Comp 590-184: Hardware Security and Side-Channels

Lecture 5: Eviction Sets

January 22, 2026
Andrew Kwong

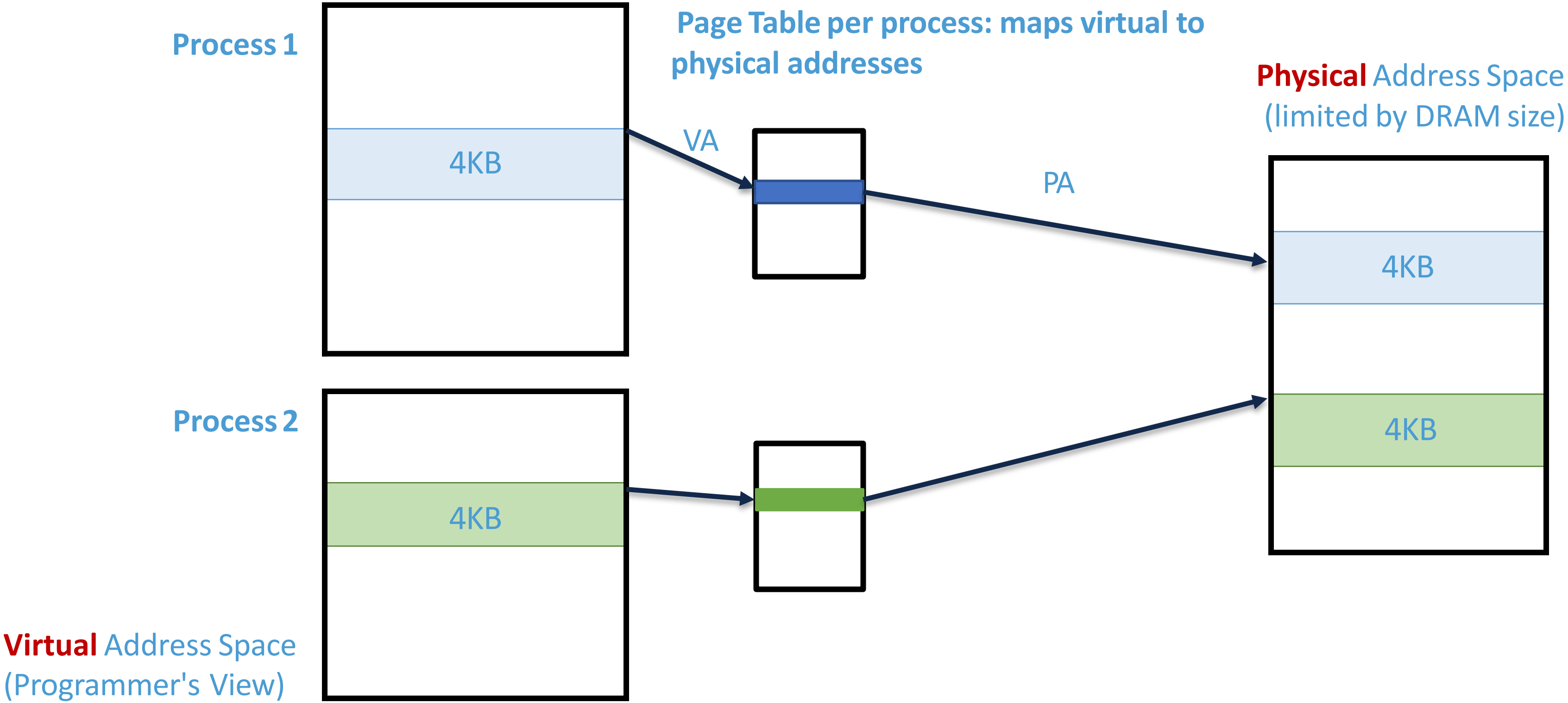
THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Slides adapted from Pepe Vila (https://vwzq.net/papers/evictionsets19.pdf)

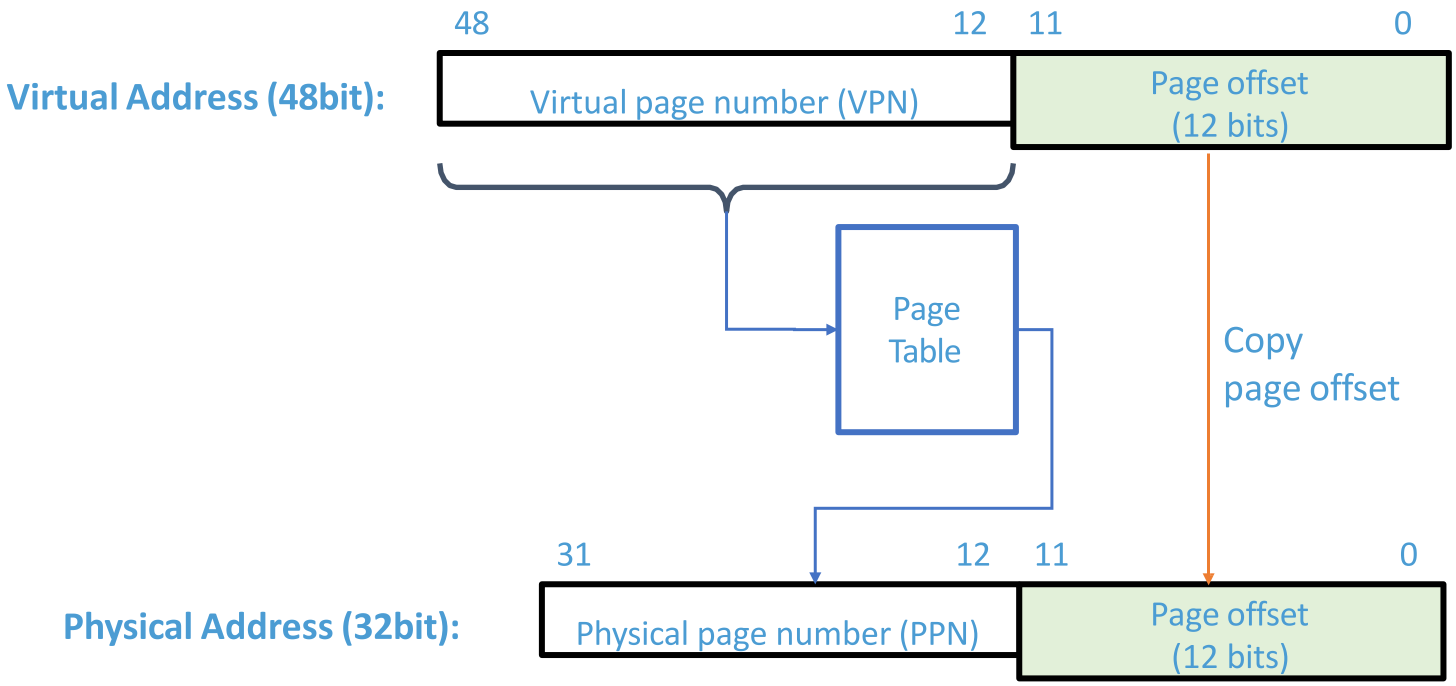
Today's Class

- How to practically build eviction sets for conducting cache attacks

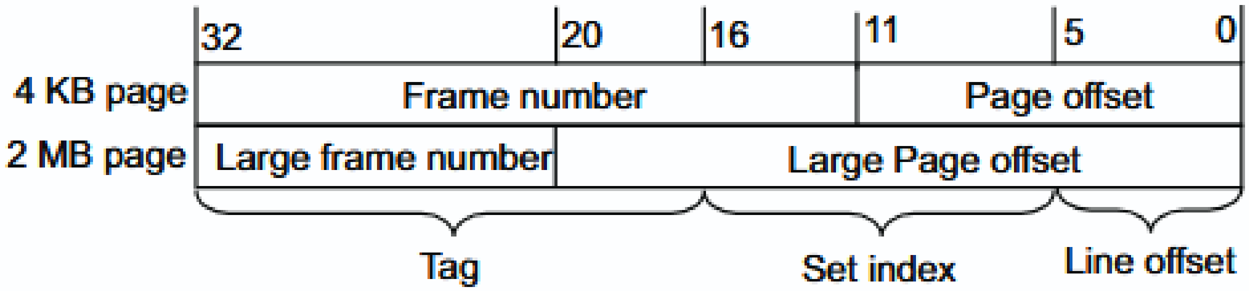
Page Mapping



Address Translation (4KB page)



Cache Indexing



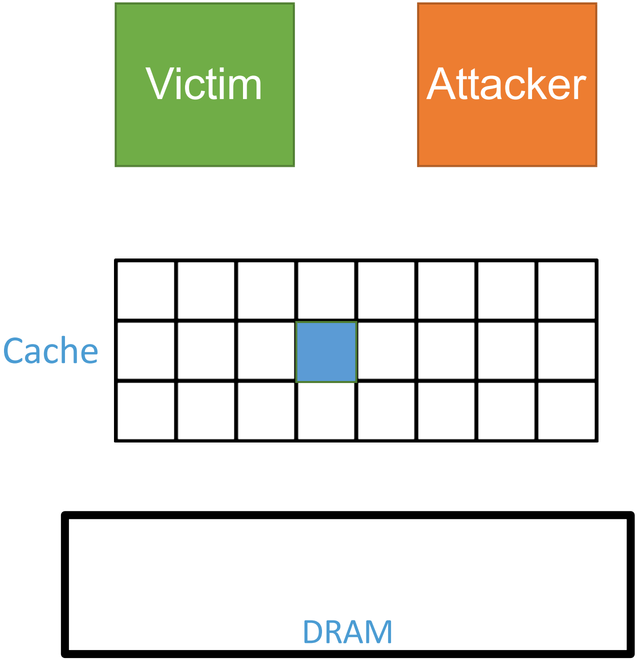
Quiz!

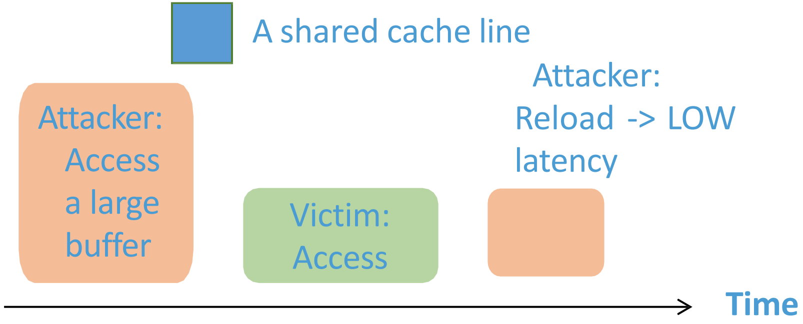
- I have a physical address: 0xAAAA
- The cache parameters are as below
  - Cache size: 32KB
  - Line size/Block size: 64B
  - Associativity: 8

Question 1:
What is the cache set index?

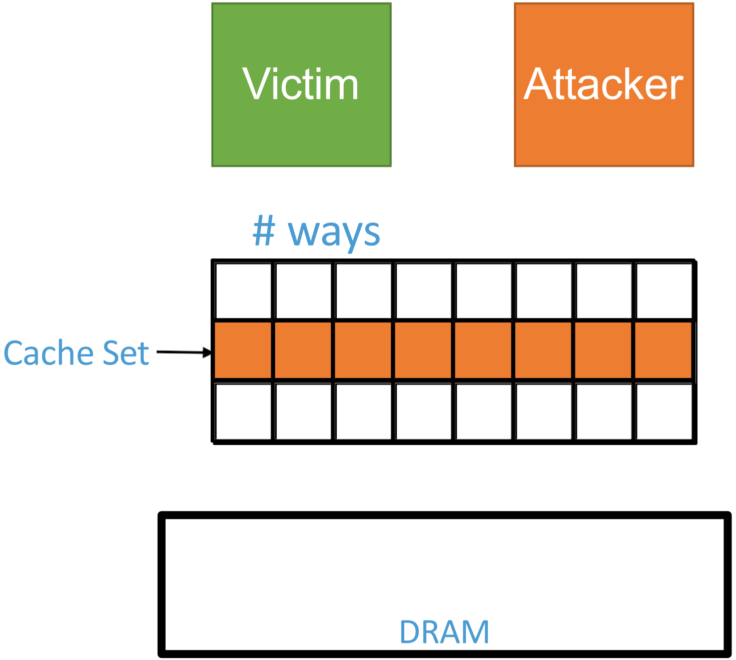
Question 2:
What is the next address that maps to **the same cache set** as this one but not the same cache line?

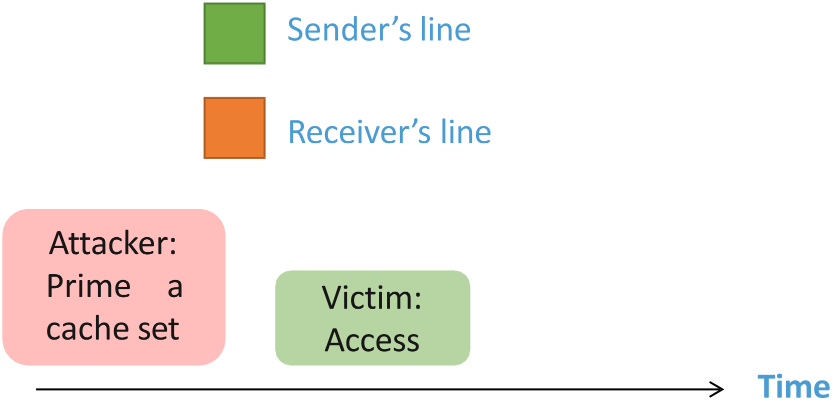
Evict+Reload



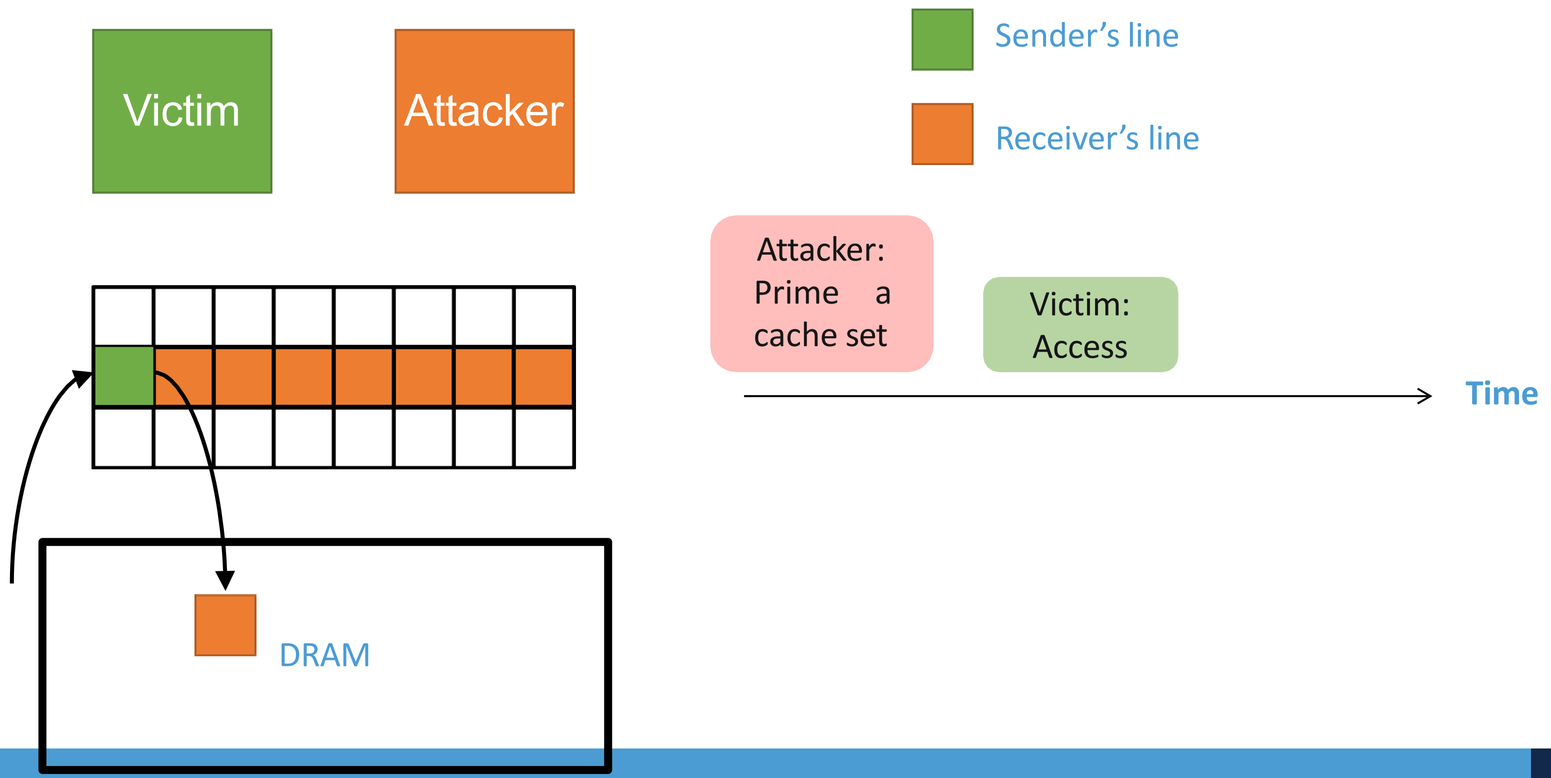


Attack Strategy #3: Prime+Probe

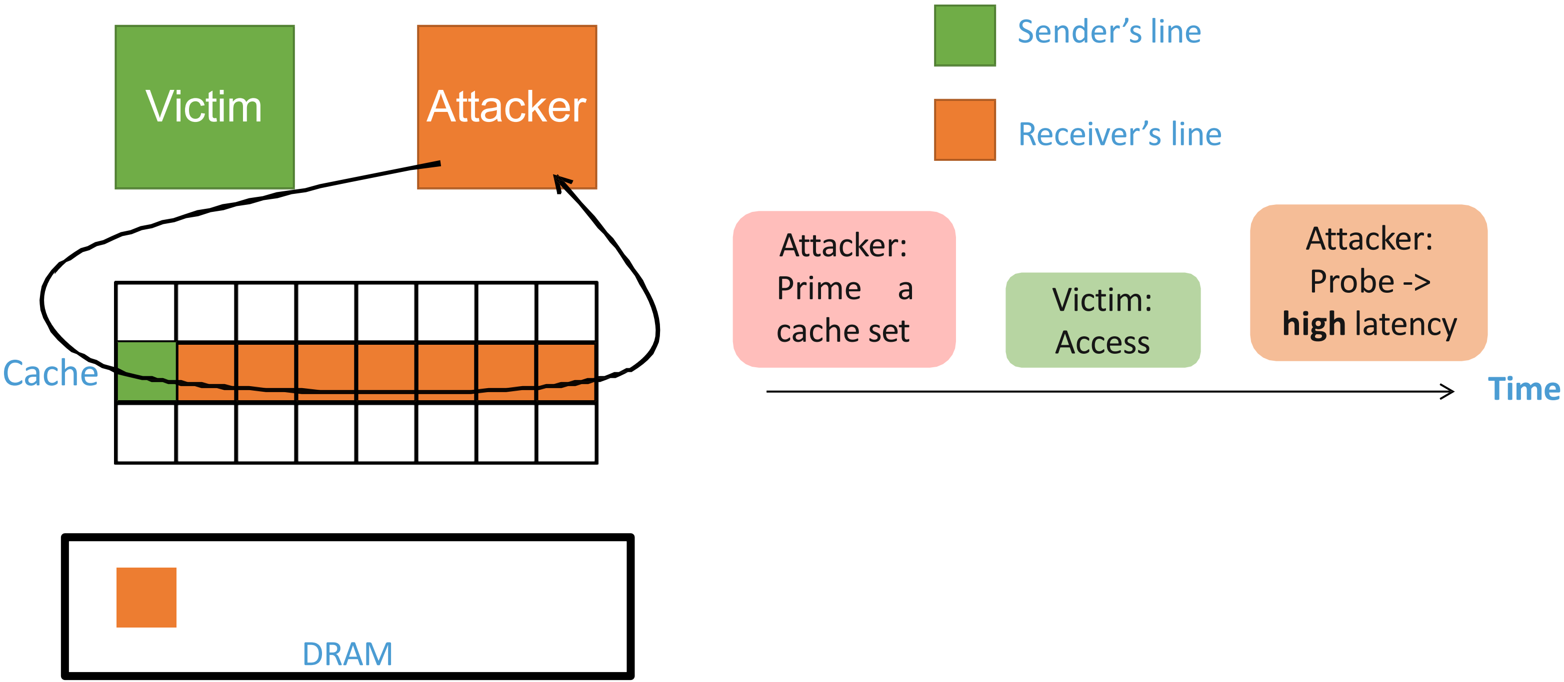




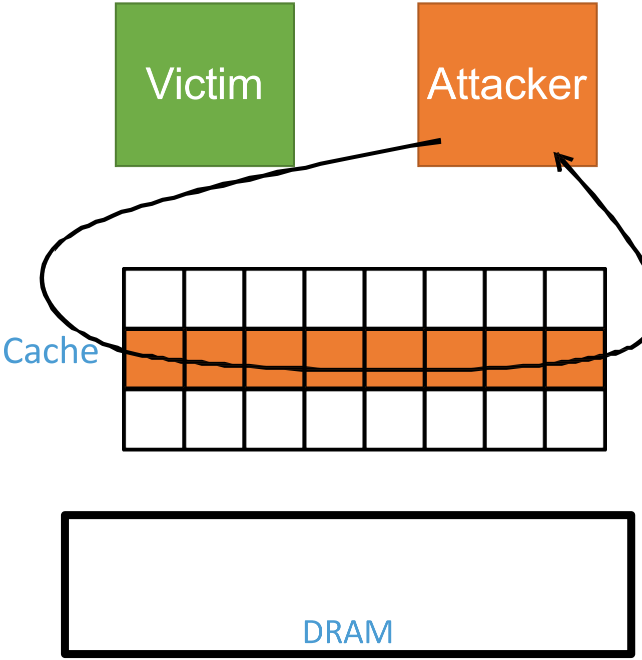
Attack Strategy #3: Prime+Probe

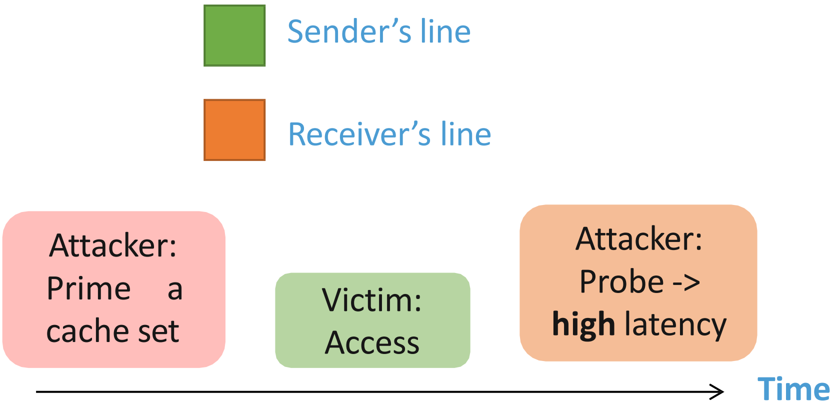


Attack Strategy #3: Prime+Probe



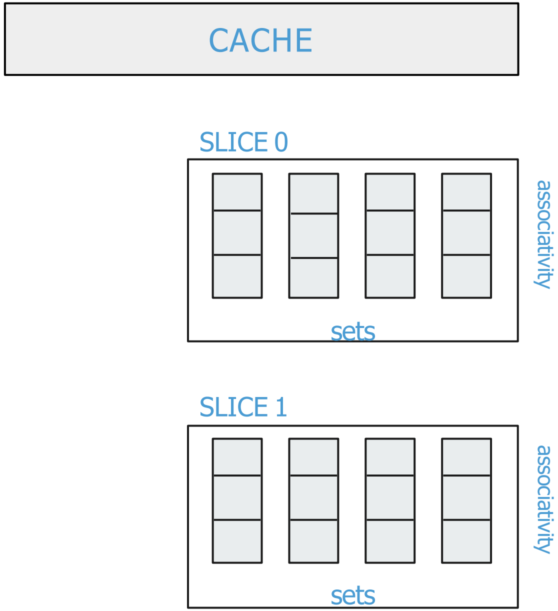
Attack Strategy #3: Prime+Probe





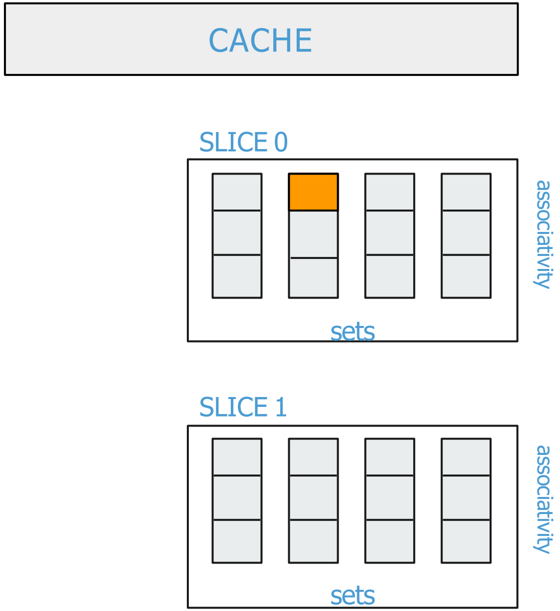
Eviction Sets

Set of addresses that collide in cache: i.e. addresses mapped into the same cache set



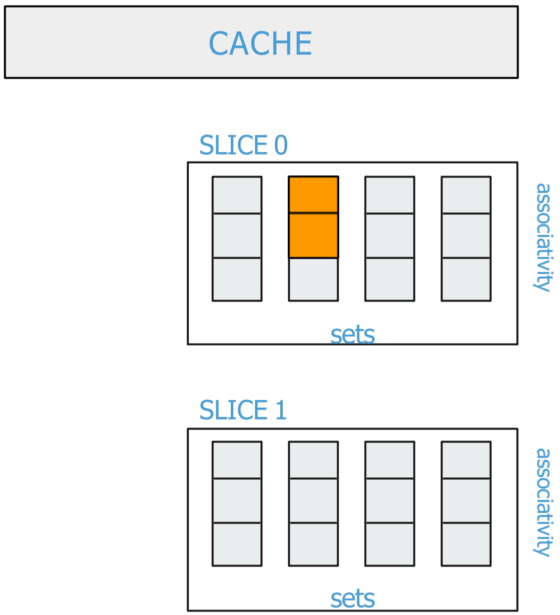
Eviction Sets

Find addresses that collide in cache: i.e. addresses mapped into the same cache set



Eviction Sets

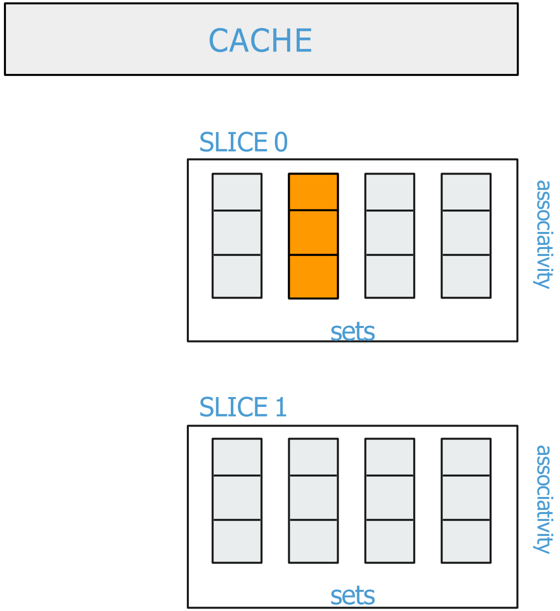
Find addresses that collide in cache: i.e. addresses mapped into the same cache set



Eviction Sets

Find addresses that collide in cache: i.e. addresses mapped into the same cache set

Find associativity many colliding addresses: i.e. an **eviction set**



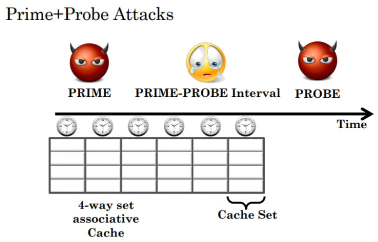
Attacks

Efficient attacks require <u>small eviction sets</u>

Attacks

Efficient attacks require <u>small eviction sets</u>

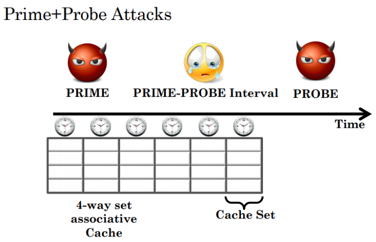
Prime+Probe



Attacks

Efficient attacks require small eviction sets

Prime+Probe

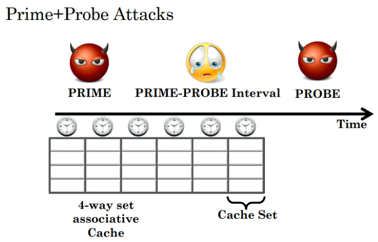


Rowhammer

Attacks

Efficient attacks require small eviction sets

Prime+Probe

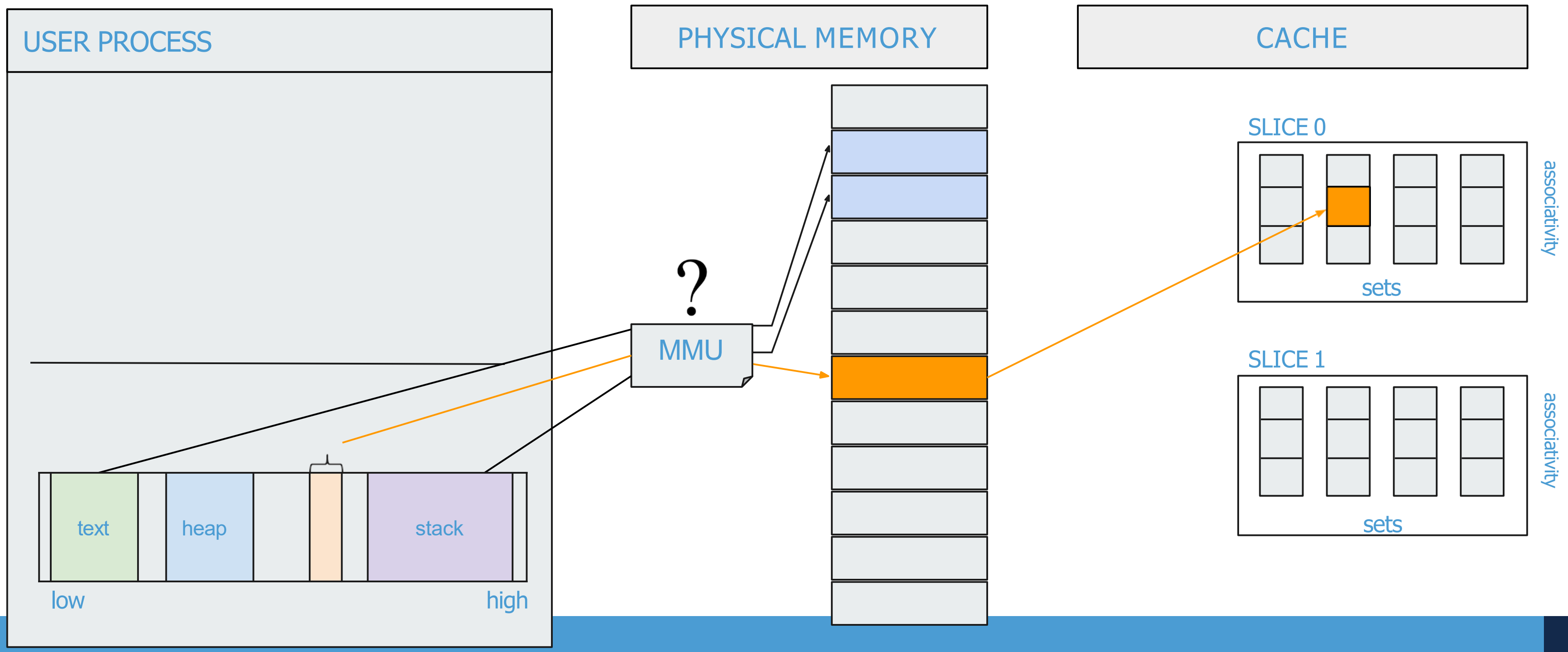


Rowhammer

Spectre

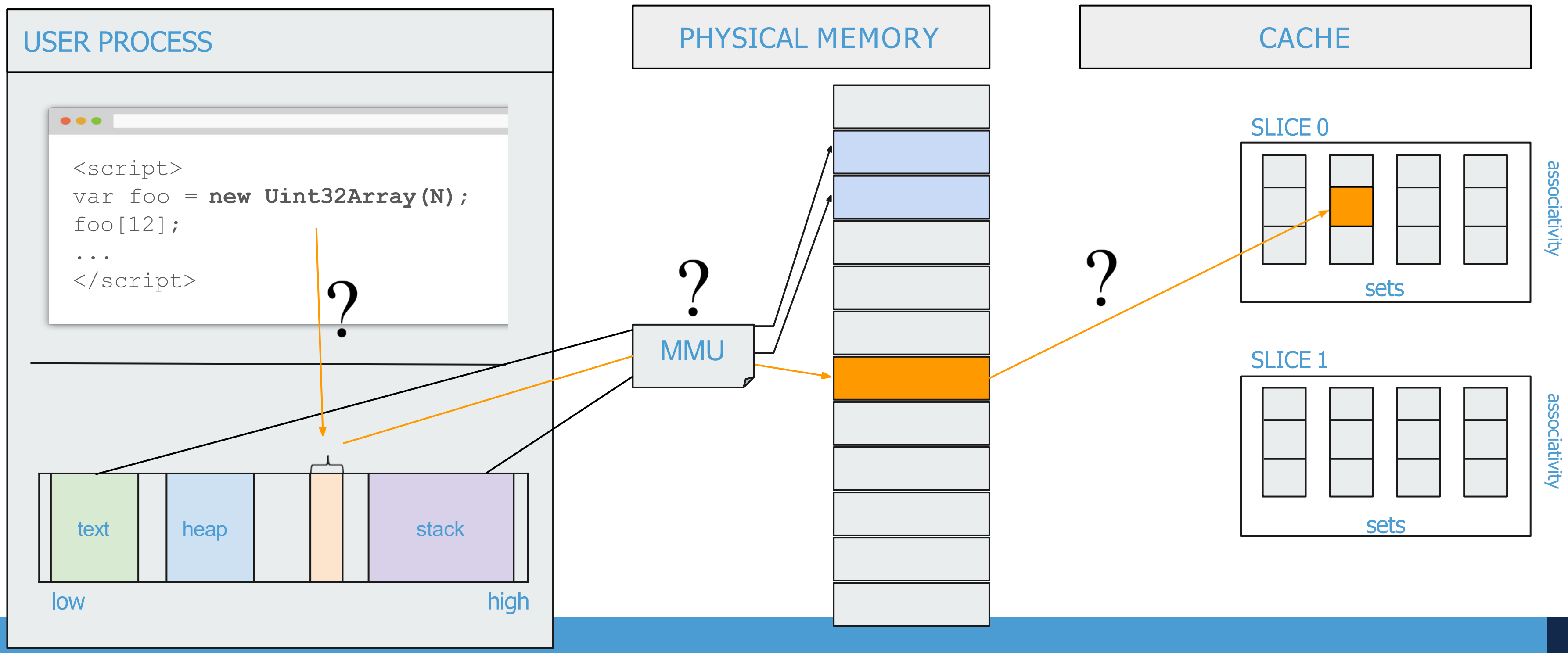
Problem

Unknown translation from virtual to physical addresses

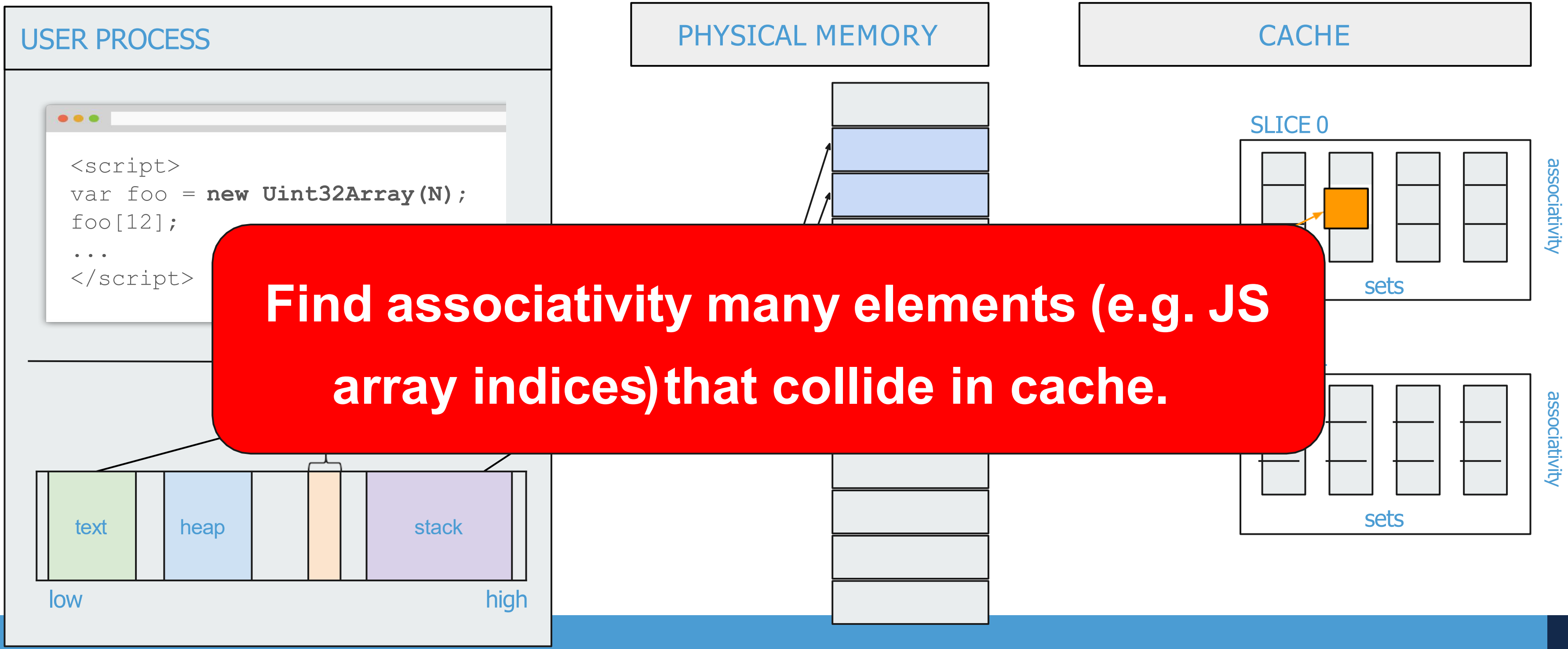


Problem

In some scenarios, even unknown virtual address



Problem



Find associativity many elements (e.g. JS array indices) that collide in cache.

Defintions

- Two virtual address x and y are congruent if they map to same cache set

  $x \simeq y$

- *The equivalence class [x] of x w.r.t.* $\simeq$ is the set of all virtual *addresses that maps to the same cache set as x*
- *set(·): set index bits*
- *pt(·): physical address*
- $x \simeq y$ *iff* $set(pt(x)) = set(pt(y))$

Definitions

- We say that a set of virtual addresses *S*, for a cache of associativity *a* is:
  - an *eviction set for* x if $x \notin S$ and at least a addresses in S map to the same cache set as x:

$$|[x] \cap S| \geq a$$

Eviction Test

- a_v is a *victim address* we want to evict
- Test to see if $S=\{a_0, a_1, \ldots , a_{n-1}\}$ is an eviction set for a_v

$$a_v \; a_0 \; a_1 \; \ldots \; a_n \; \boxed{a_v}$$

Finding minimal eviction sets

1

- Find a large eviction set for an address V:
  - Pick “enough” addresses at random

$a_v\ a_0\ a_1\ \ldots\ a_n\ \ a_v$

- Timing test:

Finding minimal eviction sets

1

- Find a large eviction set for an address V:
  - Pick “enough” addresses at random

$a_v\ a_0\ a_1\ \ldots\ a_n\ \boxed{a_v}$

- Timing test:

2

Reduce initial large eviction set into its minimal core

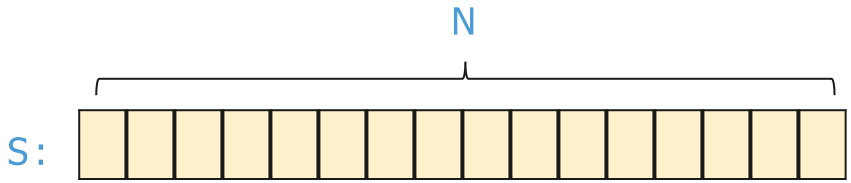
Baseline Algorithm

In: S=candidate set, x=victim address
Out: R=minimal eviction set for v

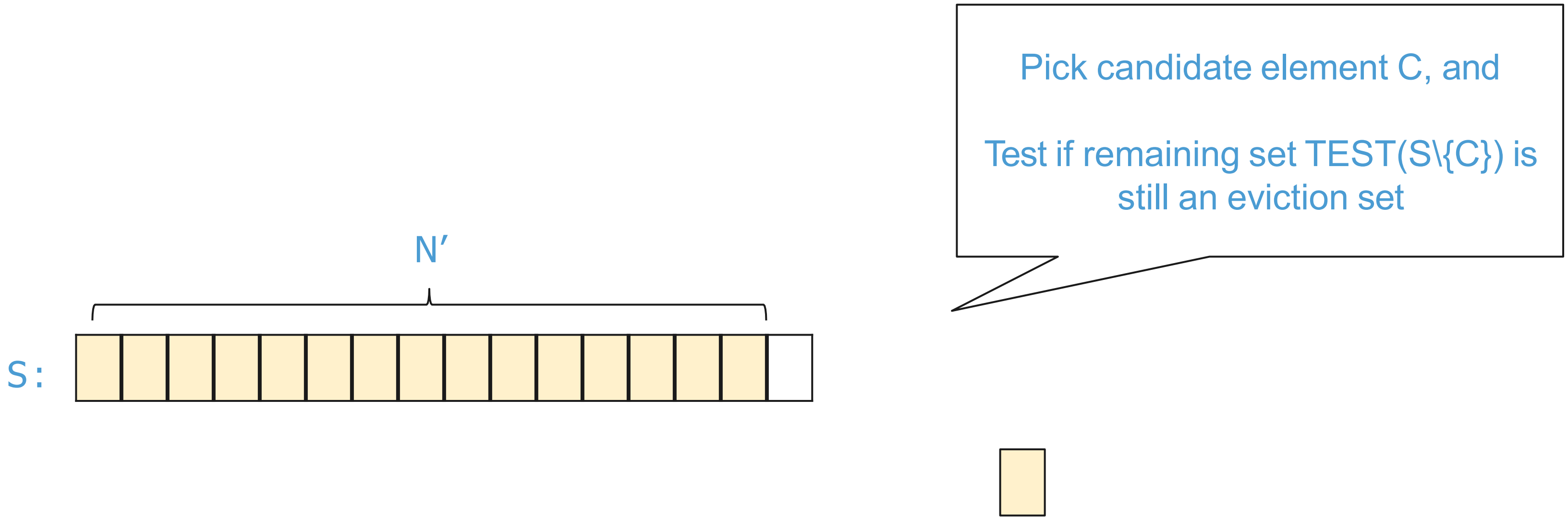
$$
\begin{array}{l}
R \leftarrow \{\} \\
\textbf{while } |R| < a \textbf{ do} \\
\quad c \leftarrow \mathit{pick}(S) \\
\quad \textbf{if } \neg\text{TEST}(R \cup (S \setminus \{c\}), x) \textbf{ then} \\
\quad\quad R \leftarrow R \cup \{c\} \\
\quad \textbf{end if} \\
\quad S \leftarrow S \setminus \{c\} \\
\textbf{end while} \\
\textbf{return } R
\end{array}
$$

Baseline algorithm

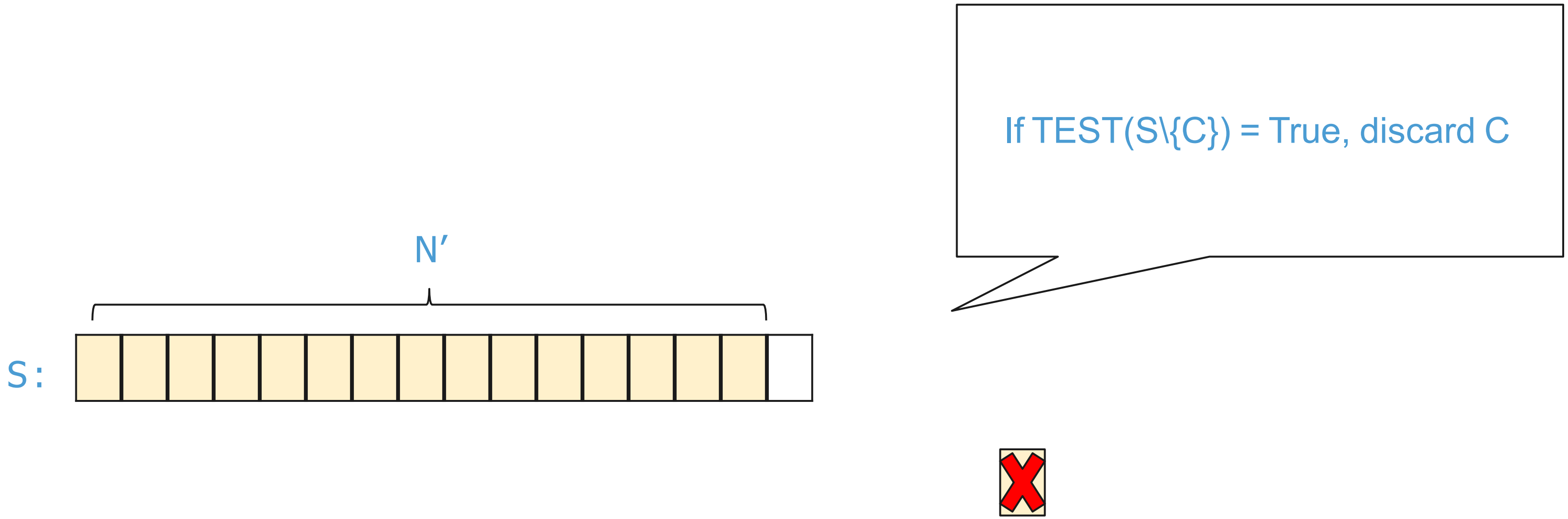
Start with large enough eviction set
S of size N



Baseline algorithm

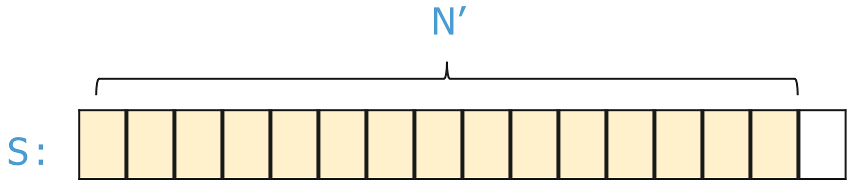


Baseline algorithm

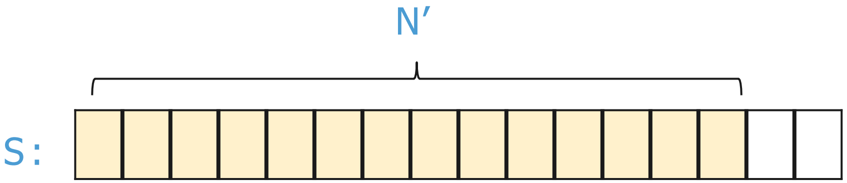


Baseline algorithm

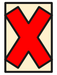
and continue with N’=N-1



Baseline algorithm

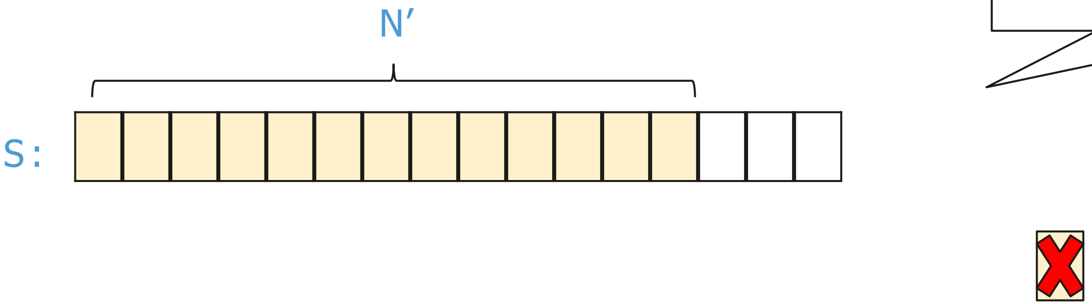


We repeat this process several times



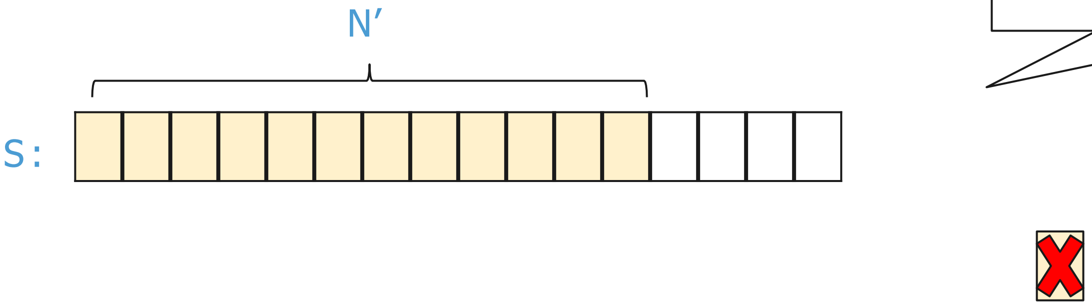
Baseline algorithm

We repeat this process several times



Baseline algorithm

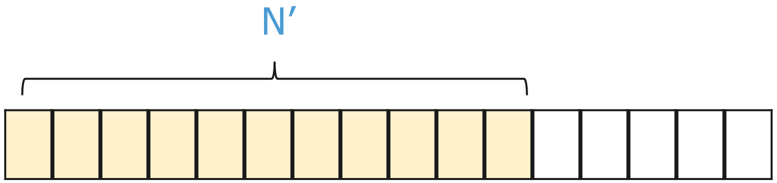
We repeat this process several times

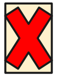


Baseline algorithm

We repeat this process several times

S:

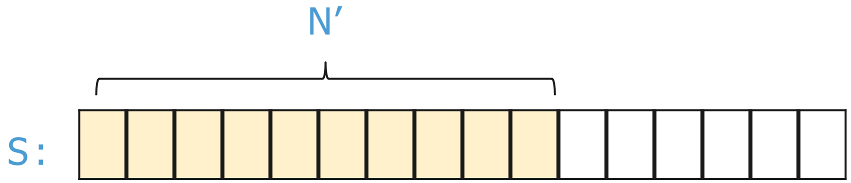


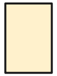


Baseline algorithm

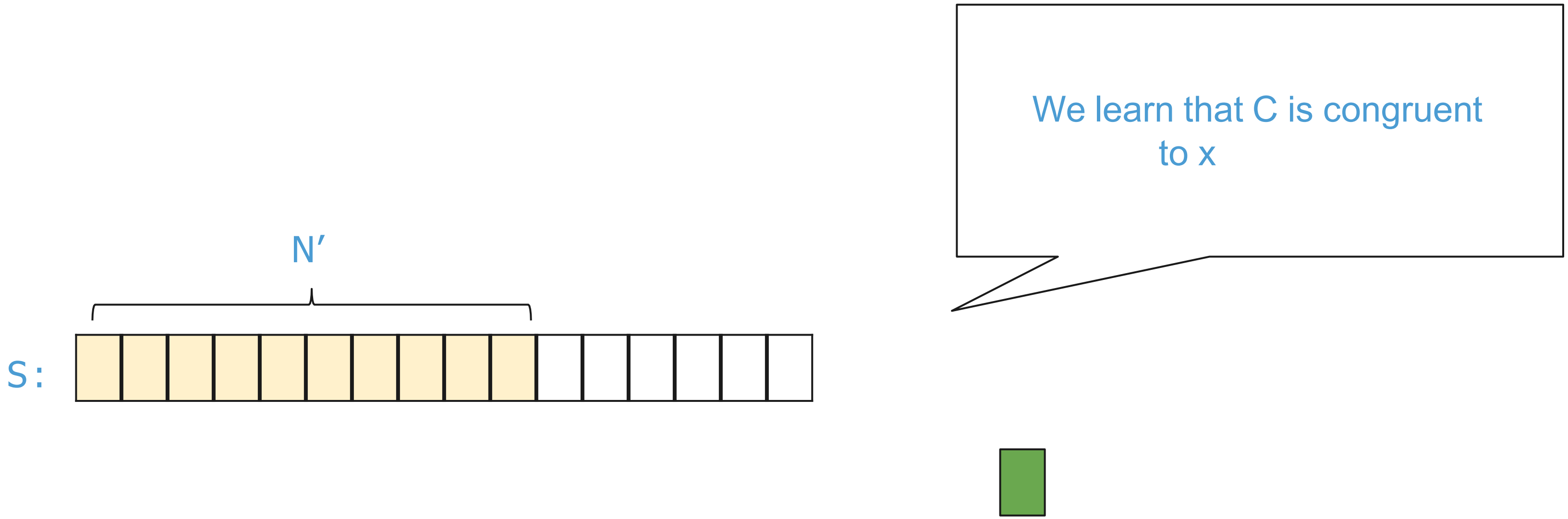
Until we find an element C such that when removed the remaining set stops being an eviction set:

TEST(S\{C}) = False



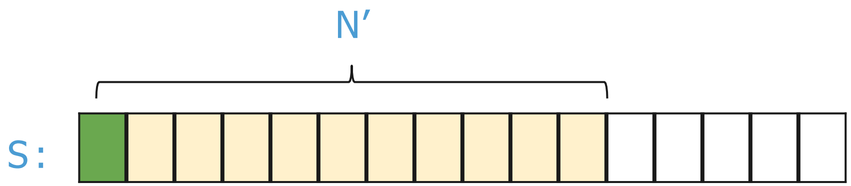


Baseline algorithm



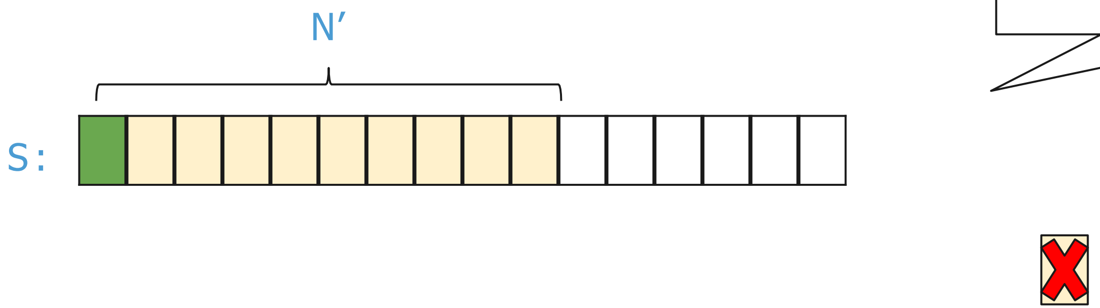
Baseline algorithm

We keep track of it, and insert it again in S

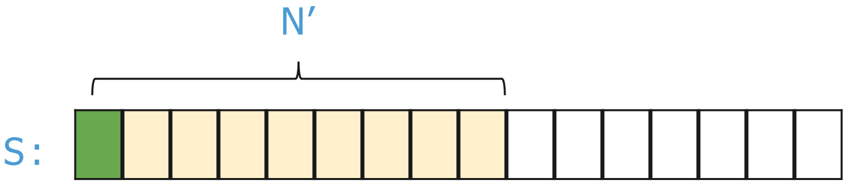


Baseline algorithm

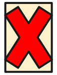
We repeat this process several times



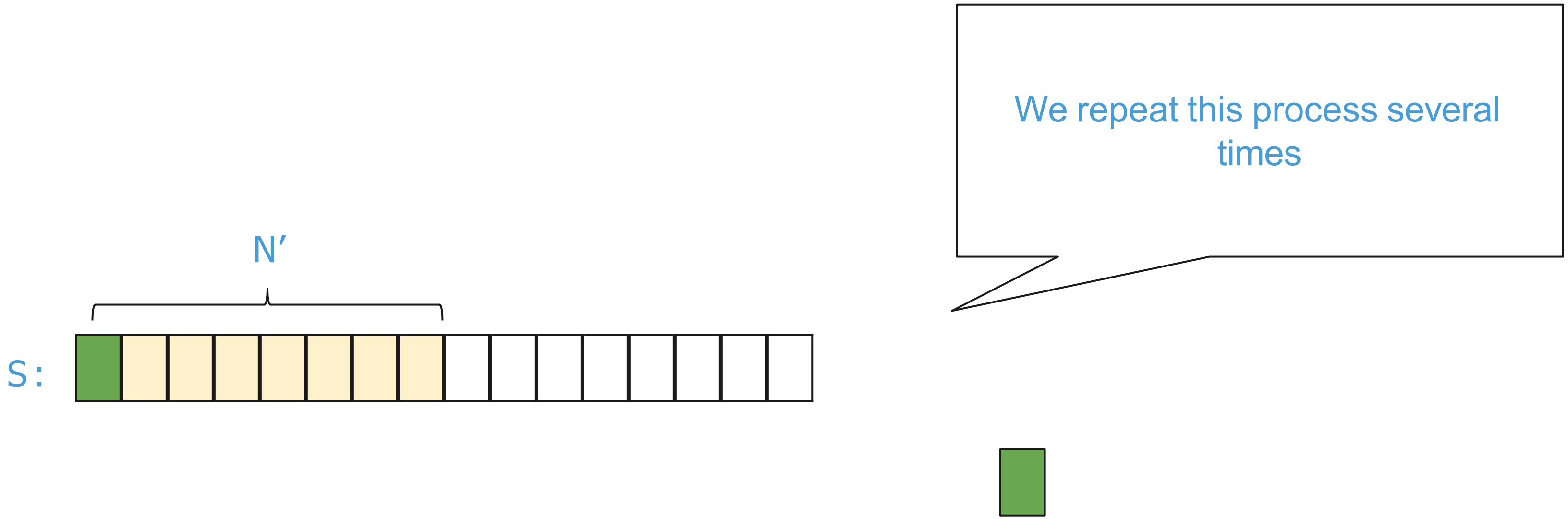
Baseline algorithm



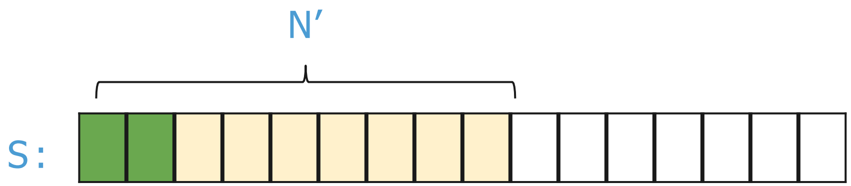
We repeat this process several times



Baseline algorithm



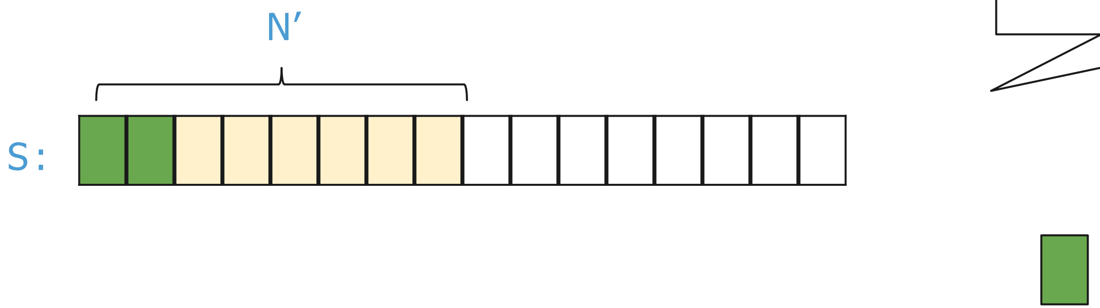
Baseline algorithm



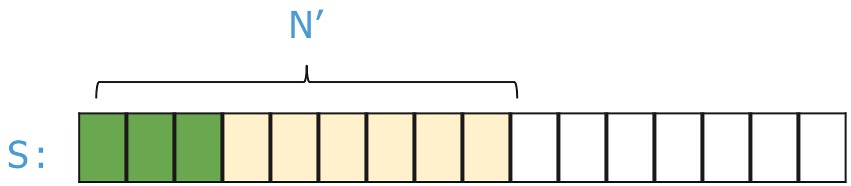
We repeat this process several times

Baseline algorithm

We repeat this process several times

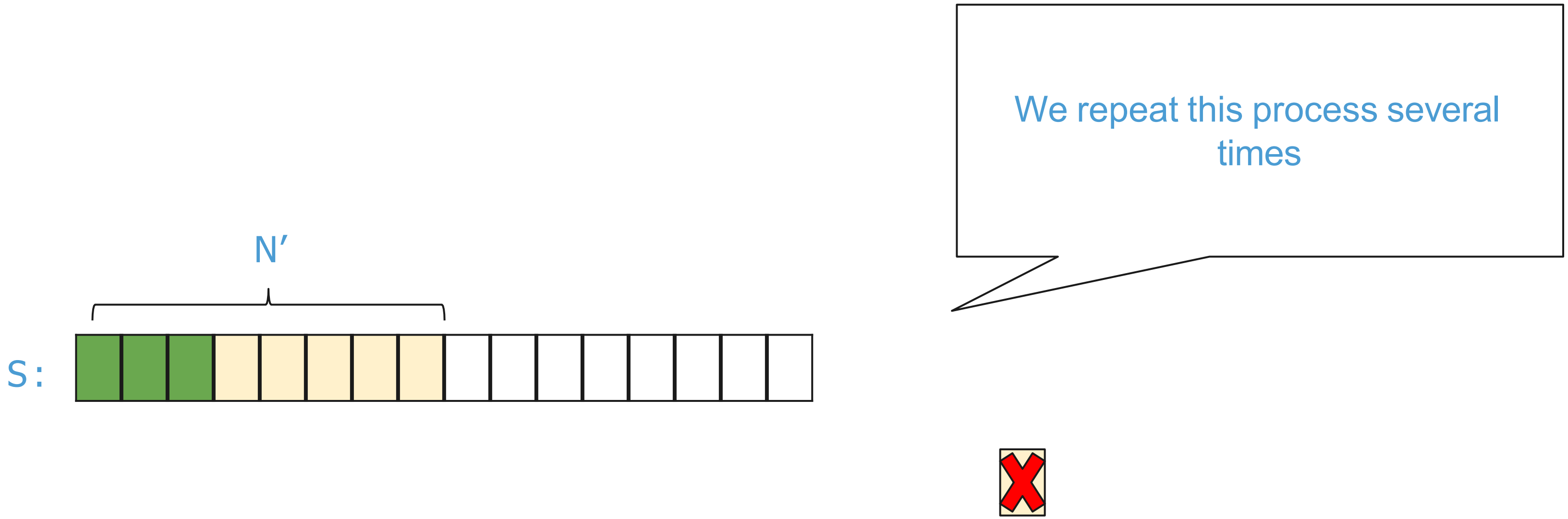


Baseline algorithm

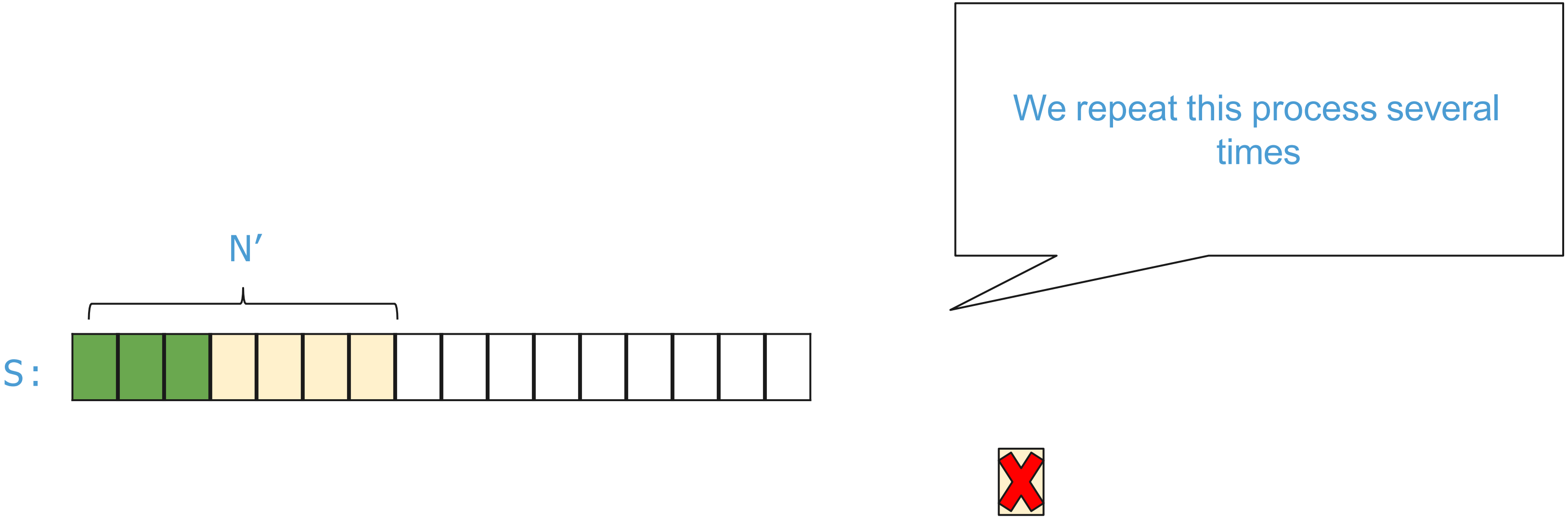


We repeat this process several times

Baseline algorithm

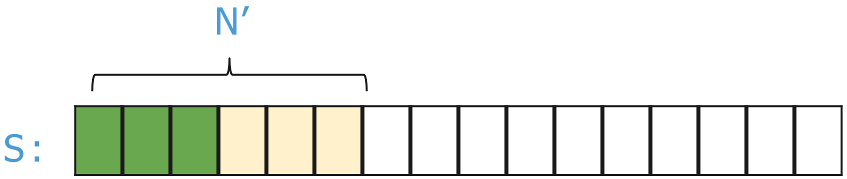


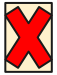
Baseline algorithm



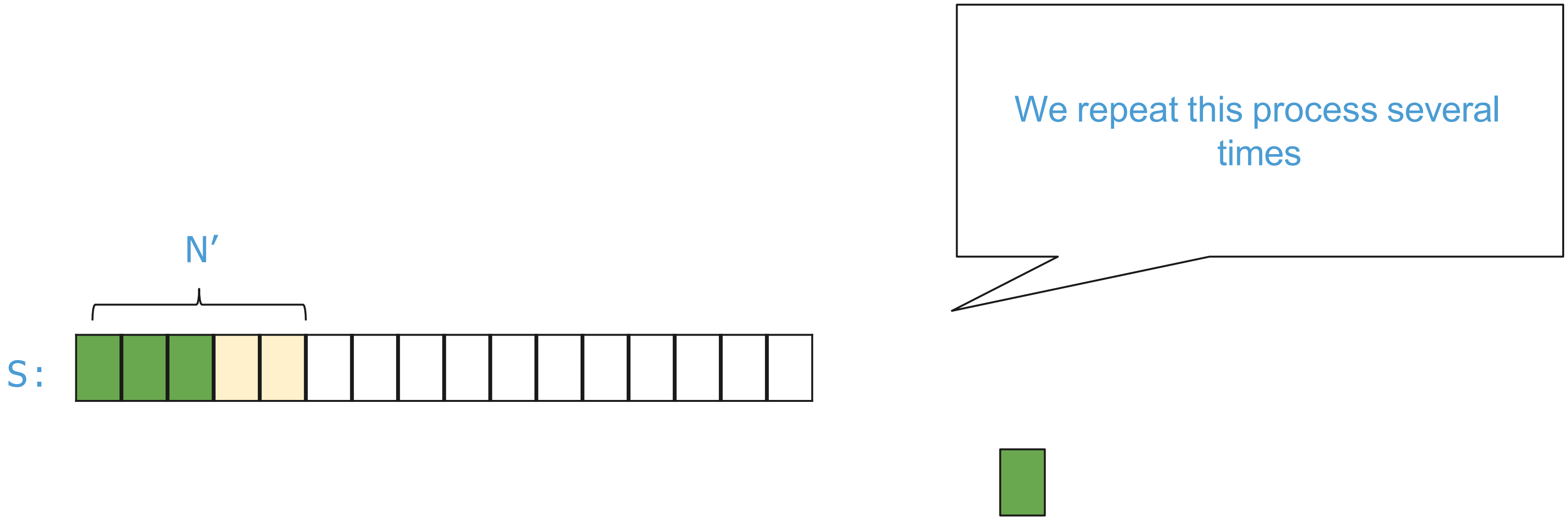
Baseline algorithm

We repeat this process several times

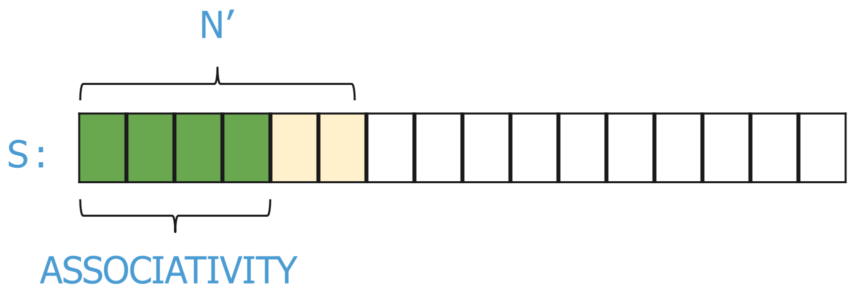




Baseline algorithm



Baseline algorithm



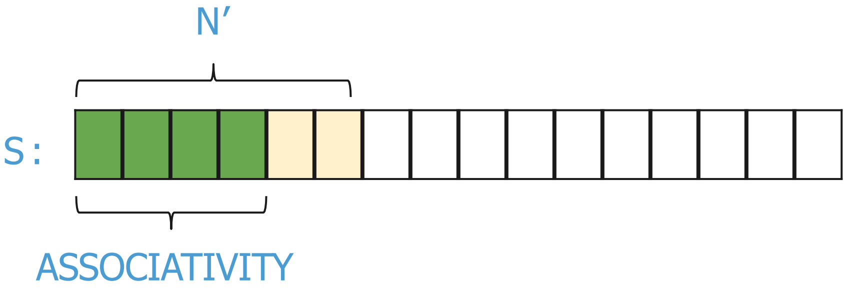
Until we have identified ASSOCIATIVITY many elements representing the eviction set’s core!

Baseline Algorithm

In: S=candidate set, x=victim address
Out: R=minimal eviction set for v

$R \leftarrow \{\}$
while $|R| < a$ **do**
    $c \leftarrow pick(S)$
    if $\neg\text{TEST}(R \cup (S \setminus \{c\}), x)$ **then**
        $R \leftarrow R \cup \{c\}$
    end if
    $S \leftarrow S \setminus \{c\}$
end while
return R

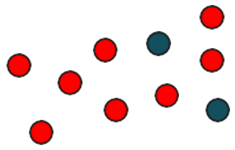
Baseline algorithm



$O(N^2)$ memory accesses

Threshold Group Testing

Group testing problem by Robert Dorfman (1943)

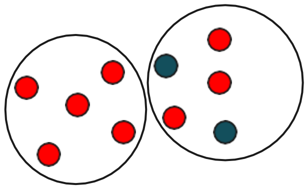


(10 individual tests)

Blood samples

Threshold Group Testing

Group testing problem by Robert Dorfman (1943)



(4 group tests +
3 individual tests)

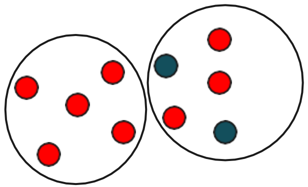
Blood samples

Threshold Group Testing

Group testing problem by Robert Dorfman (1943)

Generalization by Peter Damaschke (2006):

- Positive test only if at least "u" defectives
- Negative test only if at most "l" defectives
- Random otherwise



(4 group tests + 3 individual tests)

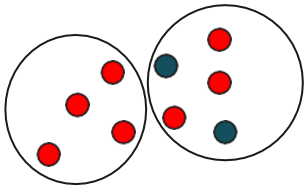
Blood samples

Threshold Group Testing

Group testing problem by Robert Dorfman (1943)

Generalization by Peter Damaschke (2006):

- Positive test if at least "u" defectives
- Negative test if at most "l" defectives
- Random answer otherwise



(4 group tests + 3 individual tests)

Blood samples

Observation: Our test is a threshold group test!

Key idea

- Start with huge eviction set S for cache of associativity *a*
  - partition S into a + 1 disjoint subsets $T_1, \ldots, T_a+1$ of (approximately) the same size.
  - A counting argument shows least one $j \in \{1, \ldots, a+1\}$ such that $S \setminus T_j$ is still an eviction set

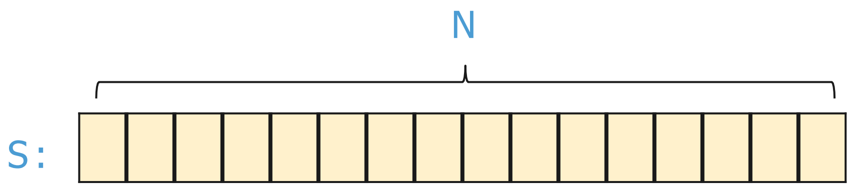
Improved Algorithm

In : S=candidate set, x=victim address
Out : R=minimal eviction set for x

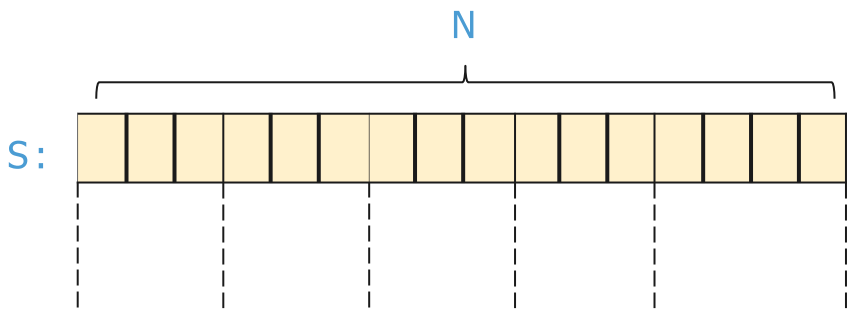
while $|S| > a$ **do**
    $\{T_1, ..., T_{a+1}\} \leftarrow split(S,\ a+1)$
    $i \leftarrow 1$
    while $\neg\text{TEST}(S \setminus T_i,\ x)$ **do**
        $i \leftarrow i + 1$
    end while
    $S \leftarrow S \setminus T_i$
end while
return S

Group-testing algorithm

Start with large enough eviction set
S of size N

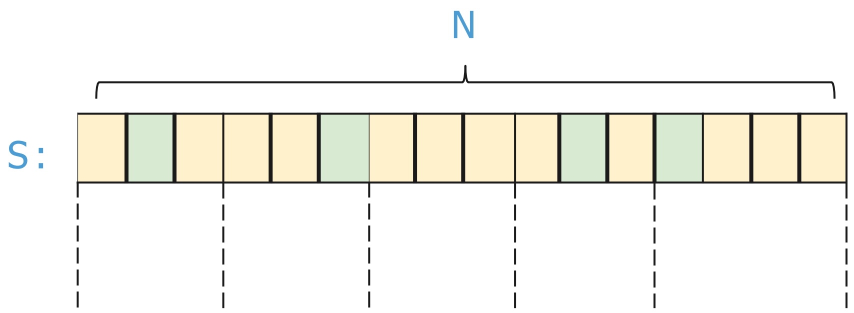


Group-testing algorithm



Split S in ASSOCIATIVITY+1 subsets

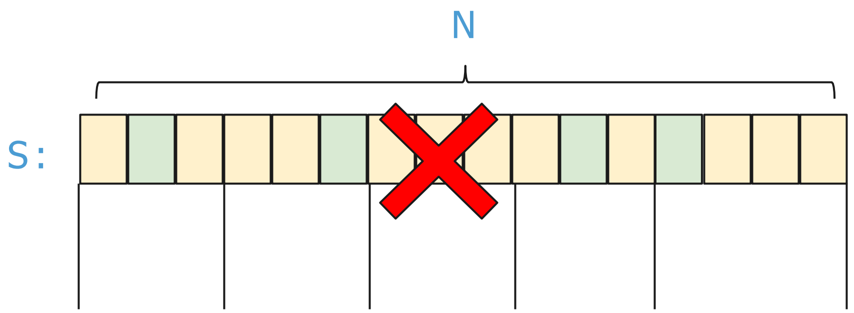
Group-testing algorithm



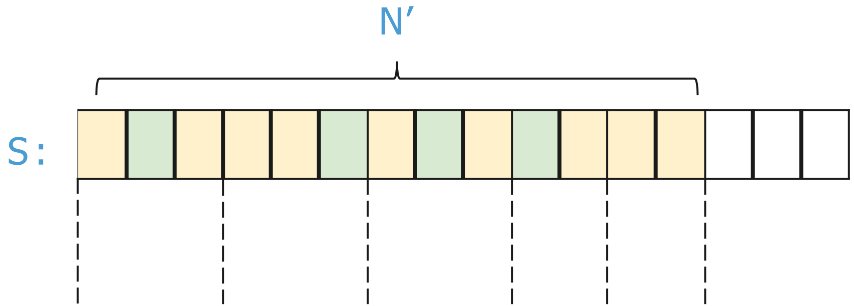
In the worst case, there exists a union of ASSOCIATIVITY subsets being an eviction set

Group-testing algorithm

We can discard N/(ASSOCIATIVITY+1) elements per iteration



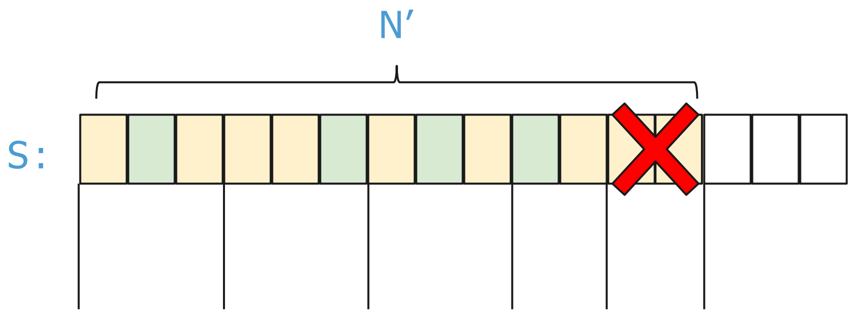
Group-testing algorithm



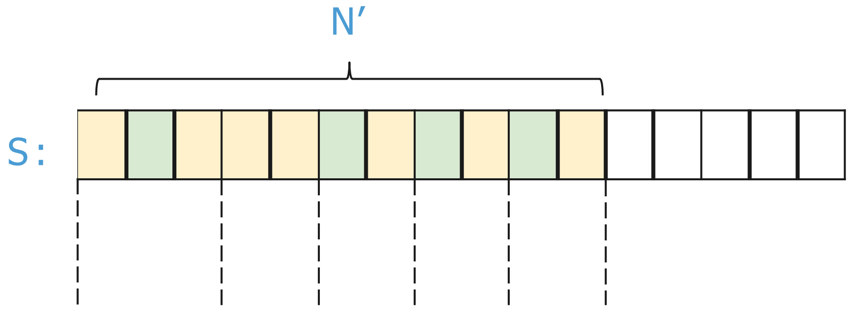
We repeat this process until we have ASSOCIATIVITY many elements

Group-testing algorithm

We repeat this process until we have ASSOCIATIVITY many elements



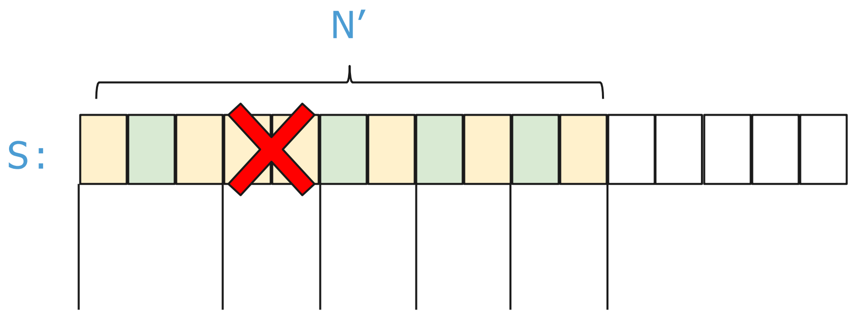
Group-testing algorithm



We repeat this process until we have ASSOCIATIVITY many elements

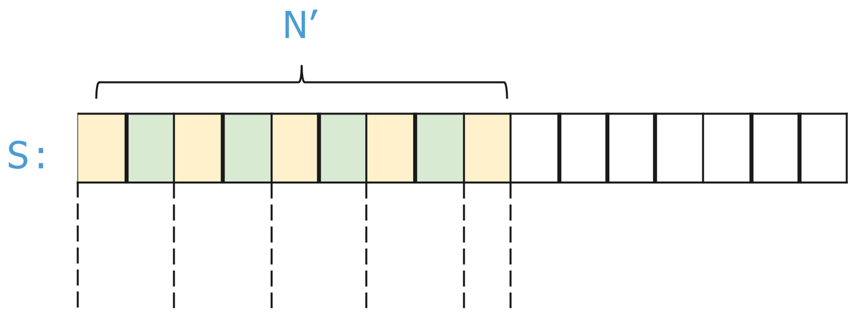
Group-testing algorithm

We repeat this process until we have ASSOCIATIVITY many elements

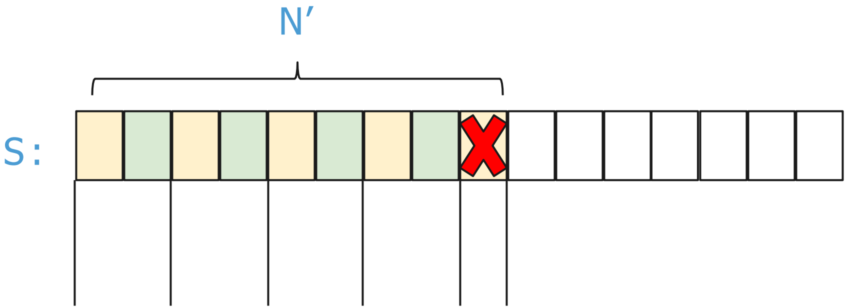


Group-testing algorithm

We repeat this process until we have ASSOCIATIVITY many elements



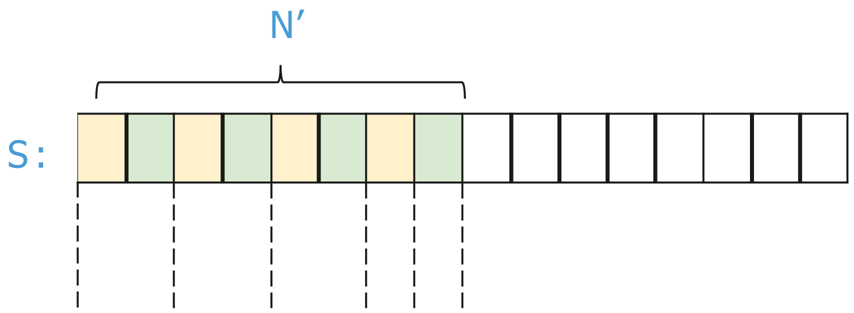
Group-testing algorithm



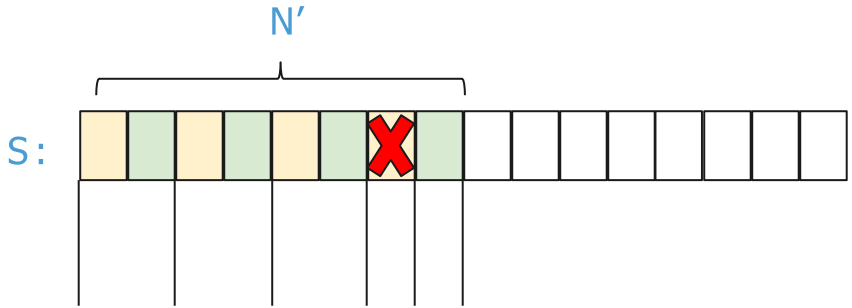
We repeat this process until we have ASSOCIATIVITY many elements

Group-testing algorithm

We repeat this process until we have ASSOCIATIVITY many elements



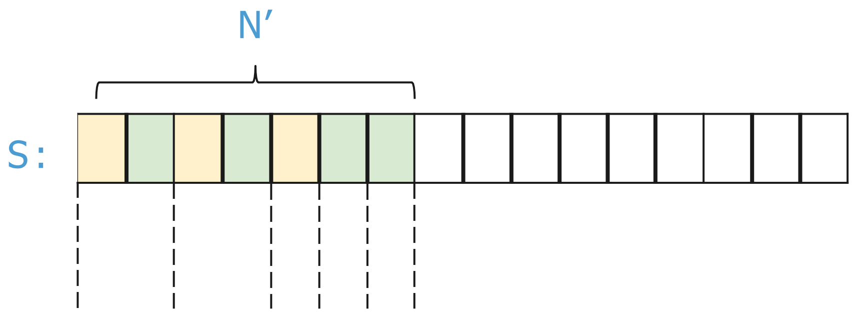
Group-testing algorithm



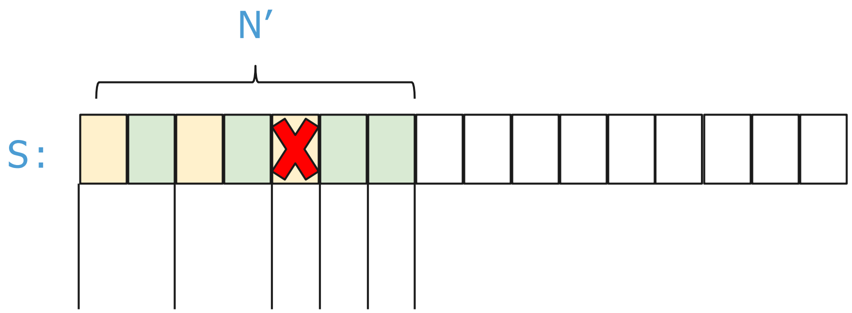
We repeat this process until we have ASSOCIATIVITY many elements

Group-testing algorithm

We repeat this process until we have ASSOCIATIVITY many elements

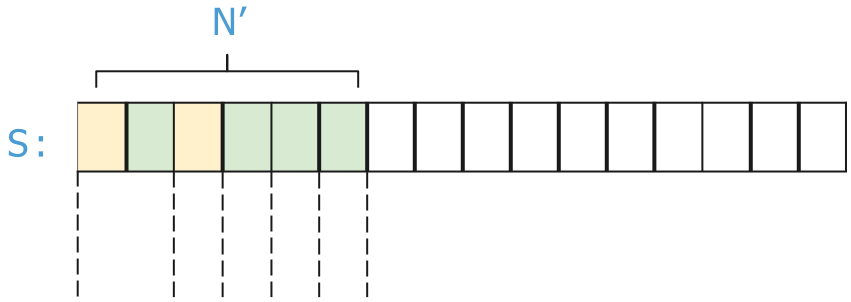


Group-testing algorithm



We repeat this process until we have ASSOCIATIVITY many elements

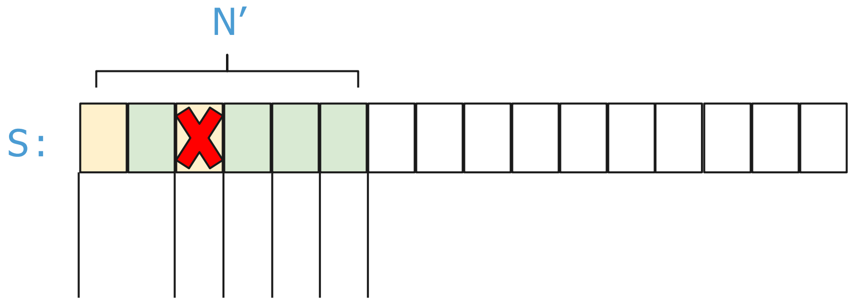
Group-testing algorithm



We repeat this process until we have ASSOCIATIVITY many elements

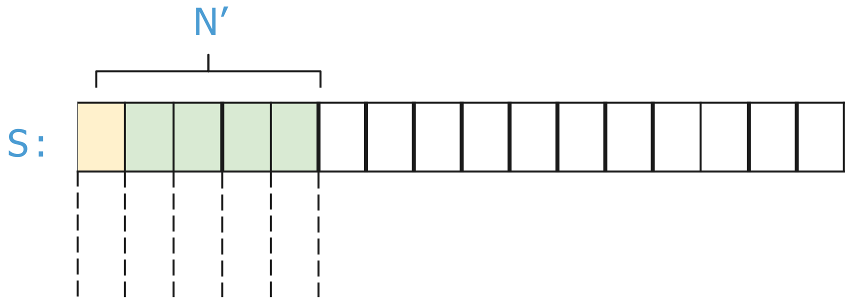
Group-testing algorithm

We repeat this process until we have ASSOCIATIVITY many elements

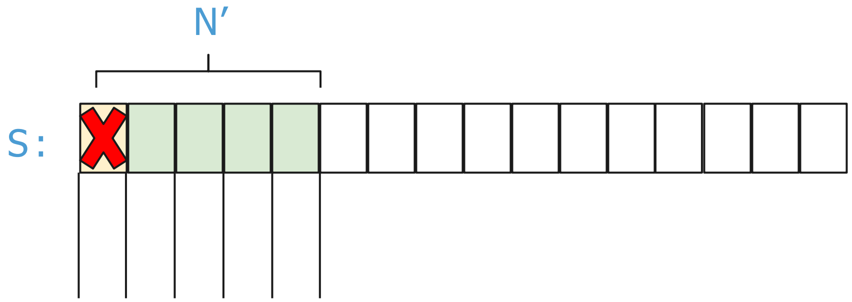


Group-testing algorithm

We repeat this process until we have ASSOCIATIVITY many elements



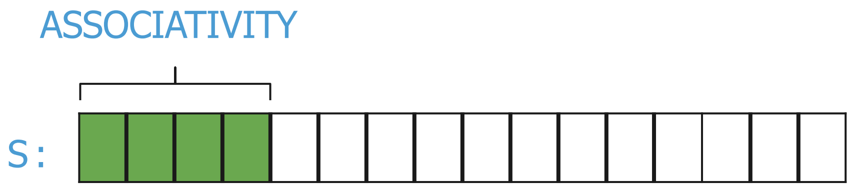
Group-testing algorithm



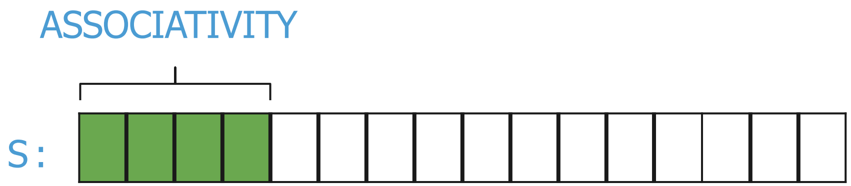
We repeat this process until we have ASSOCIATIVITY many elements

Group-testing algorithm

We find our minimal eviction set!



Group-testing algorithm



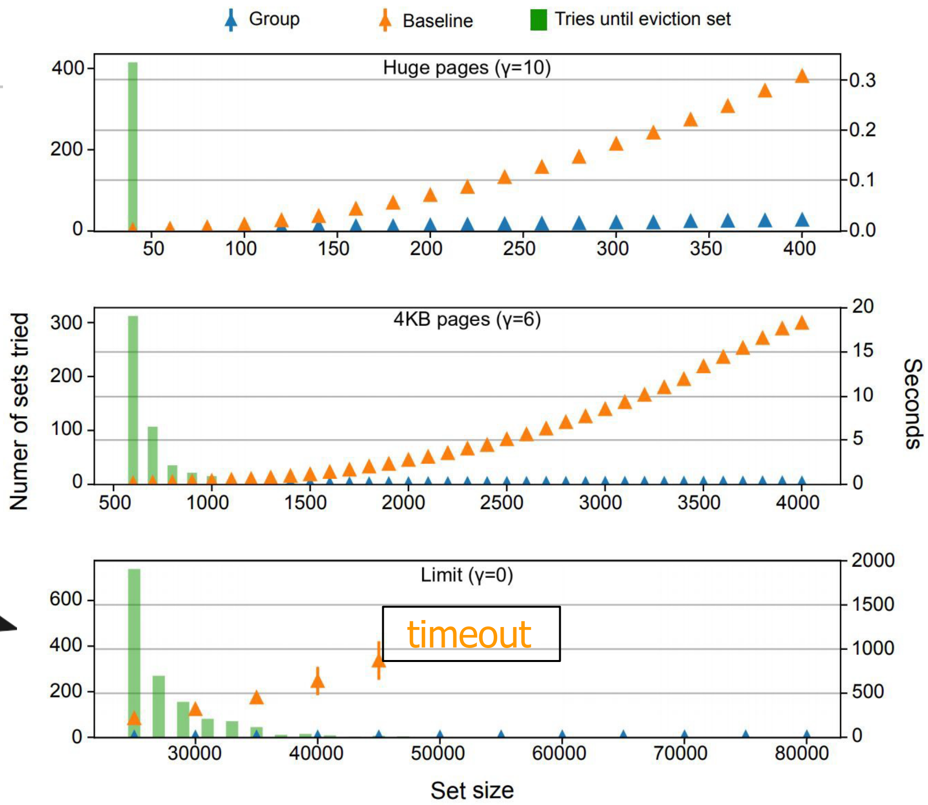
O(N) mem accesses

Performance Evaluation

O(n) vs. O(n^2) advantage shows up in practice!

Finding minimal eviction sets is practical without knowledge on any bits of the set index!

Experiments on Skylake i5-6500 with 6MB cache (8192 sets x 12 assoc)



Y-right (lines): Average running time for eviction set reduction
Y-left (columns): Cost of finding an initial eviction set of certain size

Conclusions

Finding minimal eviction sets is a threshold group-testing problem: new insight for research on principled countermeasures

Novel linear-time algorithm makes attacks faster and enables them in scenarios previously considered impractical

THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL