ALGORITHMS WITH PREDICTIONS

What Works and What Doesn't

Francesco Visona'
MENTOR: Matteo Ceccarello

Overview

Basic ideas

OMLA model - Goals

What works and what doesn’t


01
BASIC
IDEAS

Motivation

Ars Inveniendi: on the Process of Discovery in Mathematics and the Natural Sciences

Immagining how such a process could have unfolded, even just in part and even if the imagined path does not correspond to the actual history, leads to a deeper understanding of the subject itself and better prepares for the investigation of new territories.

Motivating example: Search in a sorted array

Problem

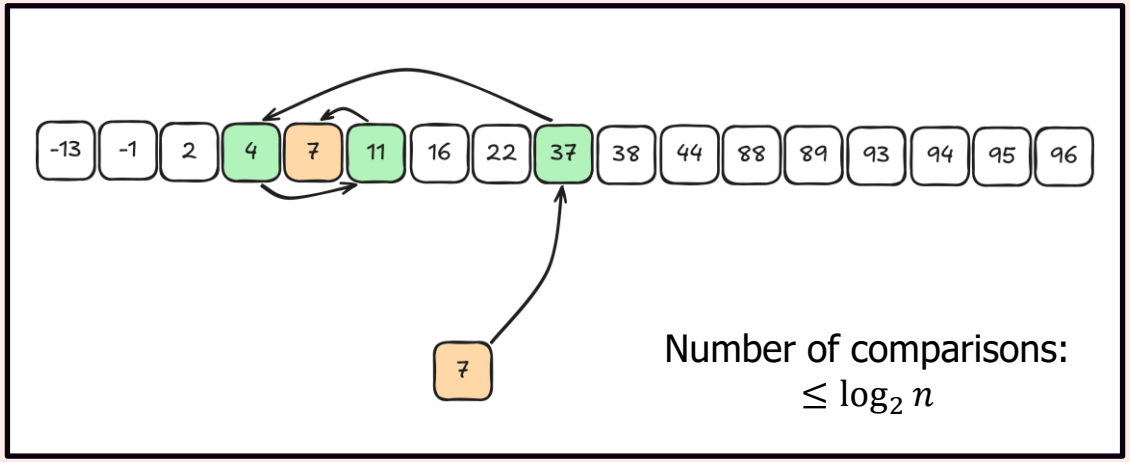
Given a sorted array of integers $A[1 \ldots n]$, and a query $q \in \mathbb{Z}$ find the index of q in the array.

Motivating example: Search in a sorted array

Problem

Given a sorted array of integers $A[1 \dots n]$, and a query $q \in \mathbb{Z}$ find the index of q in the array.

Binary search



Motivating example: Search in a sorted array

Problem

Given a sorted array of integers $A[1 \ldots n]$, and a query $q \in \mathbb{Z}$ find the index of q in the array.

Binary search

Can we make less comparisons?

Motivating example: Search in a sorted array

Motivating example: Search in a sorted array

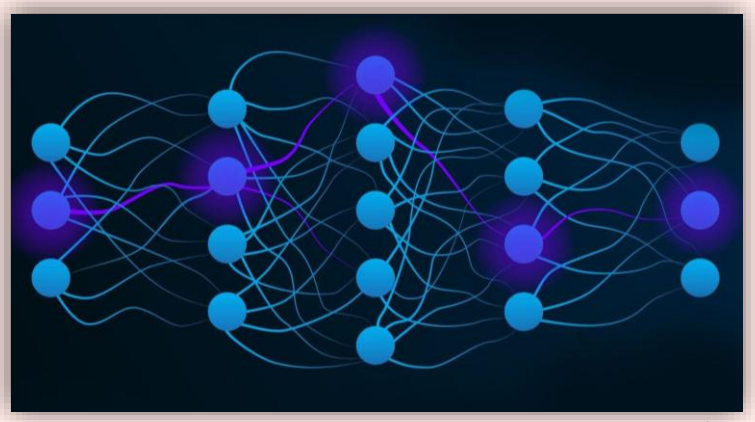
1st IDEA:

If someone tells us the correct position,
we can make only one comparison and obtain
optimal performances.

Motivating example: Search in a sorted array

1st IDEA:

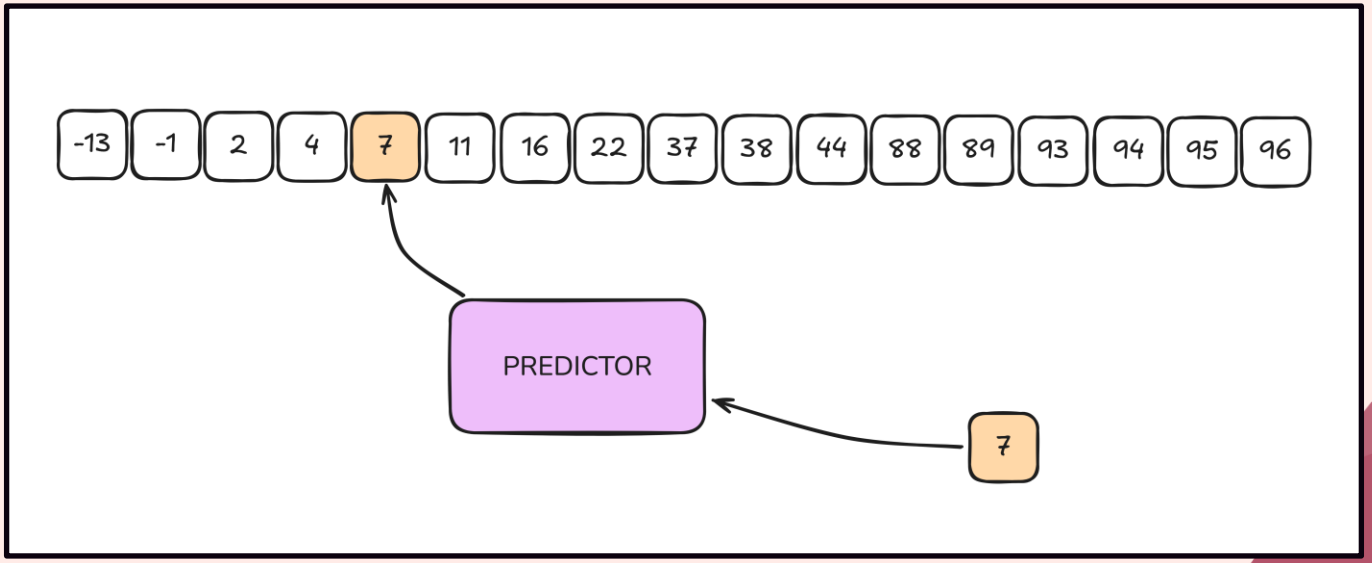
If someone tells us the correct position,
we can make only one comparison and obtain
optimal performances.



Motivating example: Search in a sorted array

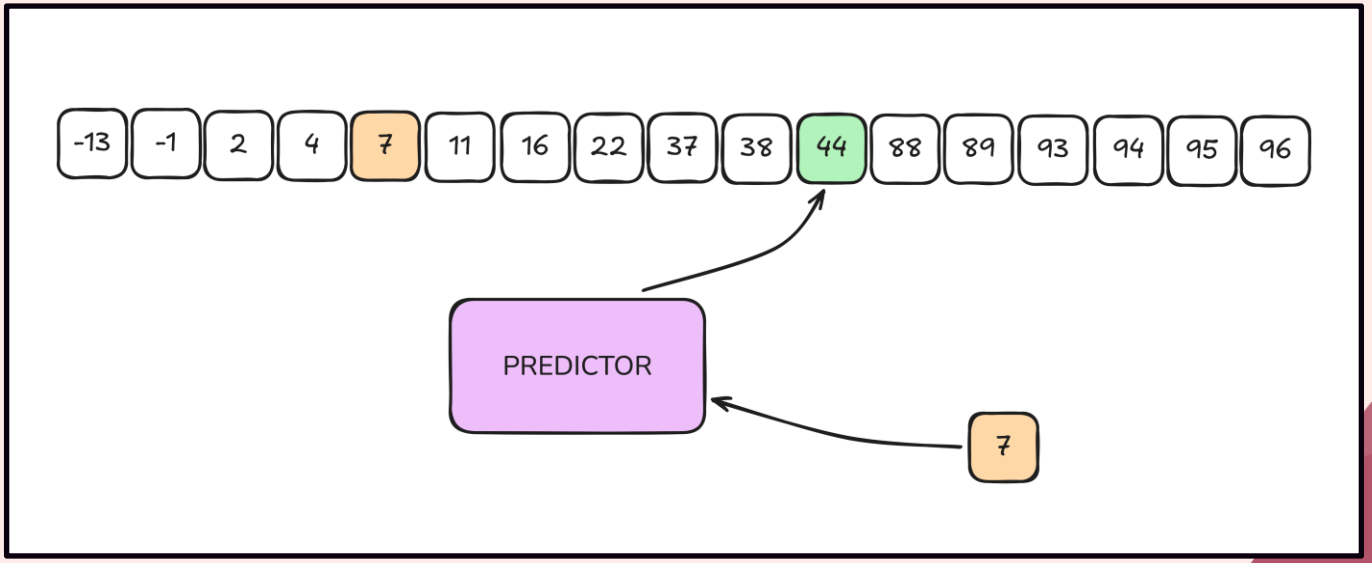
Algorithm

Check if the query element q is in the predicted position. If not, it means $q \notin A$.



Motivating example: Search in a sorted array

What happens if the predictor is wrong?



Motivating example: Search in a sorted array

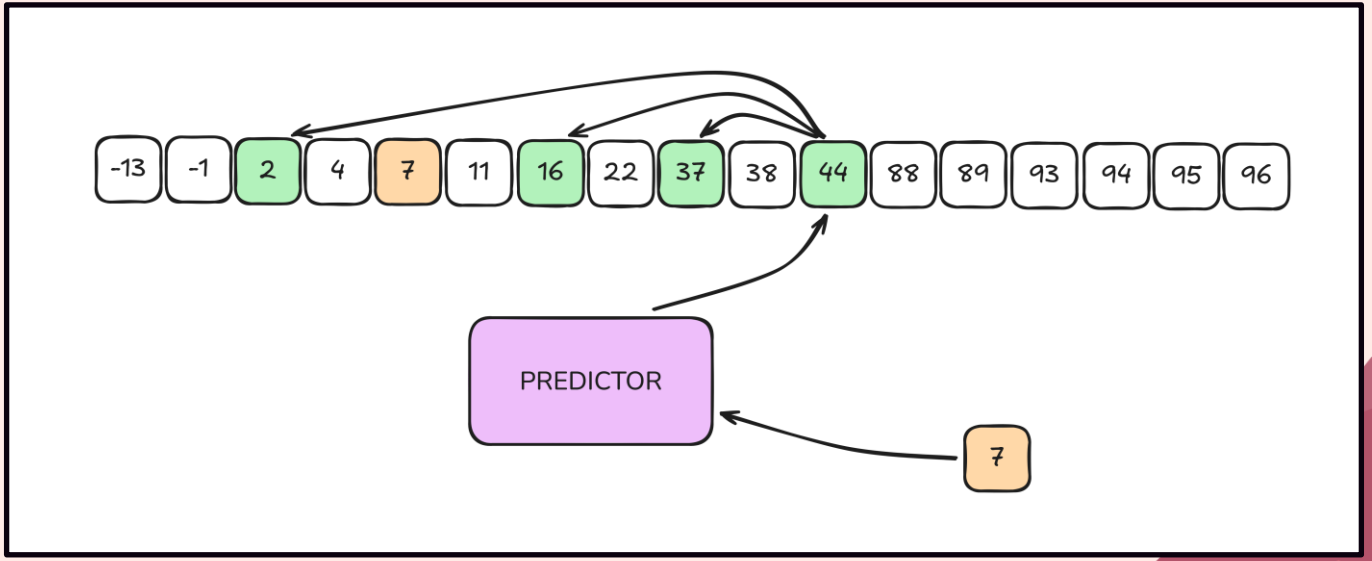
2nd IDEA:

If the predictor is wrong, our algorithm shouldn't fail or run the standard binary search, but exploit the prediction in some other way to find the item.

Motivating example: Search in a sorted array

Algorithm

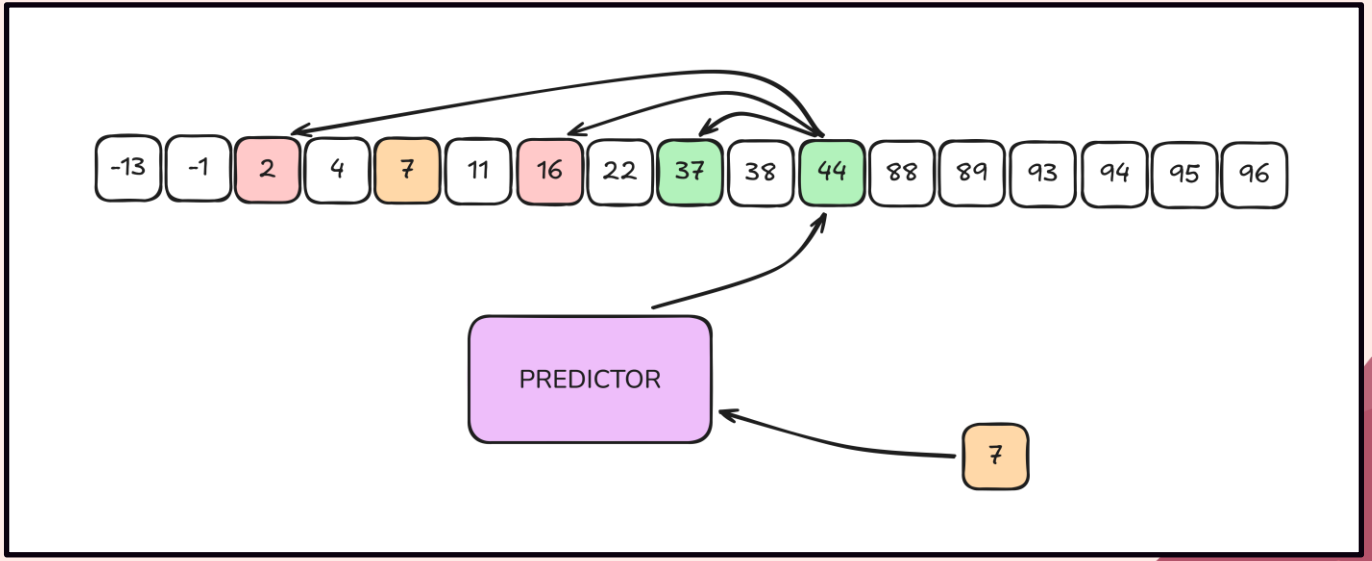
1. If the element in the predicted position is not the query element q, assess whether it's smaller or larger and proceed by doubling search (query at distance 2^i for $i = 1,2,..$) until a range is found;



Motivating example: Search in a sorted array

Algorithm

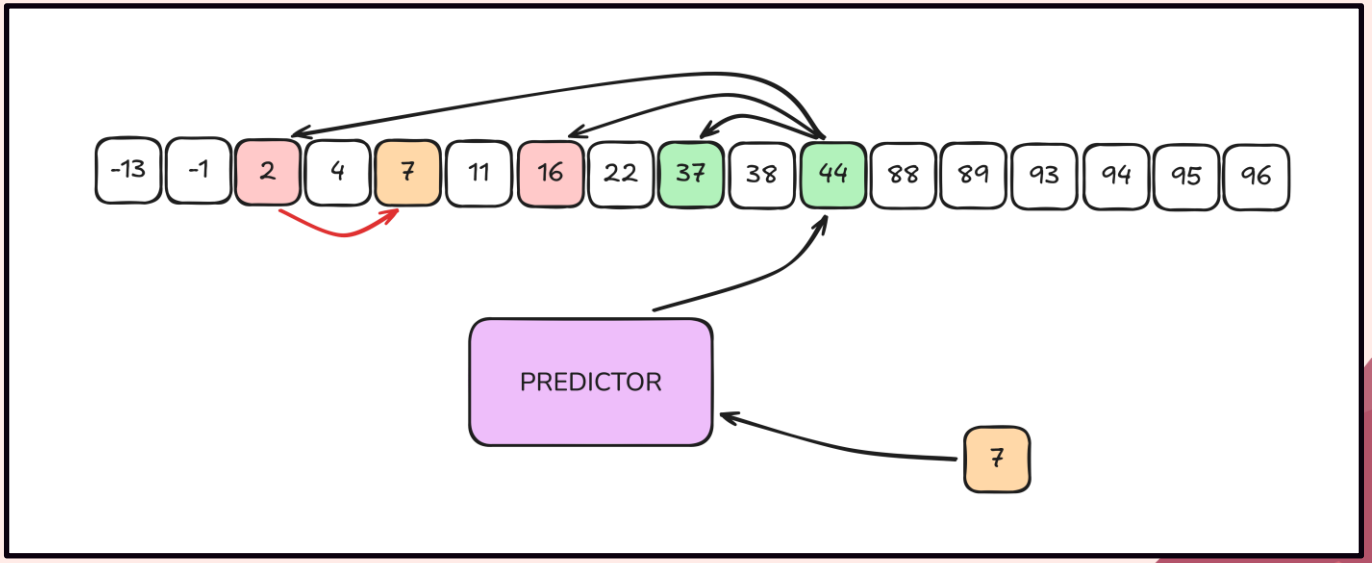
1. If the element in the predicted position is not the query element q, assess whether it's smaller or larger and proceed by doubling search (query at distance 2^i for $i = 1,2,..$) until a range is found;



Motivating example: Search in a sorted array

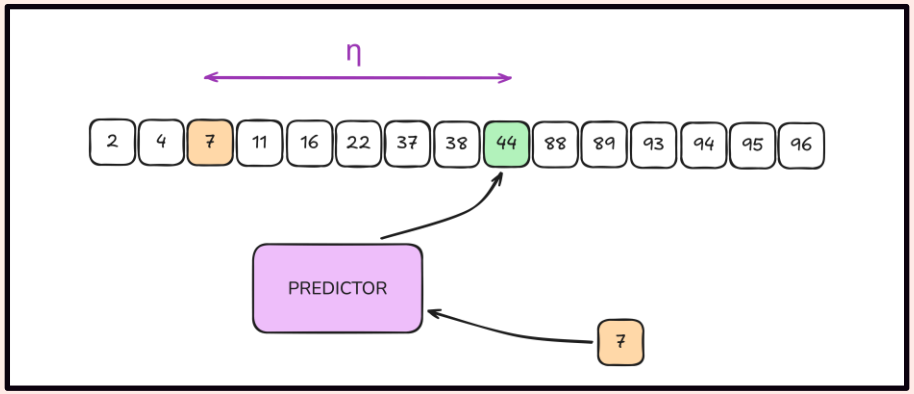
Algorithm

2. Then, proceed with binary search on that range.



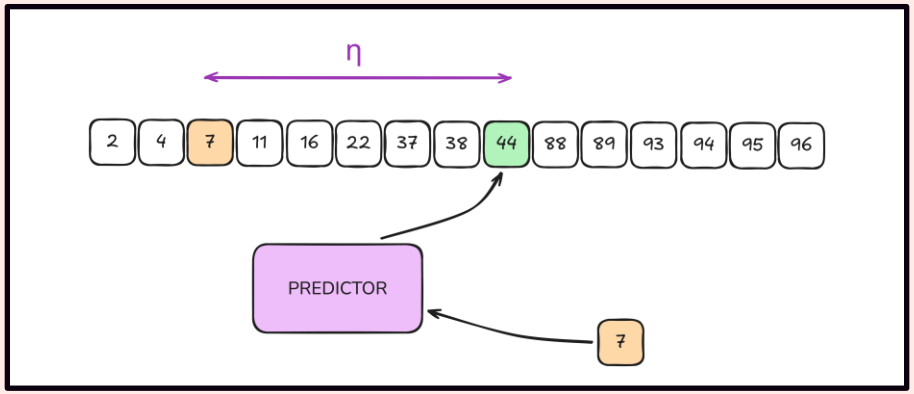
Motivating example: Search in a sorted array

Does it make less comparisons than binary search?



Motivating example: Search in a sorted array

Does it make less comparisons than binary search?



3rd IDEA:

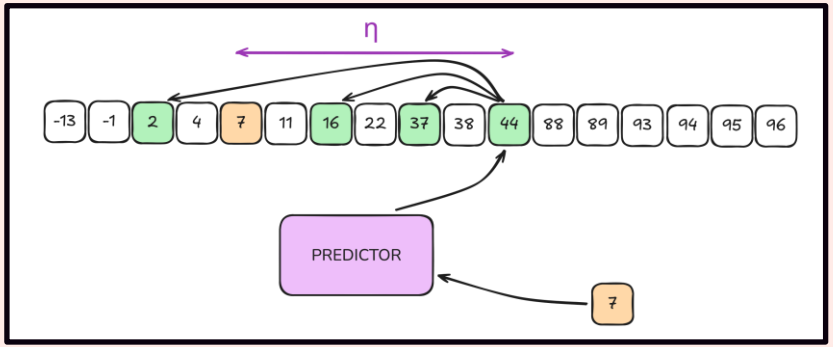
Since performance depends on prediction accuracy, we study it as a function of the prediction error.

Motivating example: Search in a sorted array

1st phase: Doubling Search

Comparisons:

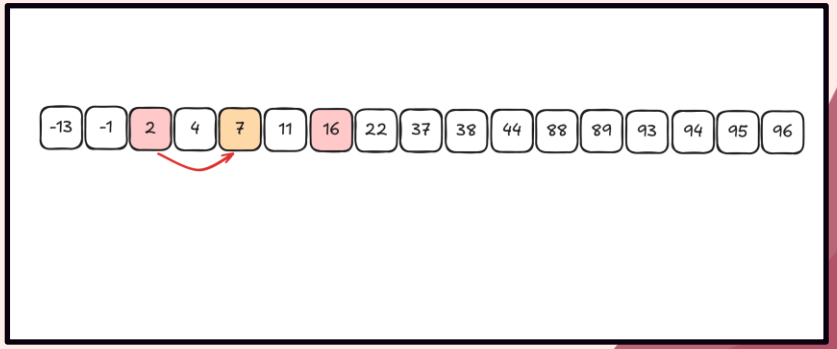
$\leq \log_2 \eta$



2nd phase: Binary Search

Comparisons:

$\leq \log_2 \eta$

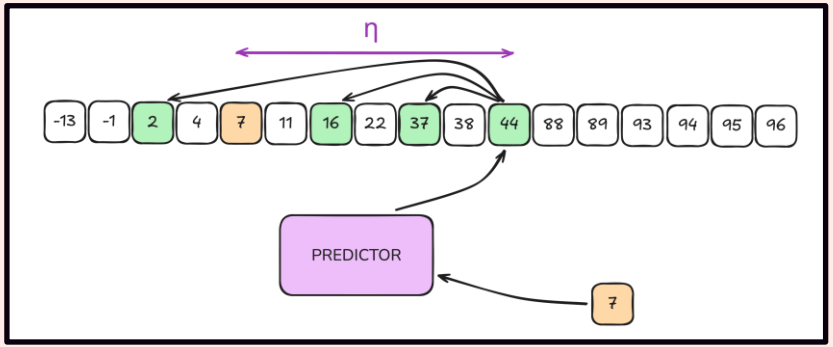


$$\text{If } \eta \leq \sqrt{n} \implies 2\log_2 \eta \leq \log_2 n$$

Motivating example: Search in a sorted array

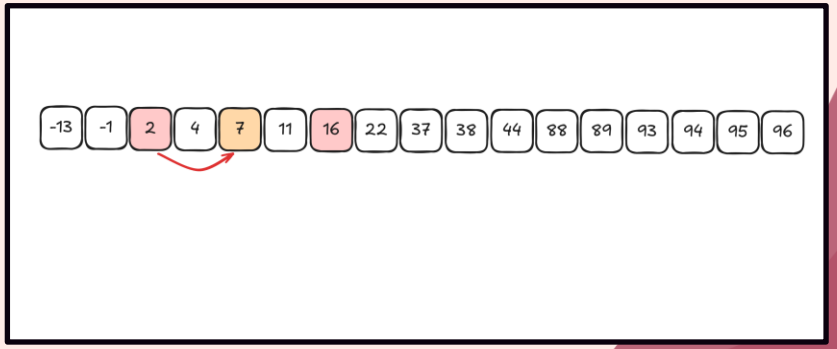
1st phase: Doubling Search

Comparisons:
$\leq \log_2 \eta$



2nd phase: Binary Search

Comparisons:
$\leq \log_2 \eta$



Always $\eta \leq n \implies 2\log_2 \eta \leq 2\log_2 n$

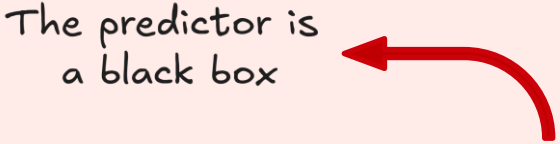
Take-away: Ideas

1st IDEA:
use a predictor to obtain better performances for our algorithms.

2nd IDEA:
integrate the predictions into an algorithm to obtain good solutions even if the predictor is wrong.

3rd IDEA:
study the performances of the algorithm with respect to the prediction error.

Take-away: Ideas



1st IDEA:

use a predictor to obtain better performances for our algorithms.

2nd IDEA:

integrate the predictions into an algorithm to obtain good solutions even if the predictor is wrong.

3rd IDEA:

study the performances of the algorithm with respect to the prediction error.

02

OMLA MODEL

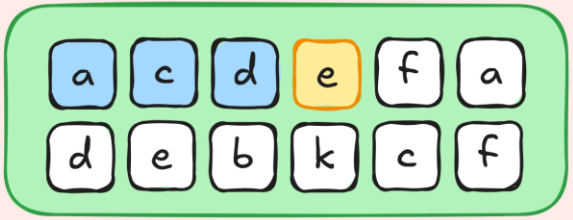
Online algorithms

OMLA = **Online** with Machine Learned Advice

Offline

Process the whole dataset and make decision knowing all the data

Online



Process dataset piece-by-piece and make decisions without knowing future data

Online algorithms

Competitive Ratio

$$\boldsymbol{COMPETITIVE\ RATIO} = \frac{c(ONL)}{c(OFF)}$$

Online algorithms

Competitive Ratio

$$\boldsymbol{COMPETITIVE\ RATIO} = \frac{c(ONL)}{c(OFF)} \leq \gamma$$

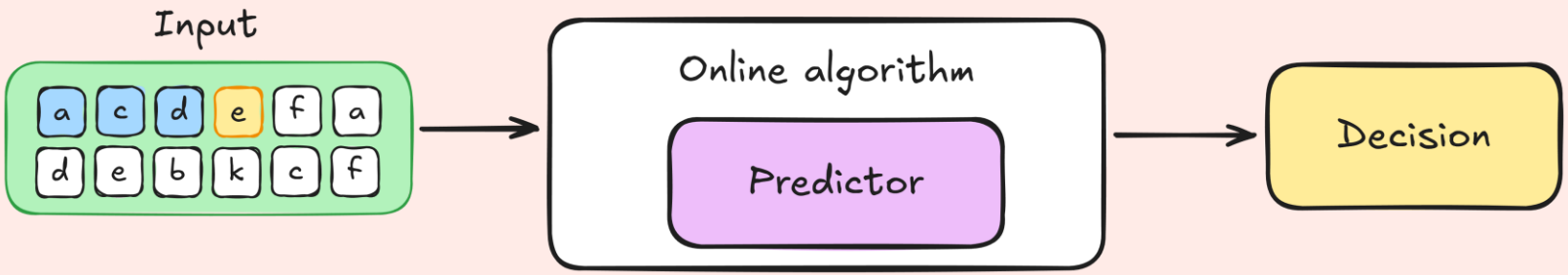
An online algorithm is ***γ* competitive** if the cost of its solution is at most γ times the cost of the corresponding offline algorithm.

OMLA model

Lykouris

Vassilvitskii

OMLA model - Algorithm



Goal is to obtain a good competitive ratio

OMLA model - Goals

Consistency

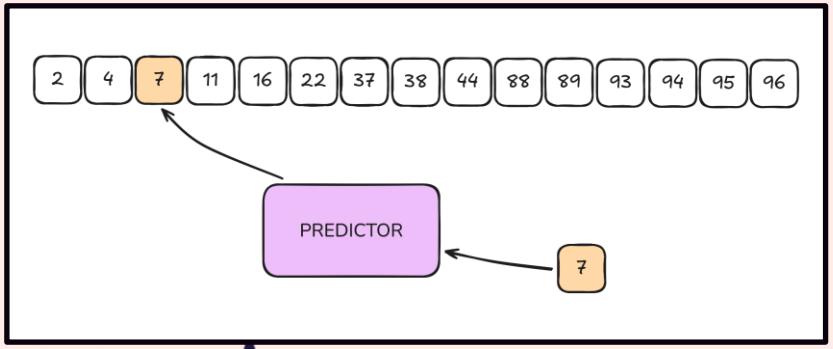
Robustness

OMLA model - Goals

Consistency

An online algorithm is β-consistent if its competitive ratio $c(\eta)$ when the prediction error η is zero is at most β.

$$c(\eta = 0) \leq \beta$$



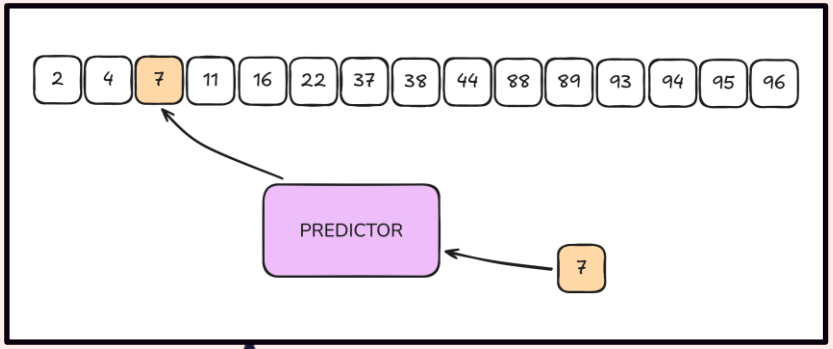
Robustness

OMLA model - Goals

Consistency

An online algorithm is β-consistent if its competitive ratio $c(\eta)$ when the prediction error η is zero is at most β.

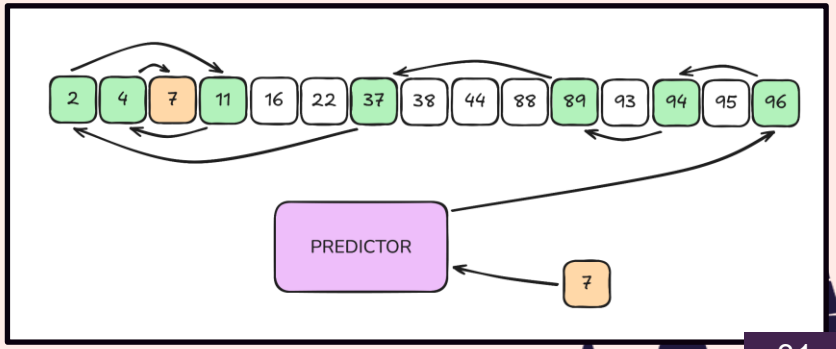
$$c(\eta = 0) \leq \beta$$



Robustness

An online algorithm is $\alpha(\eta)$-robust if its competitive ratio $c(\eta)$ is non-decreasing and

$$\forall \eta \; c(\eta) \leq \alpha(\eta)$$



Ski Rental problem

Example: Ski Rental problem

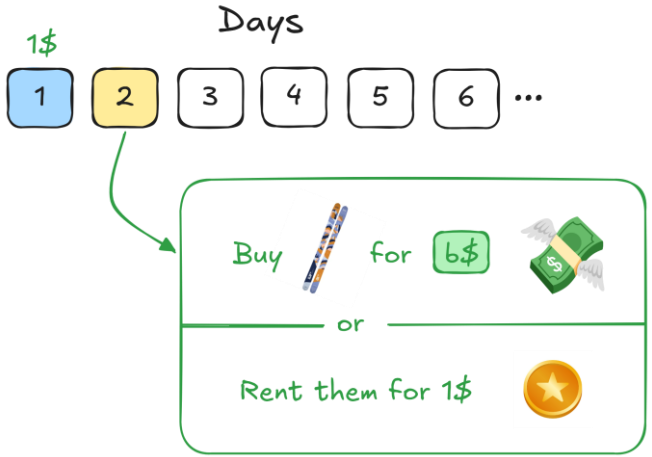
Problem

One random skier

Example: Ski Rental problem

Problem

A skier doesn't know the total number of days he will ski $t(d)$



The goal is to spend the least possible amount of money.

Example: Ski Rental problem

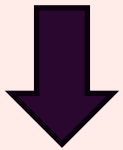
Offline problem:

the number of skiing days
is known in advance

Example: Ski Rental problem

Offline problem:

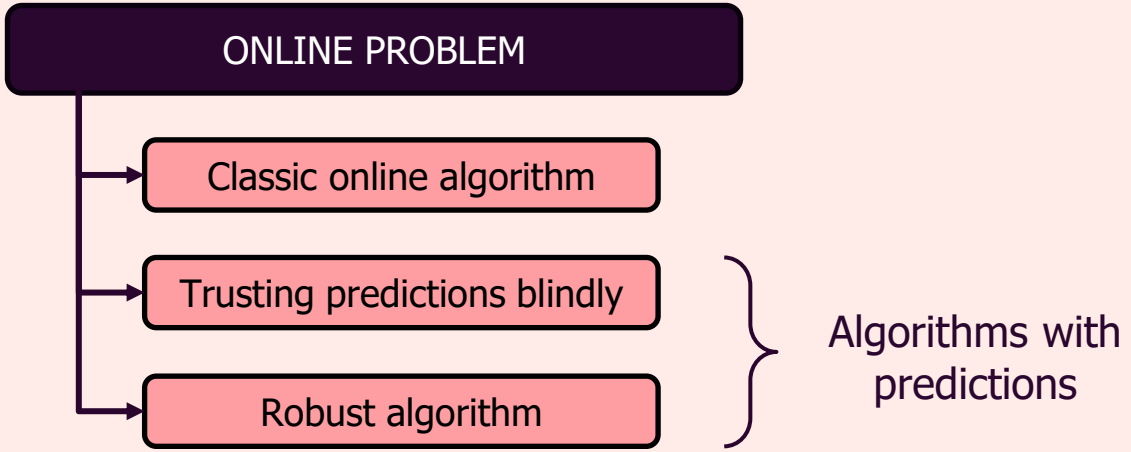
the number of skiing days is known in advance



Algorithm

Buy the skis if $t(d) \geq b$, else rent daily.

Example: Ski Rental problem



Classic online algorithm

Classic online algorithm

Algorithm

Rent skis for b days, then buy them on day $b+1$

Competitive ratio ≤ 2

Classic online algorithm

Algorithm

Rent skis for b days, then buy them on day $b+1$

Competitive ratio ≤ 2

When a skier goes skiing,
there are 2 cases:

Classic online algorithm

Algorithm

Rent skis for b days, then buy them on day $b+1$

Competitive ratio ≤ 2

When a skier goes skiing,
there are 2 cases:

1. He breaks his leg

$t(d) \leq b$

Classic online algorithm

Algorithm

Rent skis for b days, then buy them on day $b+1$

Competitive ratio ≤ 2

When a skier goes skiing,
there are 2 cases:

1. He breaks his leg — $t(d) \leq b$
2. He wants to win gold — $t(d) > b$

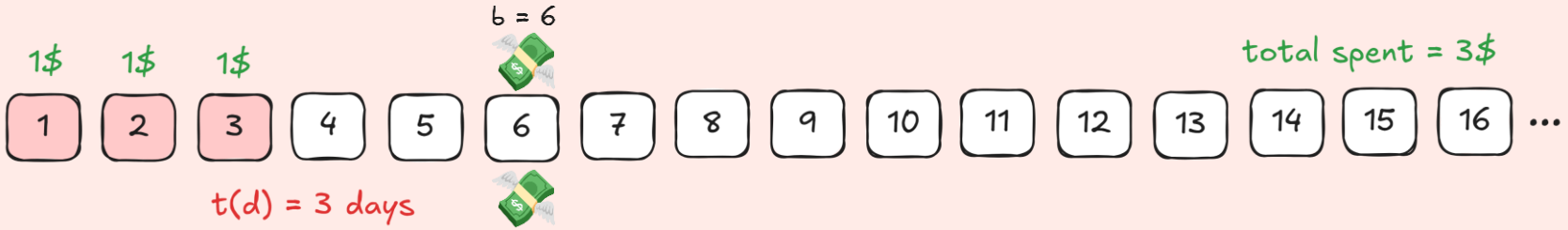
Classic online algorithm

Algorithm

Rent skis for b days, then buy them on day $b+1$

1. He breaks his leg

$t(d) \leq b$



If $t(d) \leq b$, then the algorithm is optimal.

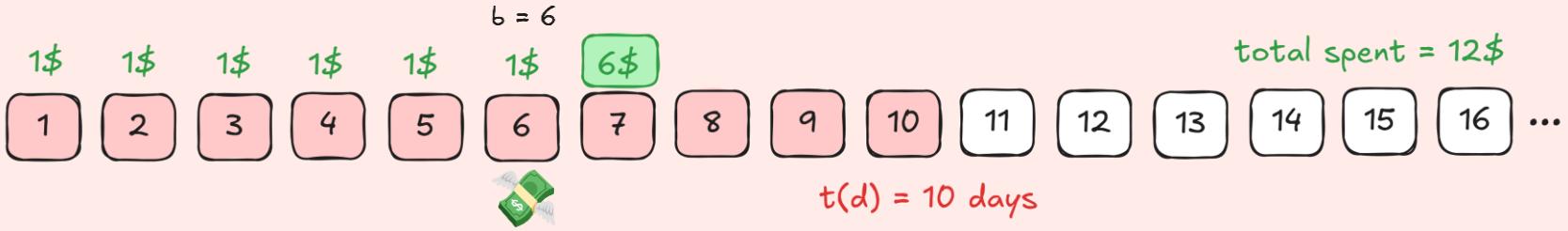
Classic online algorithm

Algorithm

Rent skis for b days, then buy them on day $b+1$

2. He wants to win gold

$t(d) > b$

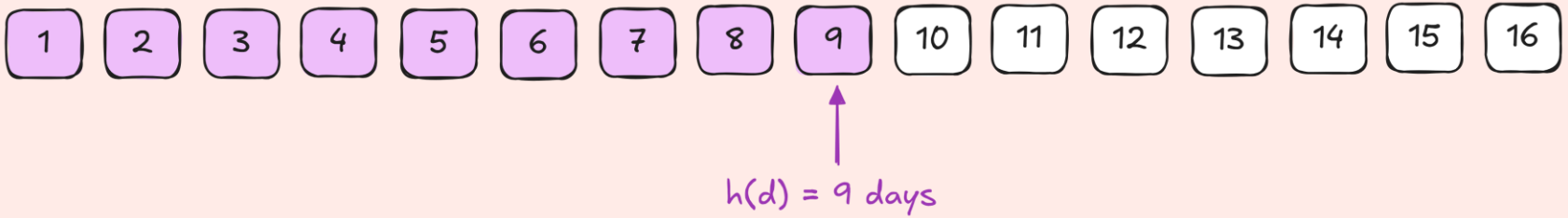


Otherwise, the cost is exactly $2b$, while the optimal is b

Algorithms with predictions

Ski Rental predictions

Prediction: total number of skiing days $h(d)$



Trusting predictions blindly

Trusting predictions blindly

Algorithm

Buy the skis if $h(d) \geq b$, else rent daily

Competitive ratio $\leq$???

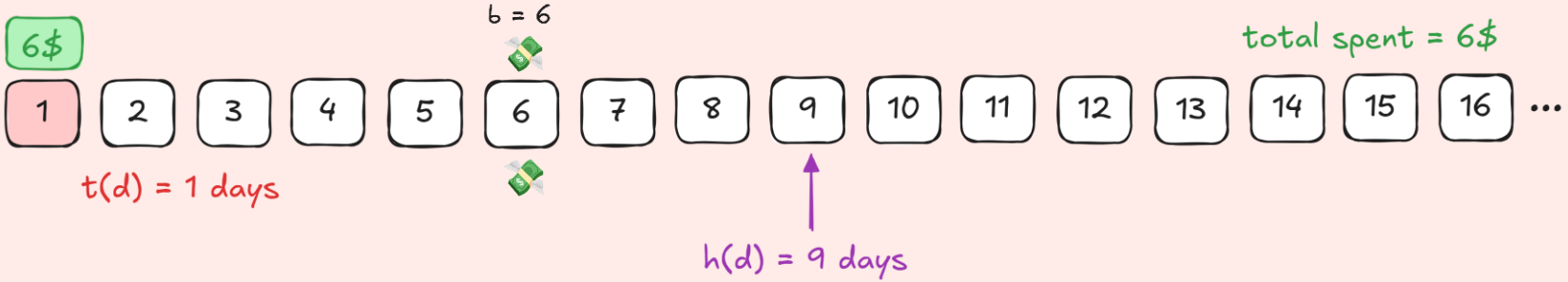
Trusting predictions blindly

Algorithm

Buy the skis if $h(d) \geq b$, else rent daily

1. He breaks his leg

$t(d) \leq b$



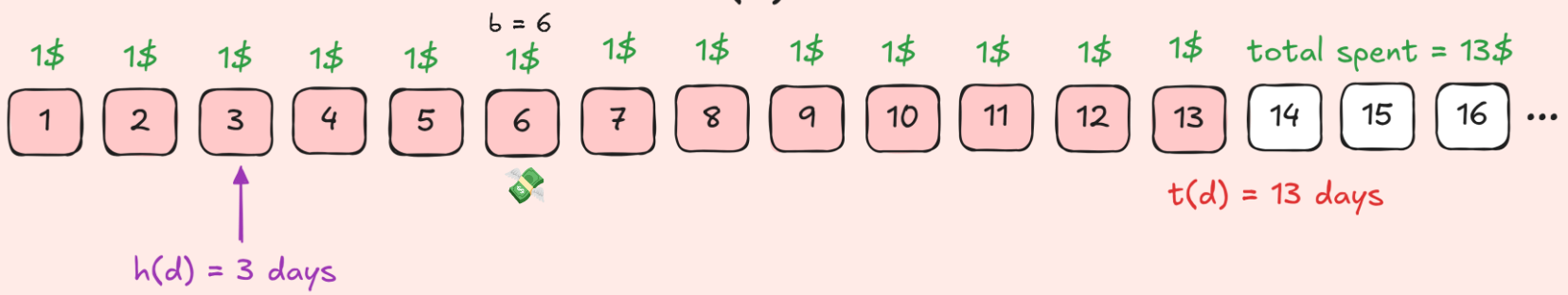
In the worst case $h(d) \geq b$, and the competitive ratio is bounded by b.

Trusting predictions blindly

Algorithm

Buy the skis if $h(d) \geq b$, else rent daily

2. He wants to win gold



In the worst case $h(d) < b$, and the competitive ratio is bounded by $\frac{t(d)}{b}$.

Trusting predictions blindly

Algorithm

Buy the skis if $h(d) \geq b$, else rent daily

Competitive ratio $\leq max\left\{b, \frac{t(d)}{b}\right\}$

The algorithm is **consistent**, but **not robust**.

Robust algorithm

Robust algorithm

Algorithm

Introduce a parameter $\lambda \in [0,1]$.

If $h(d) > b$, buy skis on day $\lceil \lambda b \rceil$, else on day $\lceil b/\lambda \rceil$

Robust algorithm

Algorithm

Introduce a parameter $\lambda \in [0,1]$.

If $h(d) > b$, buy skis on day $[\lambda b]$, else on day $[b/\lambda]$

Trade-off

Low λ -> more consistent -> similar to trusting predictions blindly

High λ -> more robust -> similar to not using predictions

Robust algorithm

Algorithm

Introduce a parameter $\lambda \in [0,1]$.

If $h(d) > b$, buy skis on day $[\lambda b]$, else on day $[b/\lambda]$

Trade-off

Low λ -> more consistent -> similar to trusting predictions blindly

High λ -> more robust -> similar to not using predictions

$\lambda = 0.7$

Robust algorithm

$\lambda = 0.7$

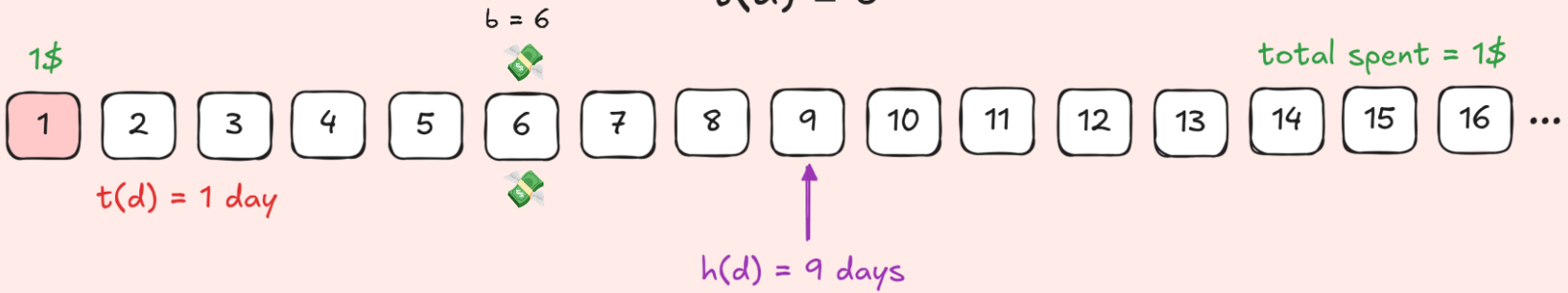
Algorithm

Introduce a parameter $\lambda \in [0,1]$.

If $h(d) > b$, buy skis on day $[\lambda b]$, else on day $[b/\lambda]$

1. He breaks his leg

$t(d) \leq b$



Even if $h(d) \geq b$, we wait day $[\lambda b] = 5$ to buy.

Robust algorithm

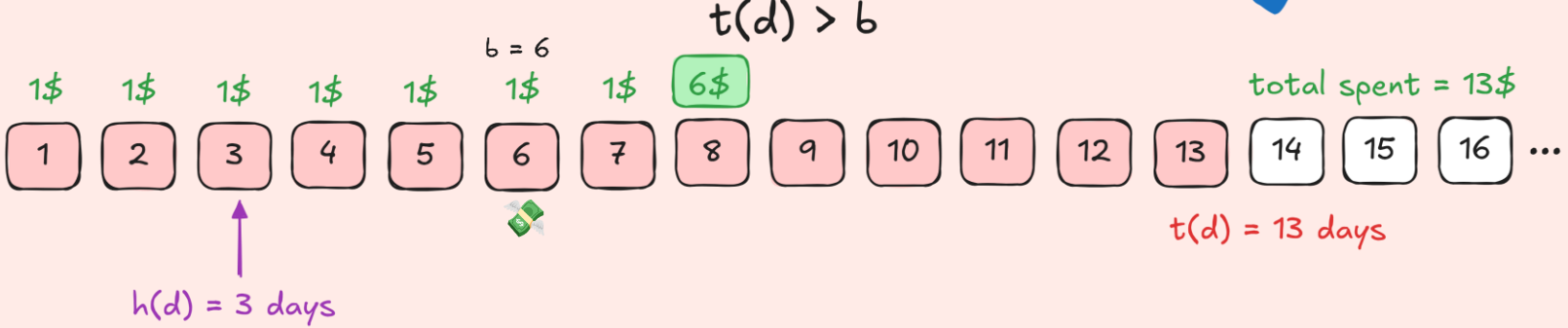
$\lambda = 0.7$

Algorithm

Introduce a parameter $\lambda \in [0,1]$.

If $h(d) > b$, buy skis on day $[\lambda b]$, else on day $[b/\lambda]$

2. He wants to win gold



Even if $h(d) < b$, we buy on day $[b/\lambda] = 8$.

Robustness and Consistency trade-off

Competitive ratio $\leq 1 + min\left(\frac{1}{\lambda}, \lambda + \frac{\eta}{(1-\lambda)OPT}\right)$

It's $(1+\lambda)$ – **consistent** and $(1+1/\lambda)$ – **robust**

Trade-off

Low λ -> more consistent -> c(0) = 1

High λ -> more robust -> c(η) ≤ 2

Take-away: Consistency and Robustness

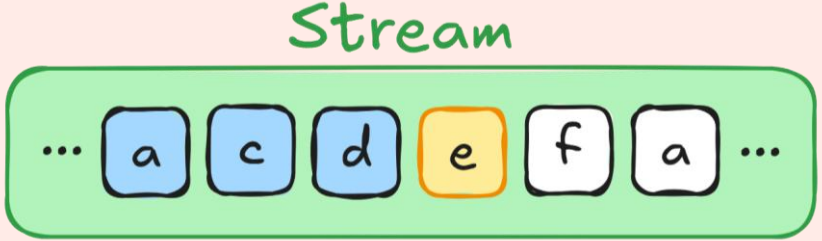
When designing Algorithms with Predictions, we shouldn't trust blindly the predictions, but always ensure robustness: our algorithms should **work well even with high prediction errors**.

03

WHAT WORKS AND WHAT DOESN’T

Streaming problems

The input is given as a stream of elements. At each point in time a data structure is updated and, if requested, an (approximate) solution for a certain problem is given.



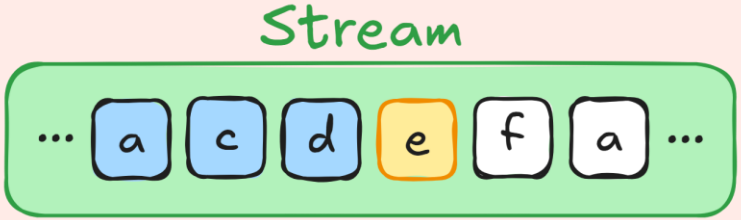
The key performance metric is memory

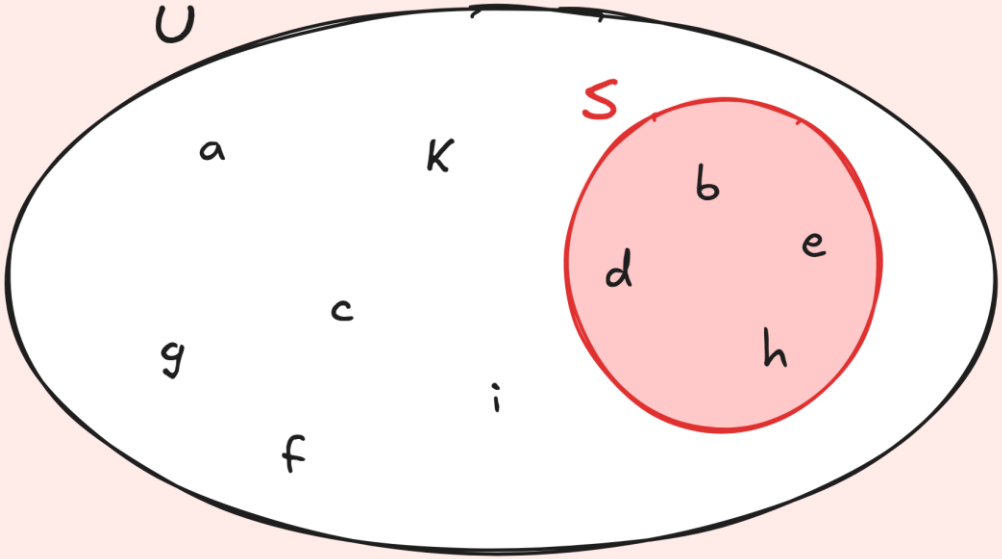
What works

Set Membership problem

Problem

Given a set of elements U, a subset $S \subseteq U$ and an item $q \in U$, assess wheter $q \in S$ or not.

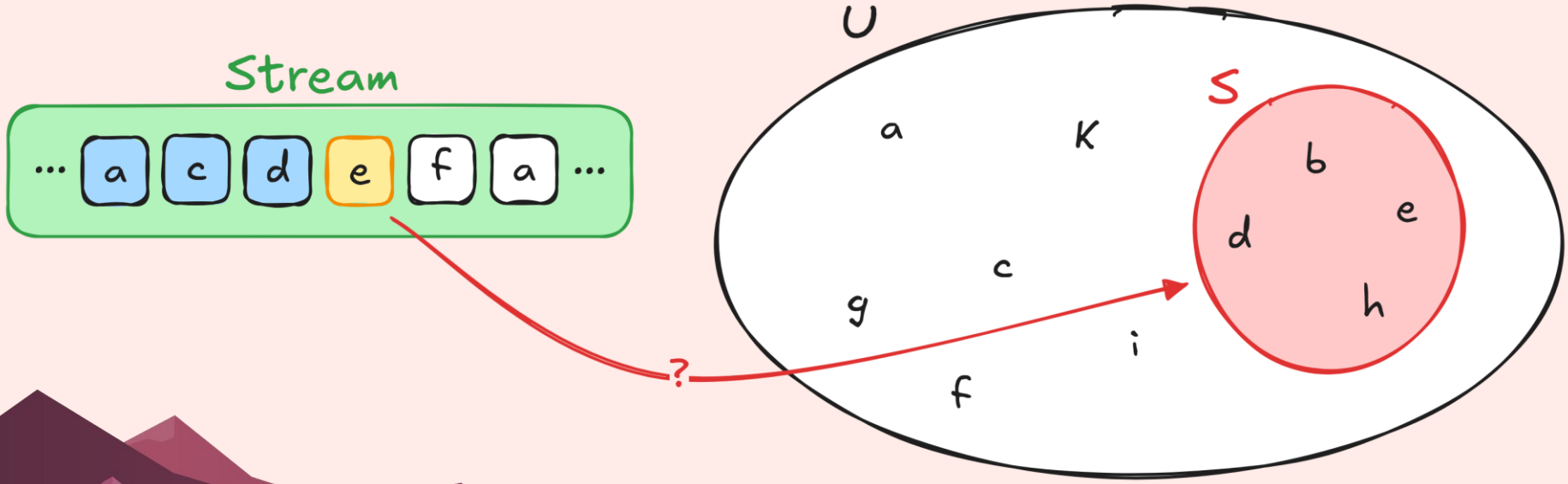




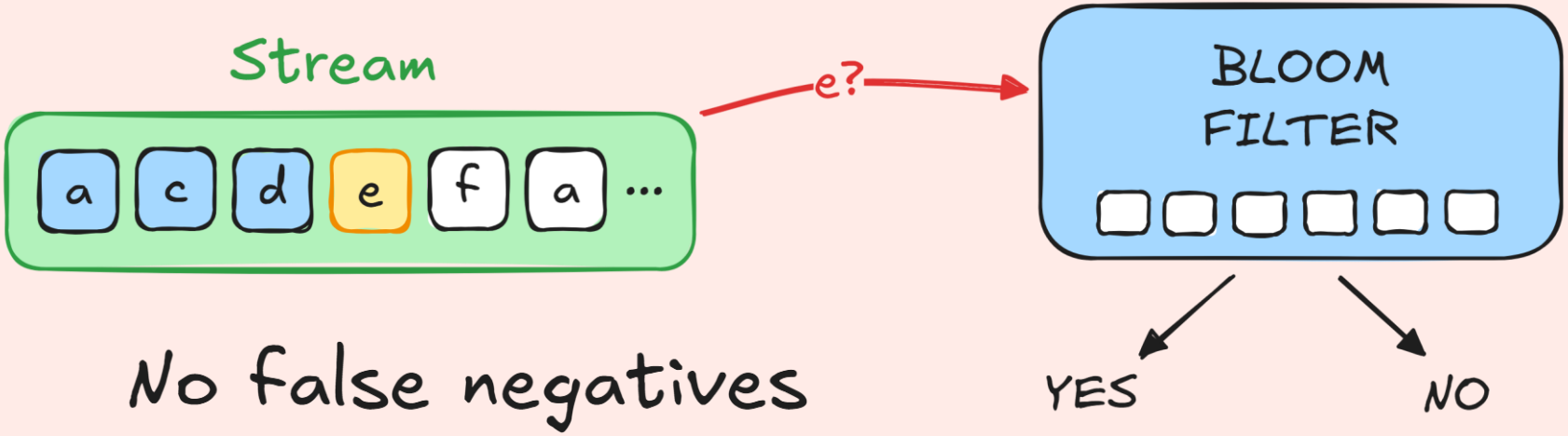
Set Membership problem

Problem

Given a set of elements U, a subset $S \subseteq U$ and an item $q \in U$, assess wheter $q \in S$ or not.

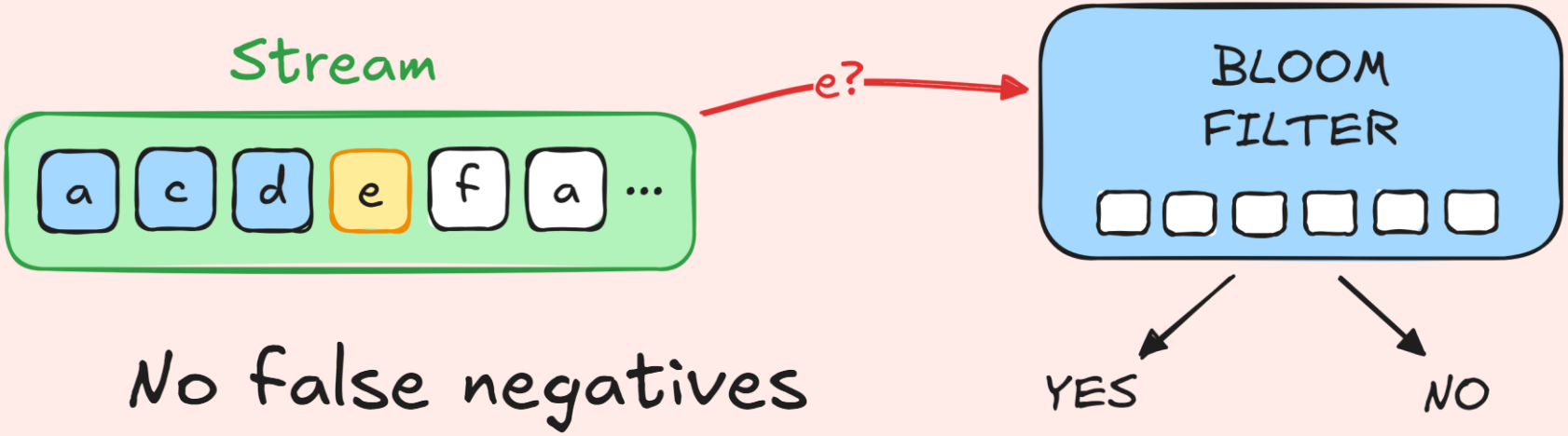


Bloom Filters



No false negatives

Bloom Filters

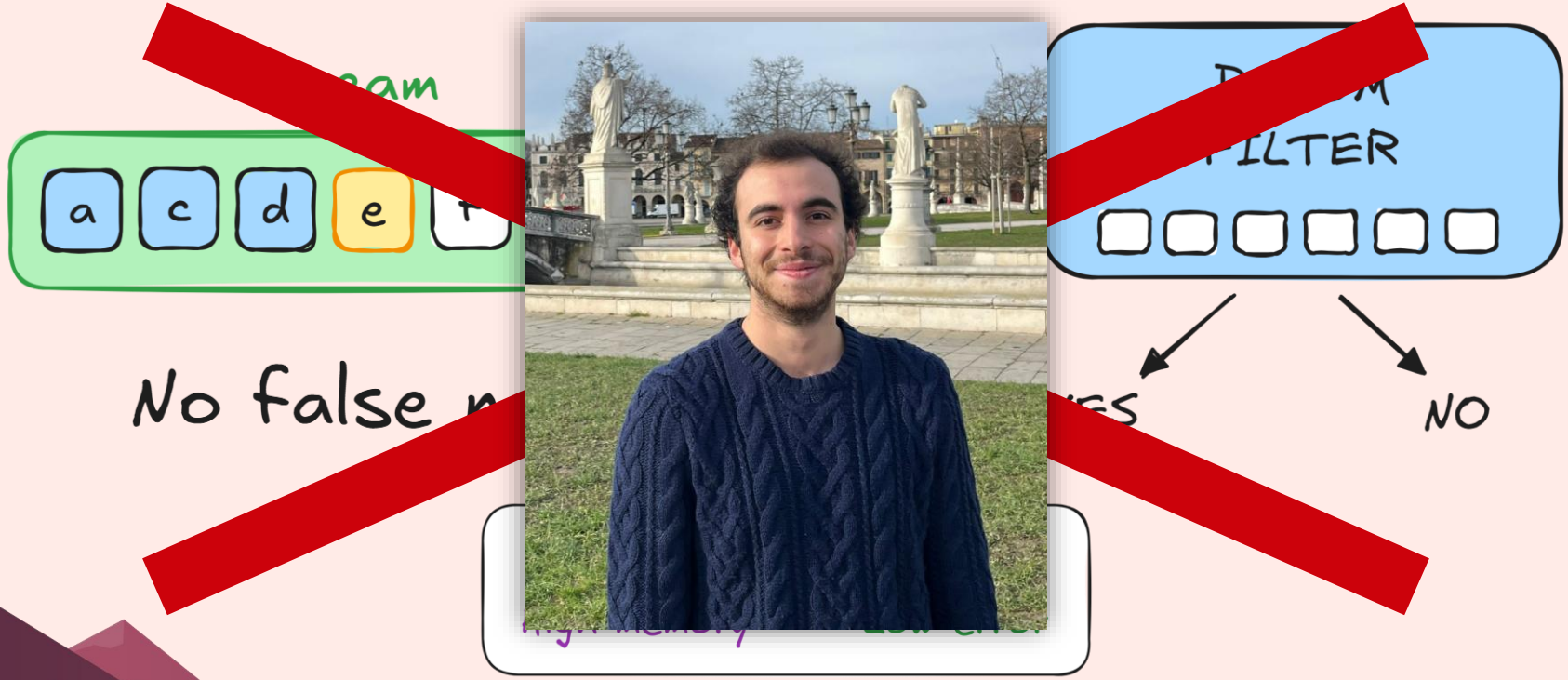


No false negatives

Trade-off

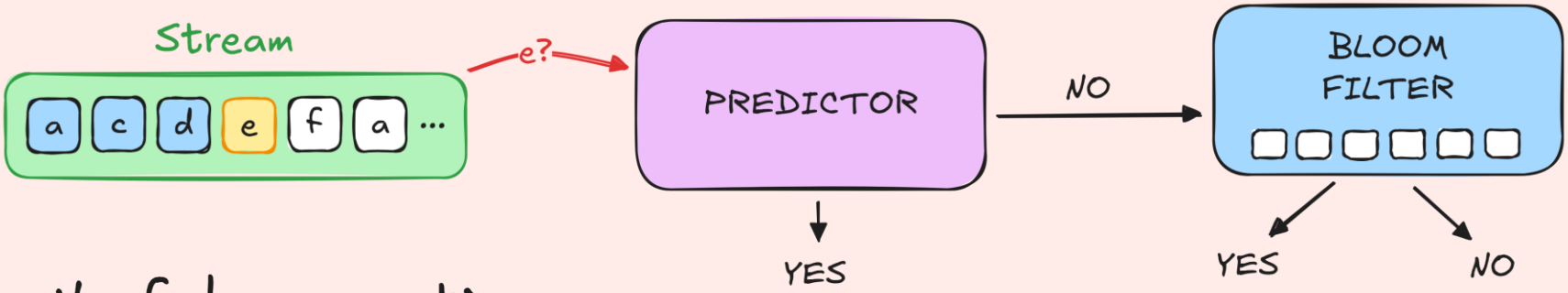
High memory - Low error

Bloom Filters



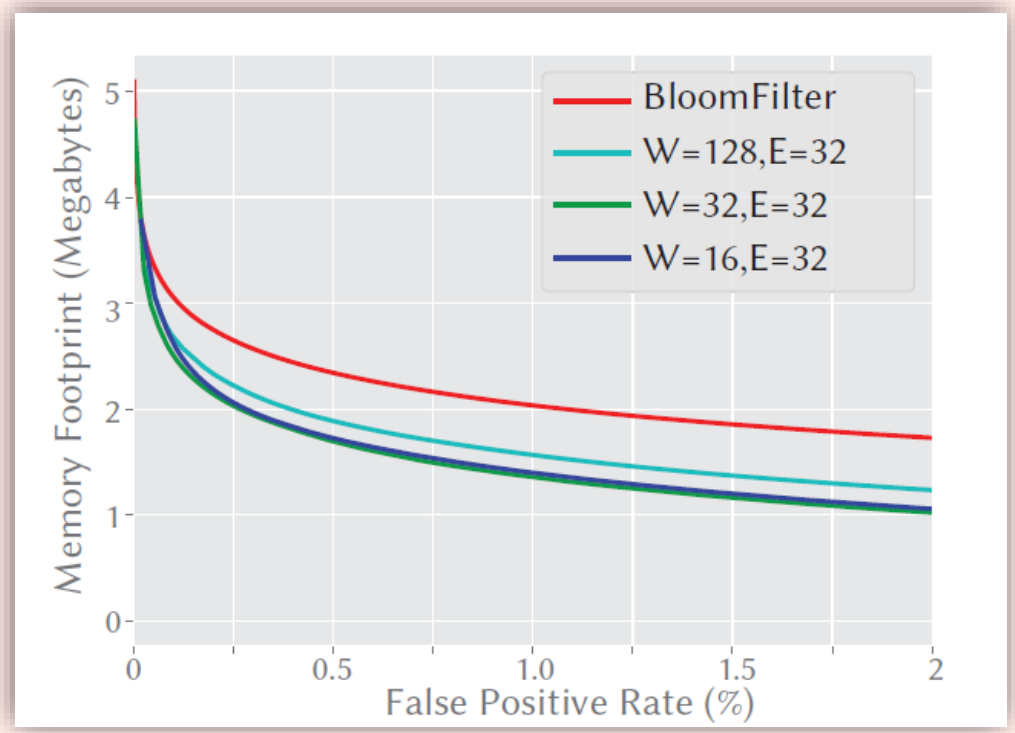
Learned Bloom Filters

Predictor classifies every item as being part of S or not



No false negatives

Learned Bloom Filters: Experiments

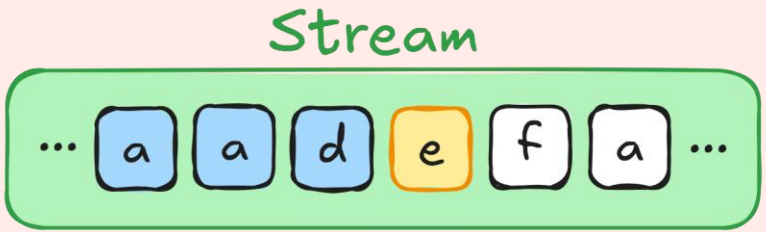


What doesn’t work

Frequency Estimation problem

Problem

Given a set of elements U, for each item $u \in U$ keep an estimate of its absolute frequency in the stream $f_u = |\{j : x_j = u, 1 \leq j \leq n\}|$

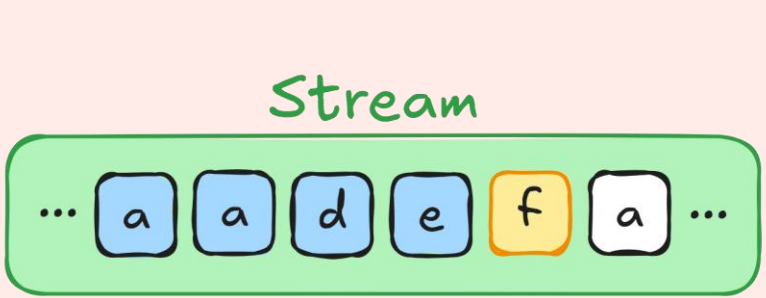


a	2
d	1
e	1
f	0

Frequency Estimation problem

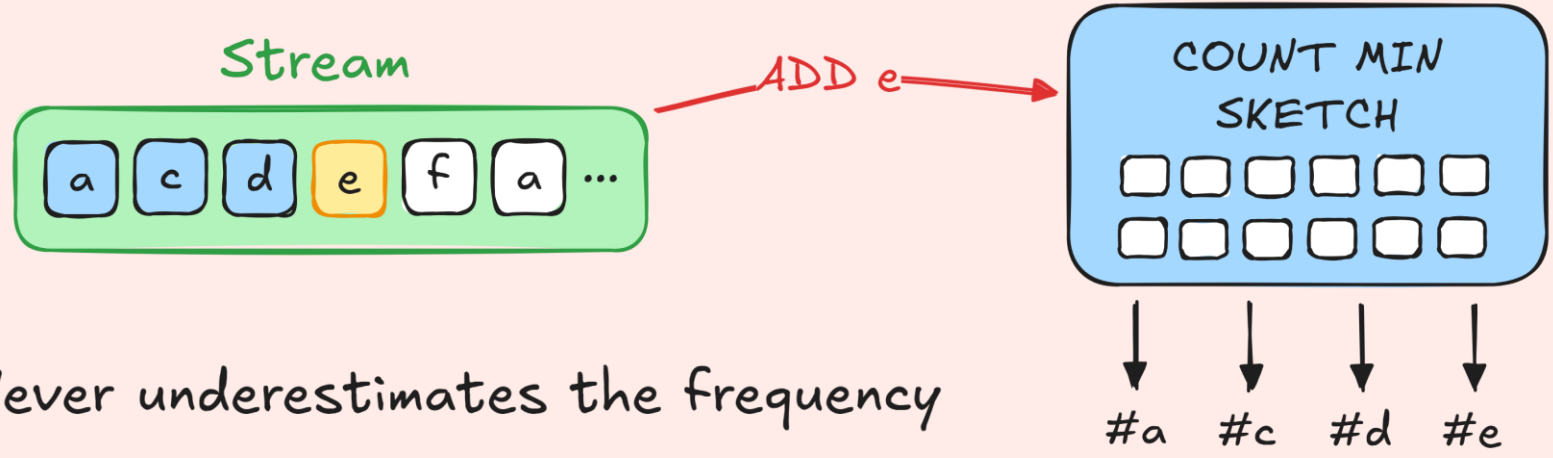
Problem

Given a set of elements U, for each item $u \in U$ keep an estimate of its absolute frequency in the stream $f_u = |\{j : x_j = u, 1 \leq j \leq n\}|$



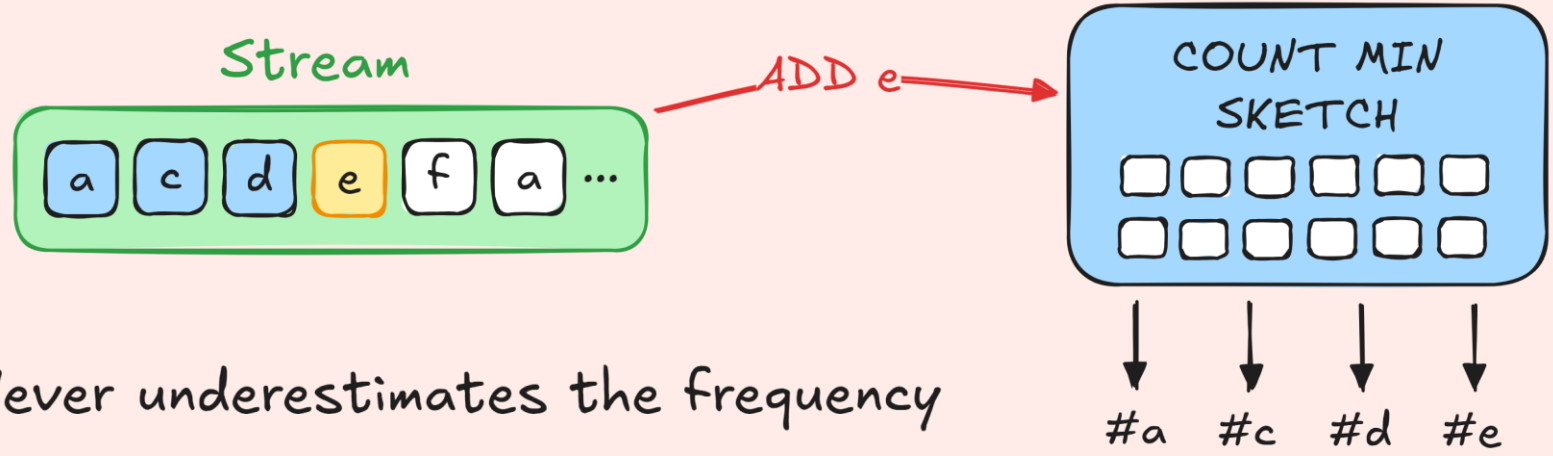
a	2
d	1
e	1
f	1

Count Min Sketch



Never underestimates the frequency

Count Min Sketch



Never underestimates the frequency

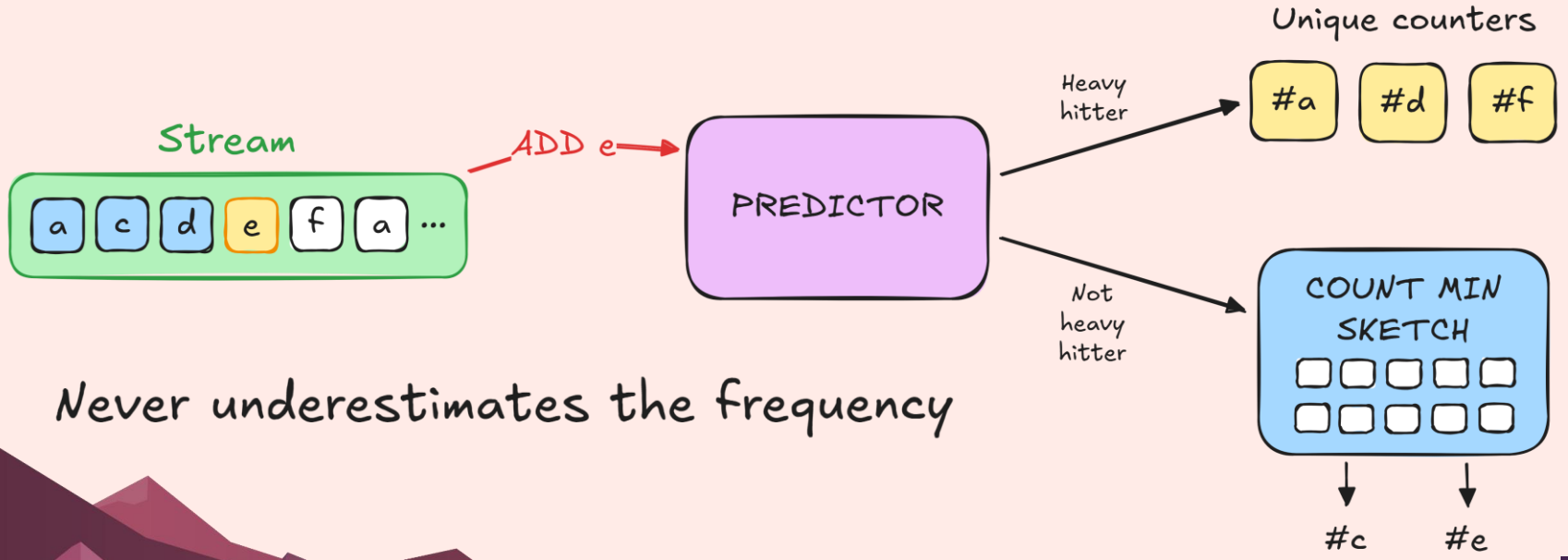
Trade-off

High memory - Low error

Count Min Sketch

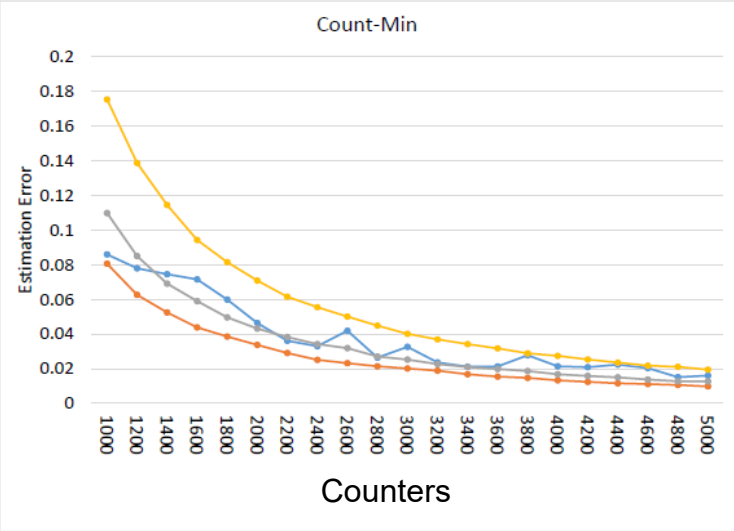
Learned Count Min Sketch

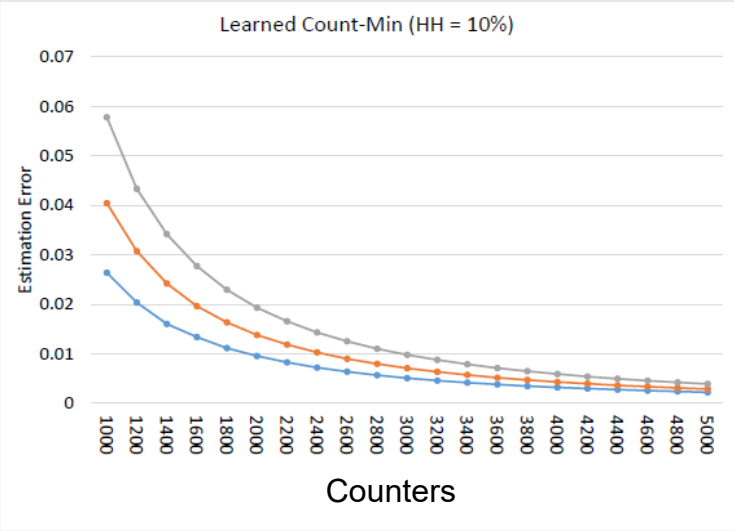
Predictor classifies an item as heavy hitter or not.



Never underestimates the frequency

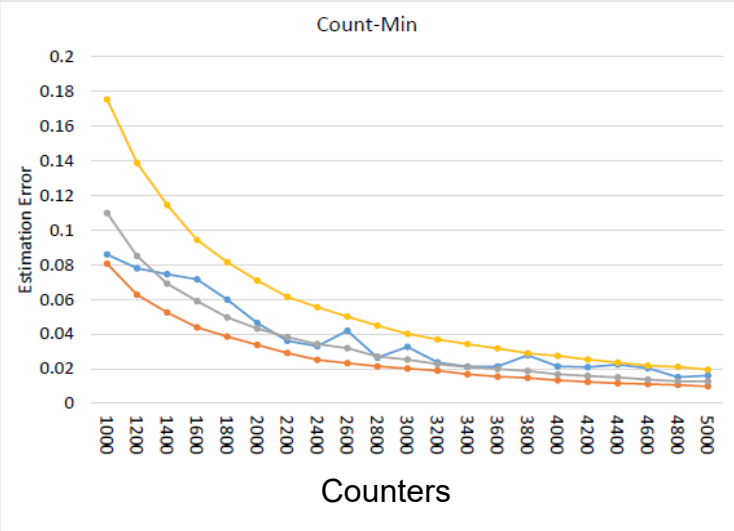
Learned CMS: Experiments

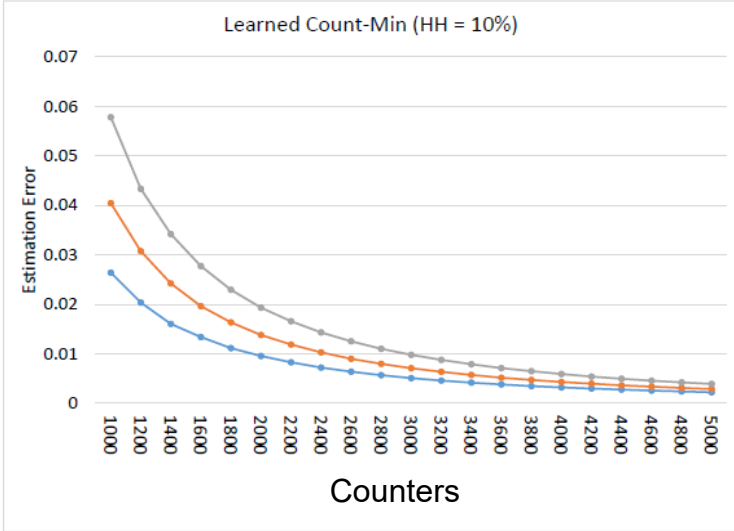




Learned CMS: Experiments

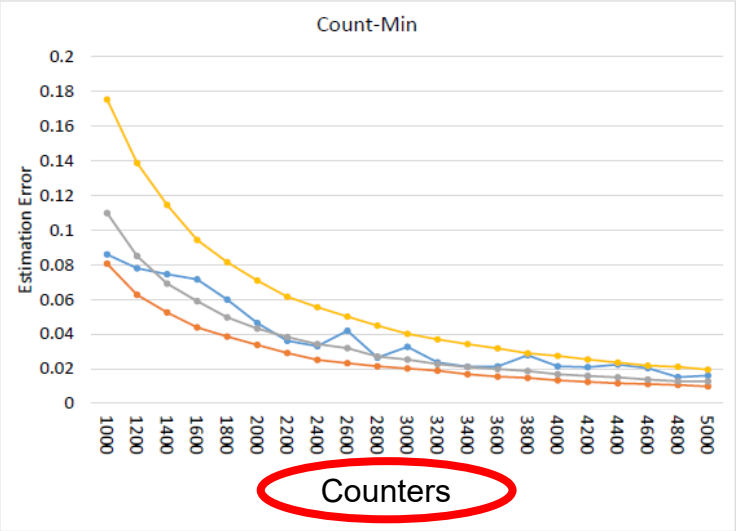
Something in this comparison is strange

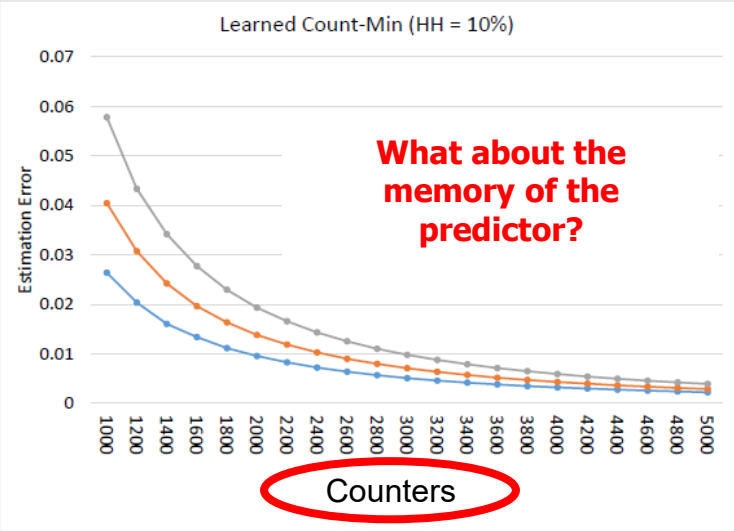




Learned CMS: Experiments

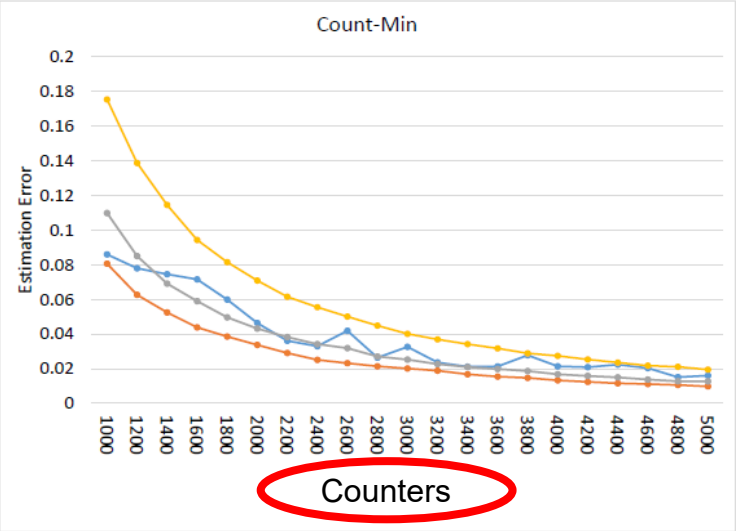
Is the **number of counters** representative of the total memory of the approach?

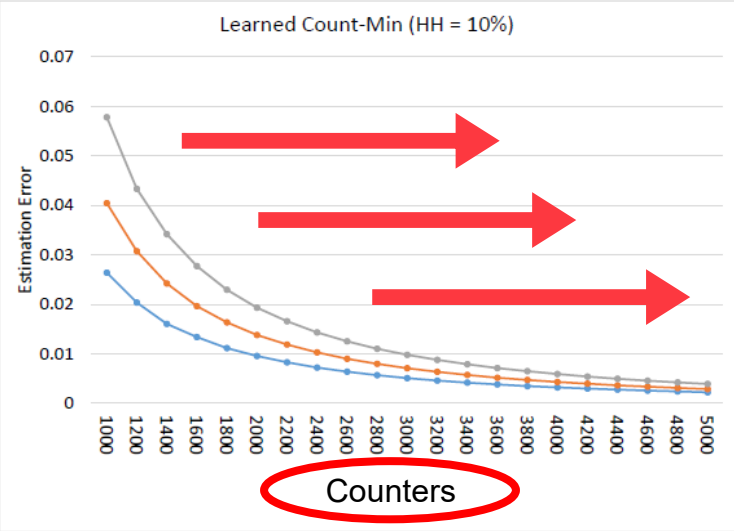




Learned CMS: Experiments

The memory needed for the RNNs implemented correspond to ≈**13.000 counters**





Take-away: Pitfalls in the Experiments

1. The **cost of the model** is not fully accounted for.
2. Authors may assume the **existence of an error-free oracle**.

Take-away

Algorithms with Predictions is a strong framework which is based on simple ideas.

When designing Algorithms with Predictions, we should always focus on both consistency and robustness.

When analysing Algorithms with Predictions, we should ensure that the comparison is fair and all costs have been taken into account.

To be continued...

References

[1] Thodoris Lykouris and Sergei Vassilvitskii. *Competitive caching with machine learned advice.* 2020.
[2] Ravi Kumar, Manish Purohit, and Zoya Svitkina. *Improving Online Algorithms via ML Predictions.* 2024.
[3] Michael Mitzenmacher and Sergei Vassilvitskii. *Algorithms with Predictions.* 2020.
[4] Burton H. Bloom. *Space/time trade-offs in hash coding with allowable errors.* 1970.
[5] Tim Kraska et al. *The Case for Learned Index Structures.* 2018.
[6] Graham Cormode and S. Muthukrishnan. *An improved data stream summary: the count-min sketch and its applications.* 2005.
[7] Chen-Yu Hsu et al. *Learning-Based Frequency Estimation Algorithms.* 2019
[8] Anders Aamand, Piotr Indyk, and Ali Vakilian. *(Learned) Frequency Estimation Algorithms under Zipfian Distribution.* 2020

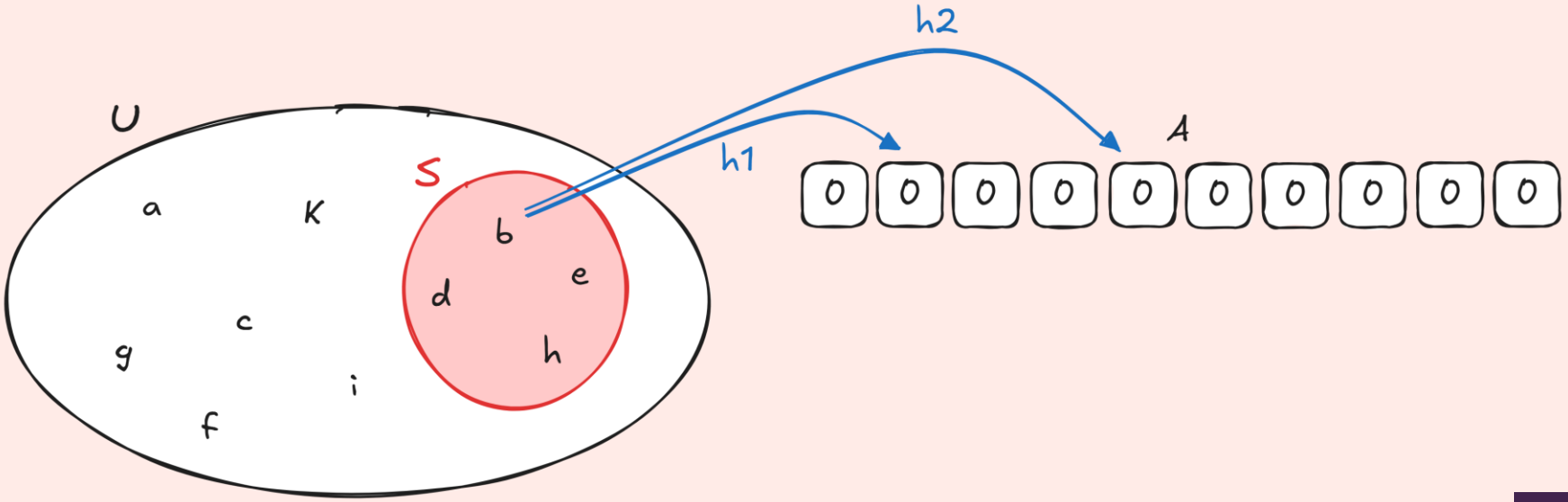
THANK YOU

APPENDIX

BLOOM FILTERS

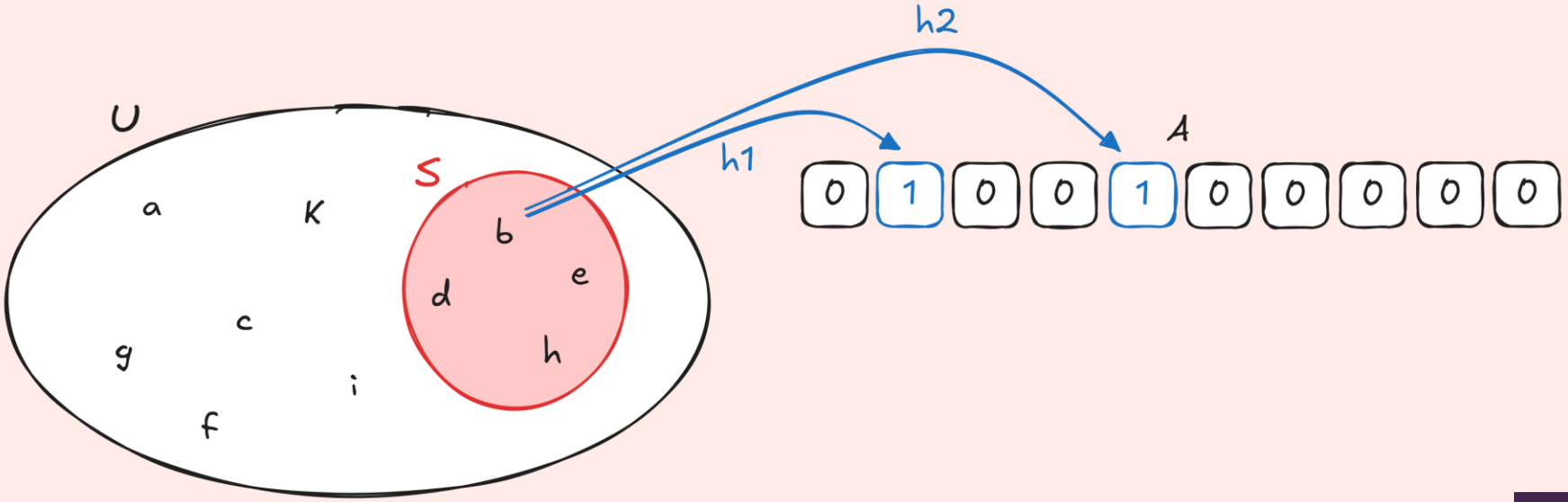
Bloom Filters

Array A of m bits
k independent hash functions $h_1, \ldots h_k$



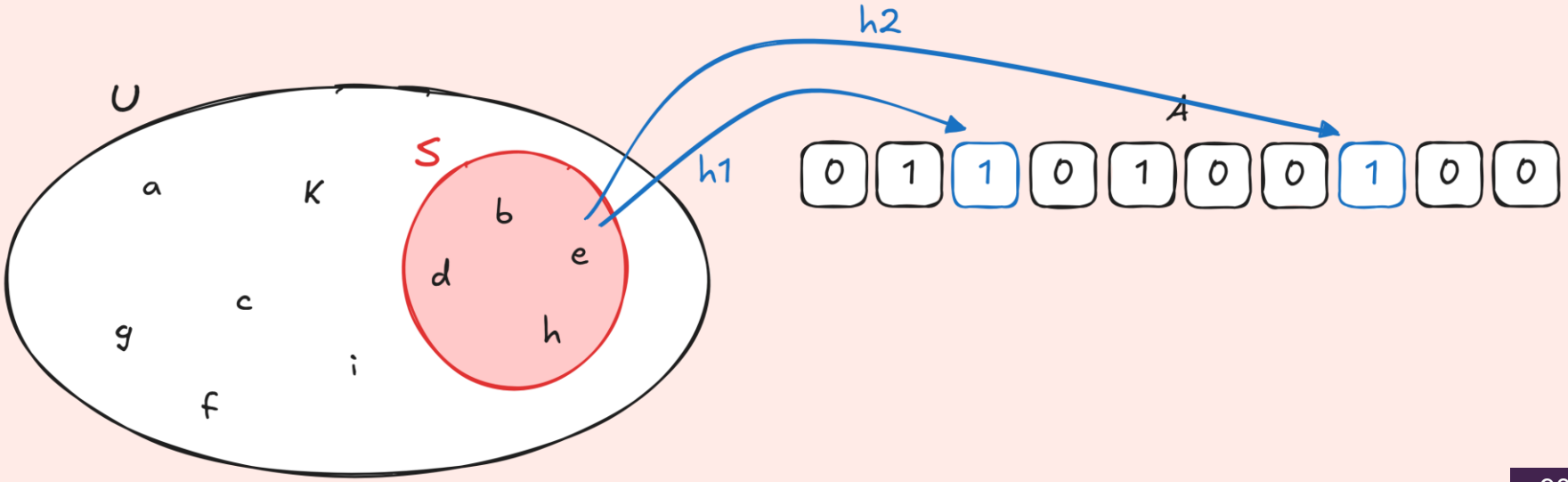
Bloom Filters

Initialization: set all bits to 0.
For each element $q \in S$, set $h_1(q), h_2(q), \ldots h_k(q)$ to 1.



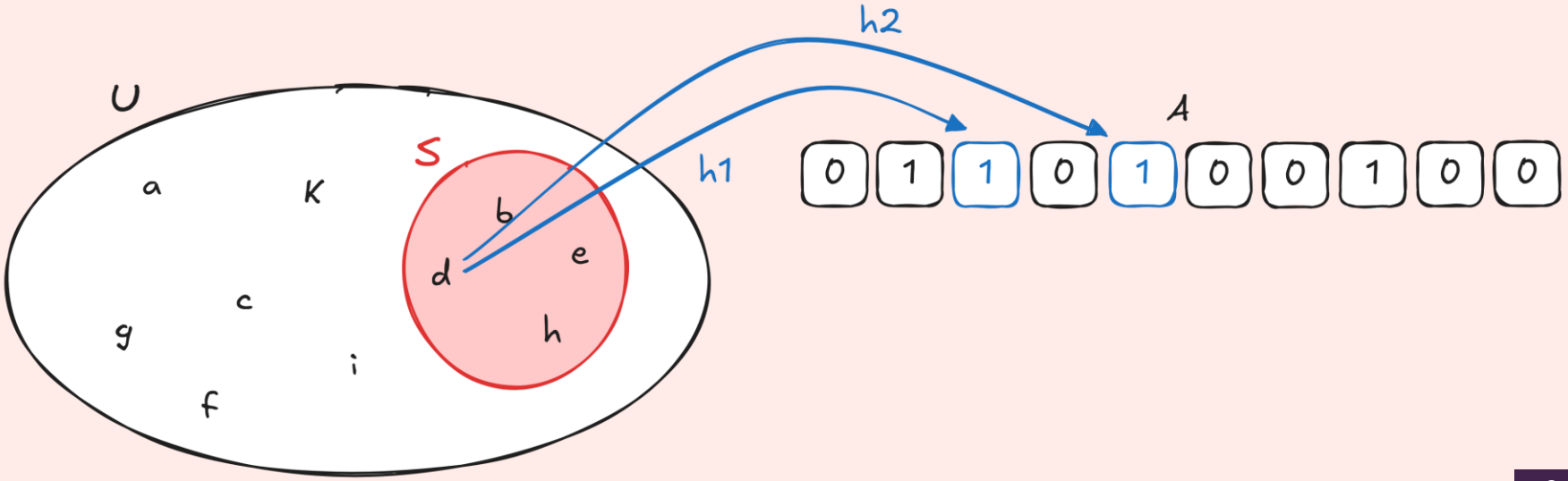
Bloom Filters

Initialization: set all bits to 0.
For each element $q \in S$, set $h_1(q), h_2(q), \ldots h_k(q)$ to 1.



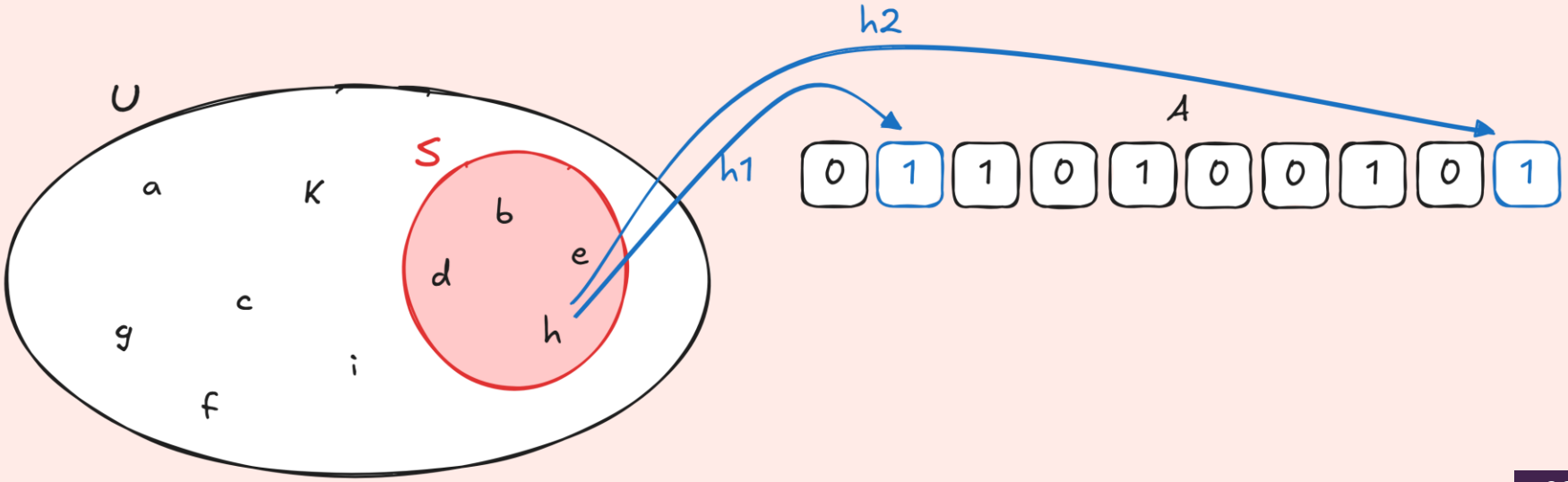
Bloom Filters

Initialization: set all bits to 0.
For each element $q \in S$, set $h_1(q), h_2(q), \ldots h_k(q)$ to 1.



Bloom Filters

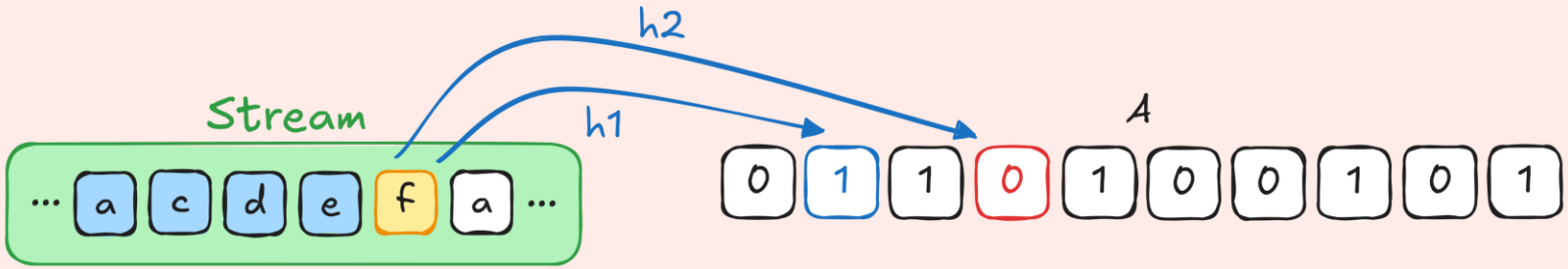
Initialization: set all bits to 0.
For each element $q \in S$, set $h_1(q), h_2(q), \ldots h_k(q)$ to 1.



Bloom Filters

Membership test:

$$q \in S \Leftrightarrow A[h_1(q)] = A[h_2(q)] = \cdots = 1$$



Bloom Filters

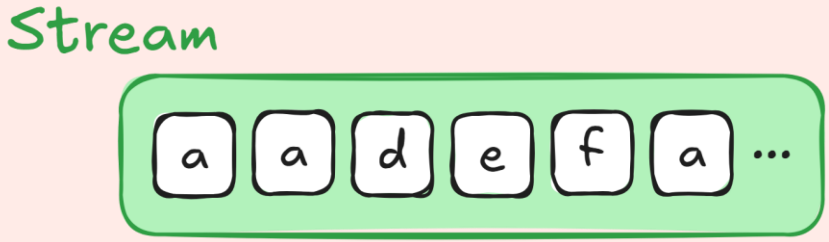
Correctness

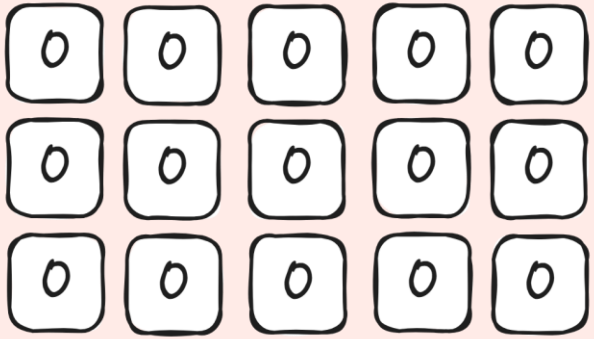
The false positive rate is:

$$p(q \text{ is said to be in } S \text{ even if it's not}) \simeq \left(1 - e^{-km/n}\right)^k$$

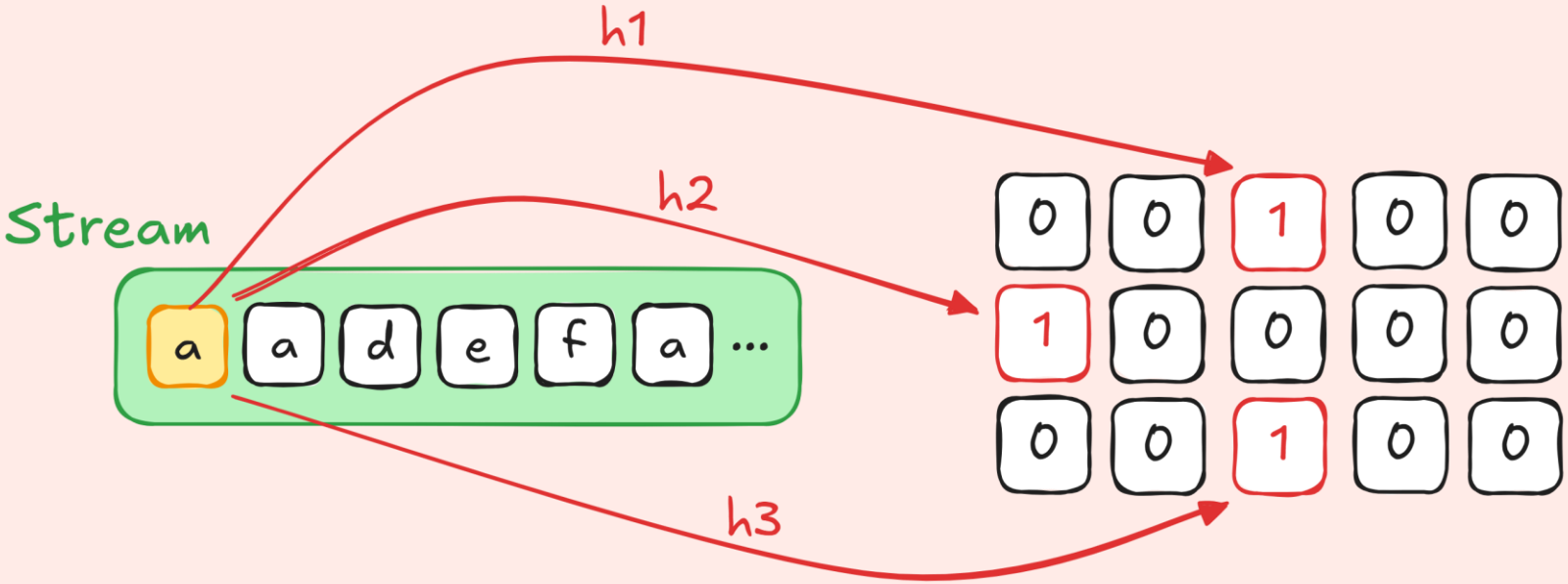
CMS

Count Min Sketch

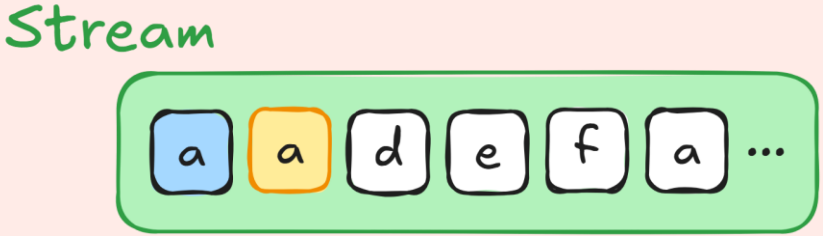


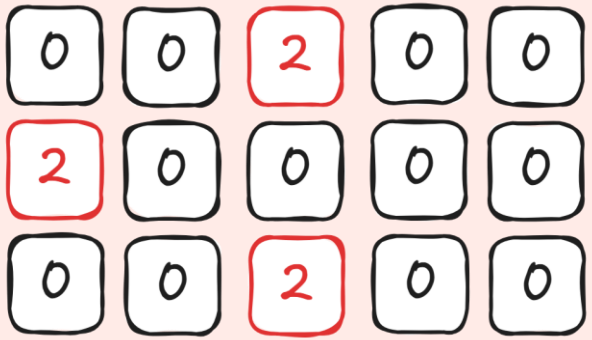


Count Min Sketch

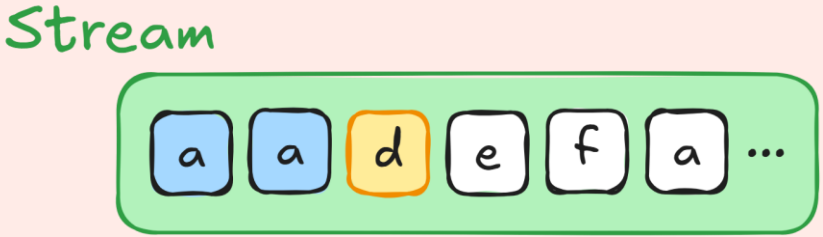


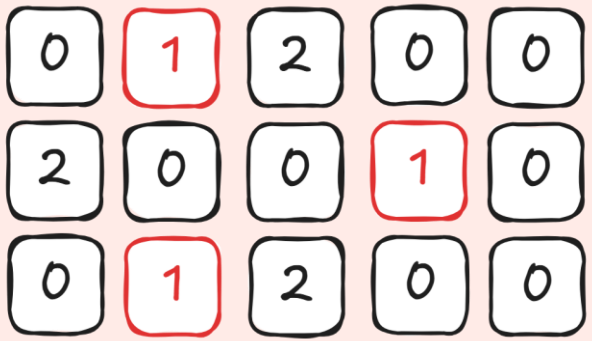
Count Min Sketch



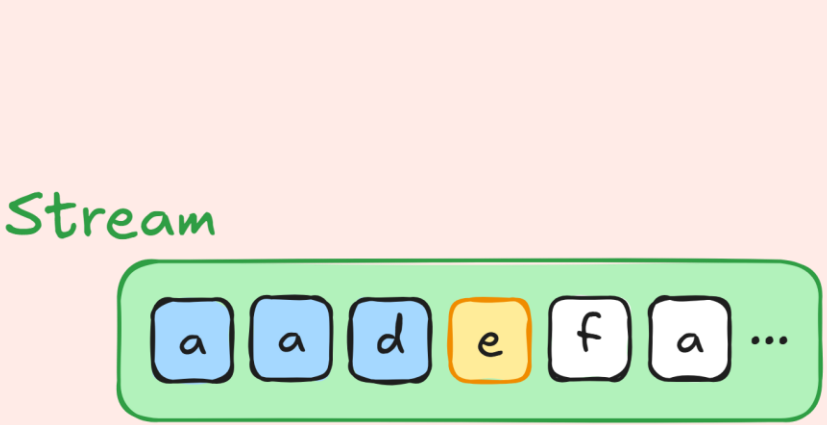


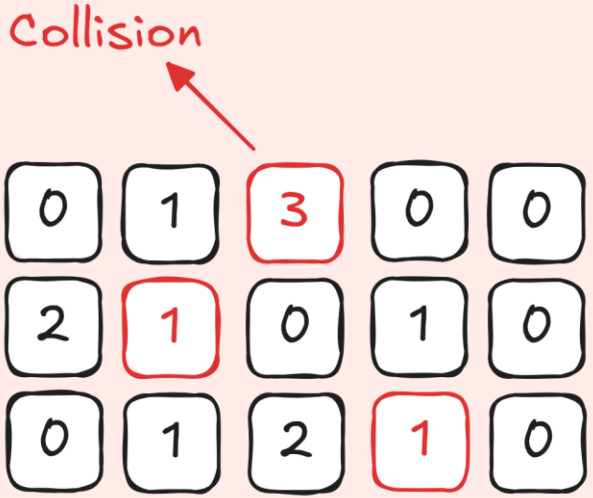
Count Min Sketch



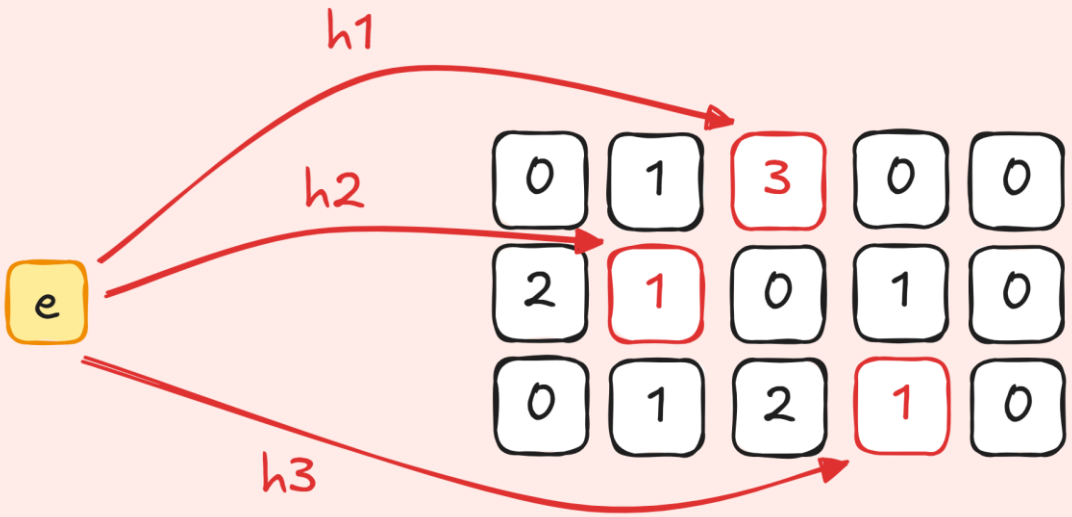


Count Min Sketch





Count Min Sketch



$|e| = \min\{3, 1, 1\} = 1$

Count Min Sketch

Correctness

If we set $r = log_2(1/\delta)$ and $w = 2/\epsilon$, the estimation error for any item of a stream of length n is:

$$\hat{f}_u - f_u \leq \epsilon \cdot n$$

with probability $\geq 1 - \delta$.