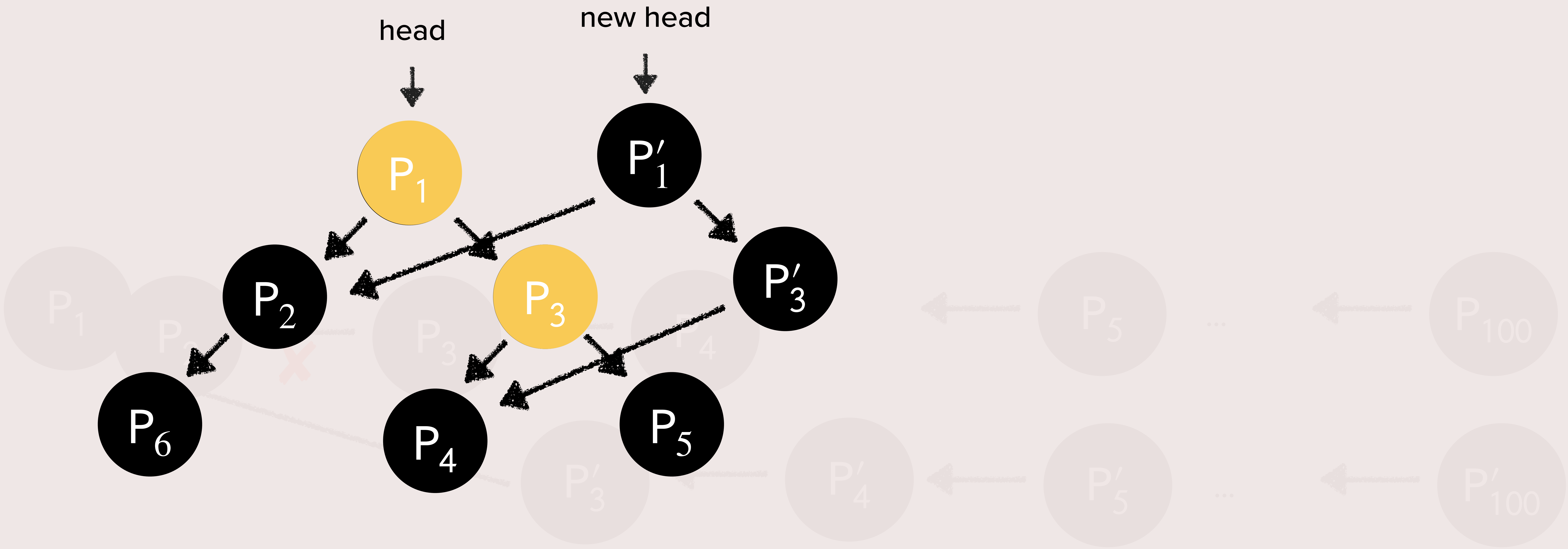


Expensive!



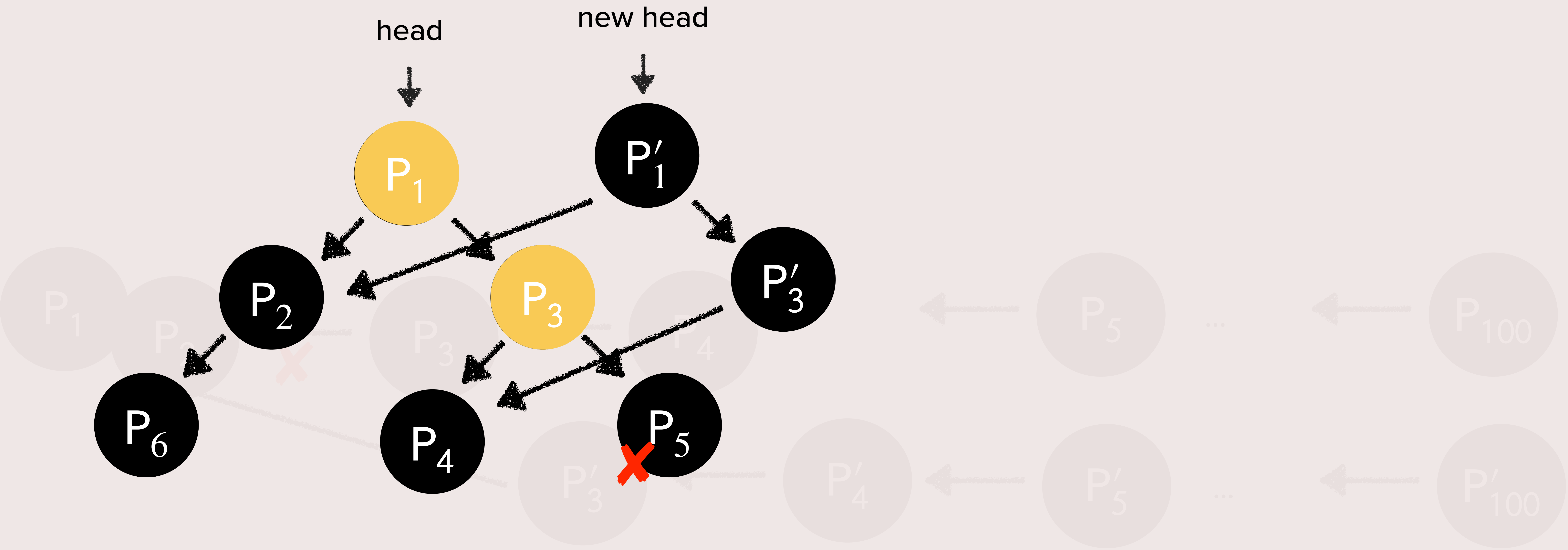
Organize patches in a BST

PAX



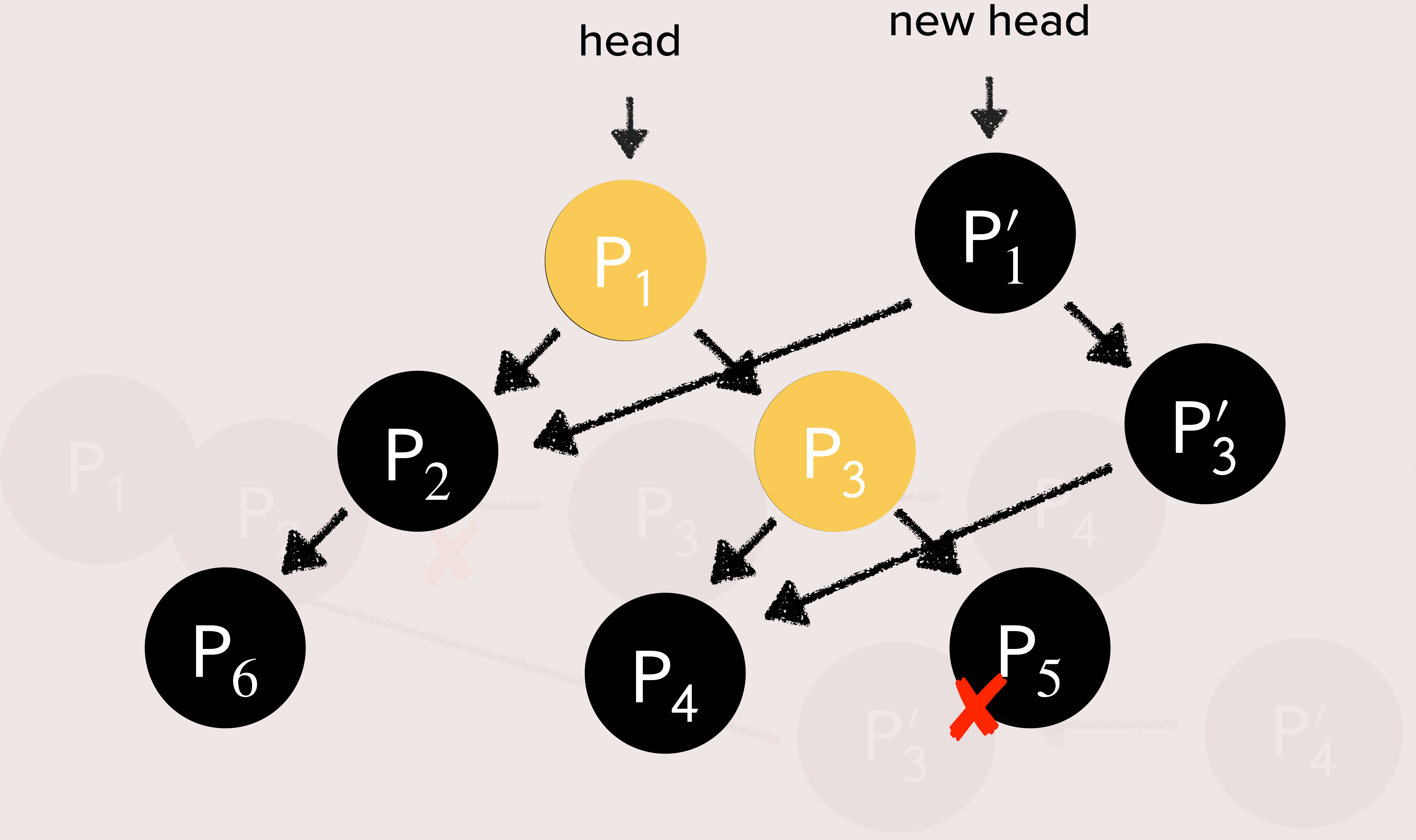
Expensive! → Organize patches in a BST

PAX



Expensive! → Organize patches in a BST

PAX



Show

nodes in BST

≤ # values in multi-map

Expensive! → Organize patches in a BST

PAX

	time	stabilization	time LSX	stabilization LSX
Add			$O(\lvert \mathbf{t} \rvert)$	$O(\lvert \mathbf{t} \rvert)$
Delete			$O(\lvert \mathbf{t} \rvert)$	$O(\lvert \mathbf{t} \rvert)$
Query			$O($ # values ever added/deleted)	

PAX

	time	stabilization	time LSX	stabilization LSX
Add	$O(\|\mathbf{t}\|)$		$O(\|\mathbf{t}\|)$	$O(\|\mathbf{t}\|)$
Delete			$O(\|\mathbf{t}\|)$	$O(\|\mathbf{t}\|)$
Query			O(# values ever added/deleted)	

PAX

	time	stabilization	time LSX	stabilization LSX
Add	$O(\|\mathbf{t}\|)$		$O(\|\mathbf{t}\|)$	$O(\|\mathbf{t}\|)$
Delete	$O(\|\mathbf{t}\| \log$ (size of MM))		$O(\|\mathbf{t}\|)$	$O(\|\mathbf{t}\|)$
Query			$O($ # values ever added/deleted)	

PAX

	time	stabilization	time LSX	stabilization LSX
Add	$O(\|\mathbf{t}\|)$		$O(\|\mathbf{t}\|)$	$O(\|\mathbf{t}\|)$
Delete	$O(\|\mathbf{t}\| \log$ (size of MM))		$O(\|\mathbf{t}\|)$	$O(\|\mathbf{t}\|)$
Query	O(size of MM)		O(# values ever added/deleted)	

PAX

	time	stabilization	time LSX	stabilization LSX
Add	$O(\lvert \mathbf{t} \rvert)$	$O(\lvert \mathbf{t} \rvert)$	$O(\lvert \mathbf{t} \rvert)$	$O(\lvert \mathbf{t} \rvert)$
Delete	$O(\lvert \mathbf{t} \rvert \log (\text{size of MM}))$	$O(\lvert \mathbf{t} \rvert)$	$O(\lvert \mathbf{t} \rvert)$	$O(\lvert \mathbf{t} \rvert)$
Query	$O(\text{size of MM})$		$O(\text{\# values ever added/deleted})$	

- What are Multi-Maps (MMs)
- What are Encrypted Multi-Maps (EMMs)
- How to store EMMs on Blockchains
  - LSX
  - TRX
  - PAX
- **Real World Deployment**

OUTLINE

Real World Deployments

Algorand

Ethereum

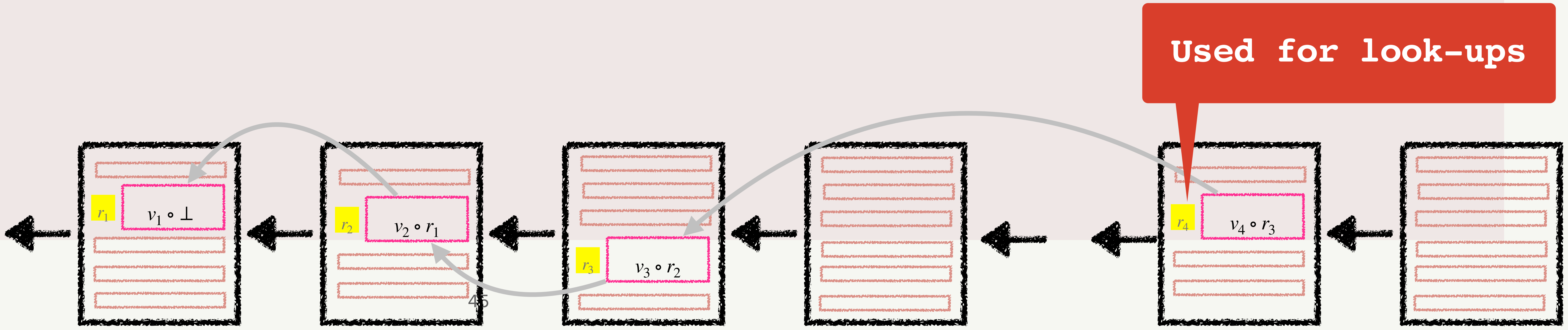
Real World Deployments

Algorand

Ethereum

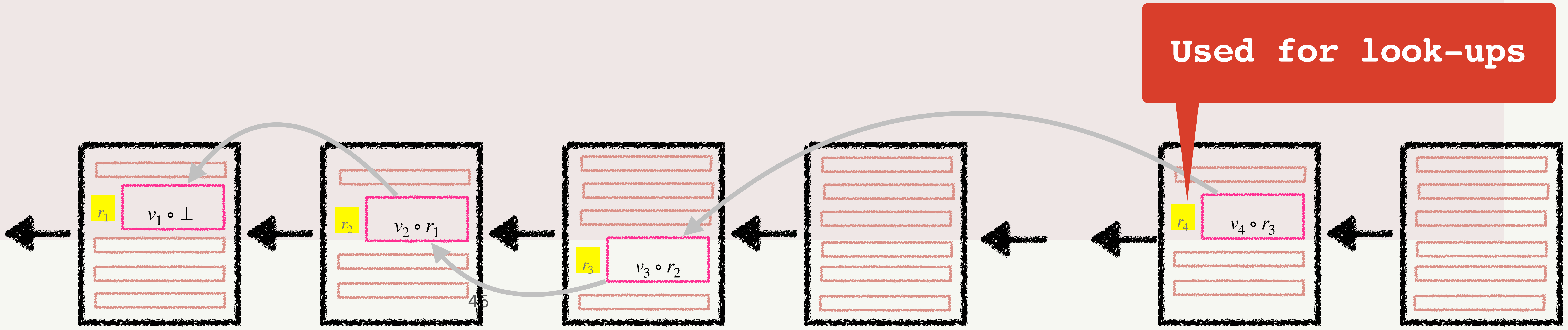
Some things to think about ...

① What should we use as addresses?



① What should we use as addresses?

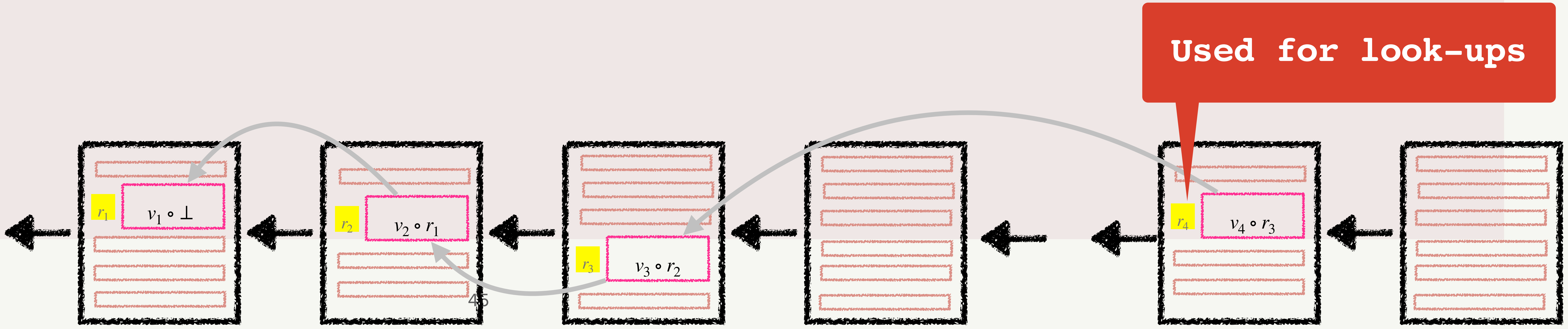
TX hashes



1 What should we use as addresses?

TX hashes

- All writes can be sent in parallel
- LSX = TRX
- Stabilization complexity = $O(1)$



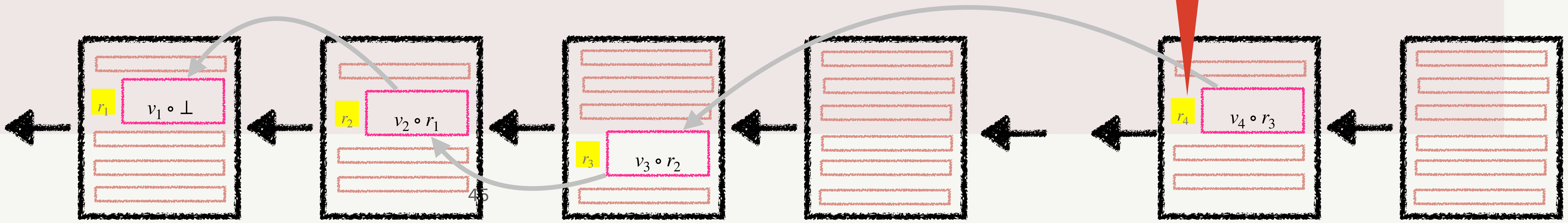
① What should we use as addresses?

TX hashes

- All writes can be sent in parallel
  - LSX = TRX
  - Stabilization complexity = $O(1)$

- Might require database indexing
- Many BCs do not support this

Used for look-ups



1 What should we use as addresses?

TX hashes

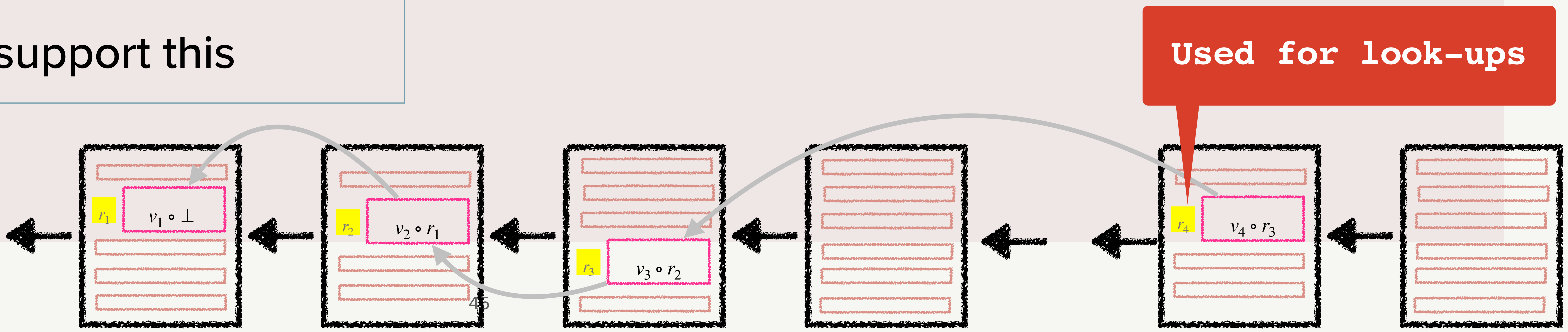
Block numbers + TX hashes

+

- All writes can be sent in parallel
- LSX = TRX
- Stabilization complexity = $O(1)$

−

- Might require database indexing
- Many BCs do not support this



① What should we use as addresses?

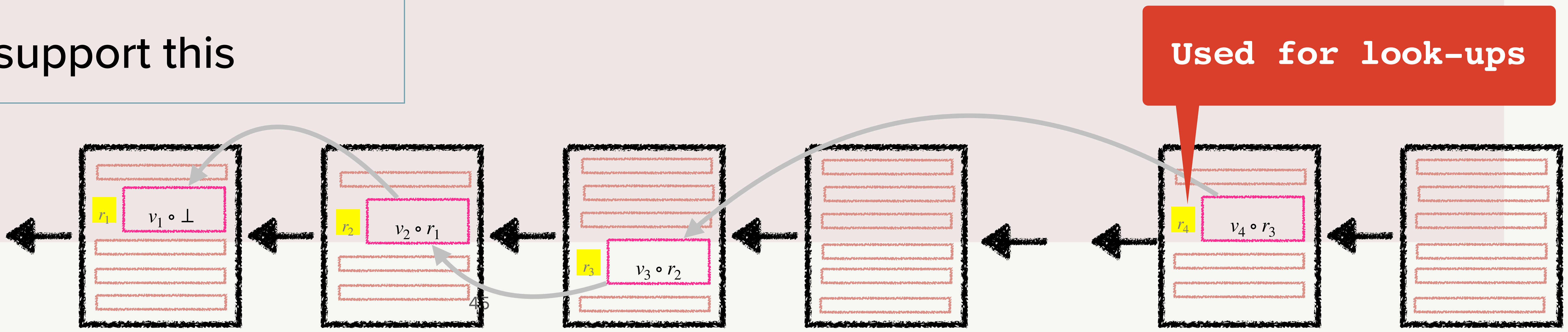
TX hashes

+
- All writes can be sent in parallel
  - LSX = TRX
  - Stabilization complexity = $O(1)$

−
- Might require database indexing
- Many BCs do not support this

Block numbers + TX hashes

+
- BCs support fast lookup by block numbers



① What should we use as addresses?

TX hashes

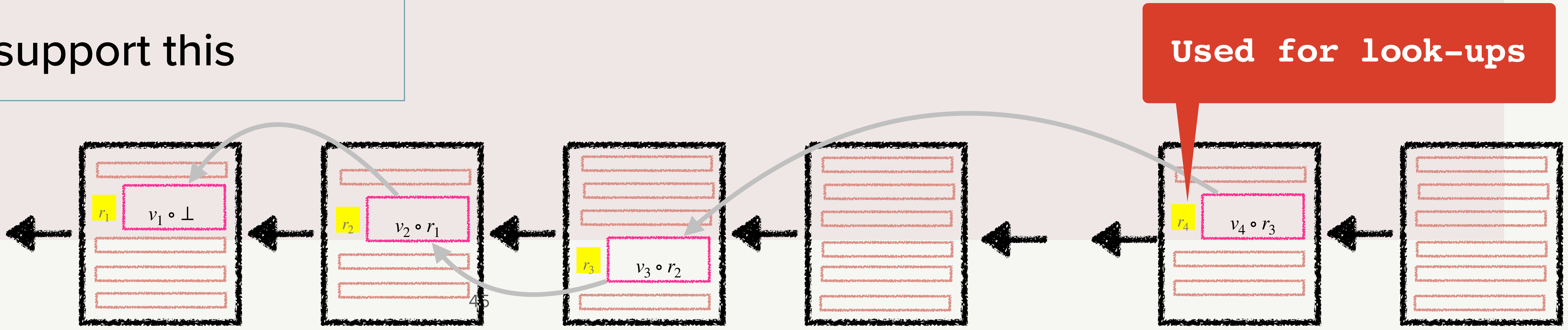
+
- All writes can be sent in parallel
  - LSX = TRX
  - Stabilization complexity = $O(1)$

−
- Might require database indexing
- Many BCs do not support this

Block numbers + TX hashes

+
- BCs support fast lookup by block numbers

−
- Higher bandwidth required



① What should we use as addresses?

Add on Algorand

TX hash

rs + TX hashes

be s

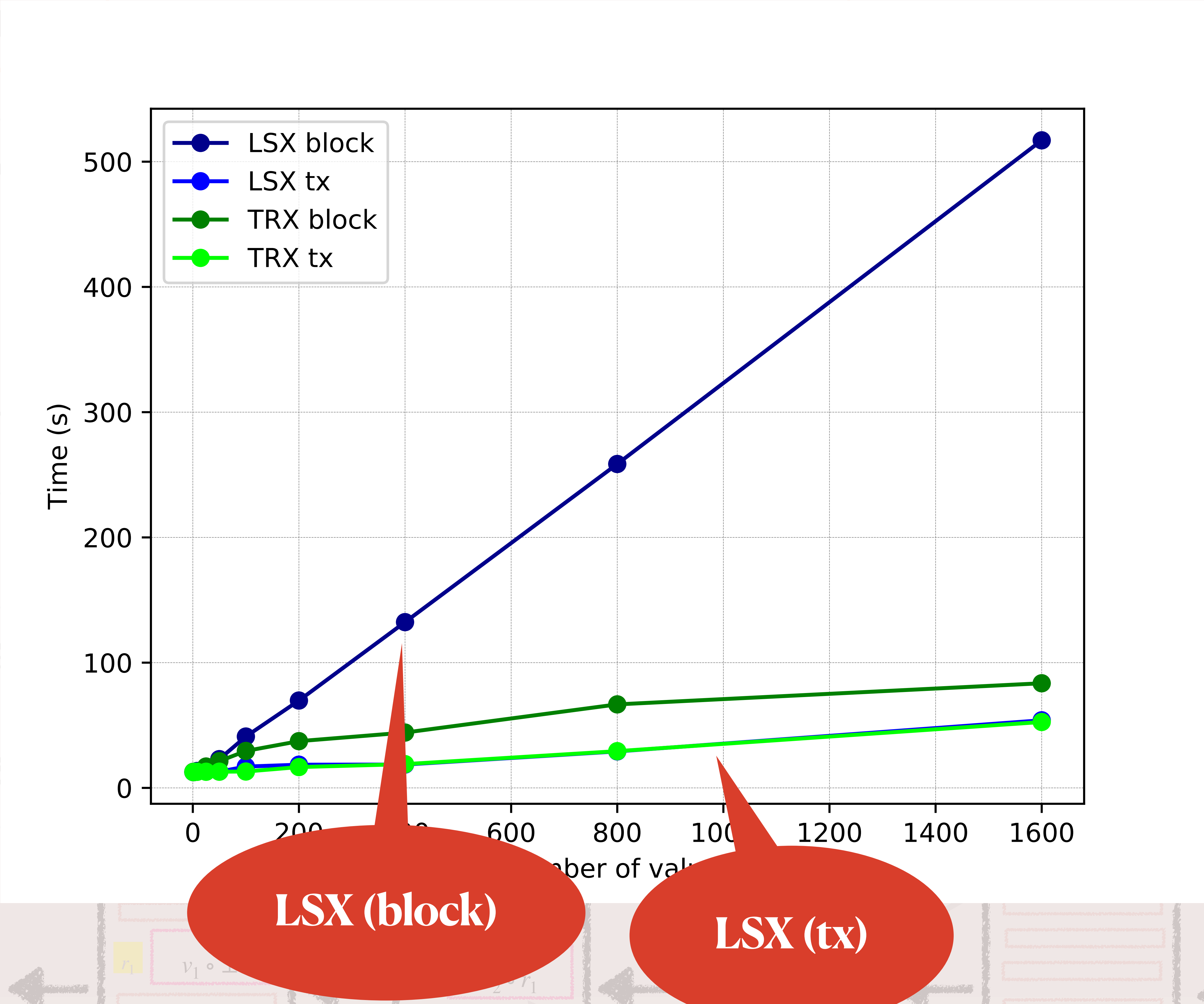
lookup by block numbers

- LSX = TRX
- Stabilization com

n required

- Might require data
- Many BCs do not s

Used for look-ups



① What should we use as addresses?

TX hash

rs + TX hashes

lookup by block numbers

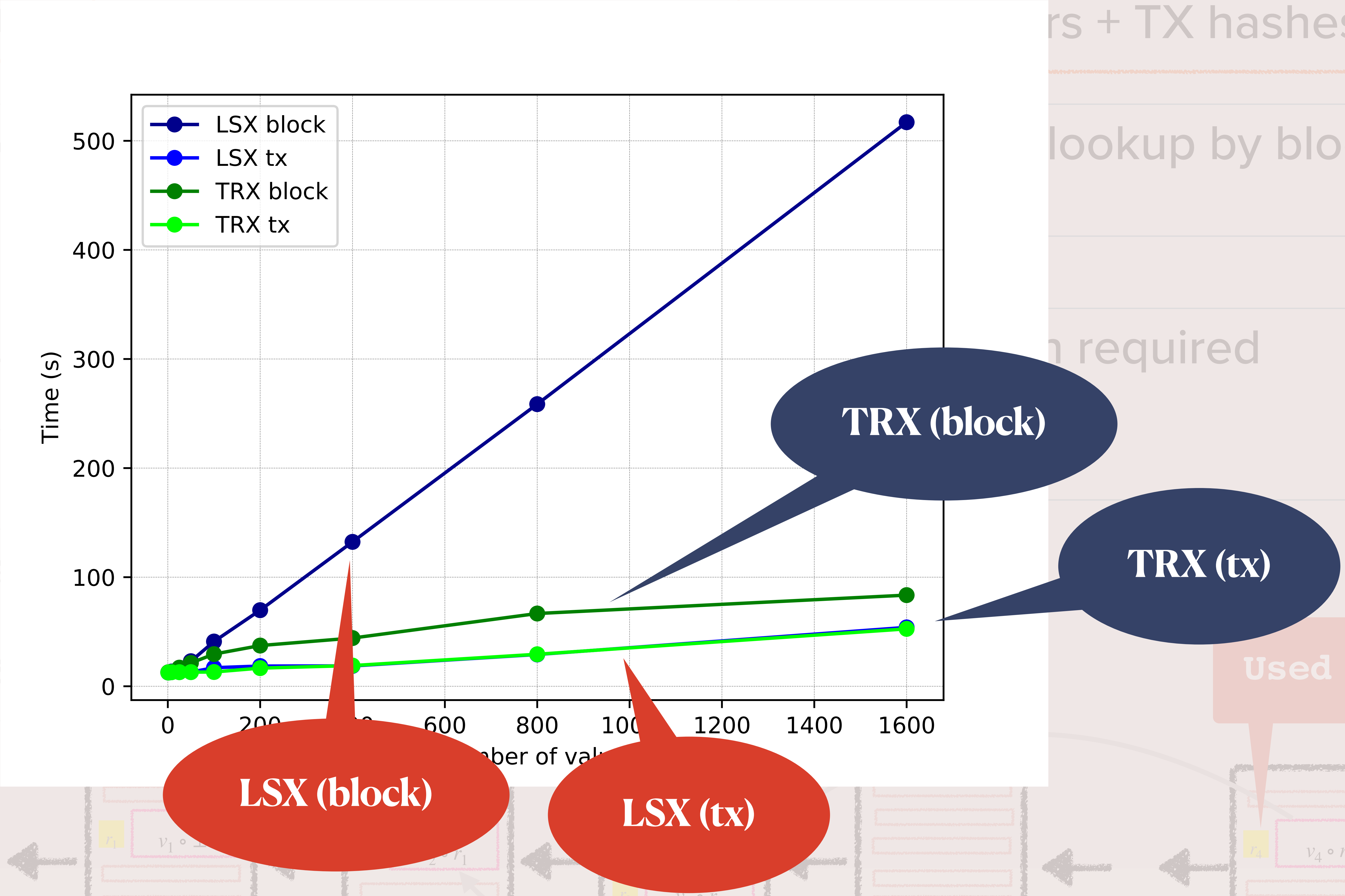
- LSX = TRX
- Stabilization com

required

- Might require data
- Many BCs do not

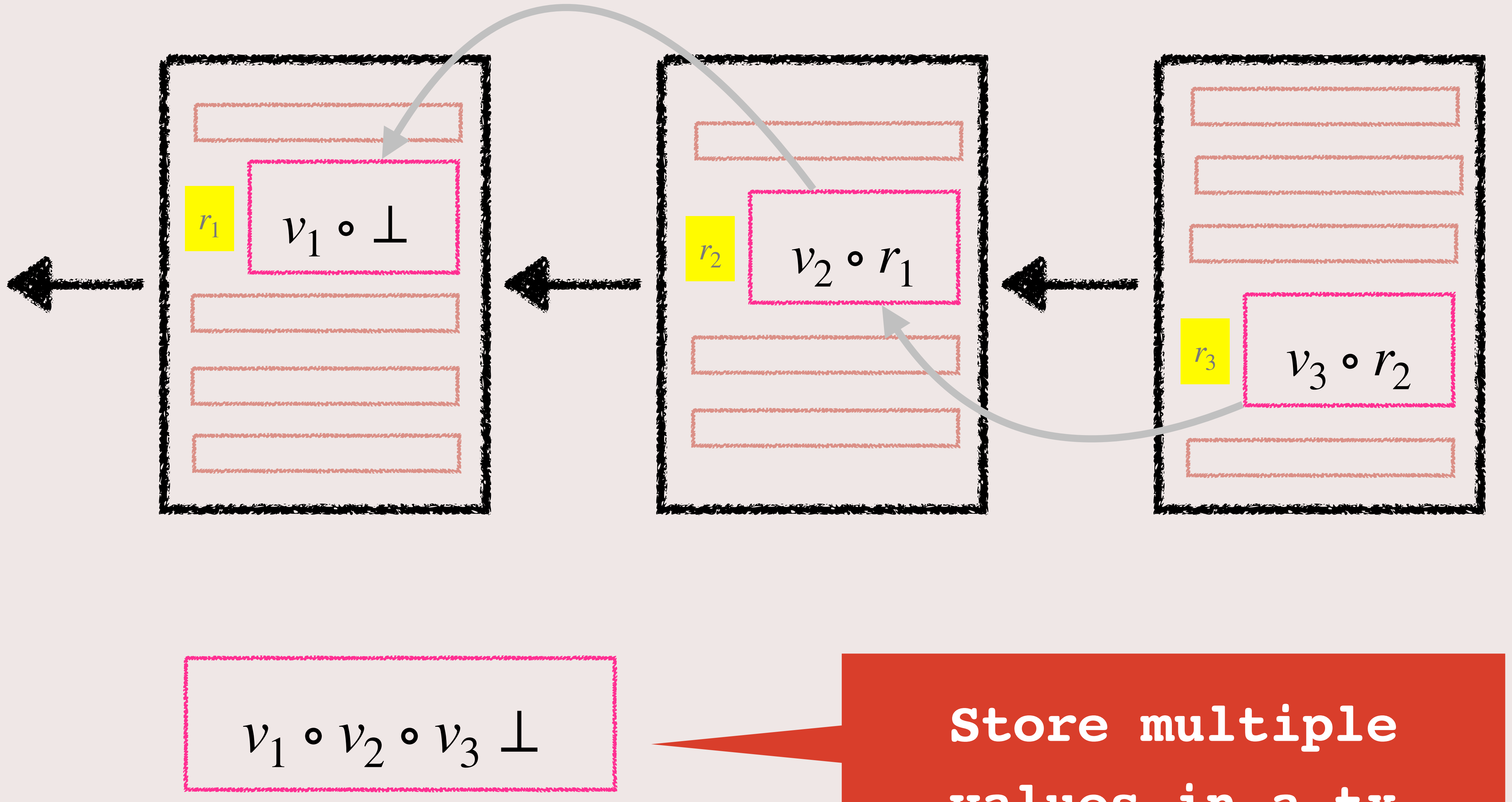
Used for look-ups

Add on Algorand



2 Packing

Saves time & money



Store multiple values in a tx

② Packing

PAX doesn't support packing

② Packing

PAX doesn't support packing

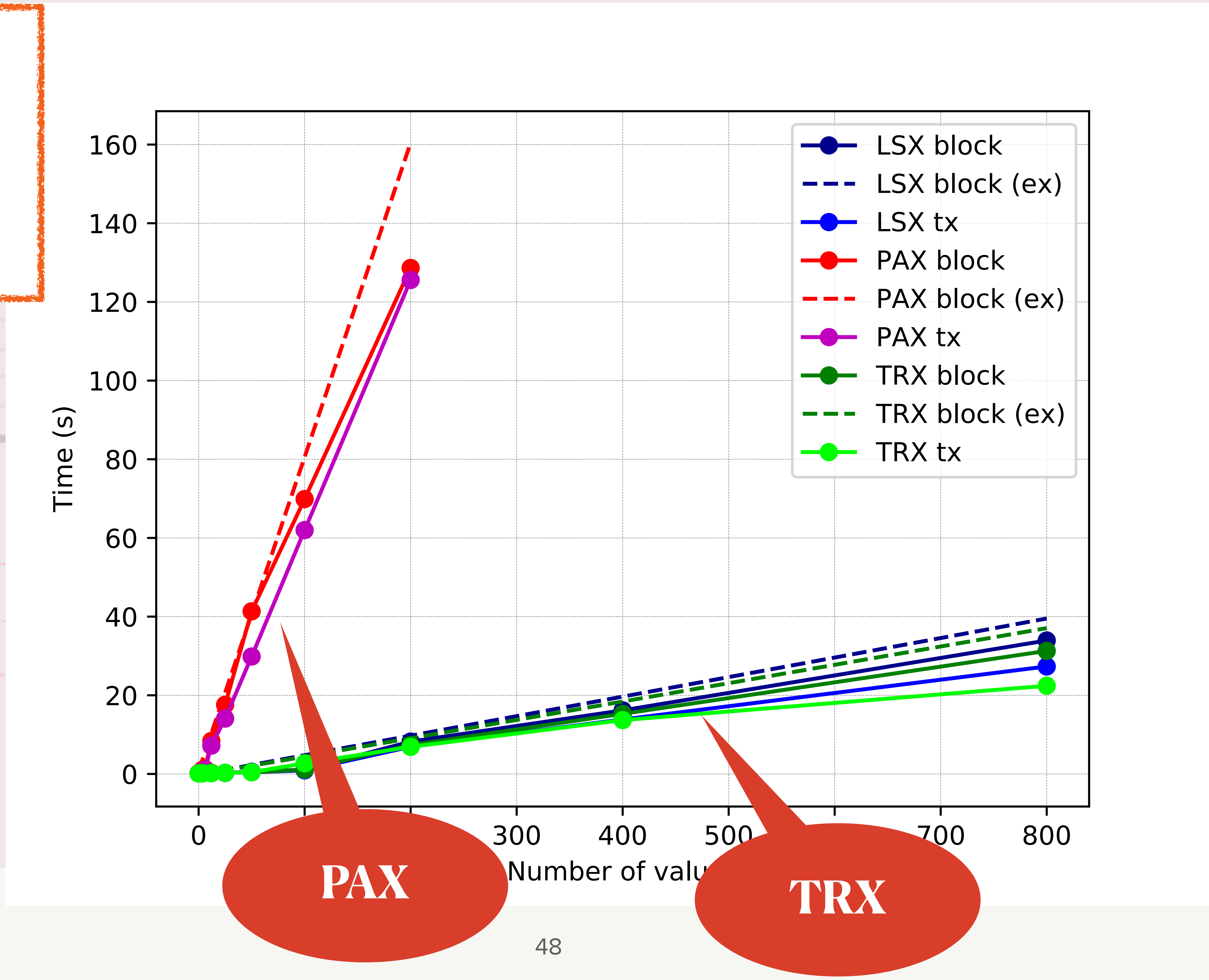
Query on Algorand

time PAX

O(size of MM)

time TRX

O(# values ever added/deleted)



② Packing

PAX doesn't support packing ?

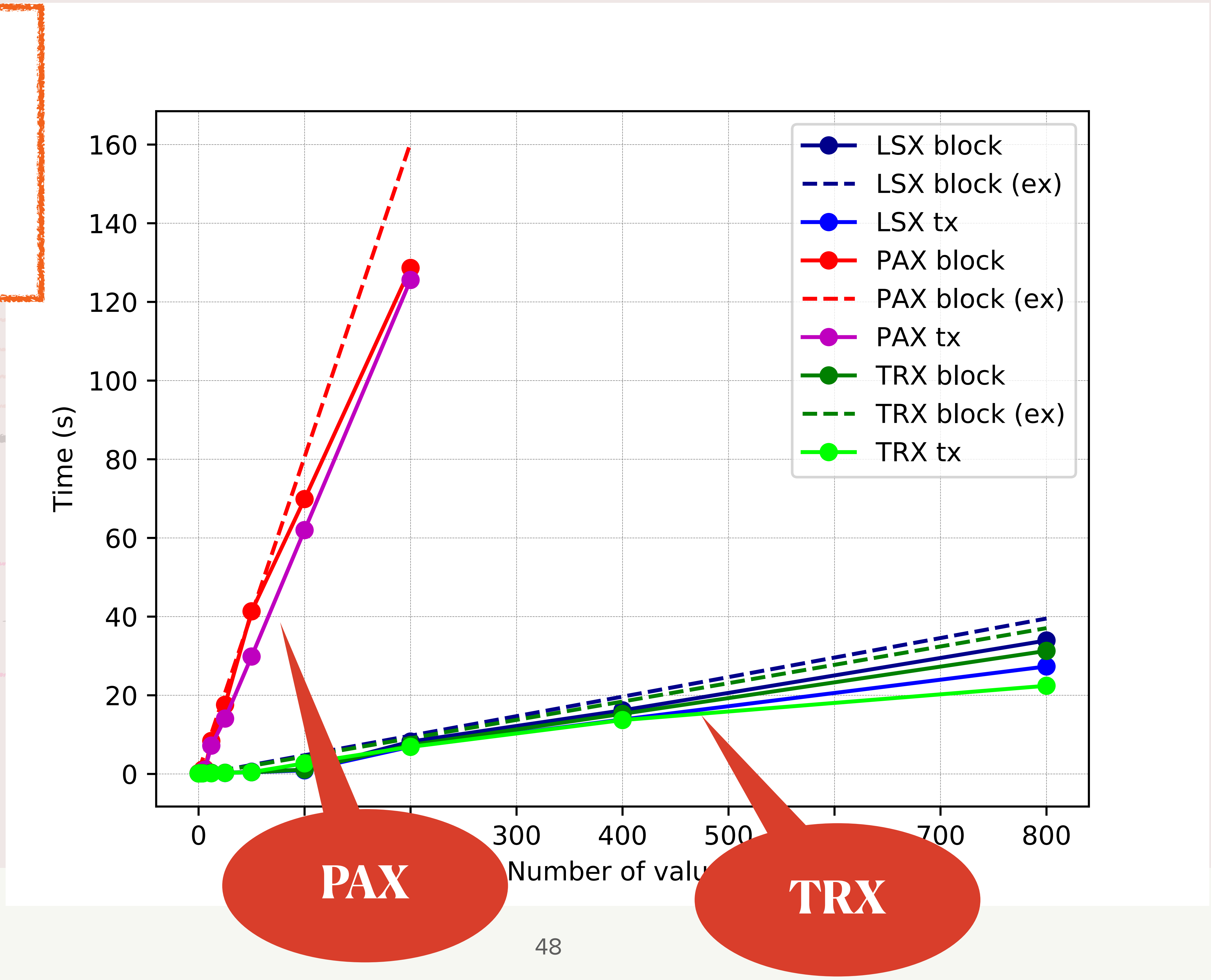
Query on Algorand

time PAX

O(size of MM)

time TRX

O(# values ever added/deleted)



② Packing

PAX doesn't support packing ?

Query on Algorand

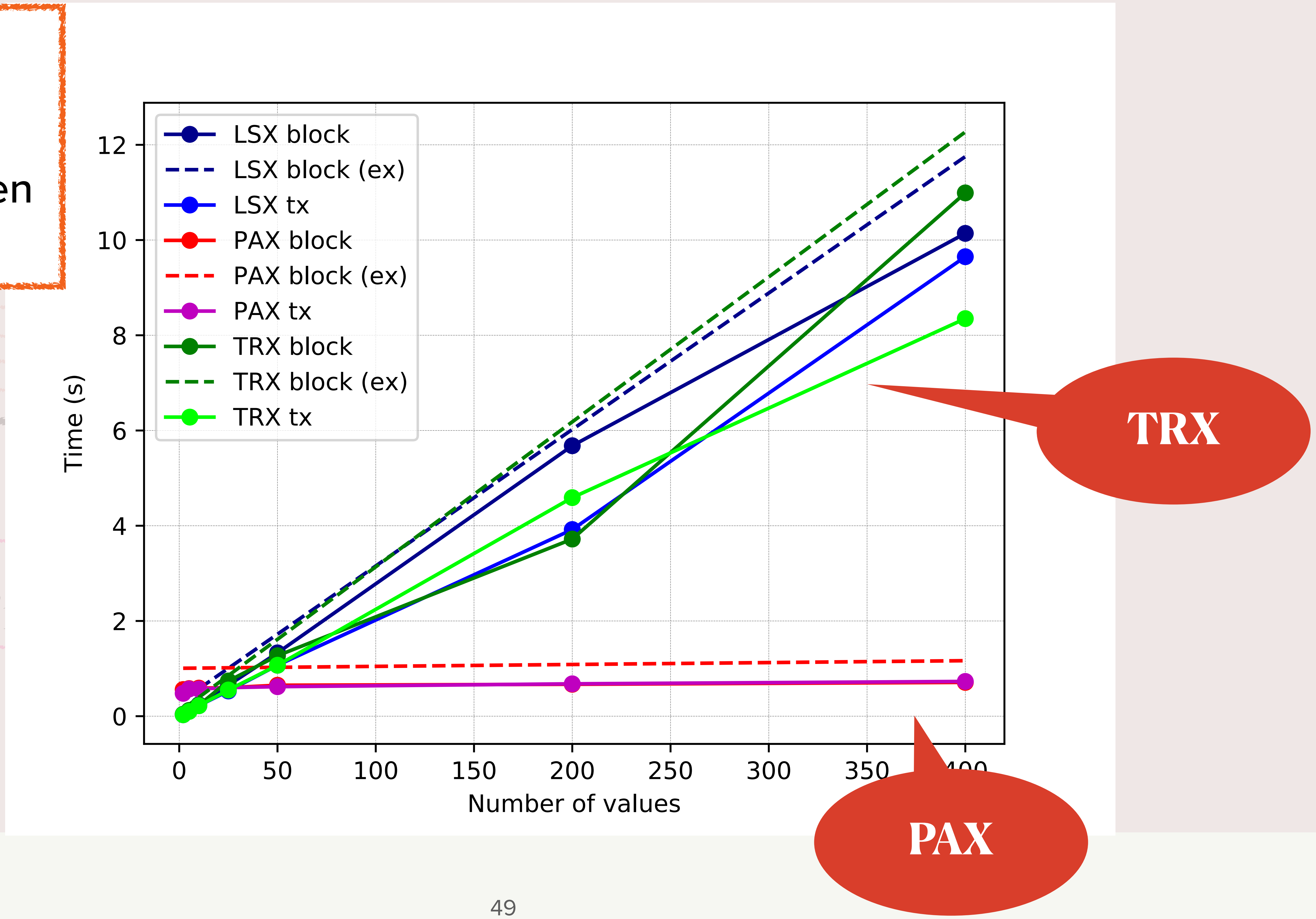
(after most values have been deleted)

time PAX

O(size of MM)

time TRX

O(# values ever added/deleted)




THANKS
"For watching!"


THANKS
"For watching!"