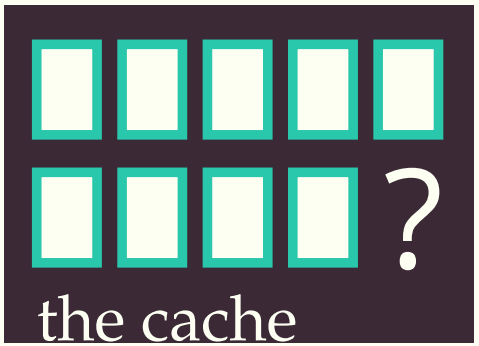
CACHE REPLACEMENT WITH MEMORY ALLOCATION

Shahram Ghandeharizadeh
Sandy Irani
Jenny Lam

ALENEX 2015 — January 5, 2015

THE PAGING PROBLEM

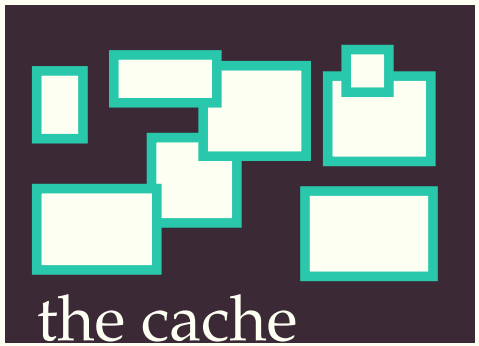


GOAL

minimize number of cache misses

THE GENERALIZED CACHING PROBLEM

variable size and cost



GOAL

minimize **total cost** of cache misses

SUBJECT TO

total size of items in cache
cannot exceed the cache size

THE MANAGED MEMORY CACHING PROBLEM

variable size and cost

every item must fit in a contiguous segment of memory

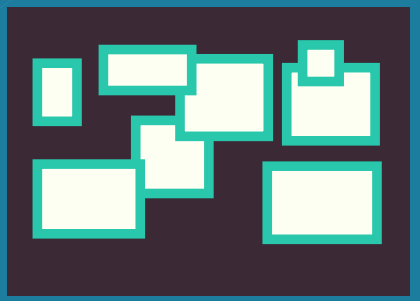
CACHE REPLACEMENT

MEMORY ALLOCATION

PAGING PROBLEM

main memory

GENERALIZED CACHING



webproxy

MANAGED MEMORY CACHING PROBLEM

key-value stores

SIMPLE ALGORITHM

for the managed memory
caching problem

COMPETITIVE RATIO

if augmented memory
against optimal offline algorithm
for the generalized caching problem

EXPERIMENTAL RESULTS

RESULTS

GDS

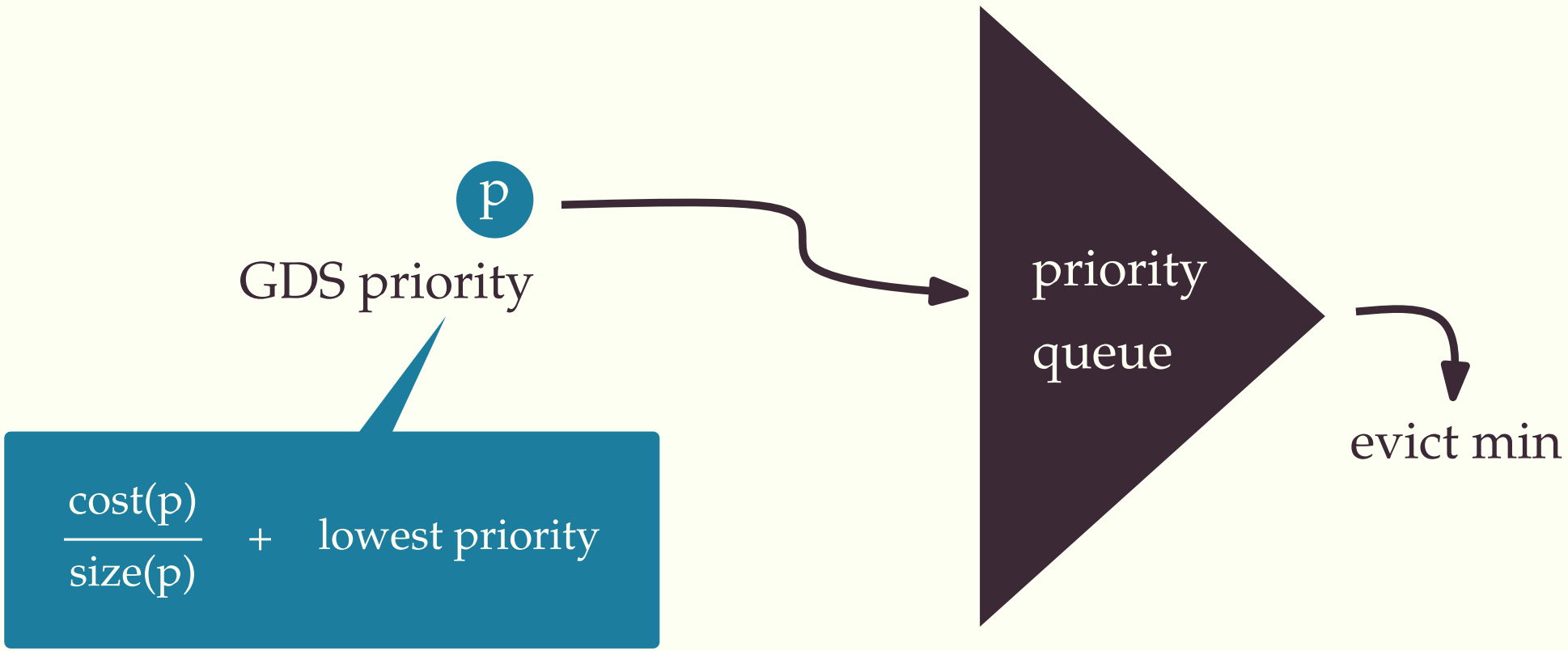
Greedy-Dual Size

CAMP

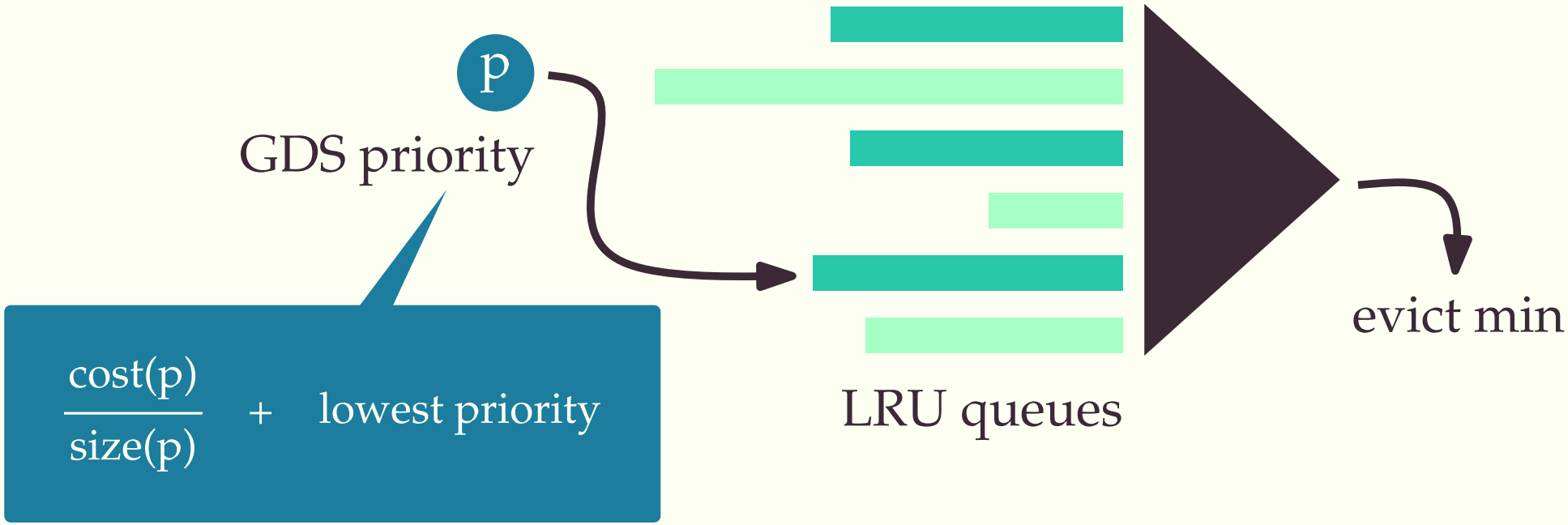
Cost-adaptive multiqueue eviction policy

OUR ALGORITHM

GDS


p
GDS priority
cost(p) / size(p) + lowest priority
priority queue
evict min

CAMP


p
GDS priority
cost(p) / size(p) + lowest priority
LRU queues
evict min

OUR ALGORITHM

LRU queues

OUR ALGORITHM

FIFO queues

OUR ALGORITHM

FIFO queue

OUR ALGORITHM

FIFO queue

OUR ALGORITHM

FIFO queue

OUR ALGORITHM

FIFO queue

OUR ALGORITHM

FIFO queue

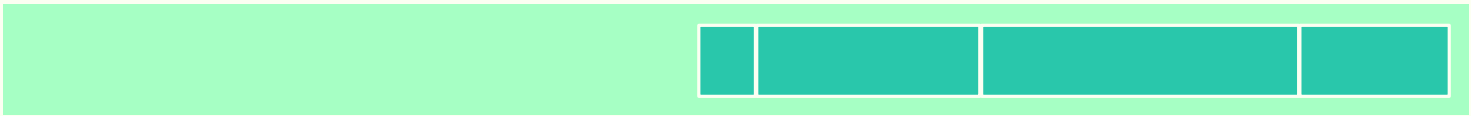
OUR ALGORITHM

FIFO queue

OUR ALGORITHM

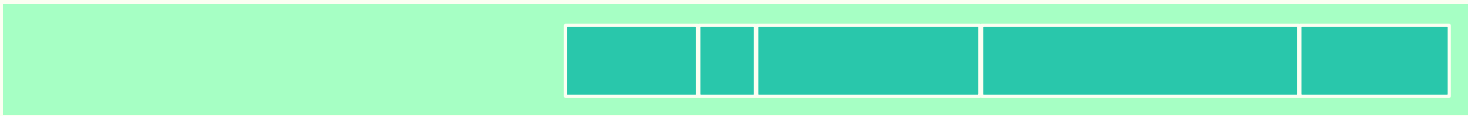
FIFO queue

OUR ALGORITHM



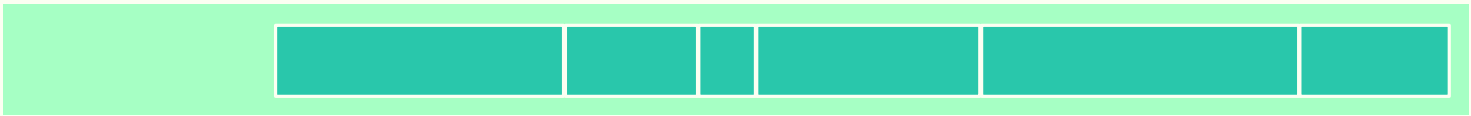
FIFO queue

OUR ALGORITHM



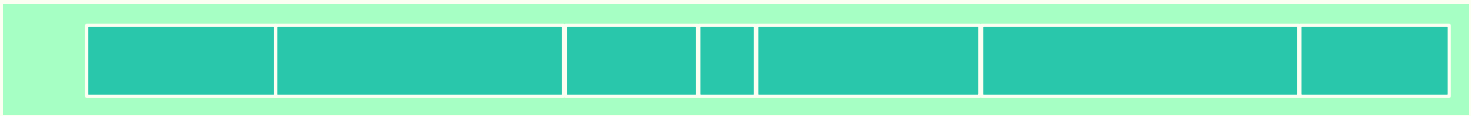
FIFO queue

OUR ALGORITHM



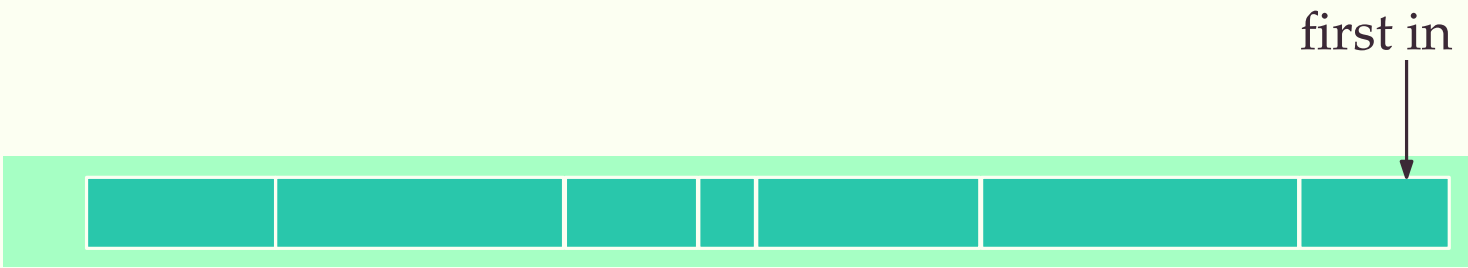
FIFO queue

OUR ALGORITHM



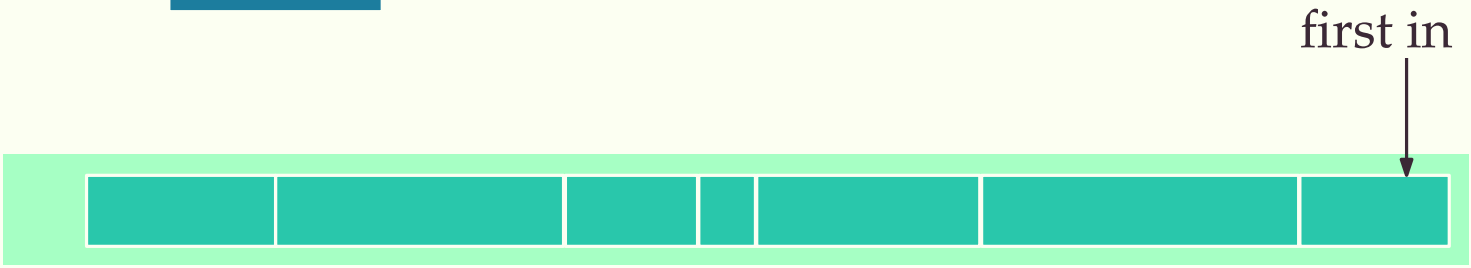
FIFO queue

OUR ALGORITHM



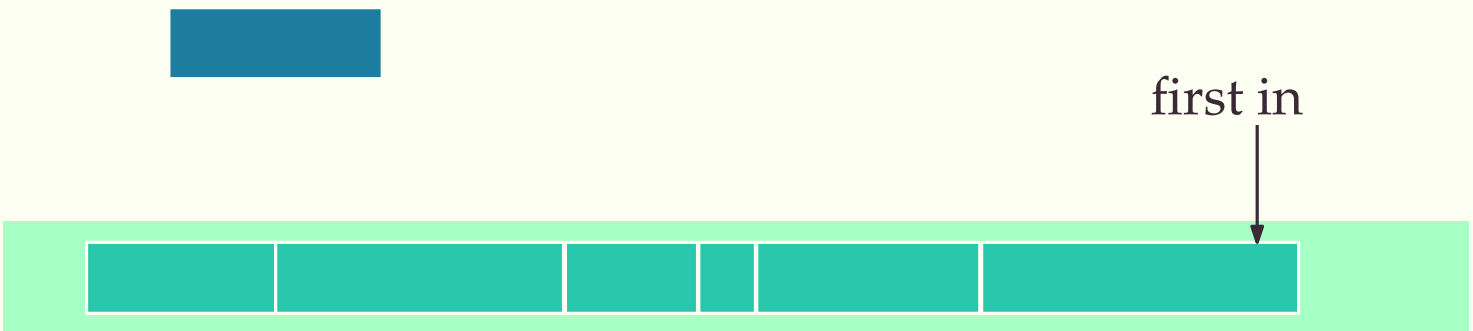
FIFO queue

OUR ALGORITHM



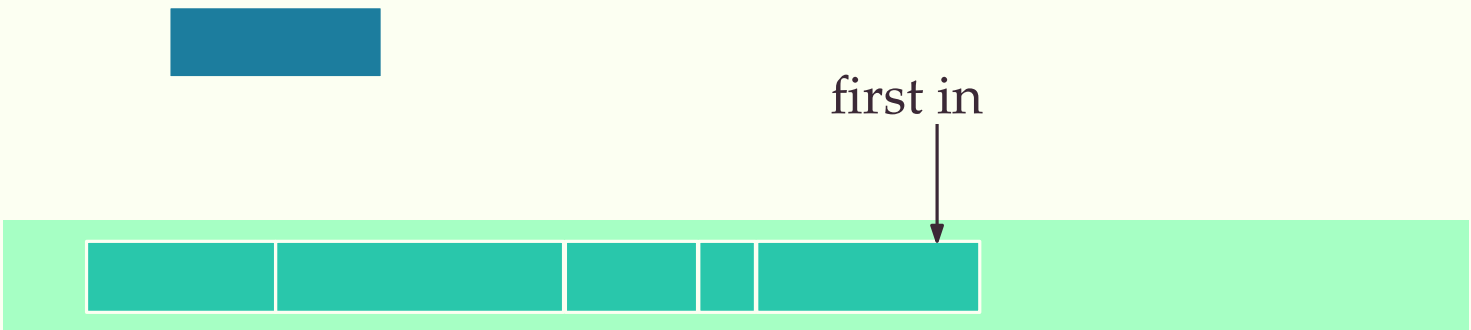
FIFO queue

OUR ALGORITHM



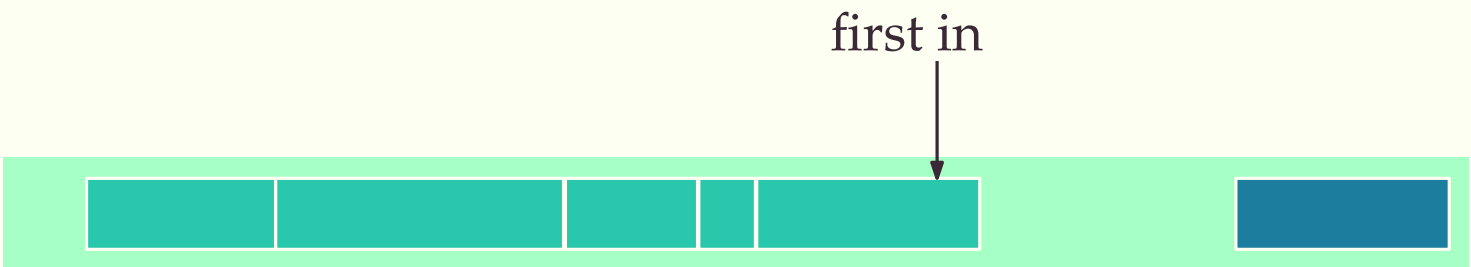
FIFO queue

OUR ALGORITHM



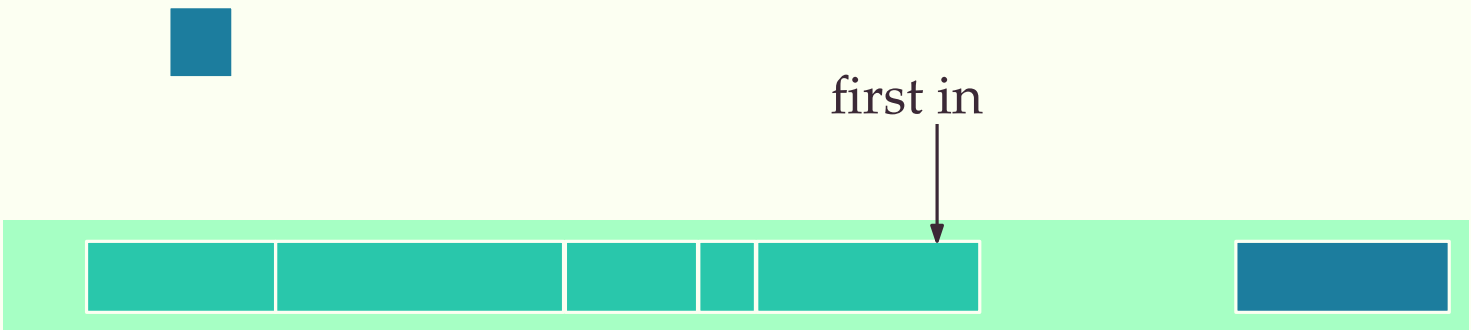
FIFO queue

OUR ALGORITHM



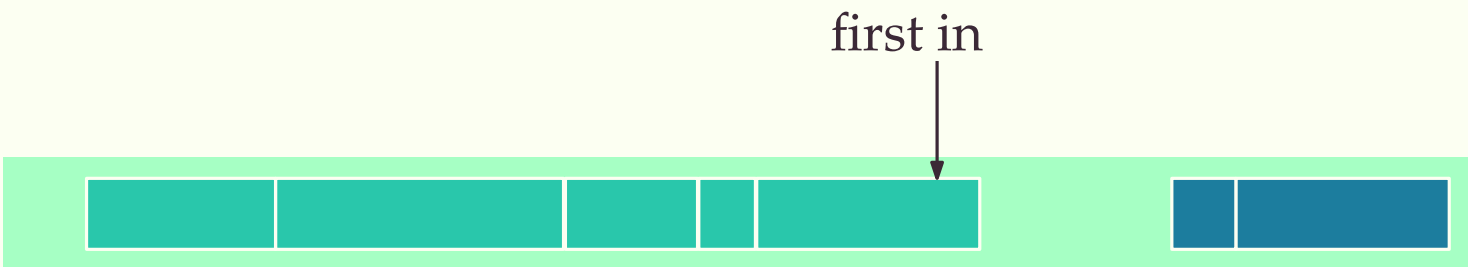
FIFO queue

OUR ALGORITHM



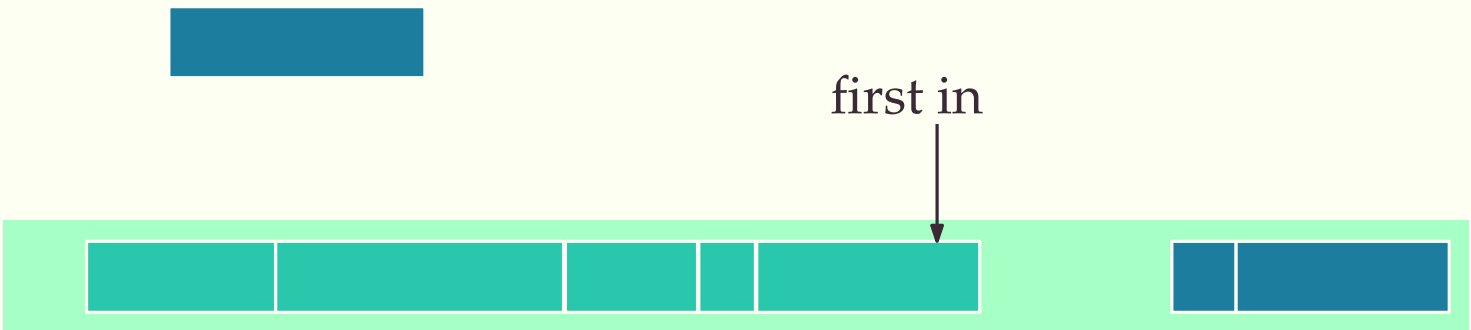
FIFO queue

OUR ALGORITHM



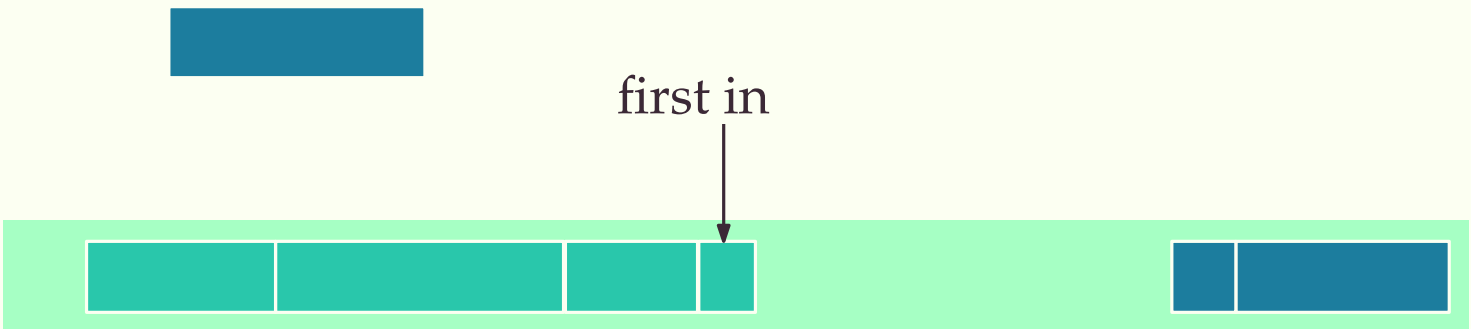
FIFO queue

OUR ALGORITHM



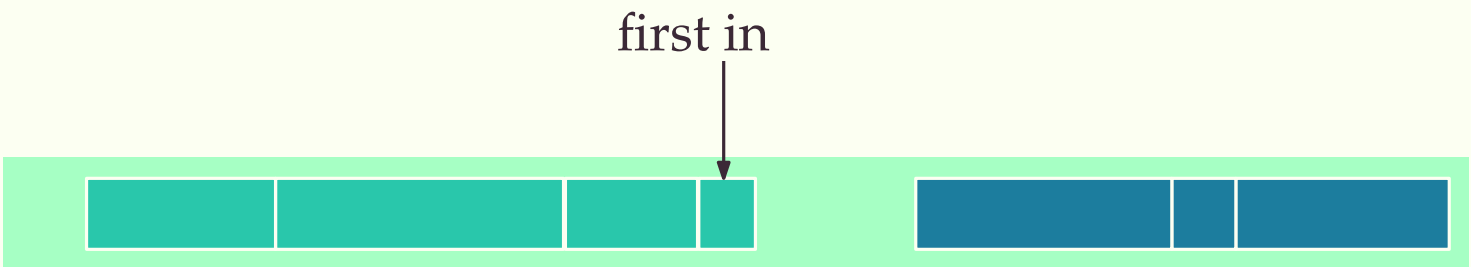
FIFO queue

OUR ALGORITHM



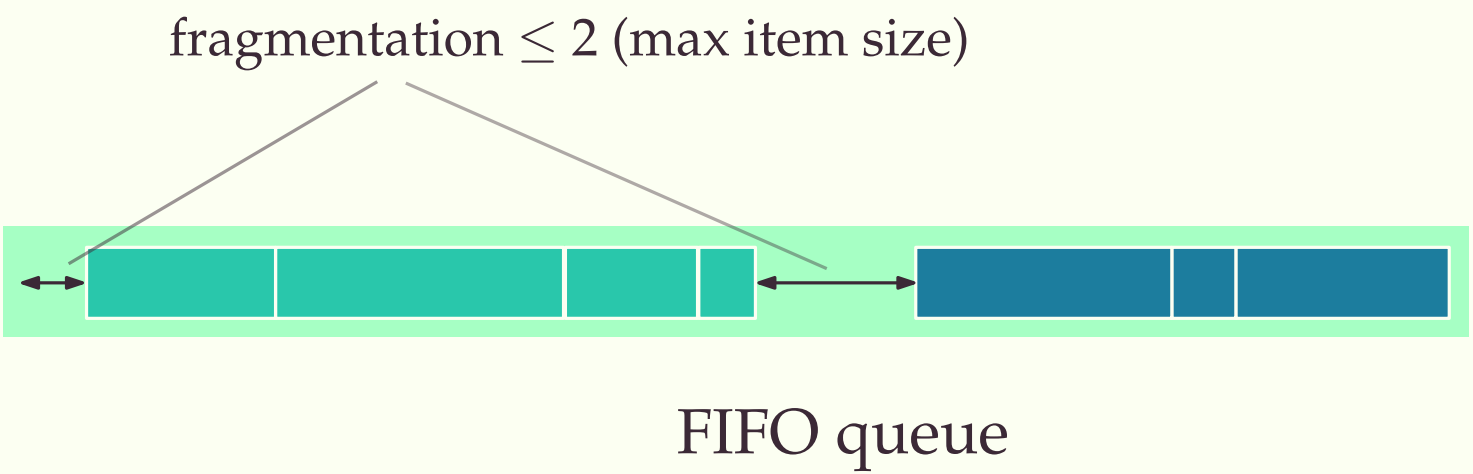
FIFO queue

OUR ALGORITHM



FIFO queue

OUR ALGORITHM



OUR ALGORITHM

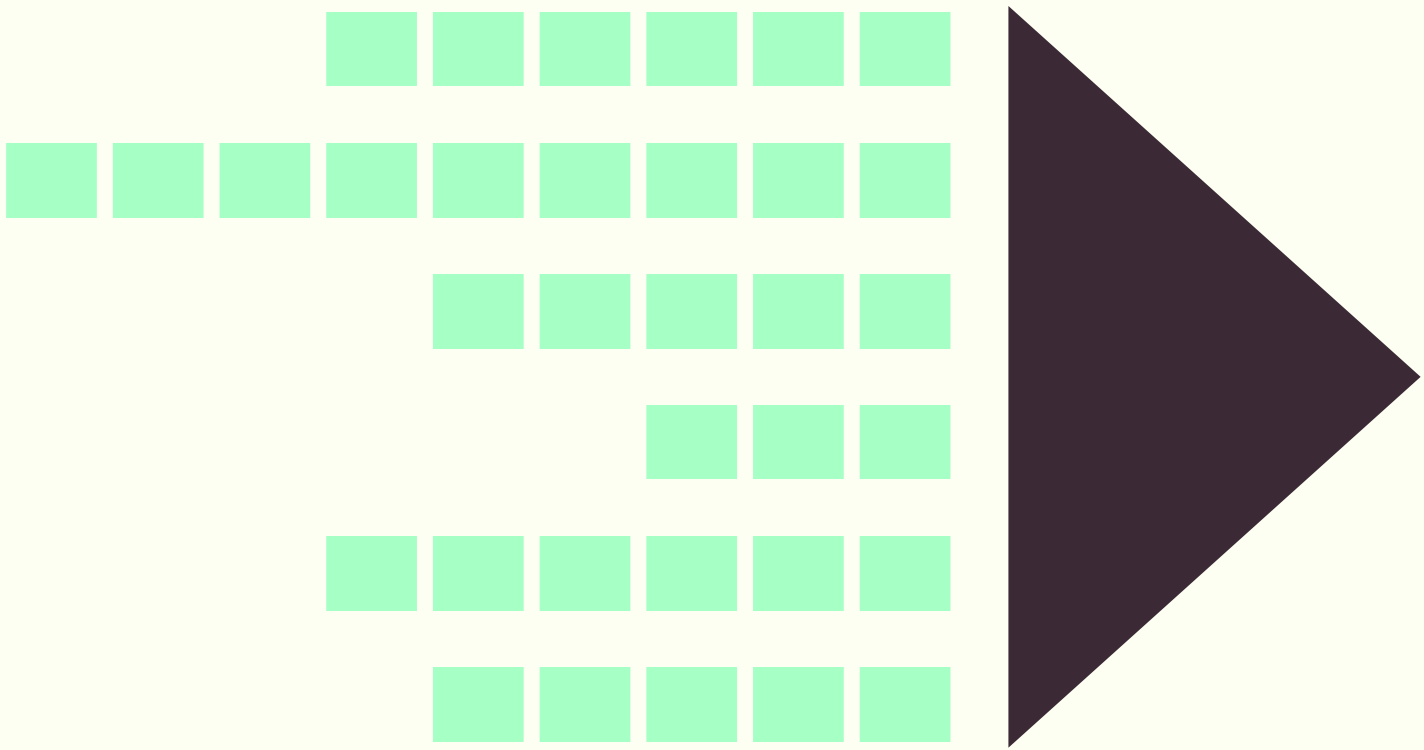
OUR ALGORITHM

OUR ALGORITHM

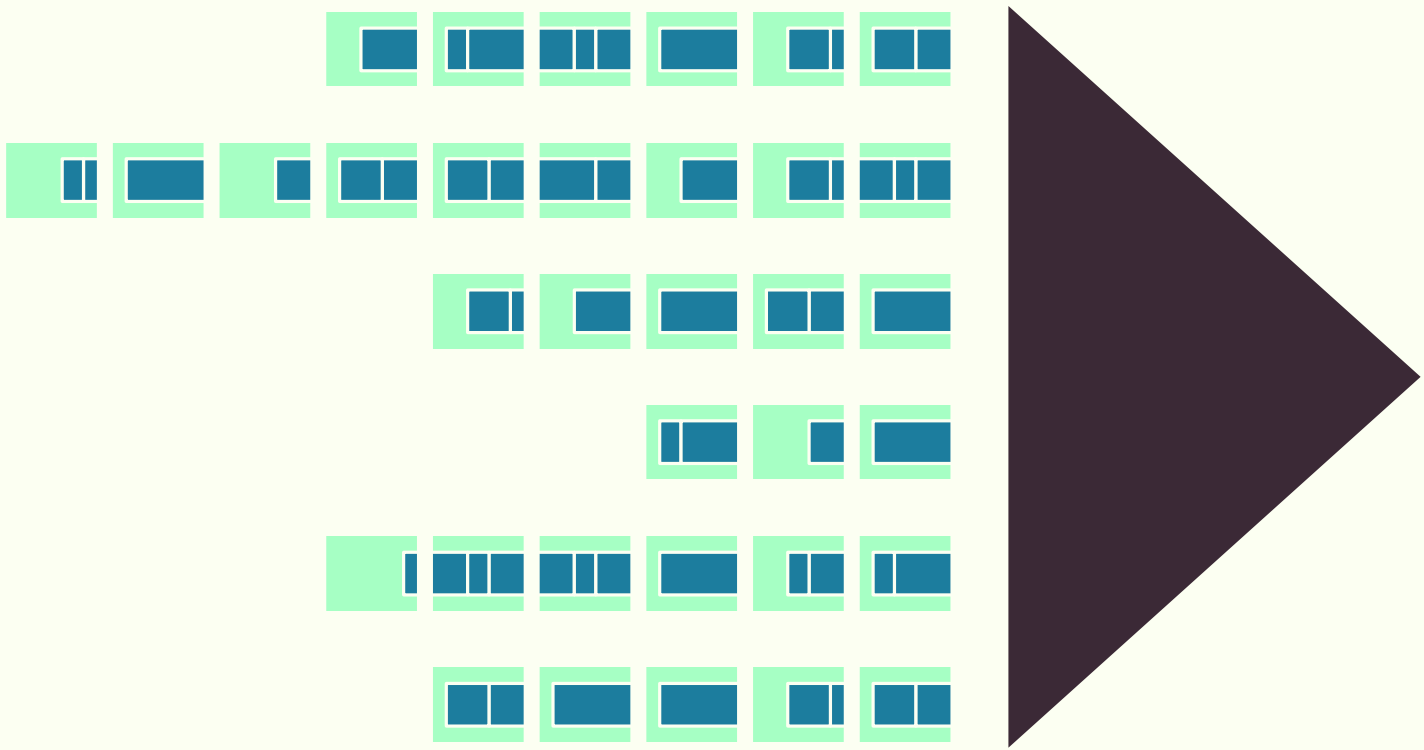
OUR ALGORITHM

OUR ALGORITHM

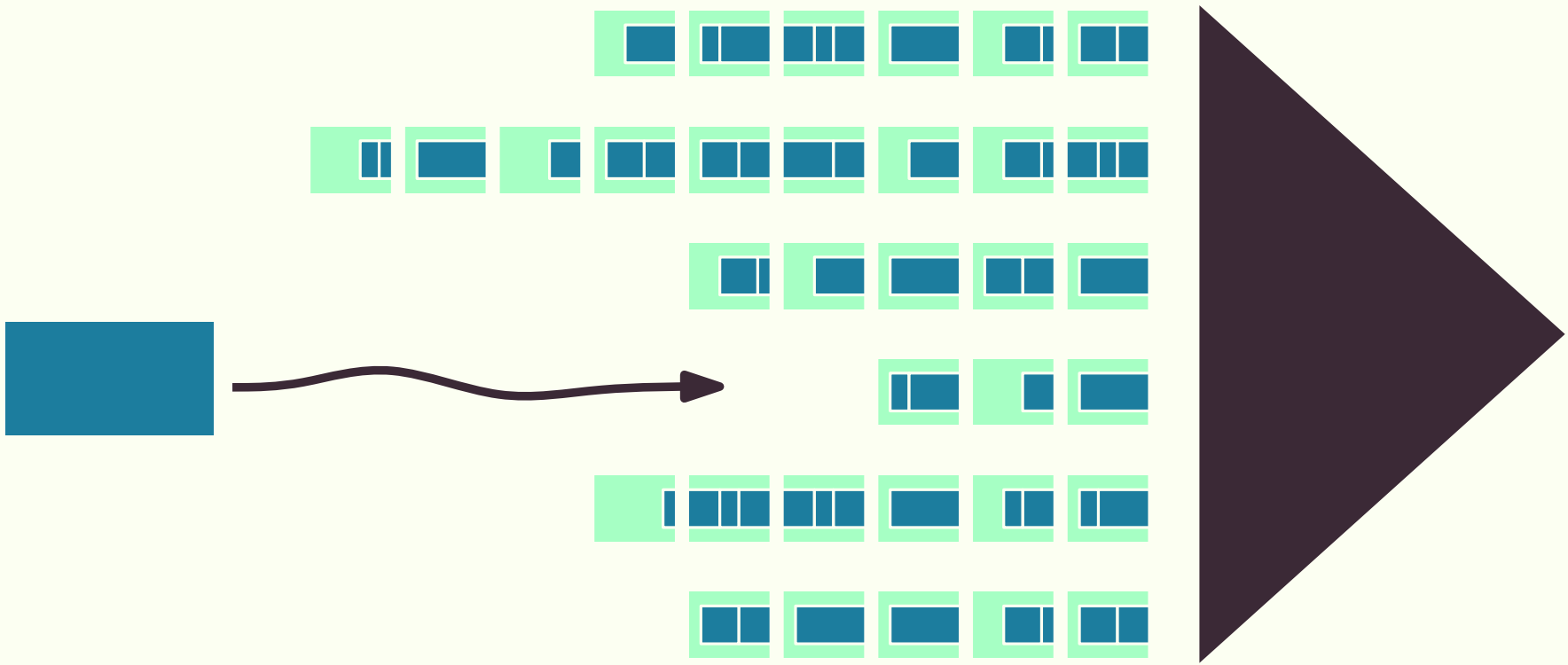
OUR ALGORITHM



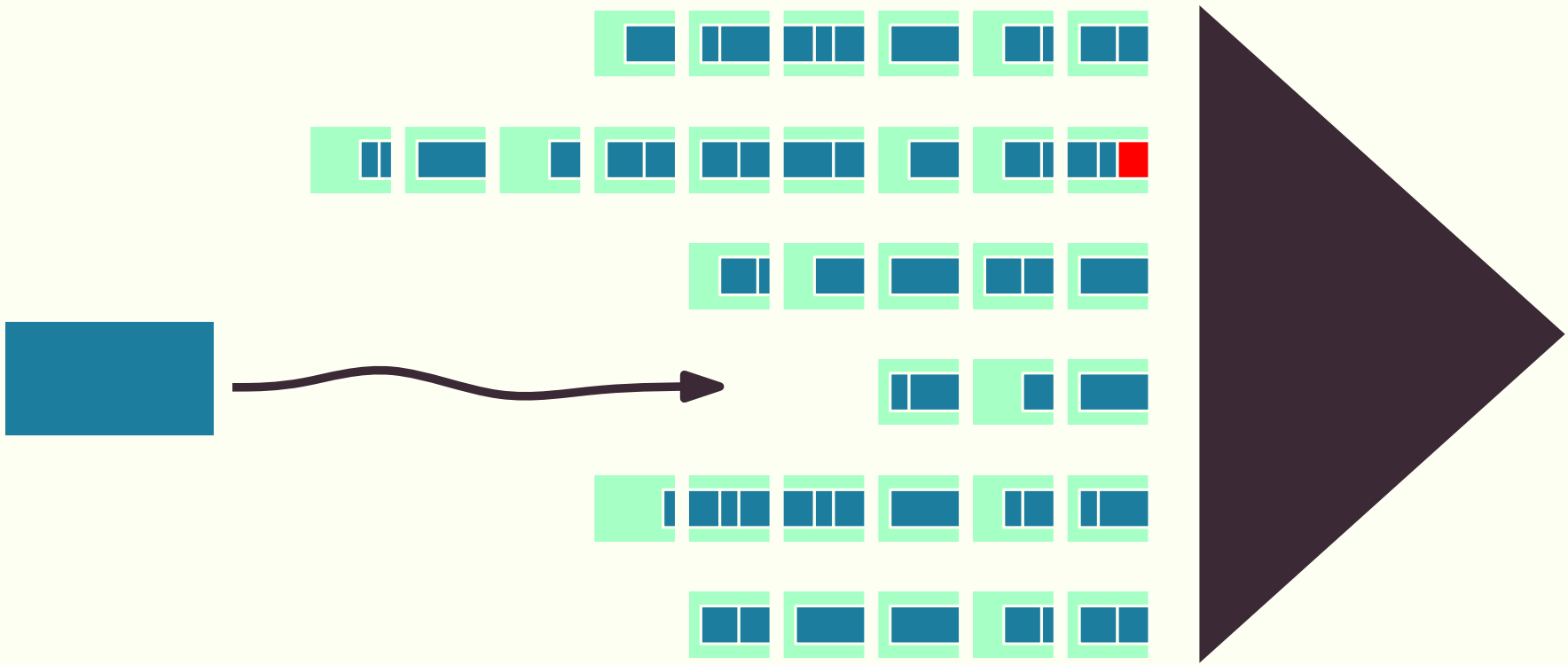
OUR ALGORITHM



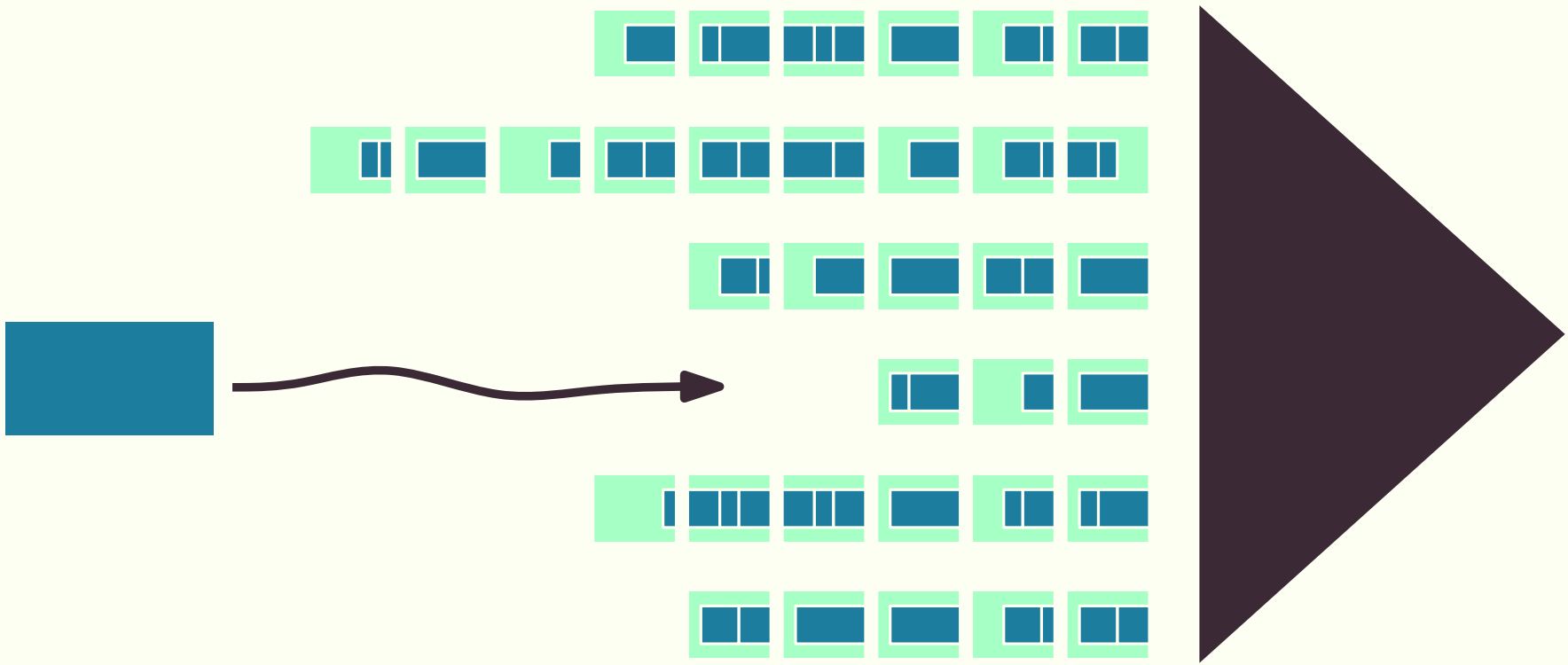
OUR ALGORITHM



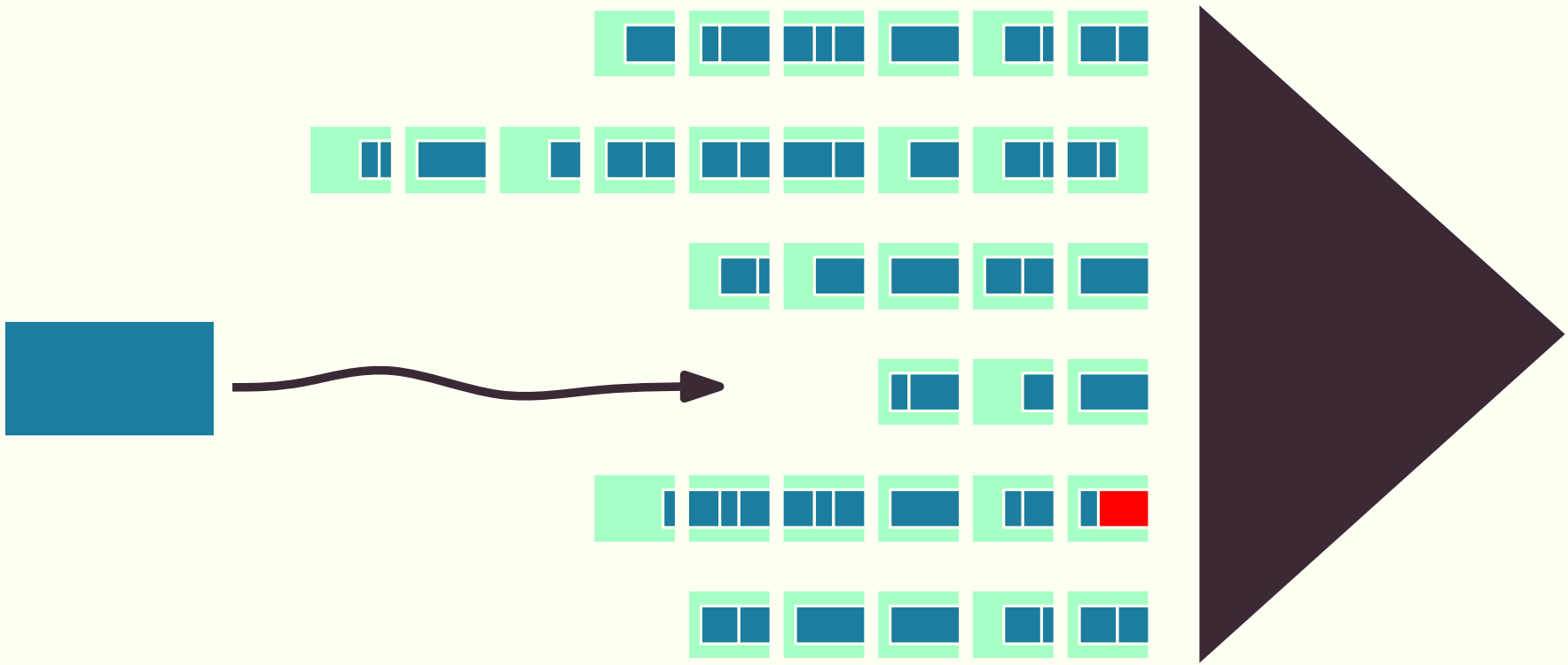
OUR ALGORITHM



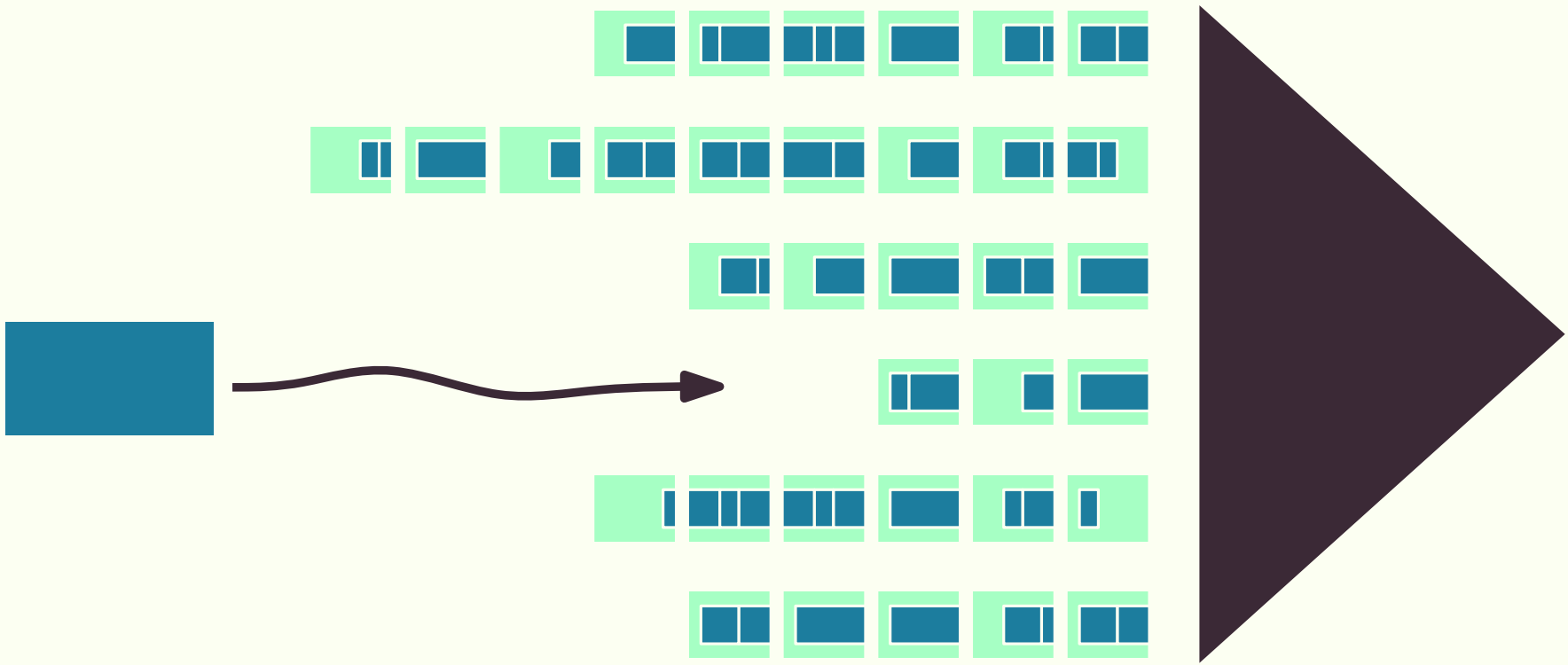
OUR ALGORITHM



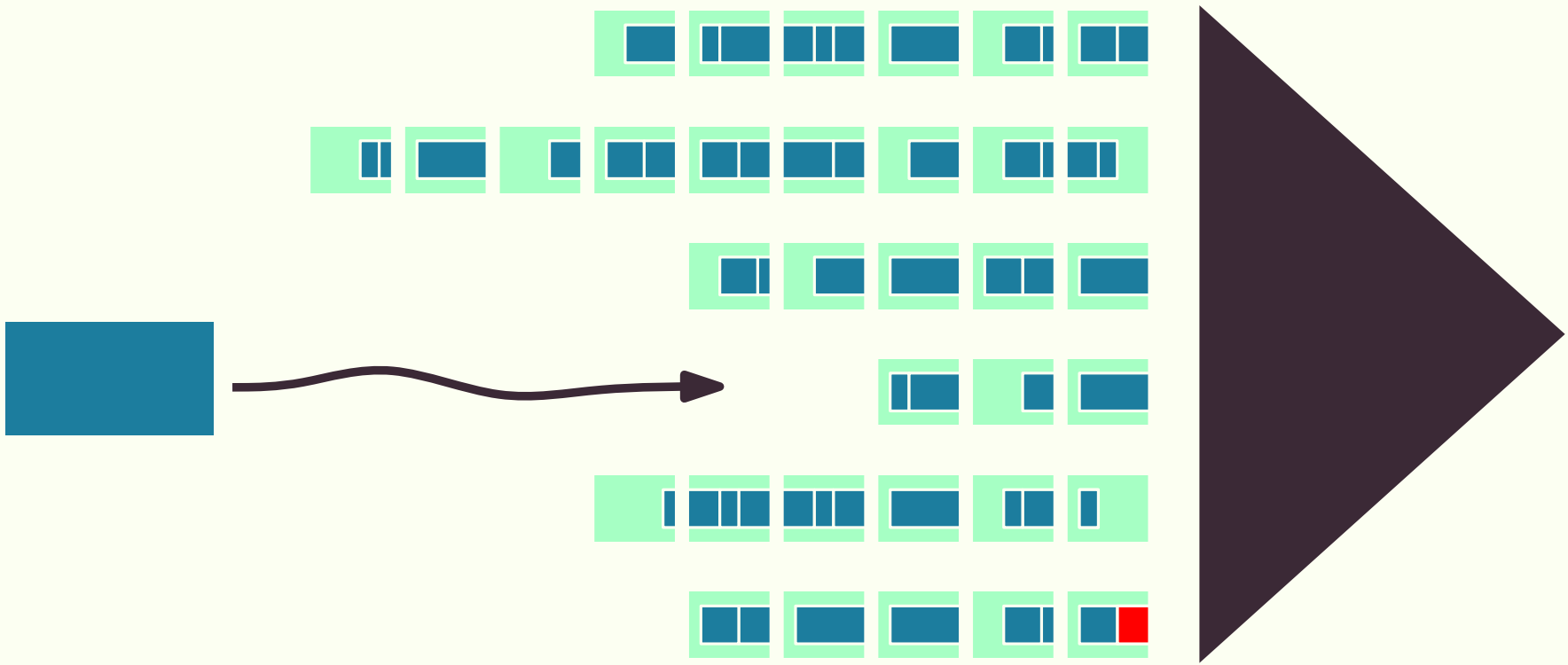
OUR ALGORITHM



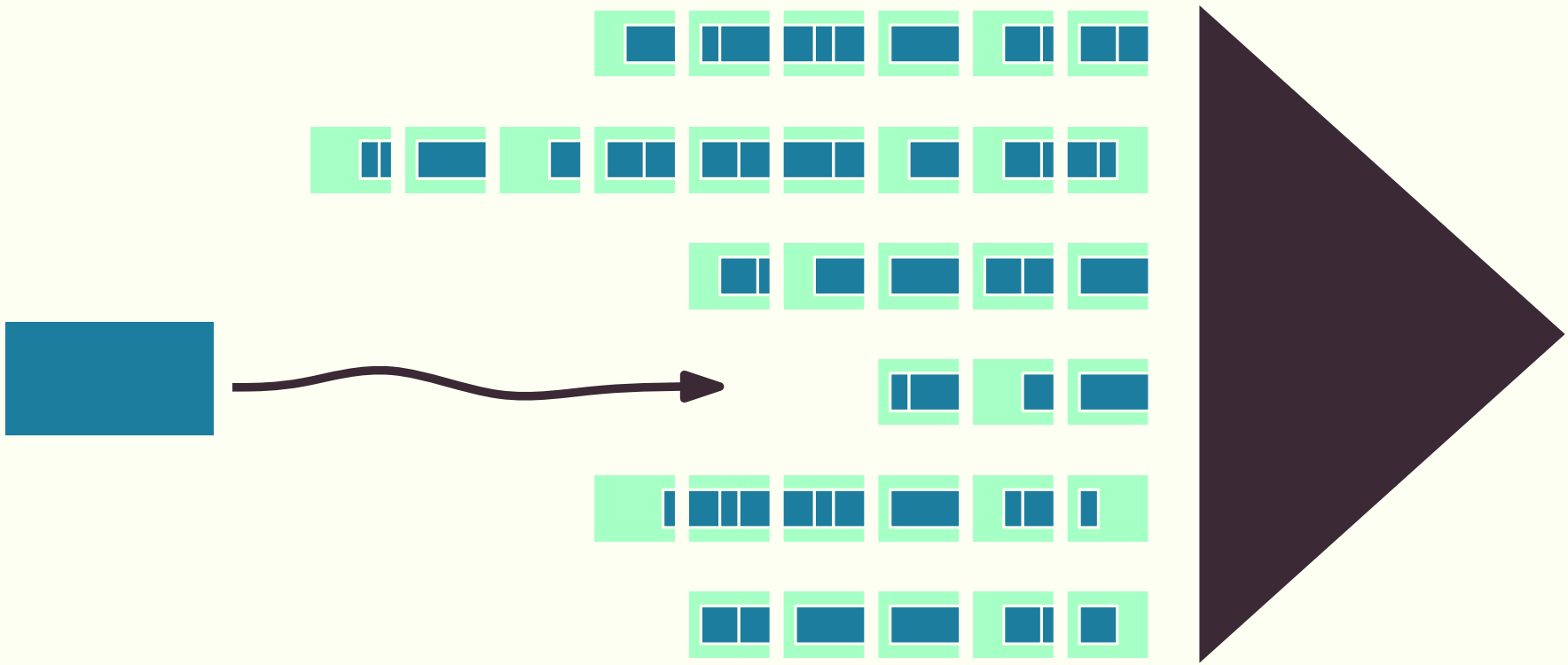
OUR ALGORITHM



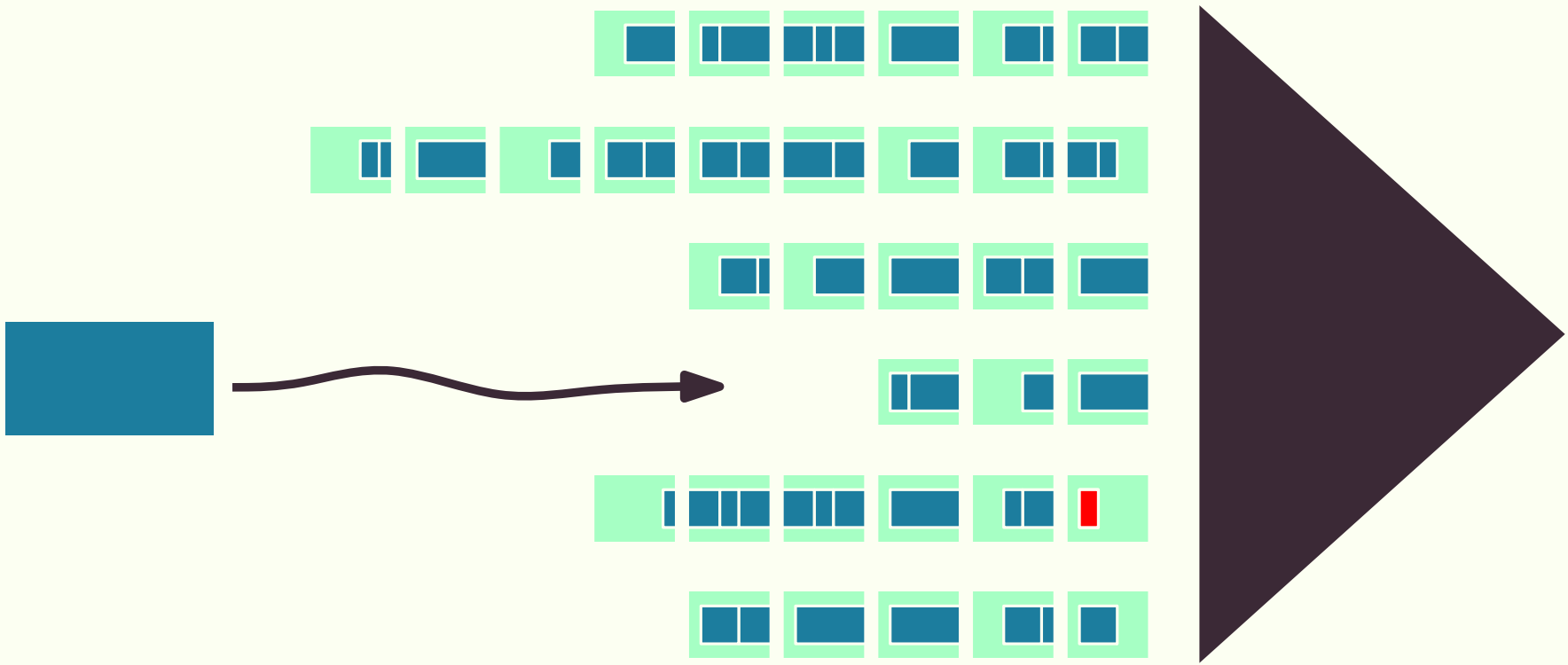
OUR ALGORITHM



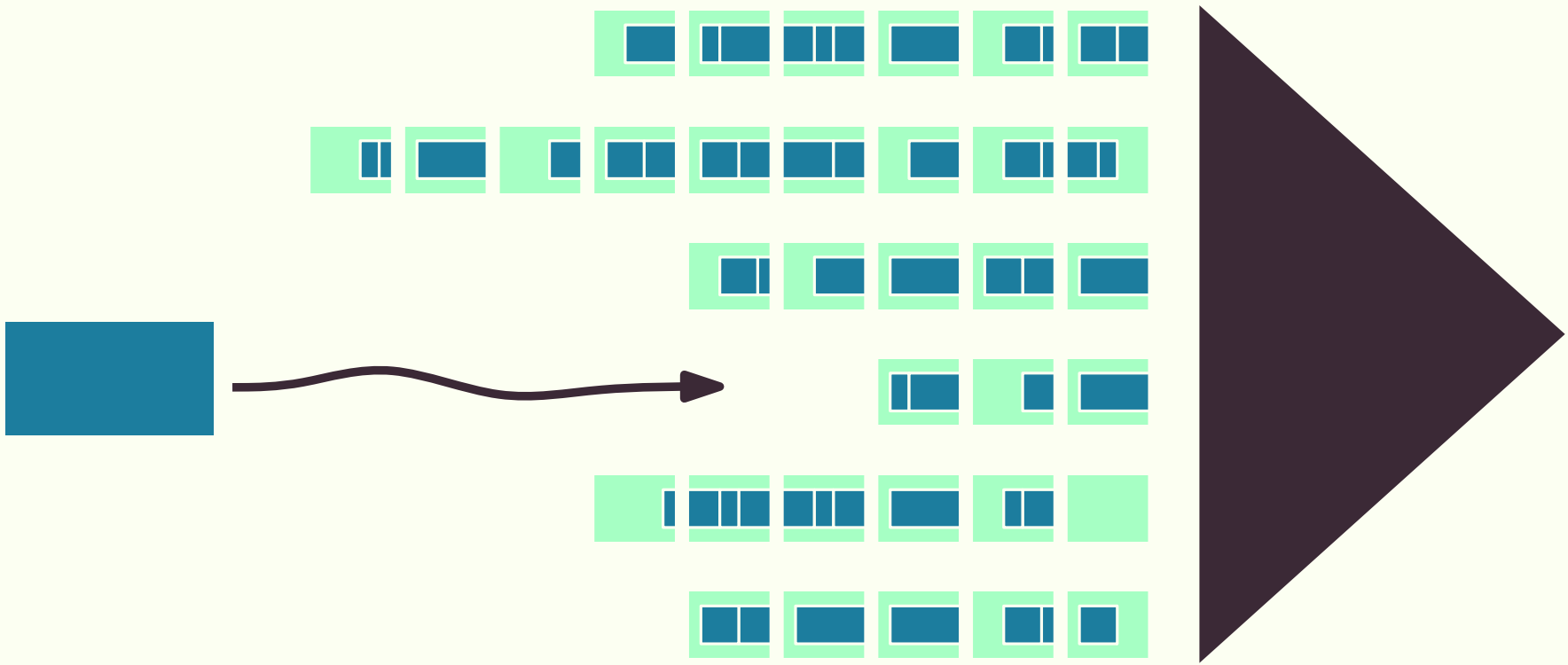
OUR ALGORITHM



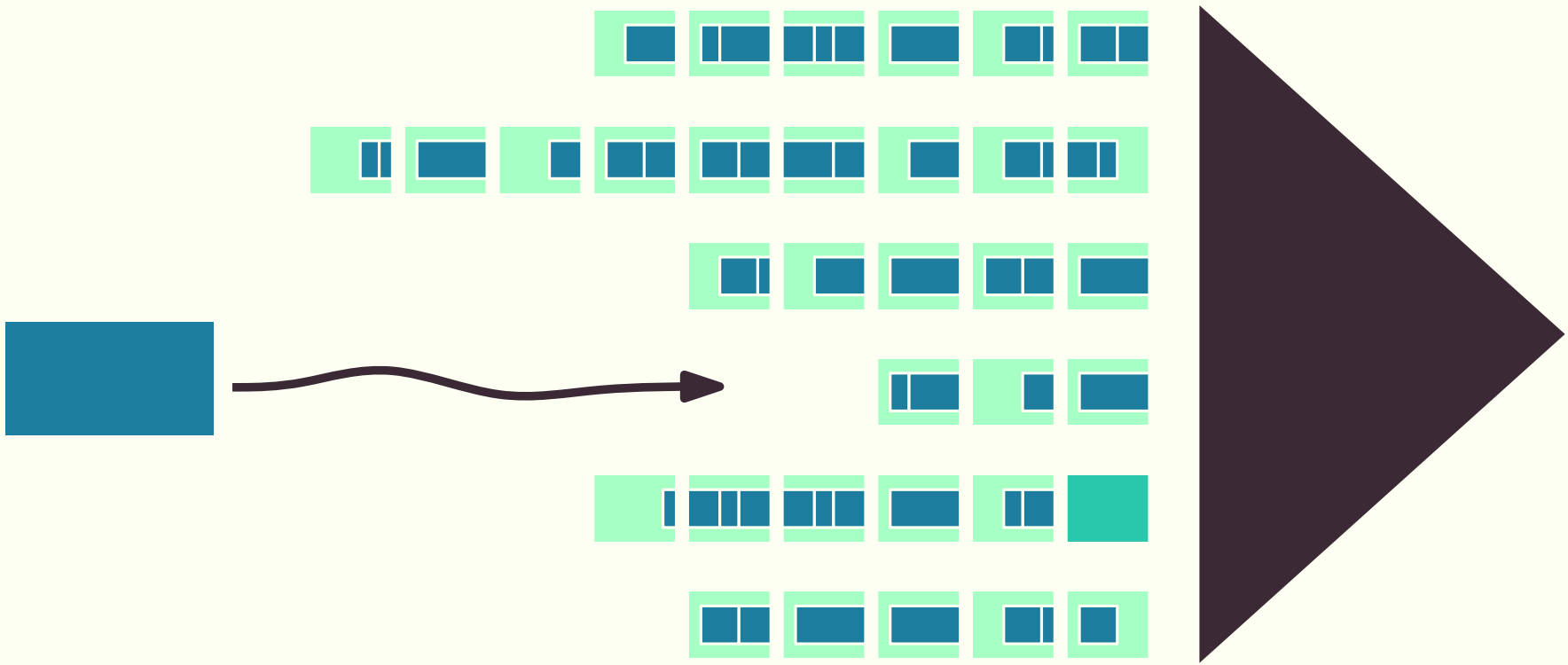
OUR ALGORITHM



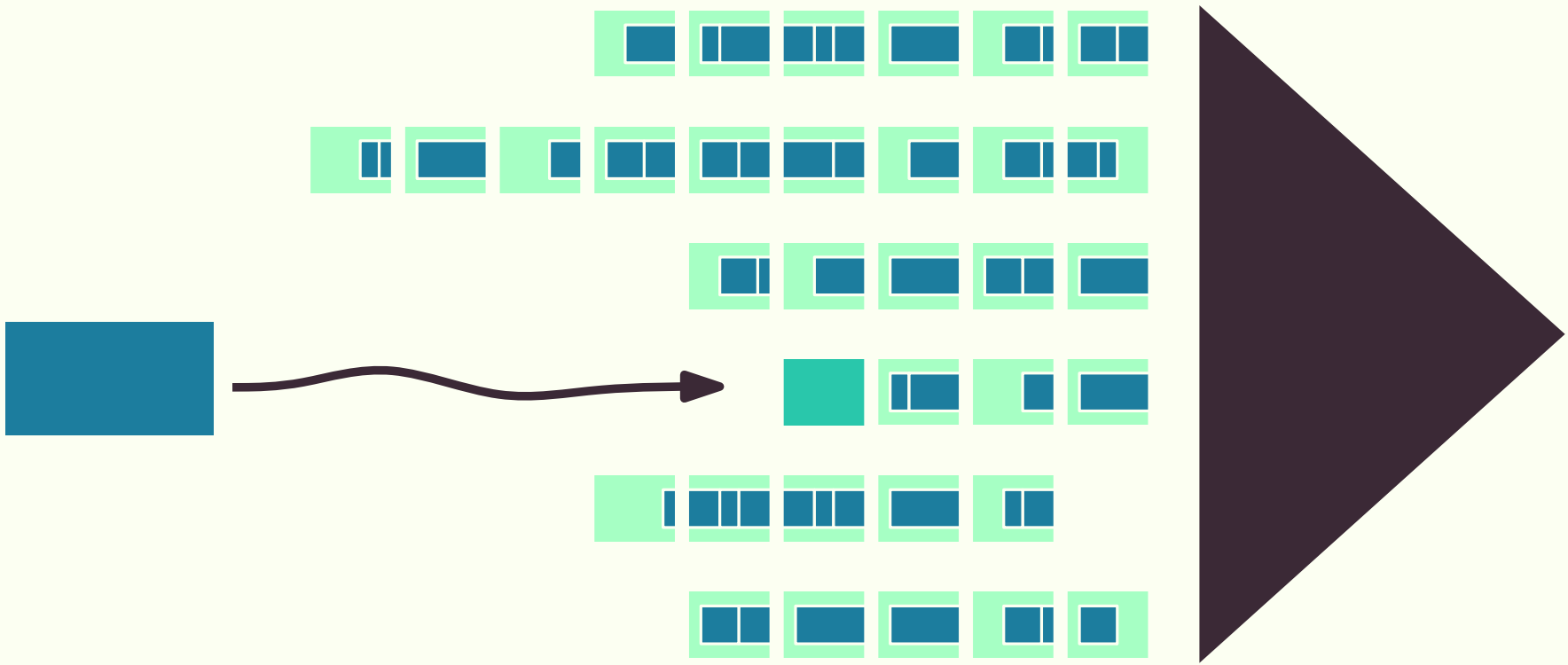
OUR ALGORITHM



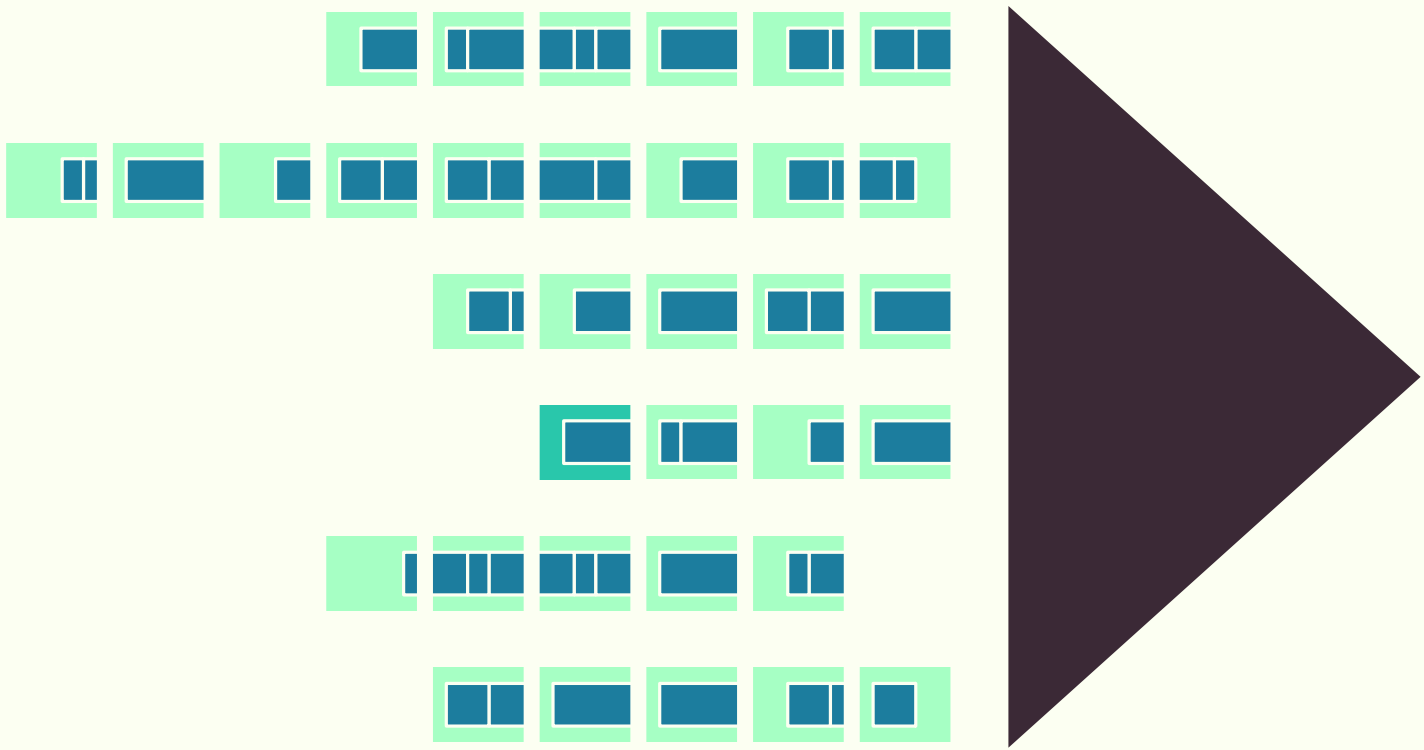
OUR ALGORITHM



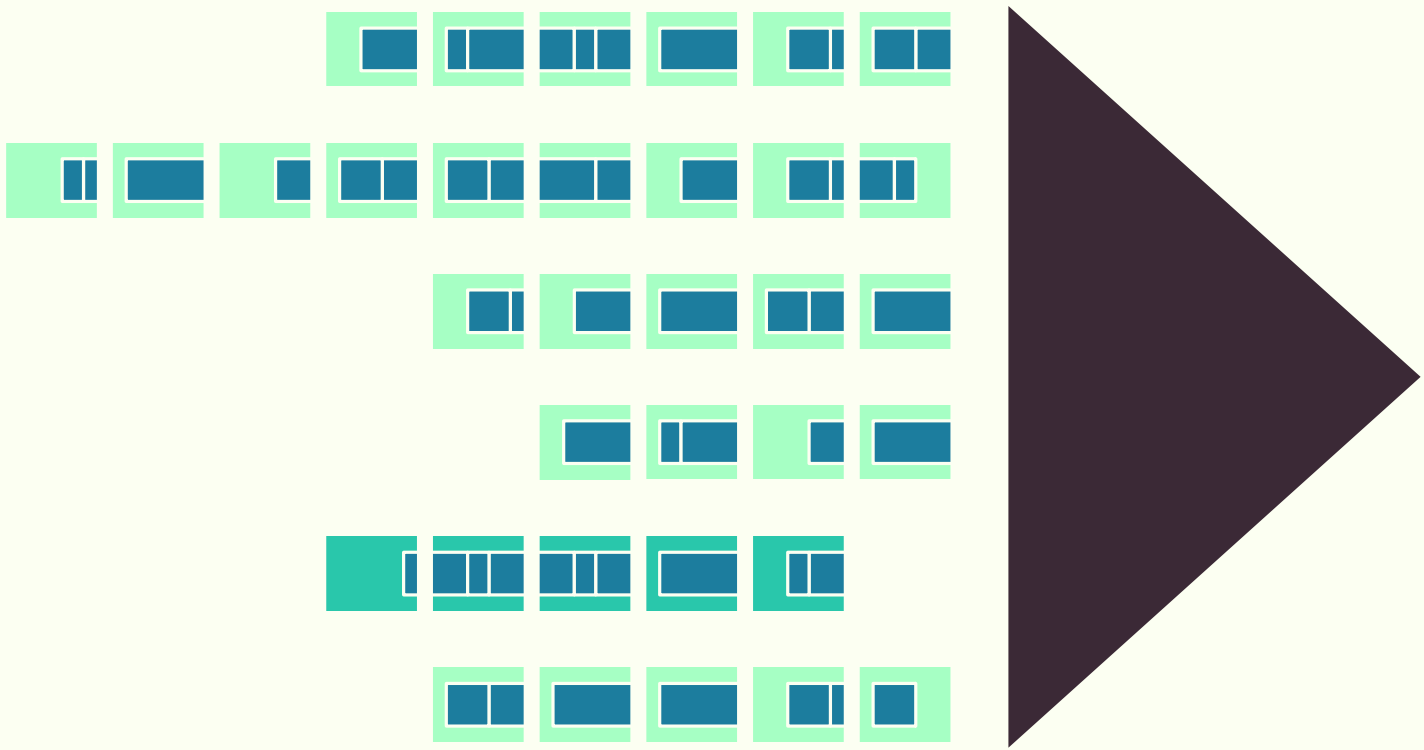
OUR ALGORITHM



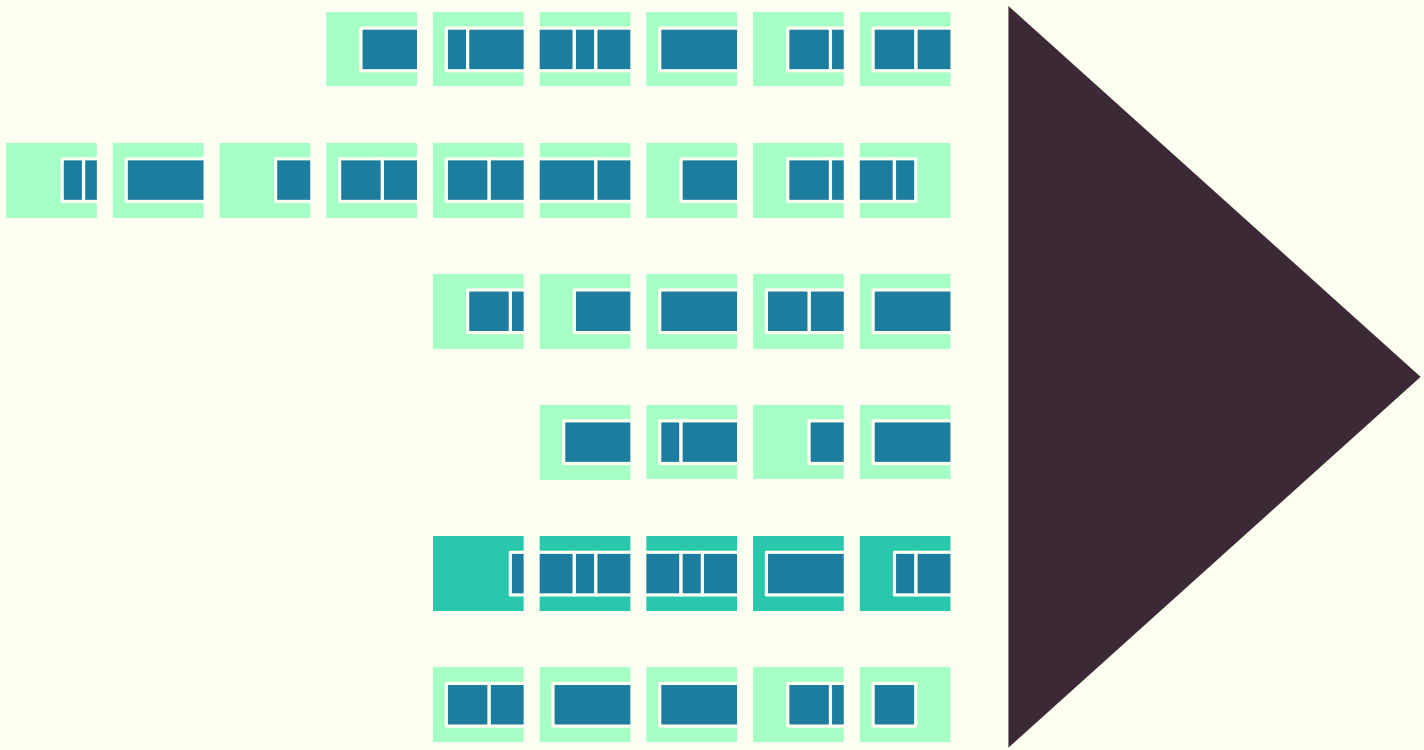
OUR ALGORITHM



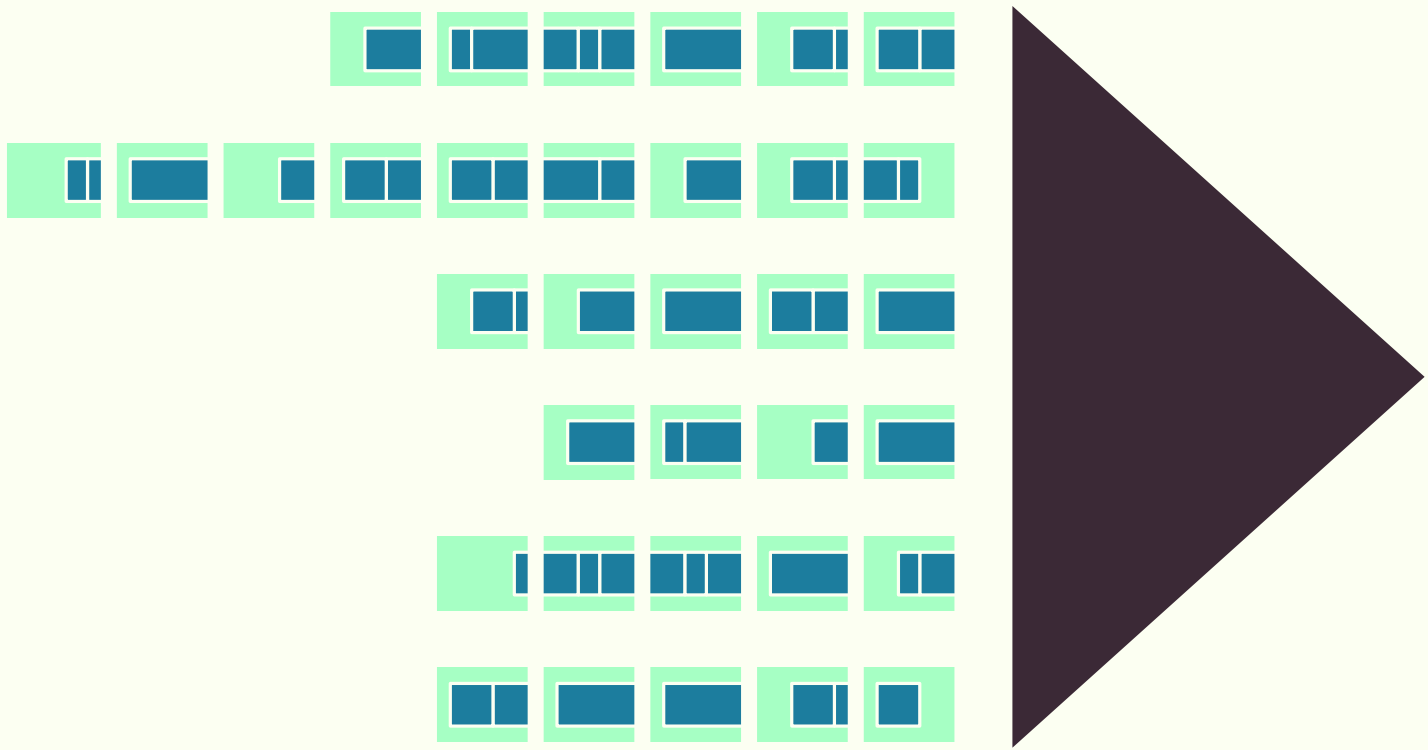
OUR ALGORITHM



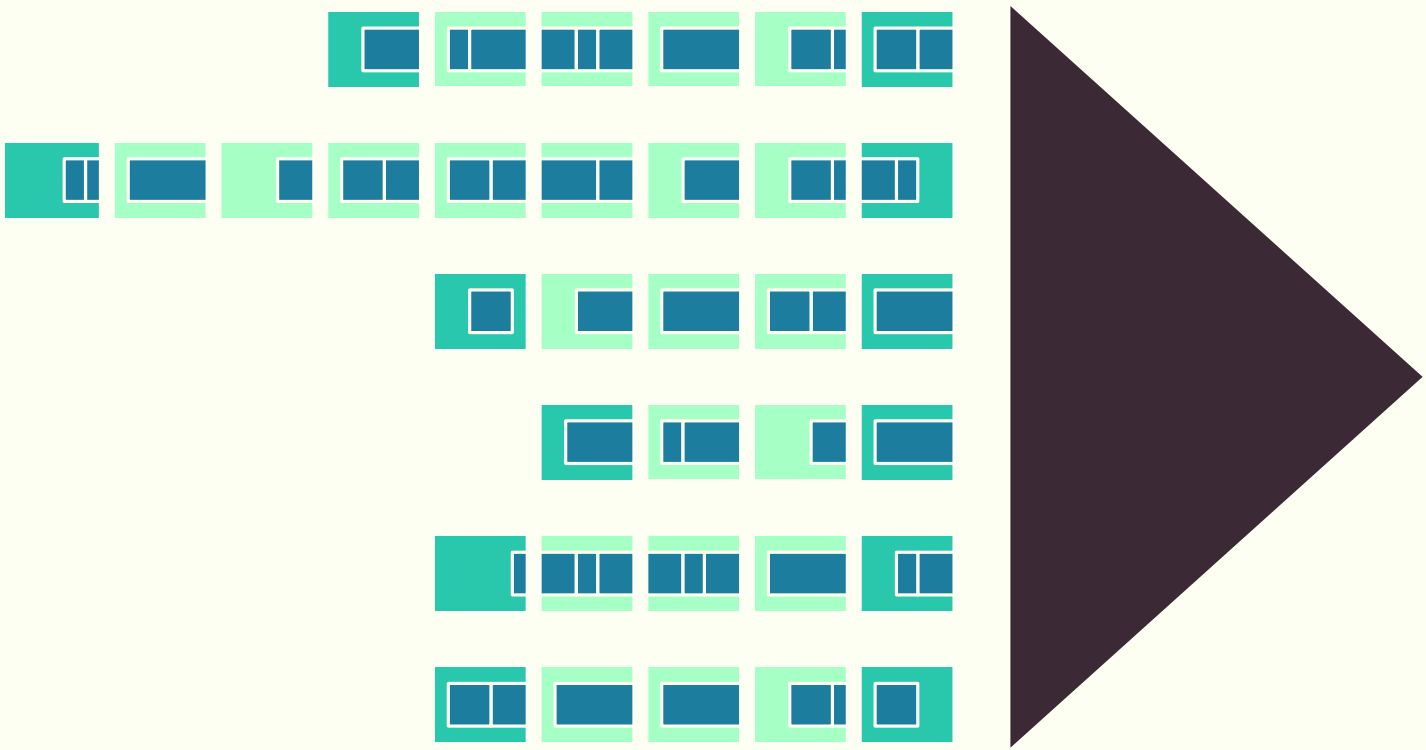
OUR ALGORITHM



OUR ALGORITHM



OUR ALGORITHM



fragmentation $\leq$ 2 (num queues) (block size) + (num blocks) (max item size)

OUR ALGORITHM

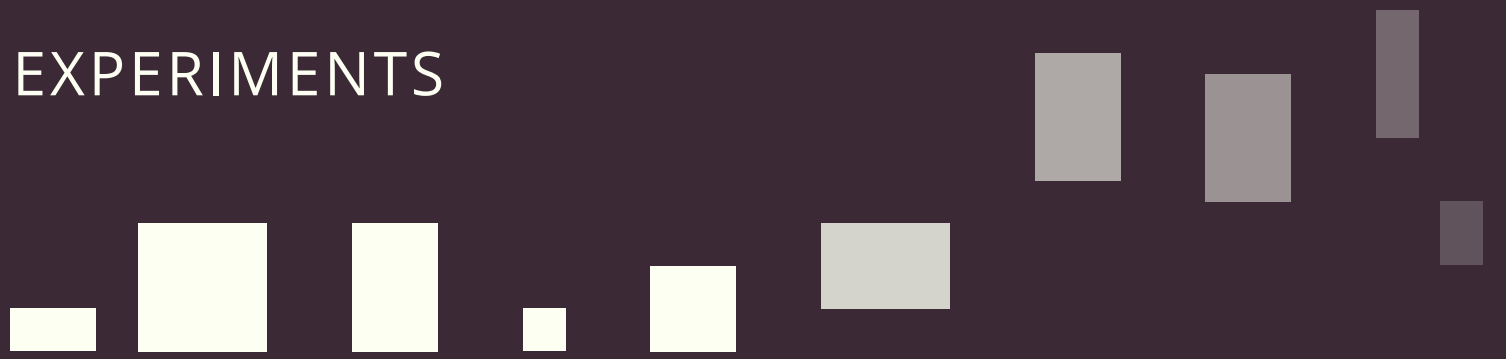
is competitive if memory augmented

$$\text{if} \quad \begin{matrix}\text{OPT's}\\ \text{cache size}\end{matrix} \leq \begin{matrix}\text{ALG's}\\ \text{cache size}\end{matrix} - \begin{matrix}\text{fragmentation}\\ \text{bound}\end{matrix}$$

$$\text{then} \quad \text{cost(ALG)} \leq \frac{\begin{matrix}\text{ALG's}\\ \text{cache size}\end{matrix}}{\text{min item size}} \text{cost(OPT)}$$

fragmentation $\leq$ 2 (num queues) (block size) + (num blocks) (max item size)

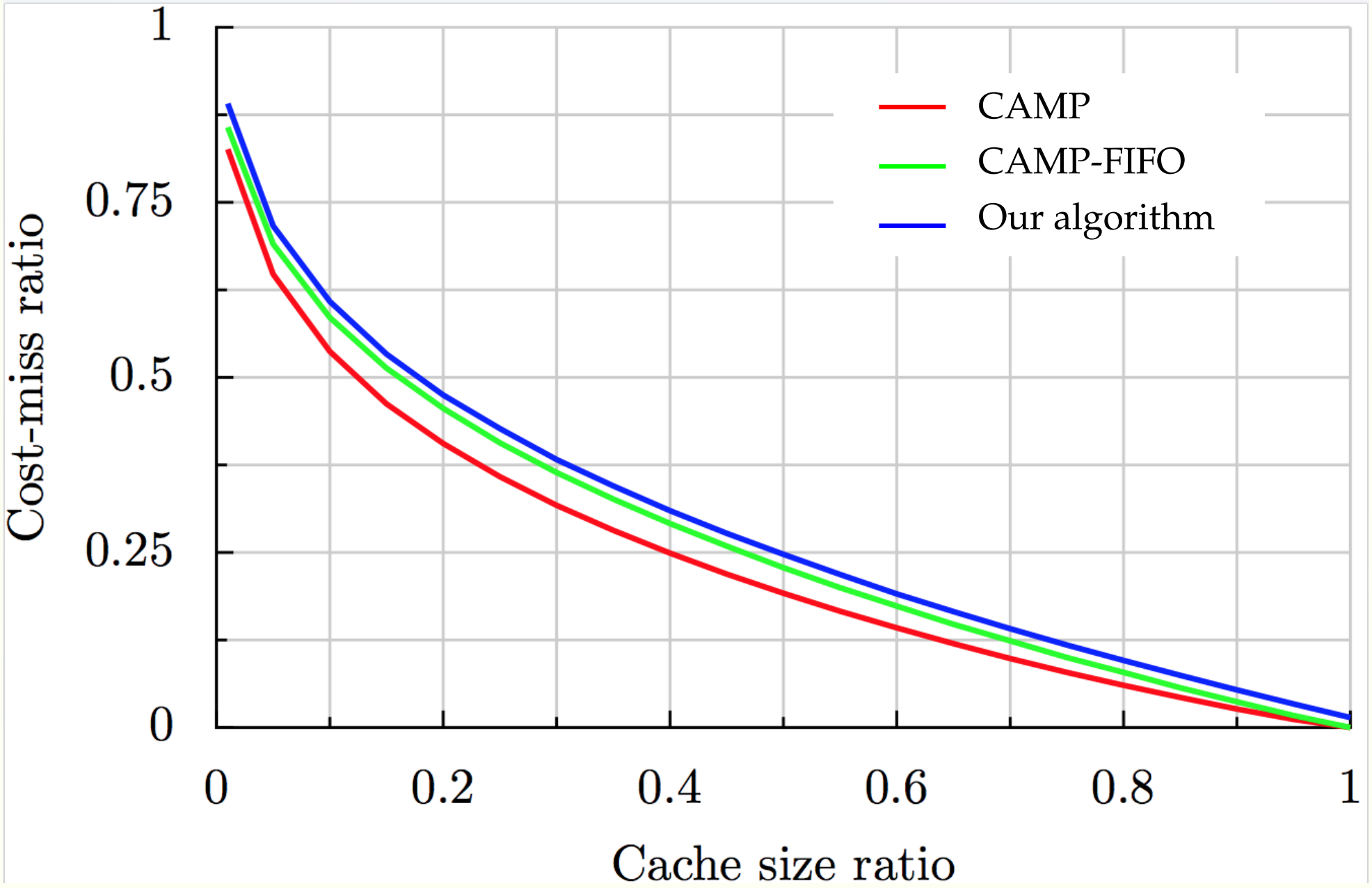
EXPERIMENTS

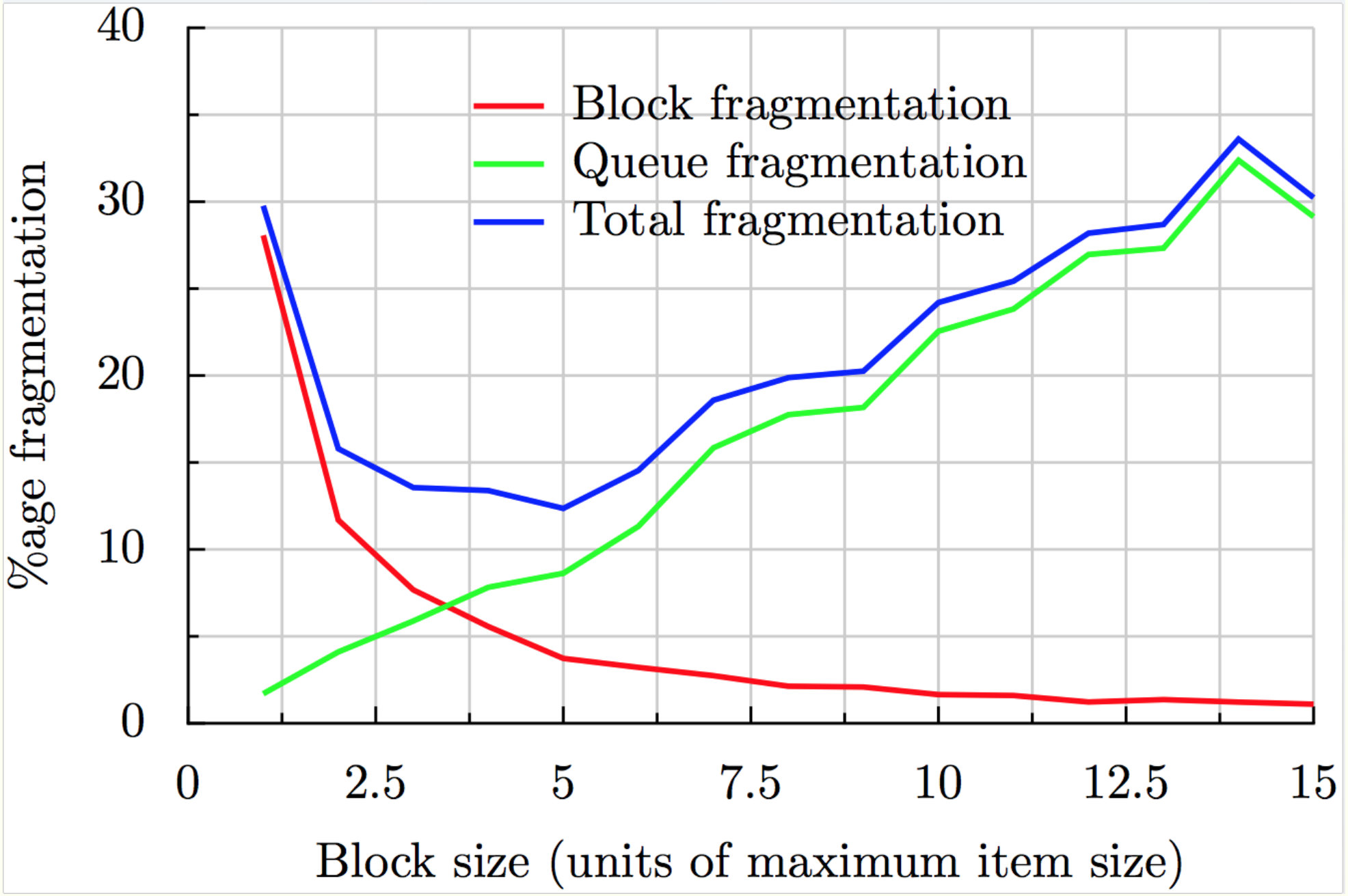


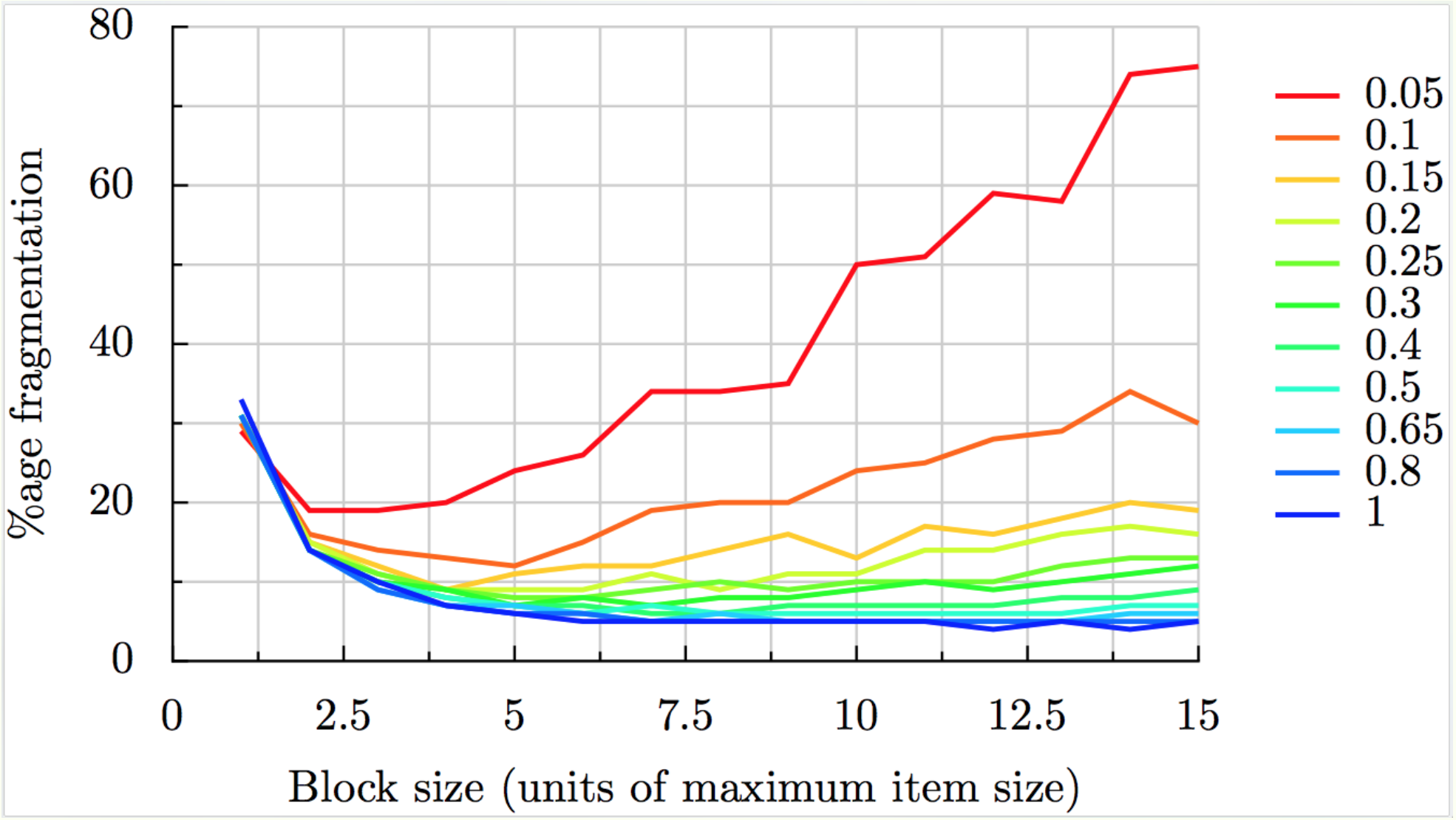
trace generated by BG, a social networking benchmark

4 million requests

i.i.d. with 70% of requests to 20% of items


CAMP
CAMP-FIFO
Our algorithm
Cost-miss ratio
Cache size ratio
0
0.25
0.5
0.75
1
0
0.2
0.4
0.6
0.8
1


Block fragmentation
Queue fragmentation
Total fragmentation
%age fragmentation
Block size (units of maximum item size)
0
10
20
30
40
0
2.5
5
7.5
10
12.5
15


80
60
40
20
0
%age fragmentation
0
2.5
5
7.5
10
12.5
15
Block size (units of maximum item size)
0.05
0.1
0.15
0.2
0.25
0.3
0.4
0.5
0.65
0.8
1

CONCLUSION

biggest performance degradation
going from LRU to FIFO queues

FUTURE RESEARCH

design a better algorithm for
the managed-memory caching problem
allowing moving items once in cache