Toward Precise PLRU Cache Analysis

Daniel Grund[1] Jan Reineke[2]

[1]Saarland University, Saarbrücken, Germany

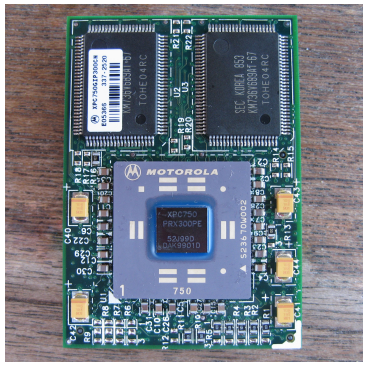
[2]University of California, Berkeley, USA

Workshop on Worst-Case Execution-Time Analysis 2010

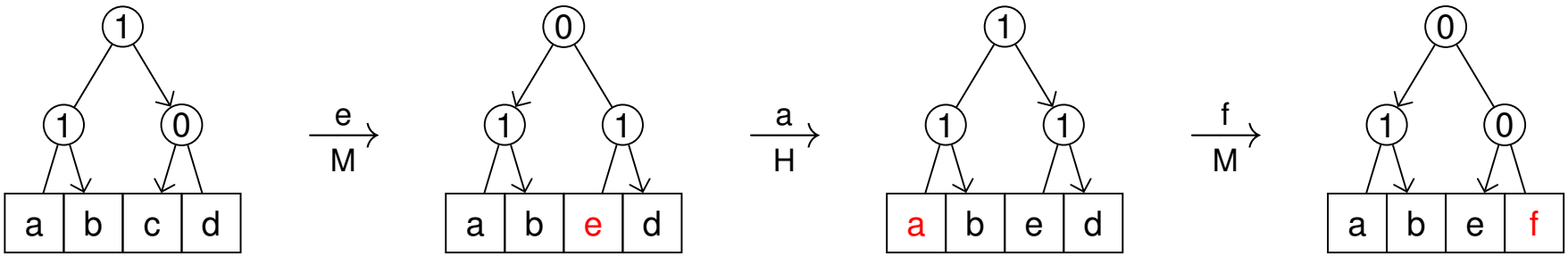
Outline

Motivation

- Cache replacement policies
  - Least-recently used
  - Pseudo-LRU
  - First-in First-out
  - Pseudo Round-Robin
  - “Random”
- PLRU
  - is flying in MPC603E
  - is flying in MPC755
  - will fly in MPC7448
  - will drive in TRICORE 1798



PLRU Replacement



- Tree bits point to next victim
- After access, tree bits on path are set to point away

Outline

Outline

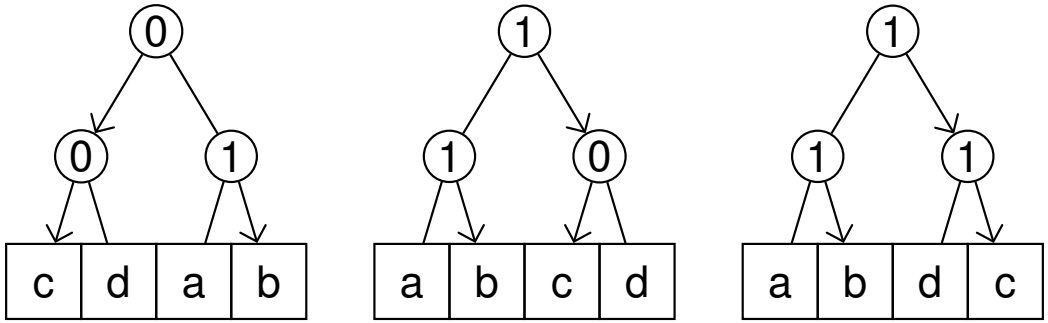
Non-Trivial Logical States

- There are cache states that
  - differ in physical arrangement of cached blocks
  - exhibit the same replacement behavior
- Inefficient to distinguish such states

$\Rightarrow$ Abstract from physical cache states to logical ones

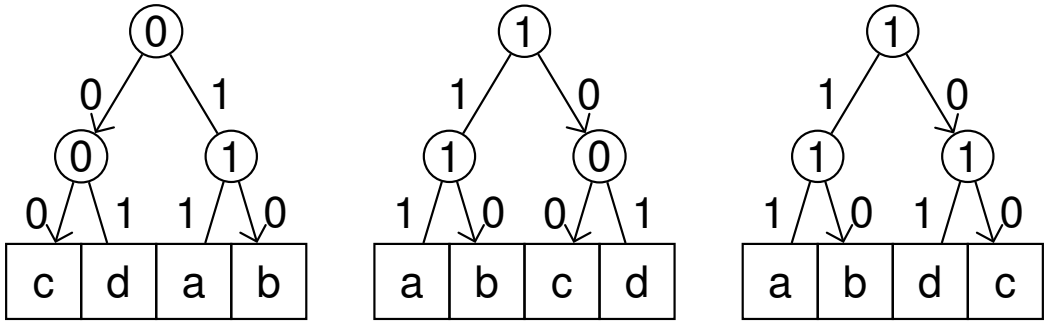
- Easy for LRU $\overset{\text{MRU}}{\downarrow}[b_1, \ldots, \overset{\text{LRU}}{b_k}]$ and for FIFO $\overset{\text{last-in}}{\downarrow}[b_1, \ldots, \overset{\text{first-in}}{b_k}]$
- How to do this for PLRU? How to abstract from the tree bits?

Equivalent States



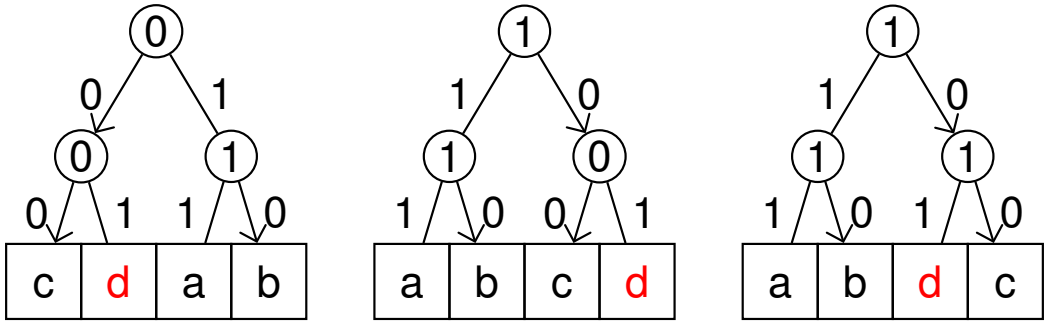
- Only differ in interchanged subtrees
- What matters? Whether tree bit points towards an element or not

Equivalent States



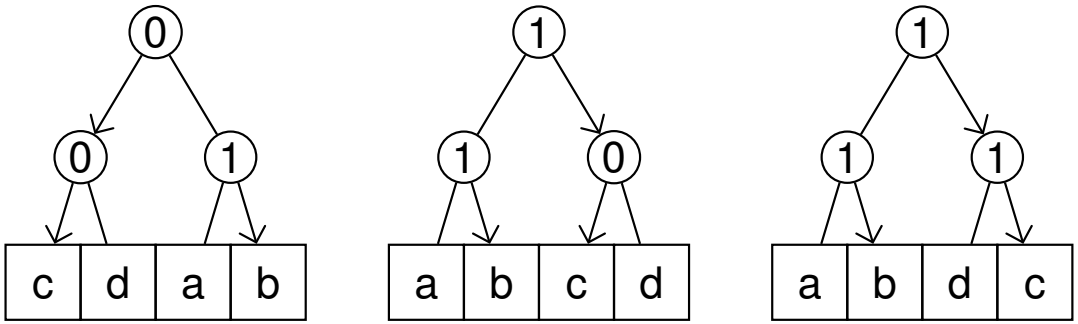
- Only differ in interchanged subtrees
- What matters? Whether tree bit points towards an element or not
- Edge bits encode this “points towards”

Equivalent States



- Only differ in interchanged subtrees
- What matters? Whether tree bit points towards an element or not
- Edge bits encode this “points towards”
- Access path consists of edge bits from leaf to root
- Access path of of *d* is 10

Coarsest Complete Abstraction



1. Same replacement behavior $\iff$ coinciding access paths

$$q_1 \sim q_2 \iff \forall b \in \mathcal{B} : ap(q_1, b) = ap(q_2, b)$$

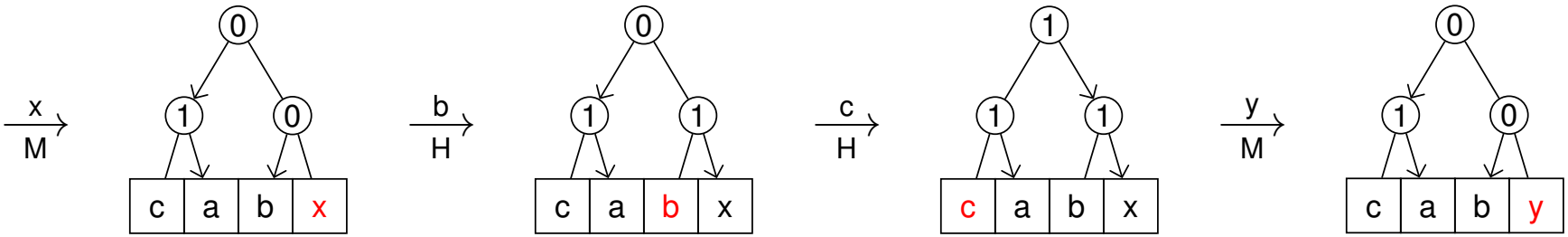
2. Imply order on access paths (*miss-replacement distance*)

$\Rightarrow$ Unique representation for all equivalent physical states

- Logical state $\tilde{q} = [a, d, b, c]^{\sim}$ represents all physical states from above

Outline

Logarithmic-Time Eviction

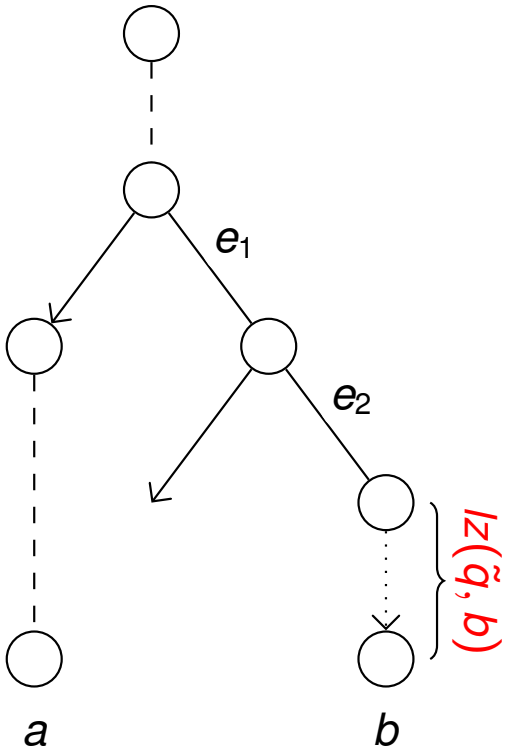


- x inserted by miss
- x evicted after $\log_2(4) + 1 = 3$ accesses
- although size $k = 4$

$\Rightarrow$ Must-analysis easy up to $\log_2(k) + 1$ blocks

$\ldots$ But hard beyond that

Leading Zeros

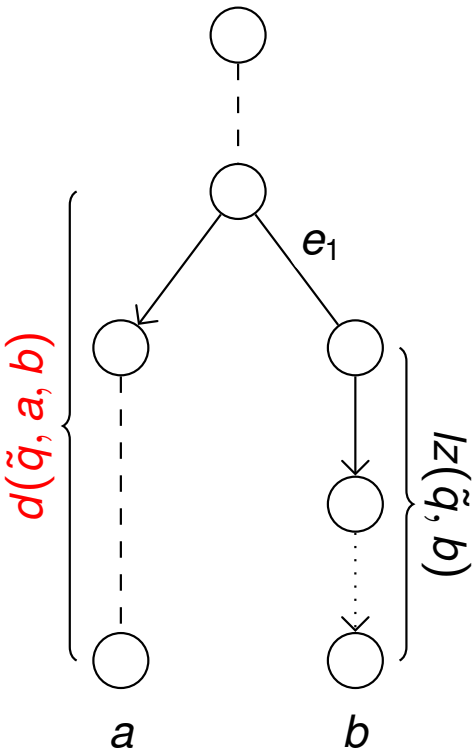


- To evict b, its access path must be $0 \ldots 0$
- Edge-bits need to be flipped bottom-up, i.e. e_2 before e_1

$$ap(\tilde{q}, b) : 111 \rightarrow 011 \rightarrow 001 \rightarrow 000$$

$\Rightarrow$ Leading zeros in access paths are interesting

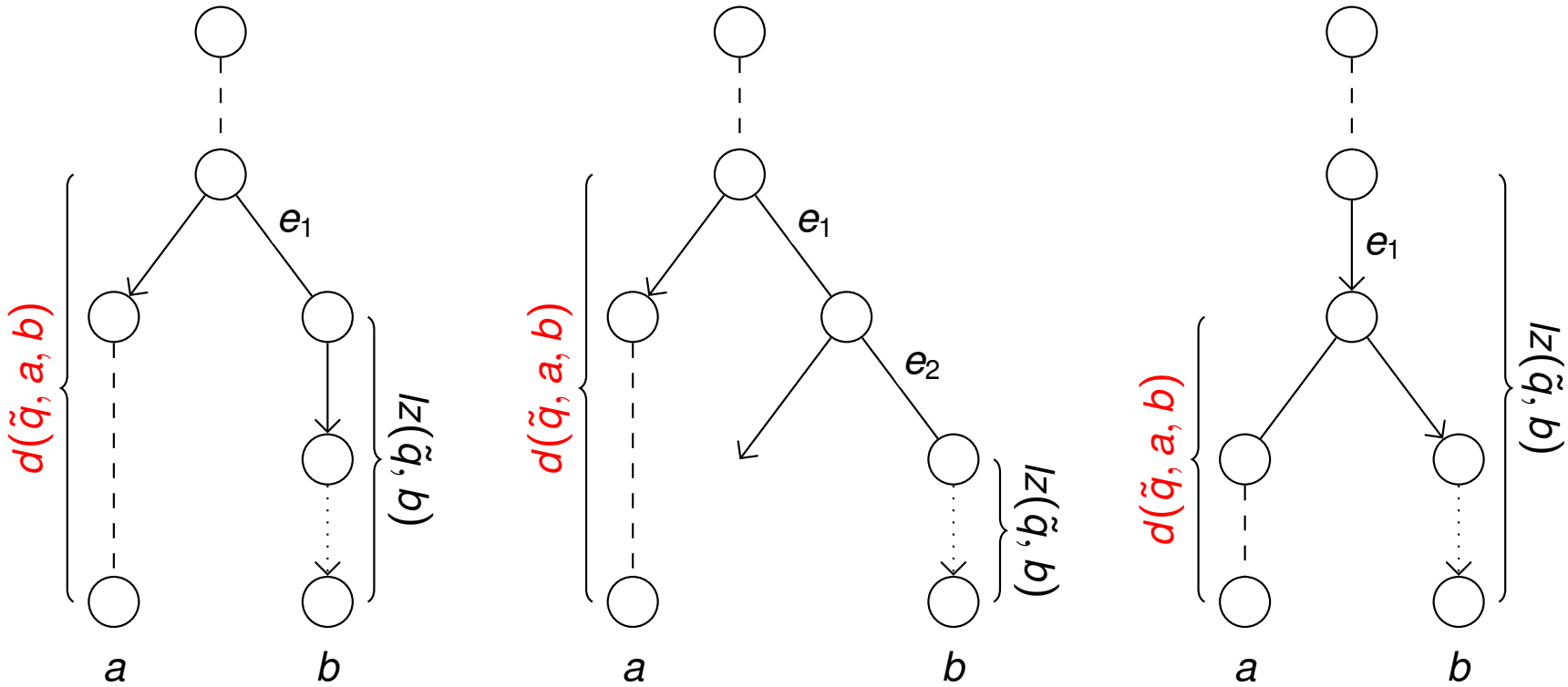
Subtree Distance



- How does $ap(\tilde{q}, b)$ change when accessing a?
- Depends on relative position of a to b

$\Rightarrow$ Subtree distance between blocks are interesting

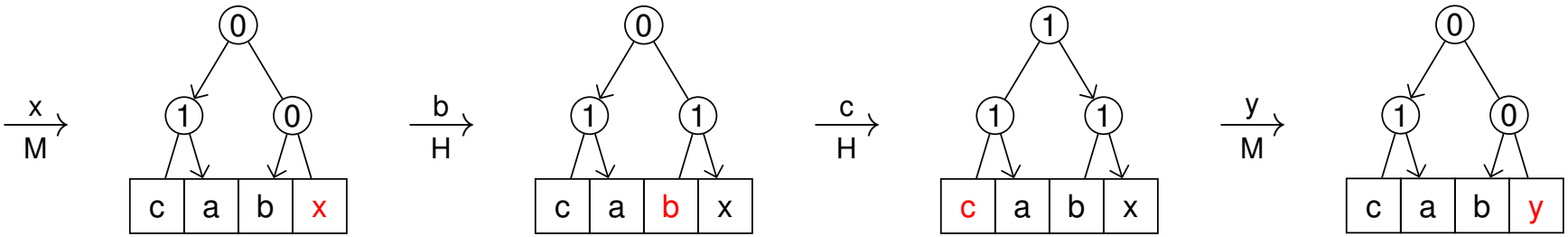
Subtree Distance



- How does $ap(\tilde{q}, b)$ change when accessing a?
- Depends on relative position of a to b

$\Rightarrow$ Subtree distance between blocks are interesting

Excluding Logarithmic-Time Eviction



- For log-time eviction, need to access blocks in increasing subtree distance
- Knowledge about subtree distances can exclude log-time eviction

Abstract Domain

- Potentially leading zeros

$$PLZ_k := \mathcal{B} \to \{0, \ldots, \log_2(k), \top\}$$

- Approximated Distance

$$AD_k := \mathcal{B} \times \mathcal{B} \to \{\, \{0\}, [1, \log_2(k)), \{\log_2(k)\}, \top \,\}$$

- The complete domain

$$Plru_k^{\Delta} := AD_k \hookrightarrow PLZ_k$$

- Tradeoff possible by plugging in different AD_k
- Formalization and details in the paper

Outline

Arbitrary Survival

- There are access sequences that
  - access arbitrarily many distinct blocks
  - do not access x
  - but x is still cached

$\Rightarrow$ Hard to prove eviction of blocks

- No viable May-Analysis, yet

Outline

Brief History of Replacement Analysis

Before ’97	LRU analyses
LCTRTS’97	Precise and efficient must- and may-analysis for LRU [1]
LCTES’08	Generic analyses for FIFO and PLRU [2]
SAS’09	Cache analysis framework and FIFO analysis [3]
WCET’10	Toward precise analysis for PLRU
ECRTS’10	Precise and efficient must- and may-analysis for FIFO [4]
	$\rightarrow$ 4pm session on Thursday

Evaluation Setup

- Analyses:
  - RC Analysis based on relative competitiveness [2]
  - Δ Analysis based on subtree distances
- Collecting semantics:
  - CS Limit for any static analysis
- Spectrum of synthetic benchmarks:
  - Random access sequences
  - Loops

Evaluation Results

SAARLAND UNIVERSITY
COMPUTER SCIENCE

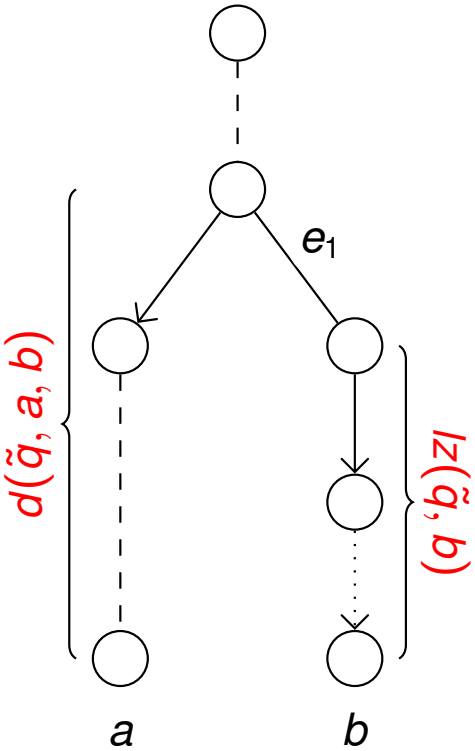
		Associativity $k = 8$						
	n	2	3	4	5	6	7	8
Loop	$Plru_k^{RC_H}$	93.8	93.8	93.8	0.0	0.0	0.0	0.0
	$Plru_k^{\Delta}$	93.8	93.8	93.8	92.5	90.6	0.0	0.0
	$Plru_k^{CS}$	93.8	93.8	93.8	92.5	91.7	90.2	86.7
Rand	$Plru_k^{RC_H}$	98.0	97.0	96.0	77.7	64.4	55.7	48.1
	$Plru_k^{\Delta}$	98.0	97.0	95.8	93.0	84.3	63.5	52.0
	$Plru_k^{CS}$	98.0	97.0	96.0	93.9	91.0	84.0	68.4

- Hit rates [%] guaranteed by the analyses and the collecting semantics
- n is number of distinct elements that get accessed

Outline

Summary

- Logical PLRU-states
  - abstract from tree-bits
  - coarsest complete abstraction
- Must-analysis / log-time eviction
  - leading zeros
  - subtree distance
- May-analysis / arbitrary survival
  - unsolved



Further Reading


C. Ferdinand
Cache Behaviour Prediction for Real-Time Systems
PhD Thesis, Saarland University, 1997


J. Reineke and D. Grund
Relative competitive analysis of cache replacement policies
LCTES 2008


D. Grund and J. Reineke
Abstract Interpretation of FIFO Replacement
SAS 2009


D. Grund and J. Reineke
Precise and Efficient FIFO-Replacement Analysis Based on Static Phase Detection
ECRTS 2010

Instruction or Data Cache?

- Both!
- Cache analysis $=$ value analysis $\oplus$ replacement analysis
- Value analysis approximates accessed addresses
- Replacement analysis approximates cache contents

- Imprecise value analysis results?
- Replacement analysis can always join over all possibilities

Do you Need a Functional Language?

- No!
- Let $f \in PLZ_k := \mathcal{B} \rightarrow \{0, \ldots, \log_2(k), \top\}$
- Let $f(a) = 0$, $f(b) = 1$, and $f(x) = \top$ otherwise
- No need to explicitly represent the default value $\top$ for all $x \in \mathcal{B}$

$\Rightarrow$ Can represent the function f as $\{(a, 0), (b, 1)\}$

Comparing $Plru_k^{\Delta}$ and $Plru_k^{RC_H}$

- $Plru_k^{\Delta}$ is not always better than $Plru_k^{RC_H}$

	a	b	b	c	d	a
$Plru_k^{RC_H}$	$\top$	$\top$	H	$\top$	$\top$	H
$Plru_k^{\Delta}$	$\top$	$\top$	H	$\top$	$\top$	$\top$

- If $d(\tilde{q}, a, b) = 1$, $Plru_k^{\Delta}$ cannot remember this (for $k \geq 8$)
- In that case, it only knows $d(\tilde{q}, a, b) \in [1, 2]$
- First access to b: $d(\tilde{q}_1, a, b) = 1$ possible
- Second access to b: $d(\tilde{q}_1, a, b) = 2$ possible