Advanced Algorithmic Techniques (COMP523)

Online Algorithms 2

Recap and plan

- **Last lecture:**
  - Online Algorithms
  - Competitive Ratio
  - Online load balancing
  - Paging
- **This lecture:**
  - Online algorithms for paging.
  - A Randomised online algorithm for paging.

Paging Algorithms

- LRU (*Least Recently Used*): Replace the page that was requested the least recently.
- FIFO (*First-In First-Out*): Replace the page that has been in the cache the longest.
- LIFO (*Last-In First-Out*): Replace the page that has been in the cache the shortest.
- LFU (*Least Frequently Used*): Replace the page that was requested the least frequently so far.
- MIN (*Offline Optimal*): Replace the page whose next request happens the furthest in the future.

Paging Algorithms

- Theorem: LRU and FIFO have competitive ratio k.

Marking algorithm

- Consider the following algorithm:

Marking algorithm

- Consider the following algorithm:
  - The algorithm proceeds in *phases*.

Marking algorithm

- Consider the following algorithm:
  - The algorithm proceeds in *phases*.
  - At the beginning of a phase, all the pages are unmarked.

Marking algorithm

- Consider the following algorithm:
  - The algorithm proceeds in *phases*.
  - At the beginning of a phase, all the pages are unmarked.
  - Whenever a page is requested, it is marked.

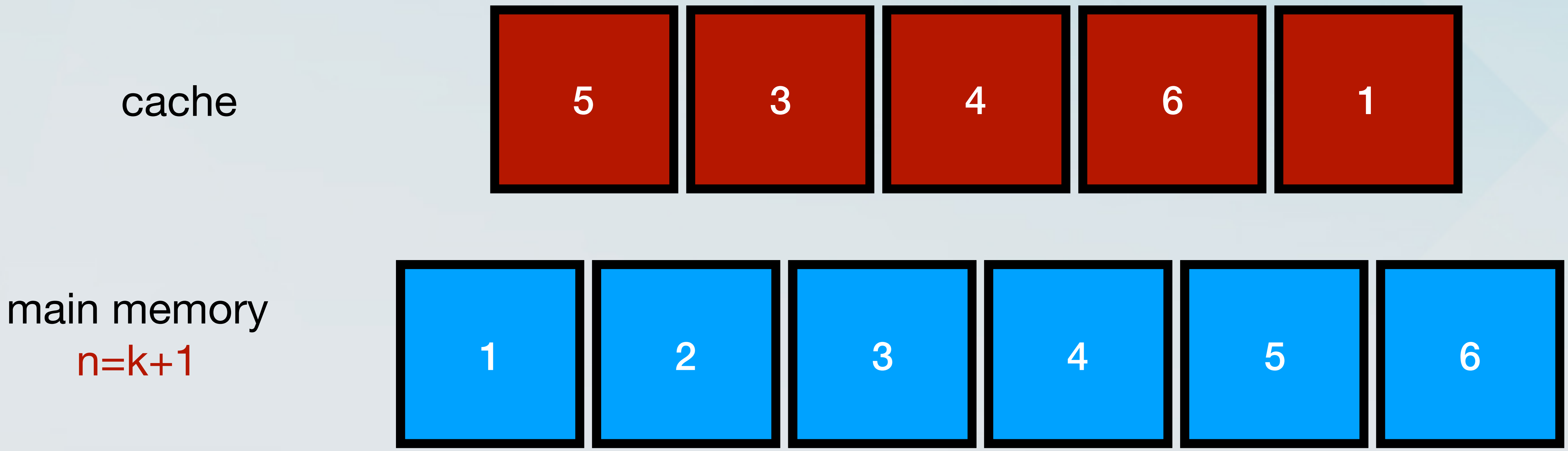
Marking algorithm

- Consider the following algorithm:
  - The algorithm proceeds in *phases*.
  - At the beginning of a phase, all the pages are unmarked.
  - Whenever a page is requested, it is marked.
  - When a "*fault*" occurs, the algorithm replaces an unmarked page.

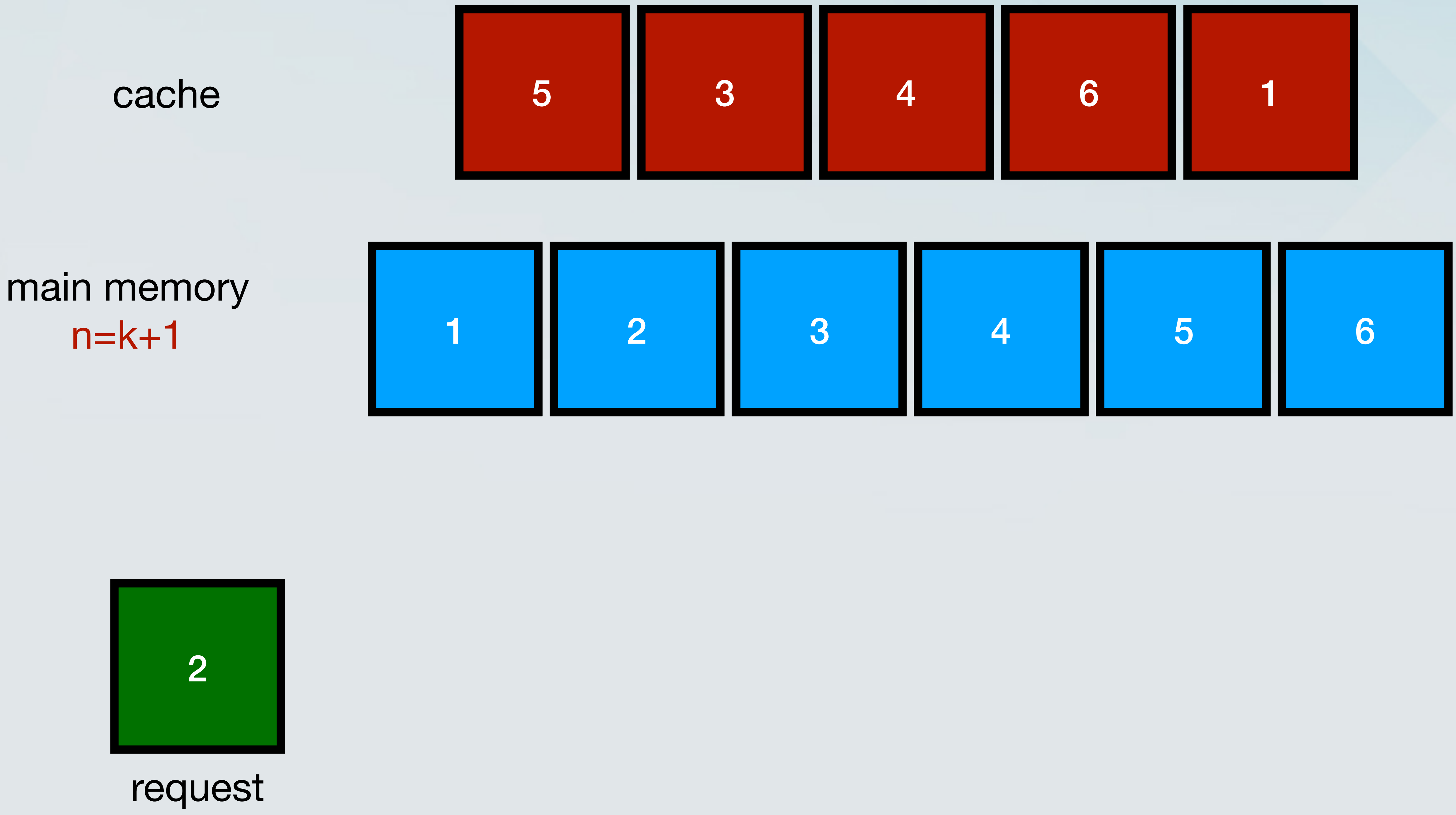
Marking algorithm

- Consider the following algorithm:
  - The algorithm proceeds in *phases*.
  - At the beginning of a phase, all the pages are unmarked.
  - Whenever a page is requested, it is marked.
  - When a "*fault*" occurs, the algorithm replaces an unmarked page.
  - When all pages in the cache are marked, and a request for an unmarked page occurs, the phase ends.

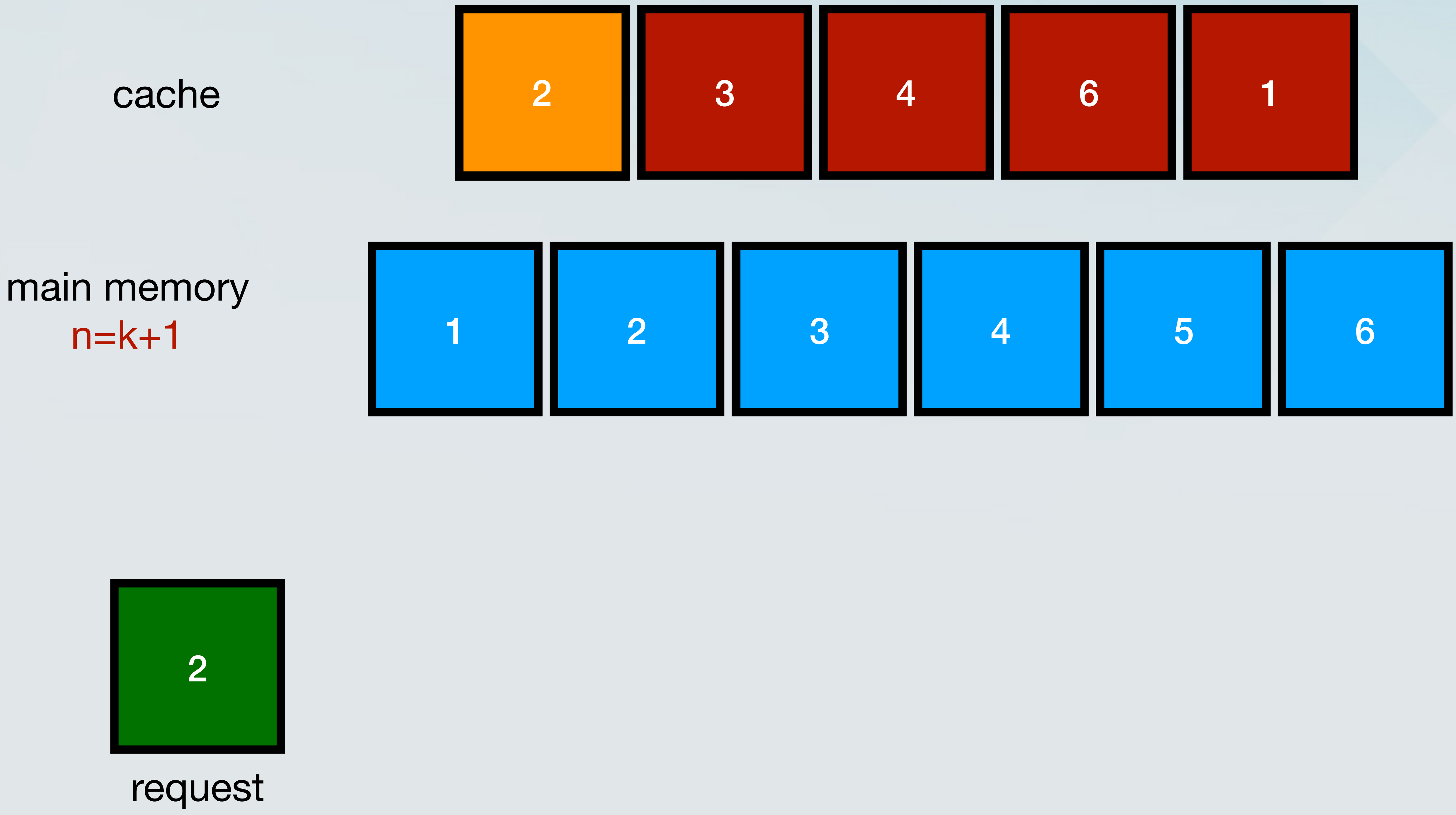
Marking algorithm



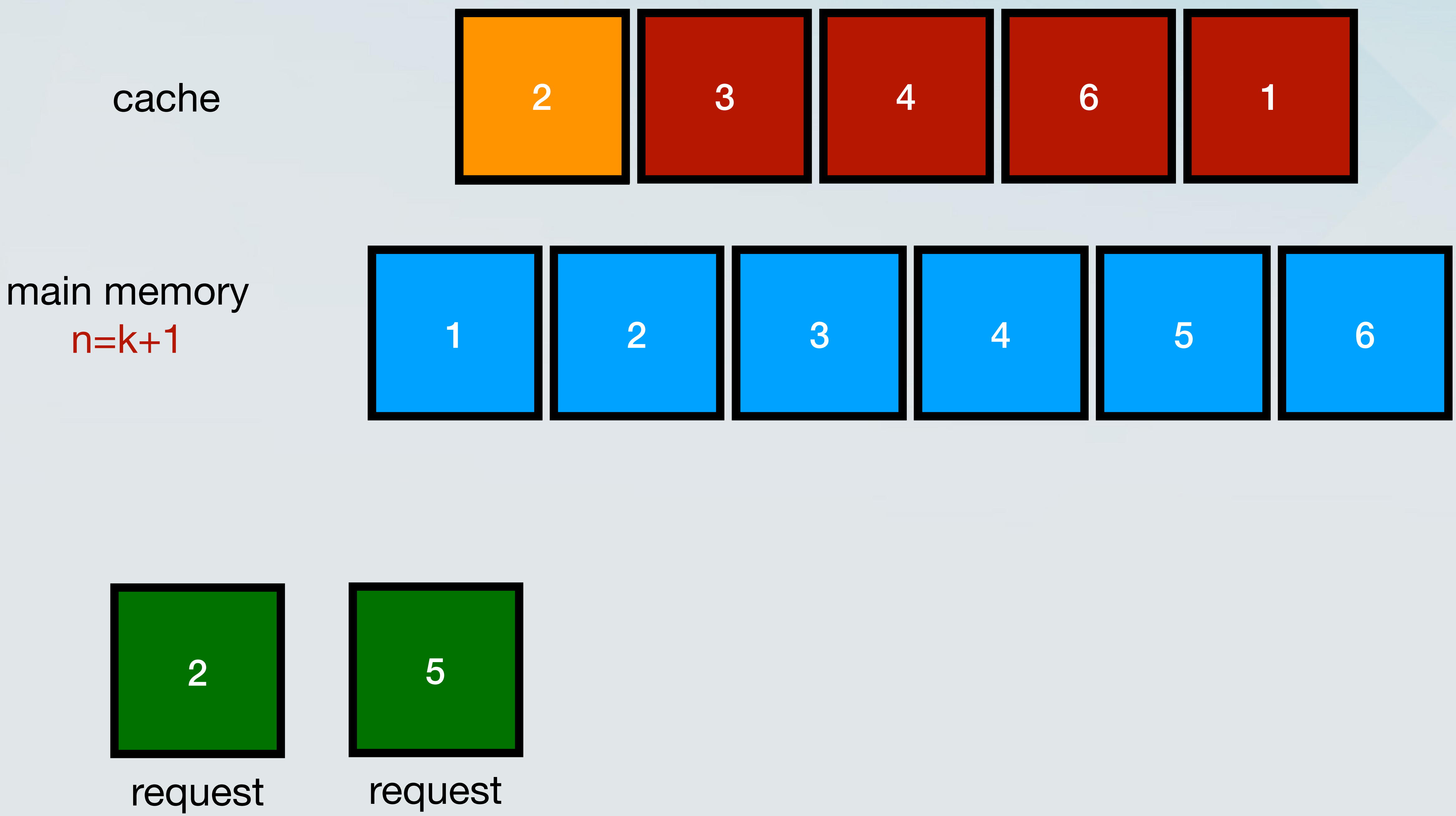
Marking algorithm



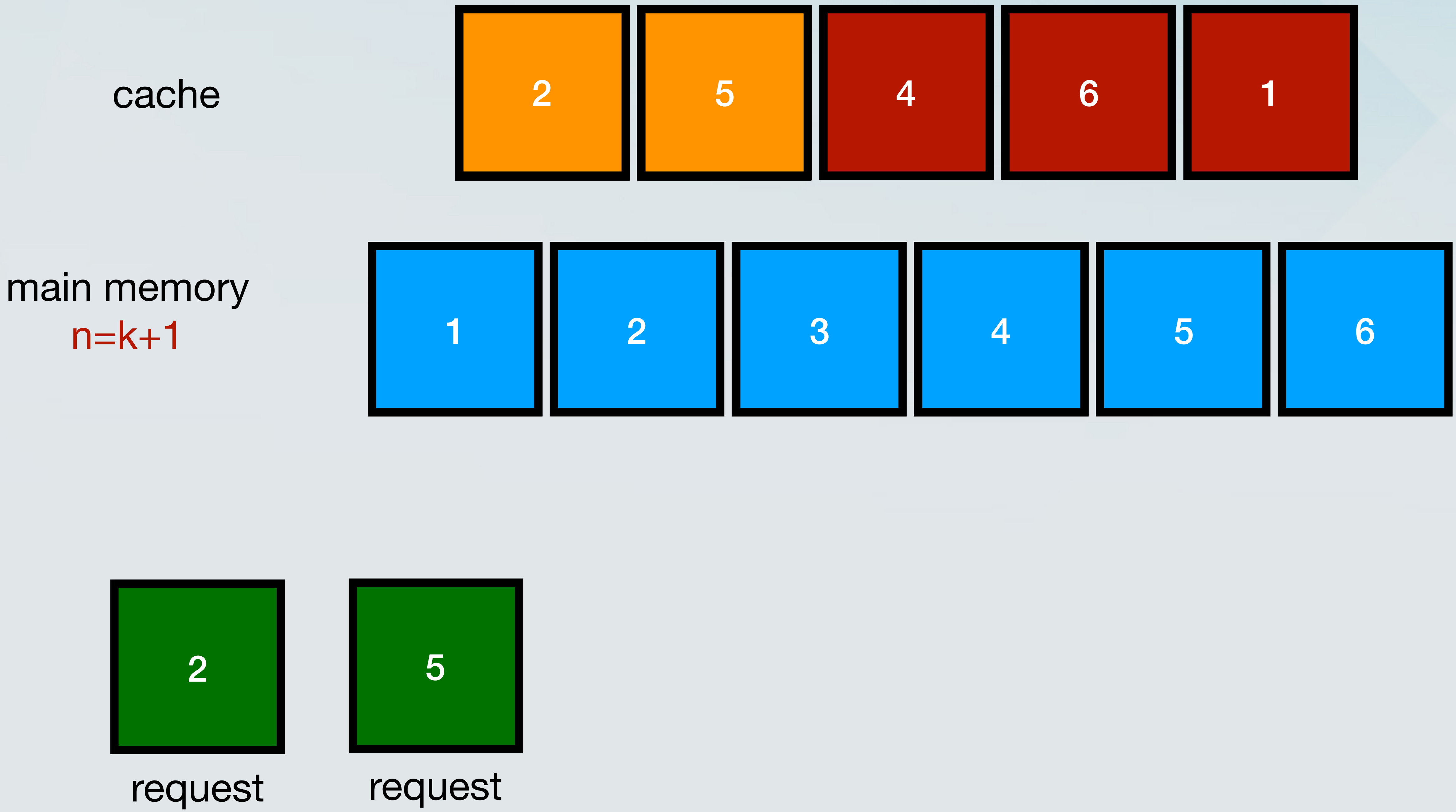
Marking algorithm



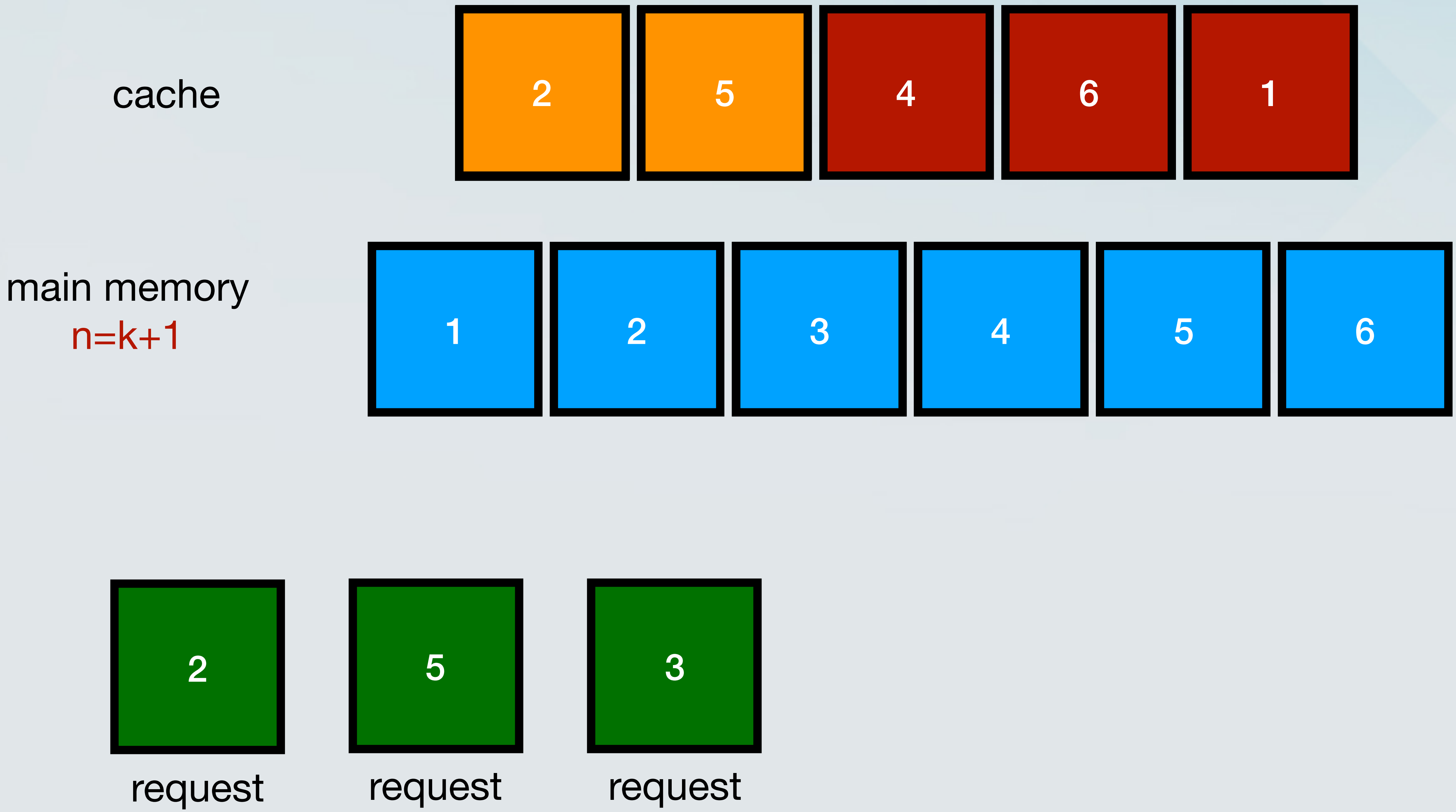
Marking algorithm



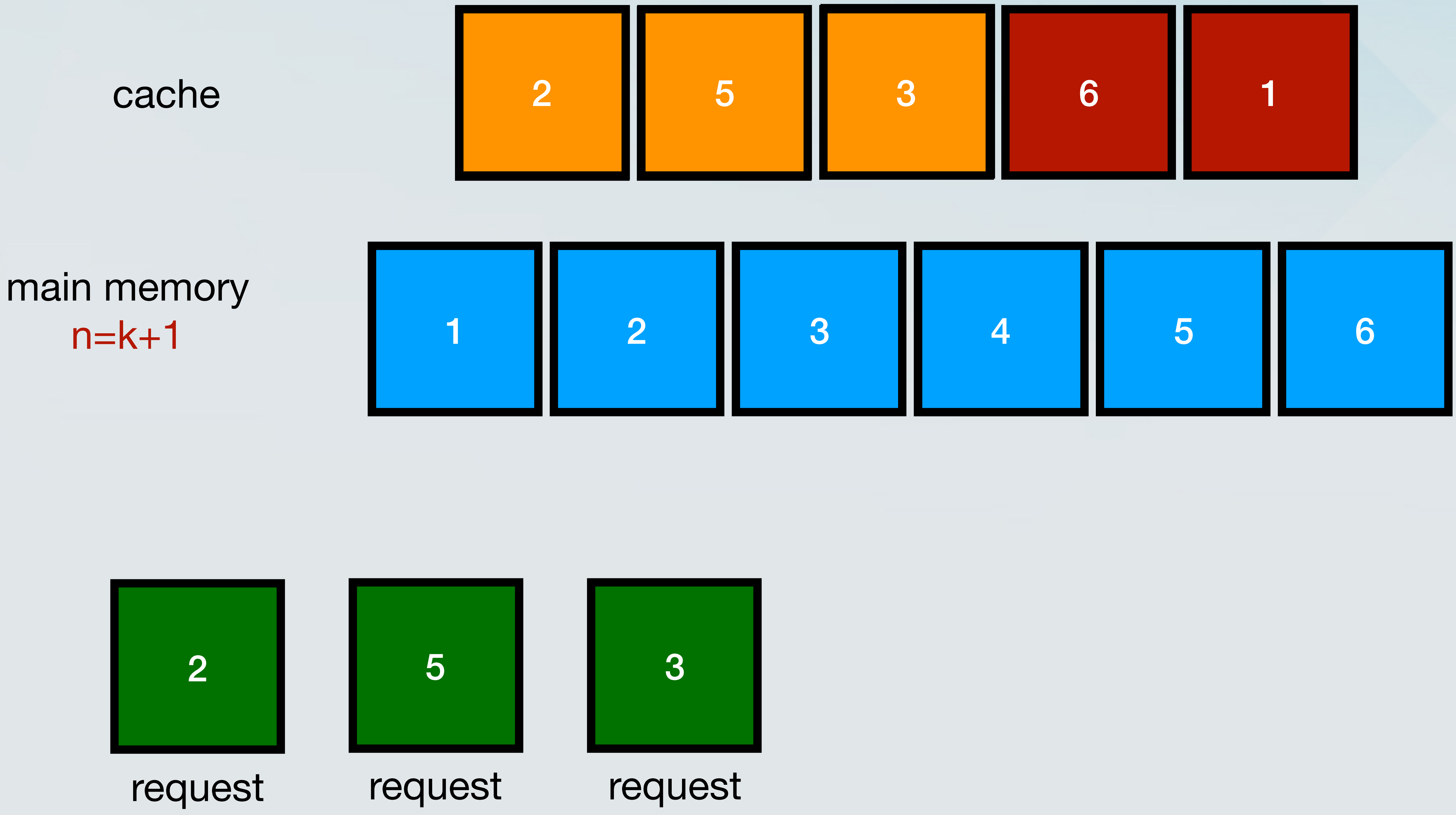
Marking algorithm



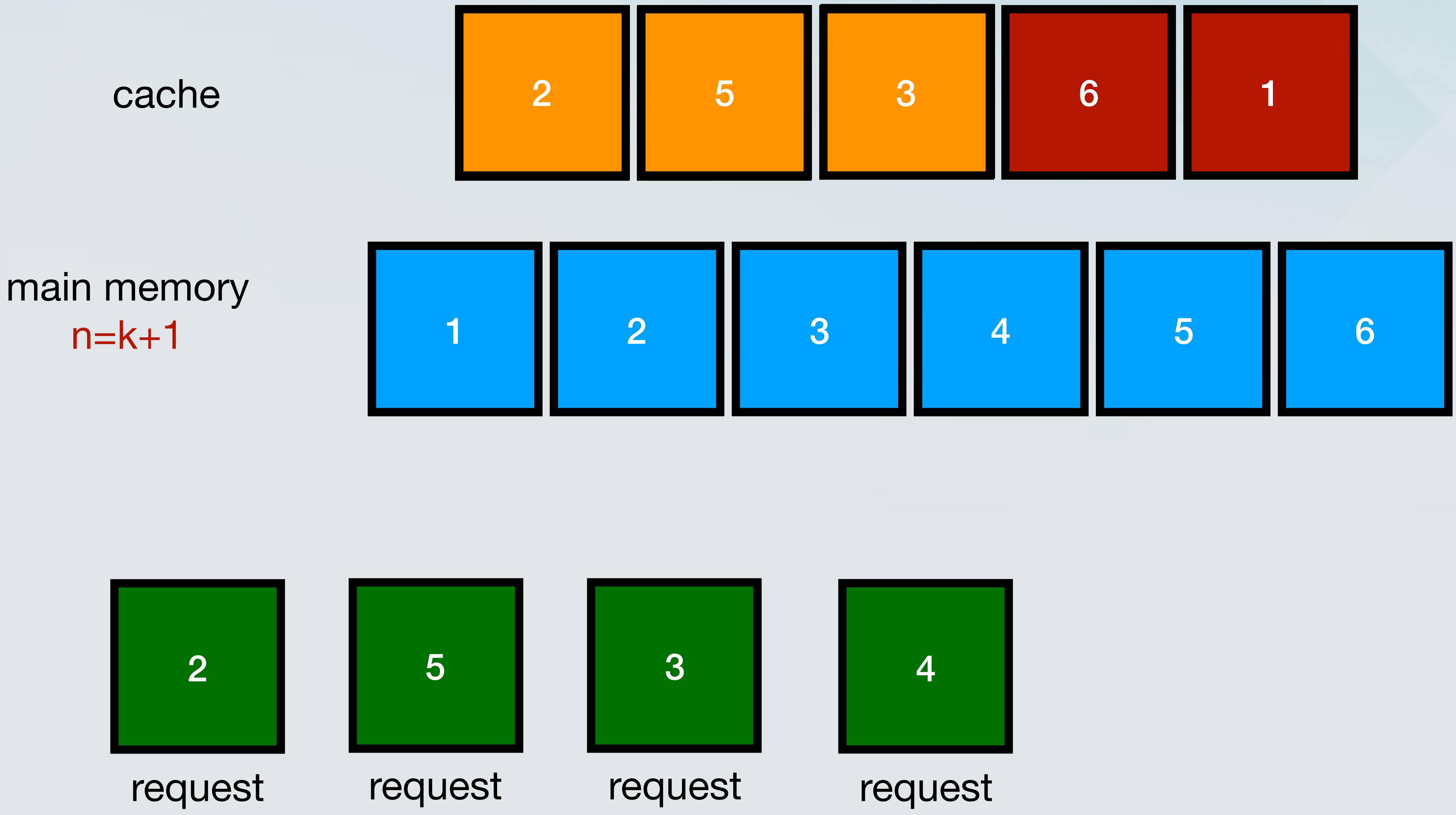
Marking algorithm



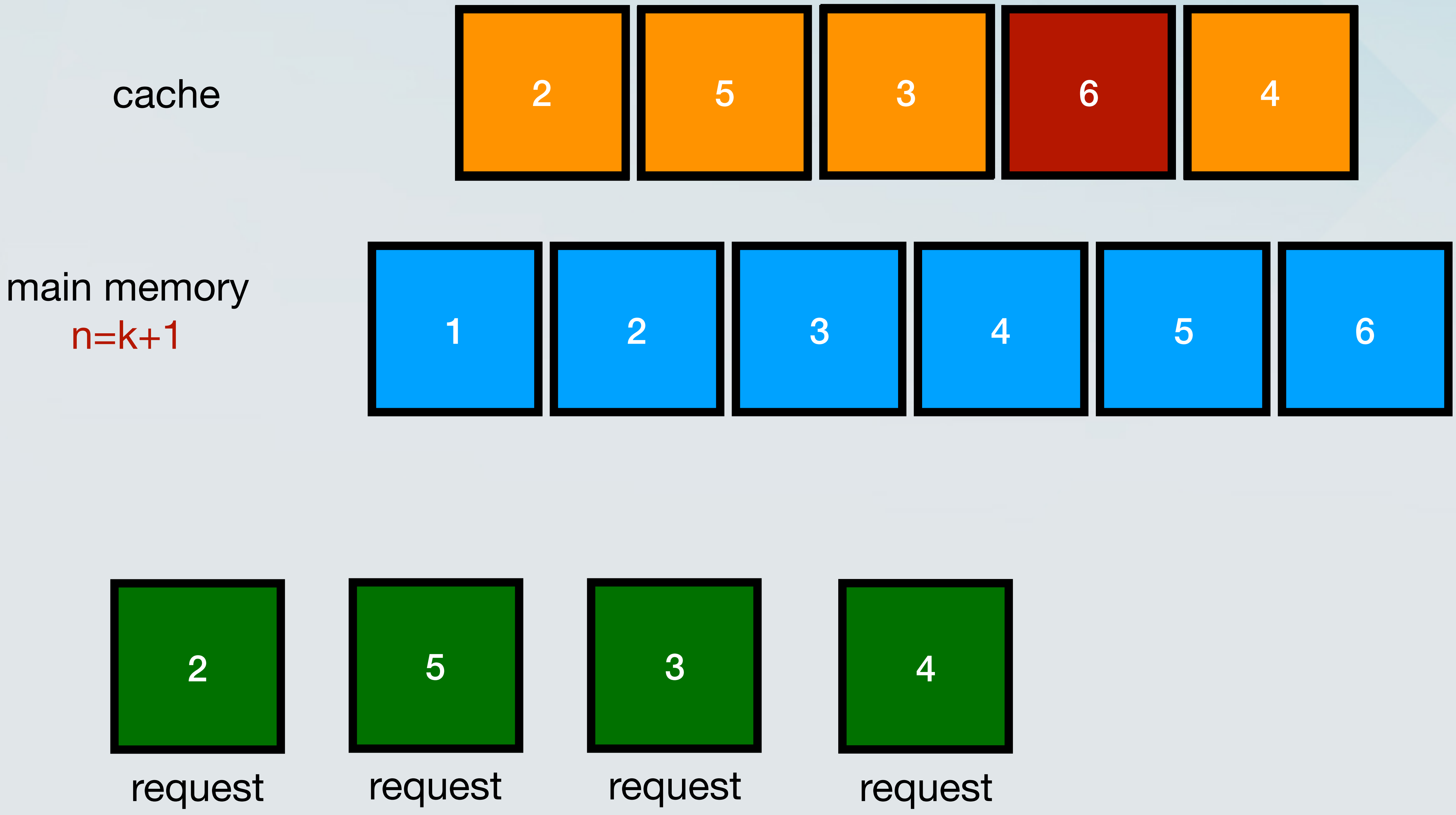
Marking algorithm



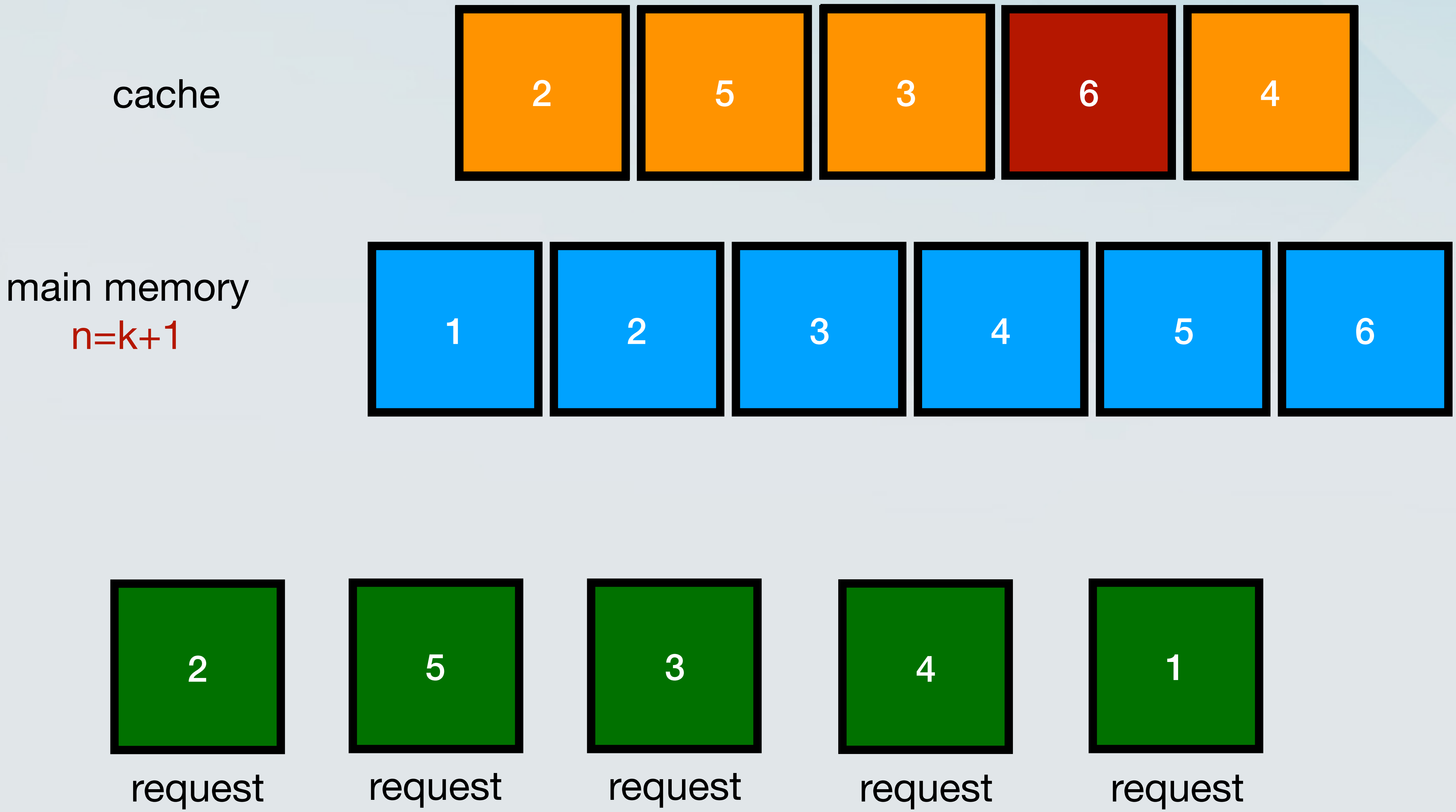
Marking algorithm



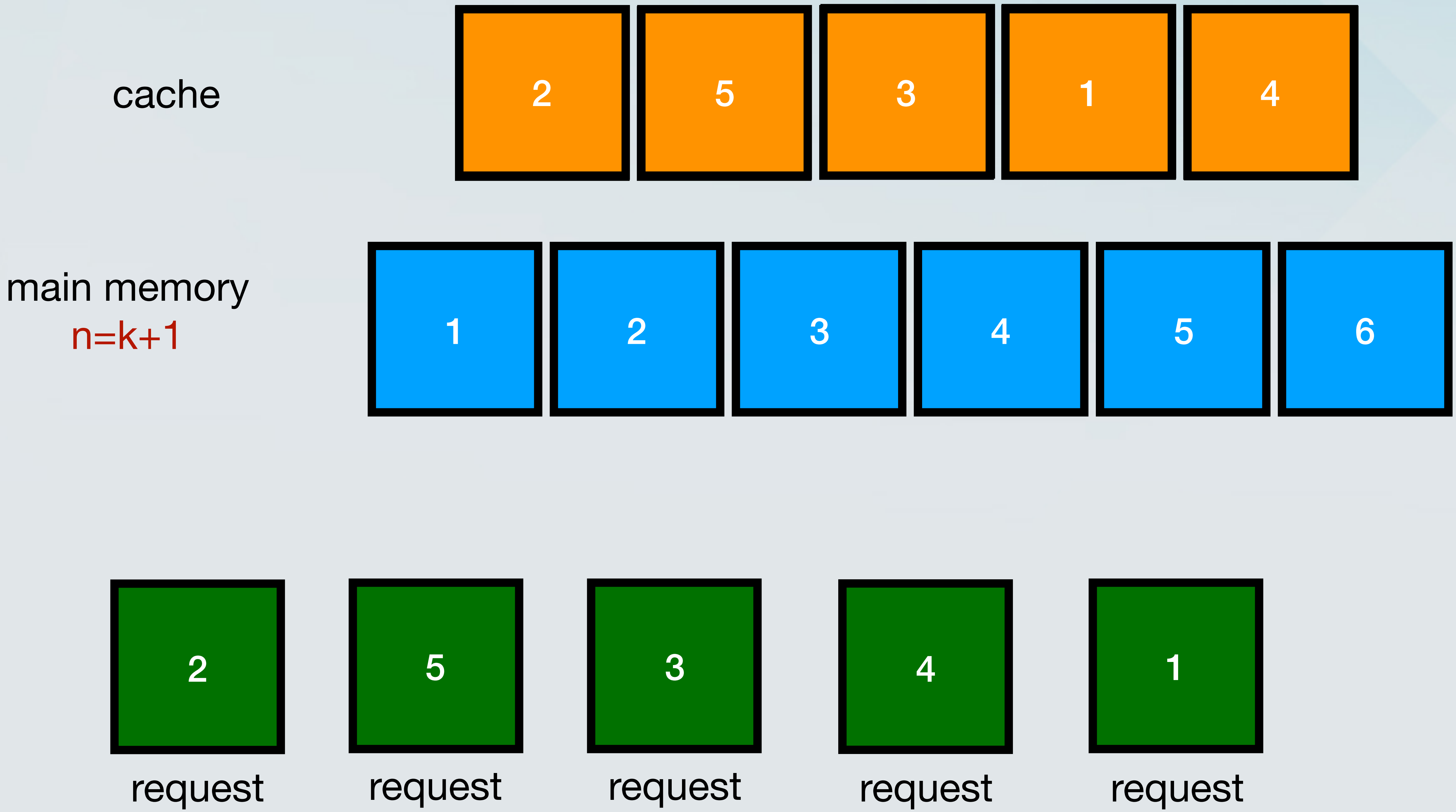
Marking algorithm



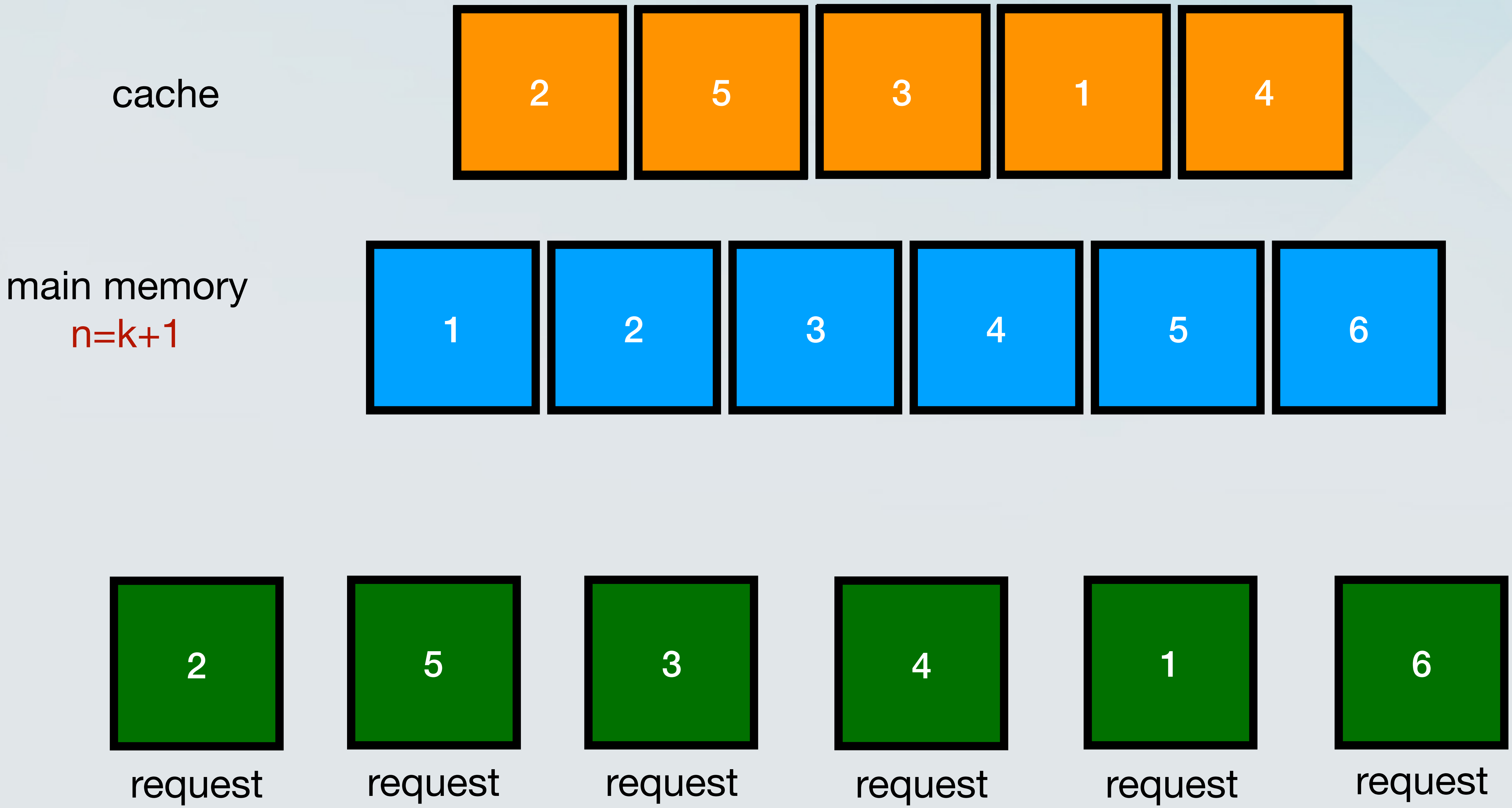
Marking algorithm



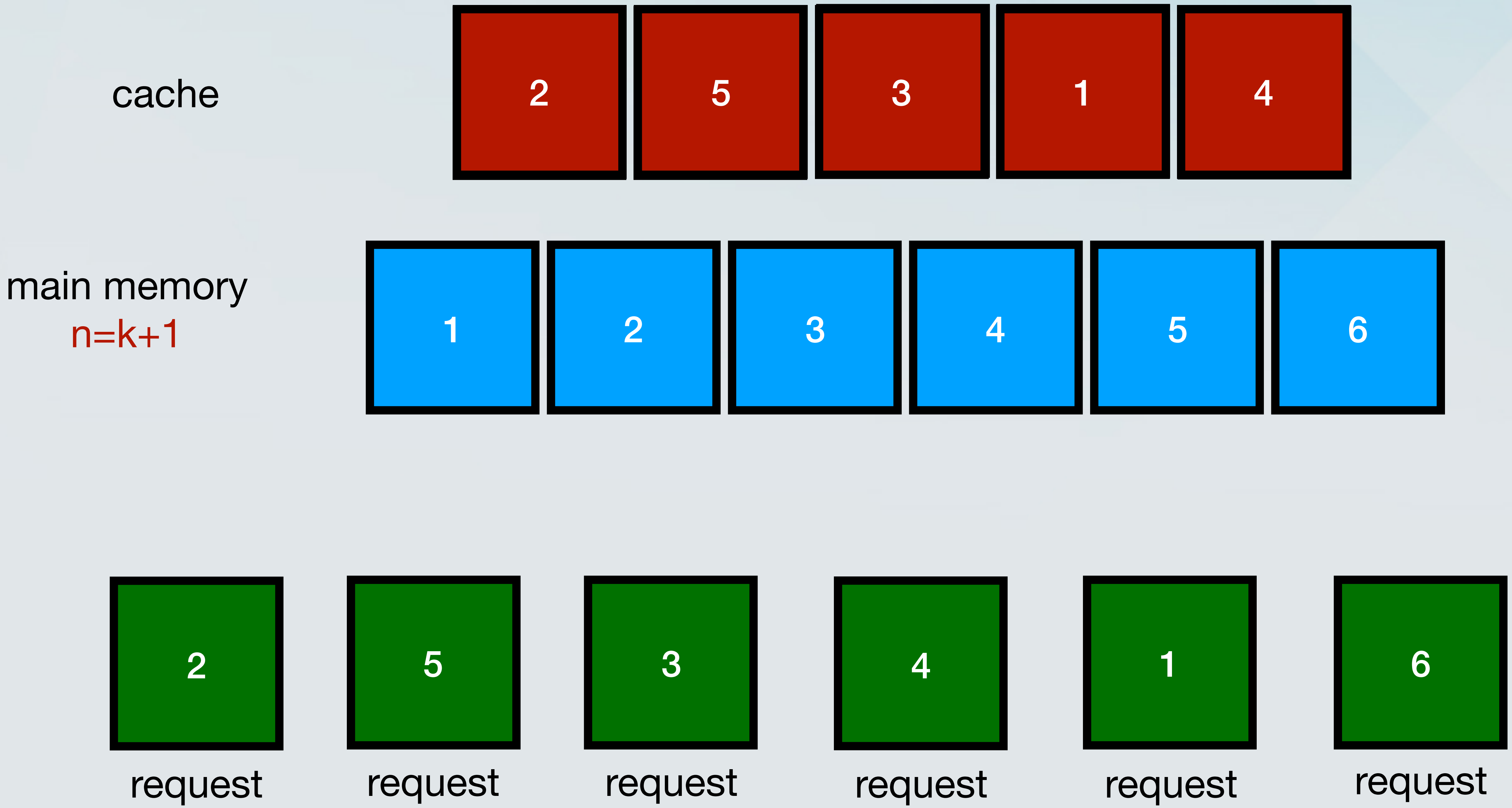
Marking algorithm



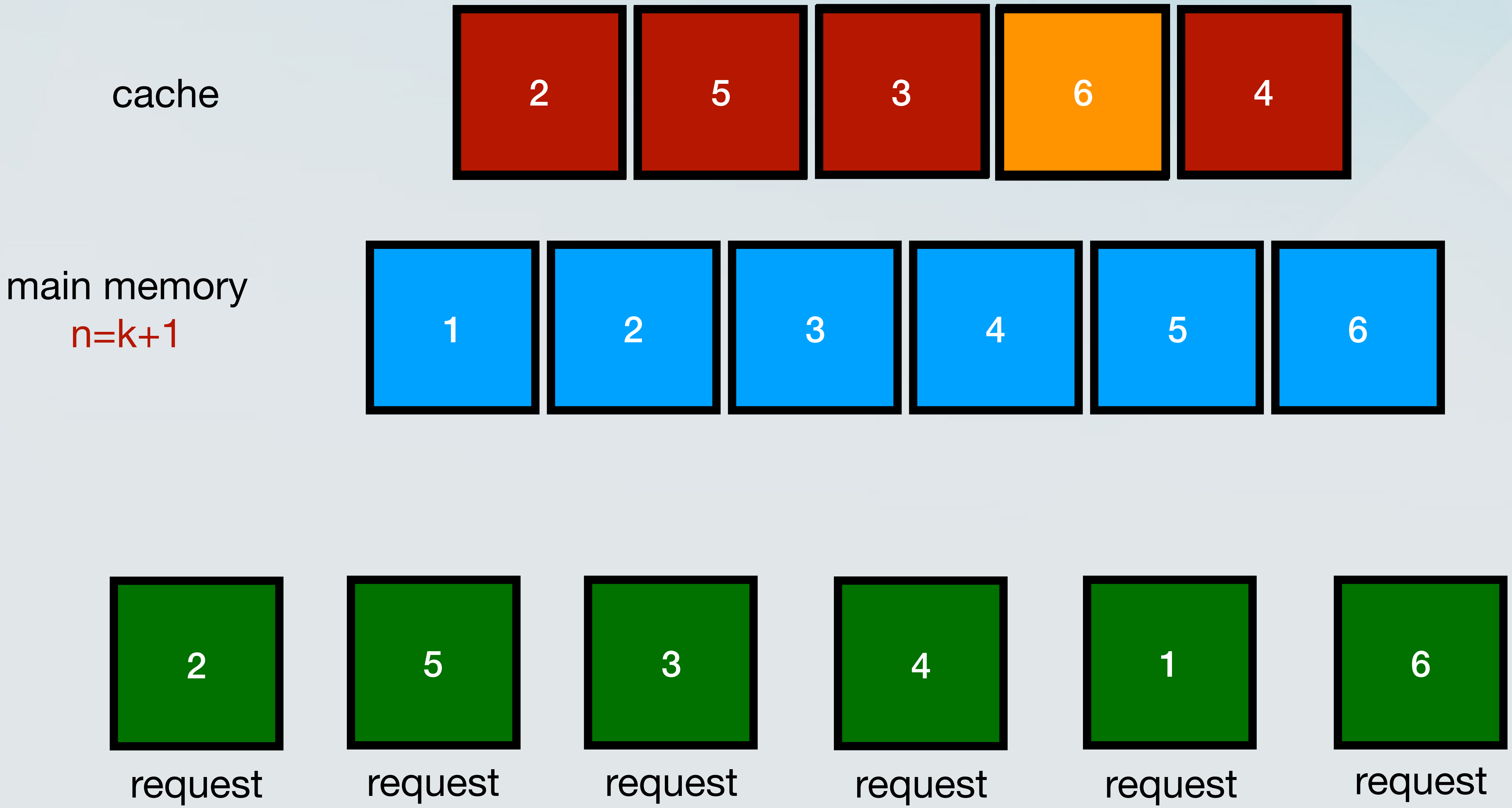
Marking algorithm



Marking algorithm



Marking algorithm



Paging Algorithms

- Theorem: The marking algorithm has competitive ratio k.

Paging Algorithms

- Theorem: The marking algorithm has competitive ratio k.
  - The algorithm "*faults*" at most k times in every phase.

Paging Algorithms

- Theorem: The marking algorithm has competitive ratio k.
  - The algorithm "*faults*" at most k times in every phase.
    - Every time it fails, the requested page is marked.

Paging Algorithms

- Theorem: The marking algorithm has competitive ratio k.
  - The algorithm "*faults*" at most k times in every phase.
    - Every time it fails, the requested page is marked.
    - If all pages in the cache are marked and a new page is requested, then the phase changes.

Paging Algorithms

- Theorem: The marking algorithm has competitive ratio k.
  - The algorithm "*faults*" at most k times in every phase.
    - Every time it fails, the requested page is marked.
    - If all pages in the cache are marked and a new page is requested, then the phase changes.
  - The optimal offline algorithm "*faults*" at least once in every phase.

Paging Algorithms

- Theorem: The marking algorithm has competitive ratio k.
  - The algorithm "*faults*" at most k times in every phase.
    - Every time it fails, the requested page is marked.
    - If all pages in the cache are marked and a new page is requested, then the phase changes.
  - The optimal offline algorithm "*faults*" at least once in every phase.
    - The phase ends when k+1 different pages are requested.

Paging Algorithms

- Theorem: The marking algorithm has competitive ratio k.
  - The algorithm "*faults*" at most k times in every phase.
    - Every time it fails, the requested page is marked.
    - If all pages in the cache are marked and a new page is requested, then the phase changes.
  - The optimal offline algorithm "*faults*" at least once in every phase.
    - The phase ends when k+1 different pages are requested.
    - The optimal offline algorithm can only keep at most k of those in the cache.

Paging Algorithms

- Theorem: LRU and FIFO have competitive ratio k.

Paging Algorithms

- Theorem: LRU and FIFO have competitive ratio k.
- Proof: LRU and FIFO are marking algorithms.

Paging Algorithms

- Theorem: LRU and FIFO have competitive ratio k.
- Proof: LRU and FIFO are marking algorithms.
- Corollary: LRU and FIFO are the best online algorithms for the paging problem.

Randomisation

Randomisation

- We will use randomisation to achieve a better competitive ratio.

Randomisation

- We will use randomisation to achieve a better competitive ratio.
- We have to make a distinction, when it come to the power of the adversary:

Randomisation

- We will use randomisation to achieve a better competitive ratio.

- We have to make a distinction, when it come to the power of the adversary:

    - Oblivious Adversary: The adversary fixes an input sequence in advance.

Randomisation

- We will use randomisation to achieve a better competitive ratio.
- We have to make a distinction, when it come to the power of the adversary:
  - Oblivious Adversary: The adversary fixes an input sequence in advance.
  - Adaptive Adversary: The adversary can change the input sequence based on the realisations of randomness of the choices of the algorithm.

Marking algorithm

- Consider the following algorithm:
  - The algorithm proceeds in *phases*.
  - At the beginning of a phase, all the pages are unmarked.
  - Whenever a page is requested, it is marked.
  - When a “*fault*” occurs, the algorithm replaces an unmarked page.
  - When all pages in the cache are marked, and a request for an unmarked page occurs, the phase ends.

Randomised Marking algorithm

- Consider the following algorithm:
  - The algorithm proceeds in *phases*.
  - At the beginning of a phase, all the pages are unmarked.
  - Whenever a page is requested, it is marked.
  - When a "*fault*" occurs, the algorithm replaces an unmarked page, *selecting one uniformly at random*.
  - When all pages in the cache are marked, and a request for an unmarked page occurs, the phase ends.

Randomised Marking algorithm

- Theorem: The Randomised Marking algorithm has competitive ratio $2H_k$ against *oblivious adversaries*.

The proof

The proof

- Assume without loss of generality that RMA makes a “*fault*” on the first request.

The proof

- Assume without loss of generality that RMA makes a “*fault*” on the first request.

- Consider phase i,

The proof

- Assume without loss of generality that RMA makes a “*fault*” on the first request.

- Consider phase i,

  - let m_i be the number of “*new*” pages in the phase, i.e., pages which were not requested in phase i-1.

The proof

- Assume without loss of generality that RMA makes a "*fault*" on the first request.

- Consider phase i,

  - let m_i be the number of "*new*" pages in the phase, i.e., pages which were not requested in phase i-1.

  - call the remaining $k-m_i$ pages "*old*".

The proof

- Assume without loss of generality that RMA makes a "*fault*" on the first request.
- Consider phase i,
  - let m_i be the number of "*new*" pages in the phase, i.e., pages which were not requested in phase i-1.
  - call the remaining k-m_i pages "*old*".
- Every time a "*new*" page is requested, we have a "*fault*".

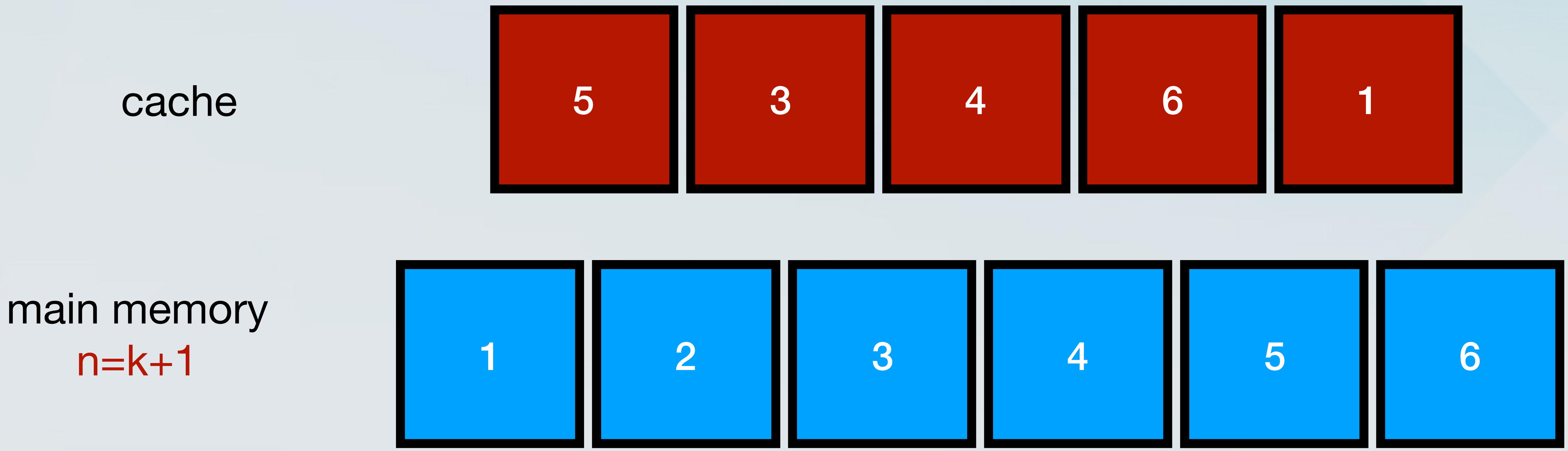
The proof

- Assume without loss of generality that RMA makes a "*fault*" on the first request.
- Consider phase i,
  - let m_i be the number of "*new*" pages in the phase, i.e., pages which were not requested in phase i-1.
  - call the remaining $k-m_i$ pages "*old*".
- Every time a "*new*" page is requested, we have a "*fault*".
- Every time an "*old*" page is requested, we may have "*fault*".

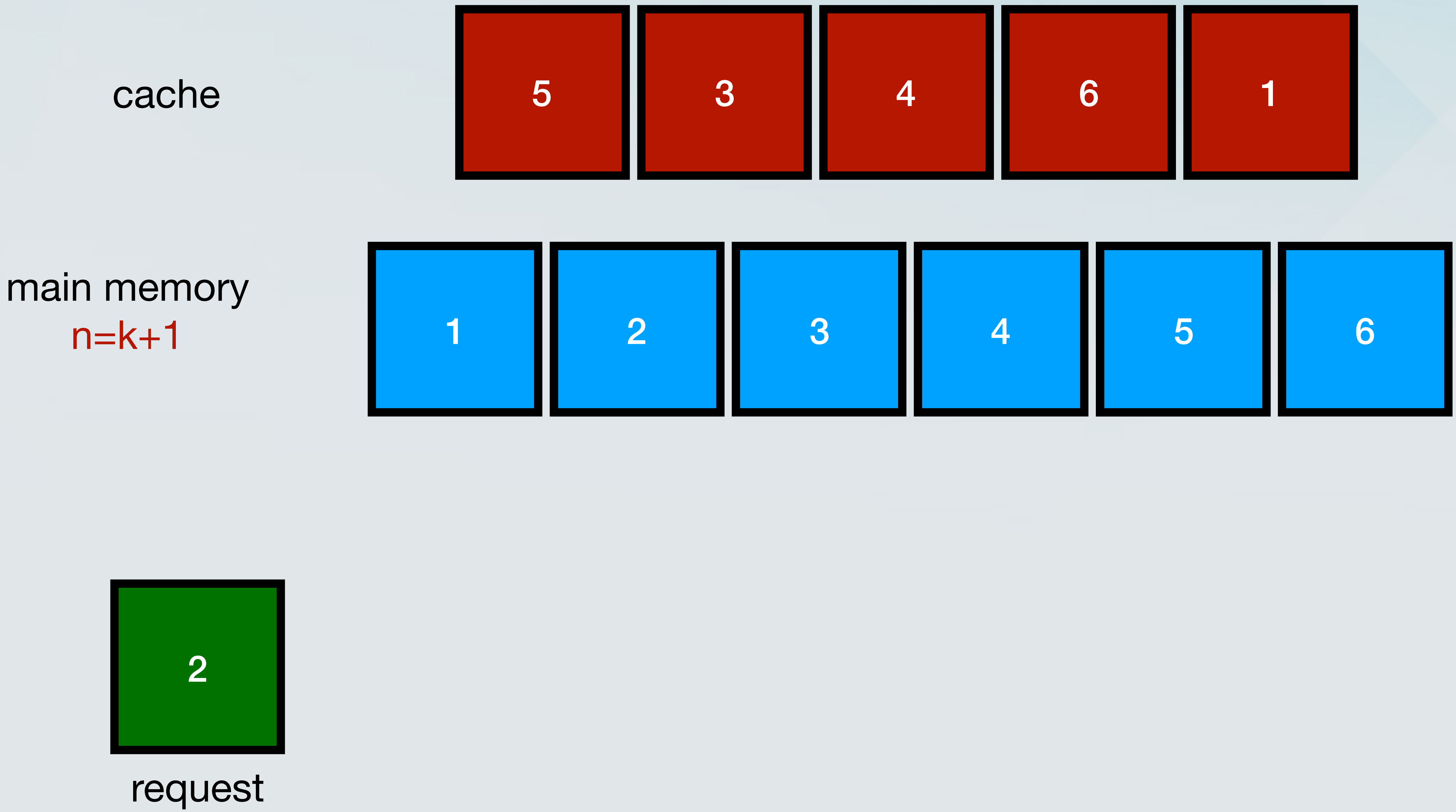
The proof

- Assume without loss of generality that RMA makes a "*fault*" on the first request.
- Consider phase i,
  - let m_i be the number of "*new*" pages in the phase, i.e., pages which were not requested in phase i-1.
  - call the remaining $k-m_i$ pages "*old*".
- Every time a "*new*" page is requested, we have a "*fault*".
- Every time an "*old*" page is requested, we may have "*fault*".
  - The "*fault*" happens if we replaced the "*old*" page with a "*new*" one.

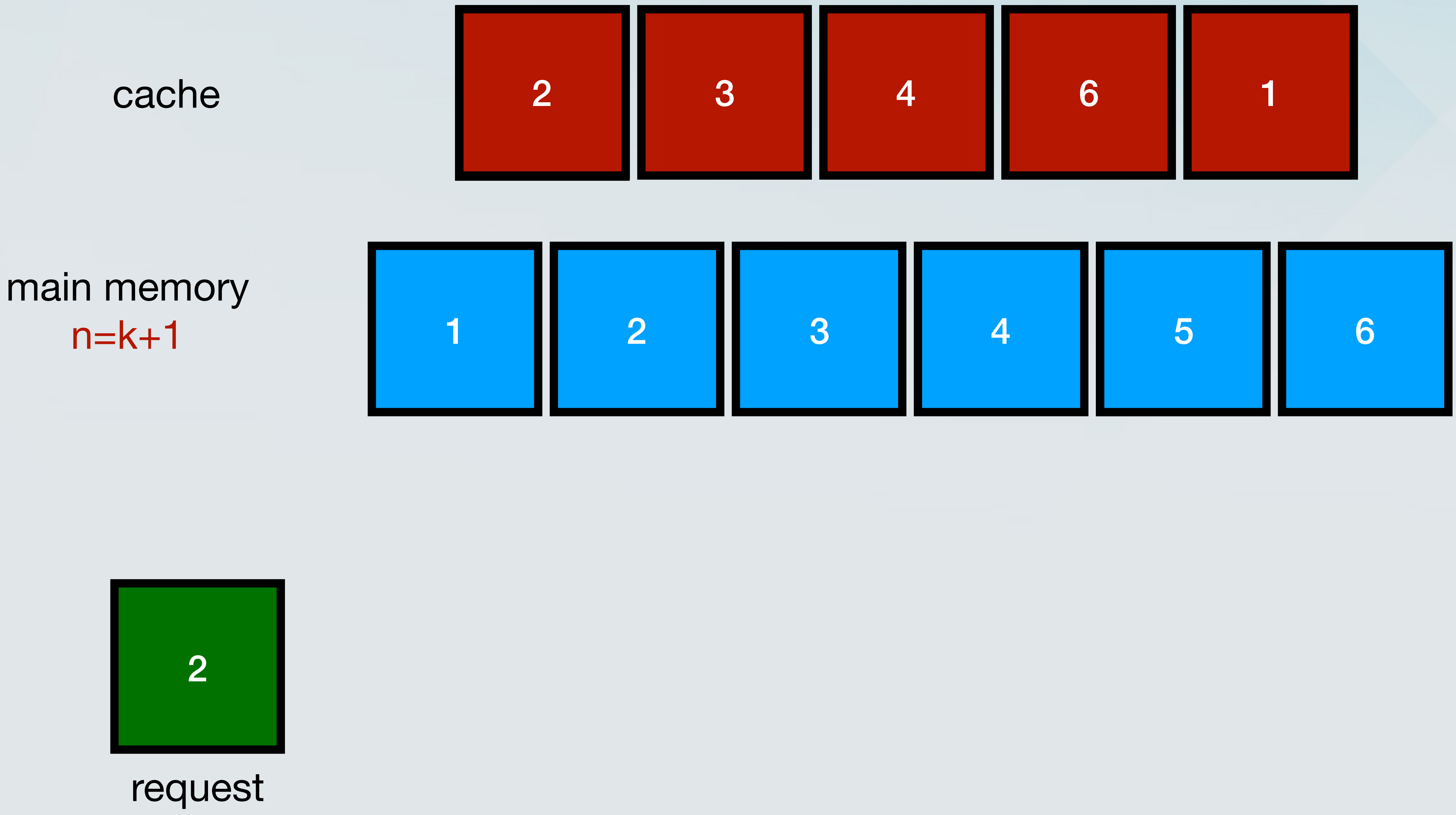
Randomised Marking



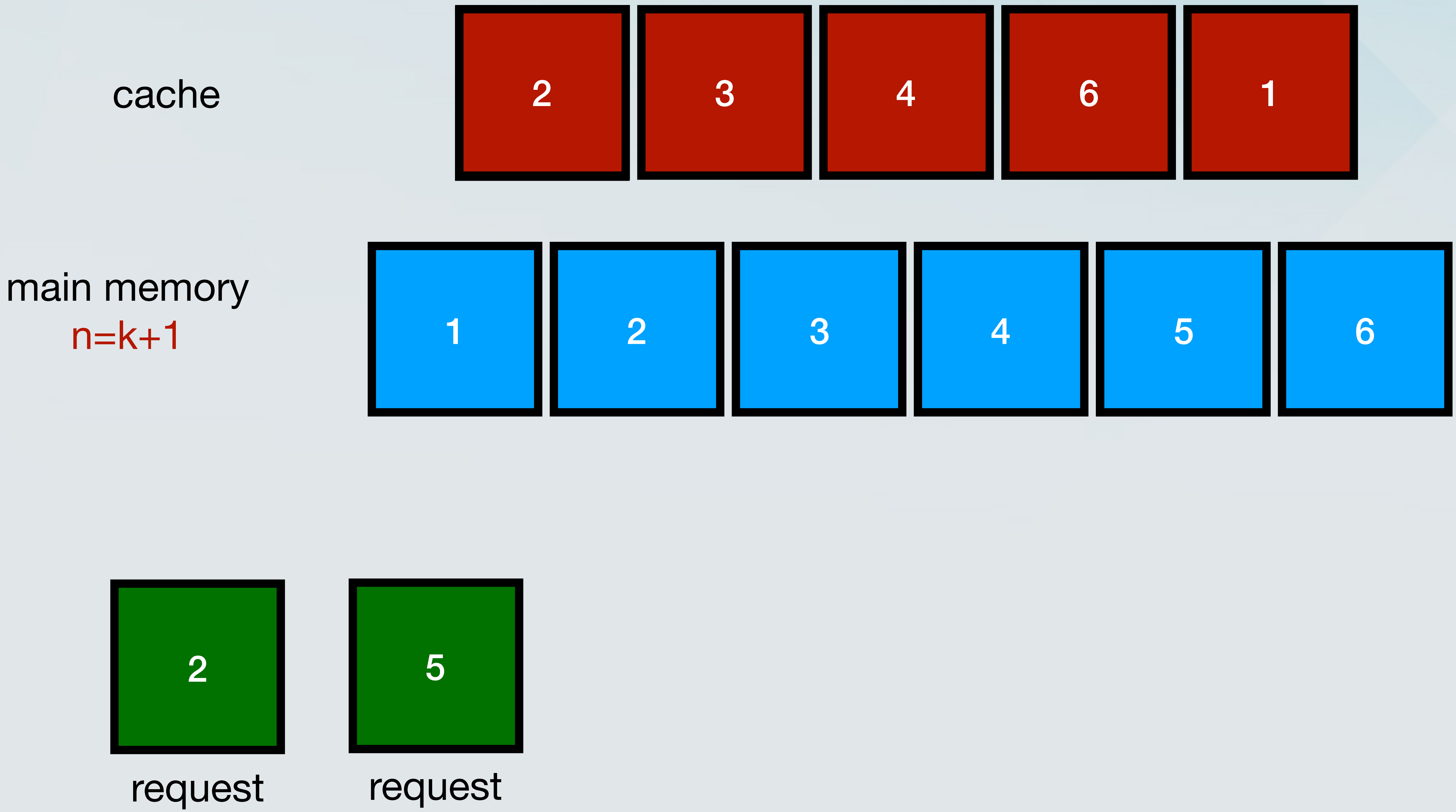
Randomised Marking



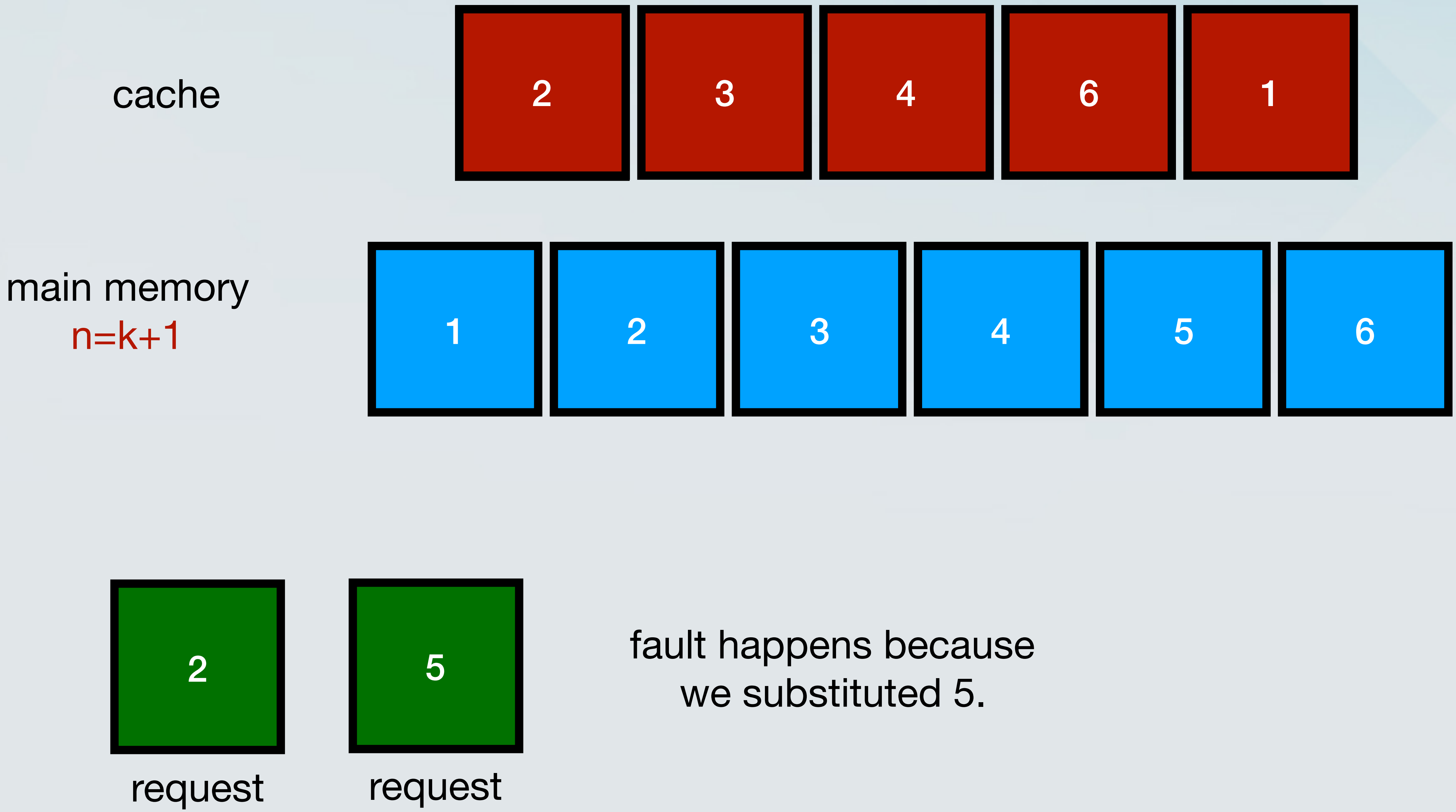
Randomised Marking



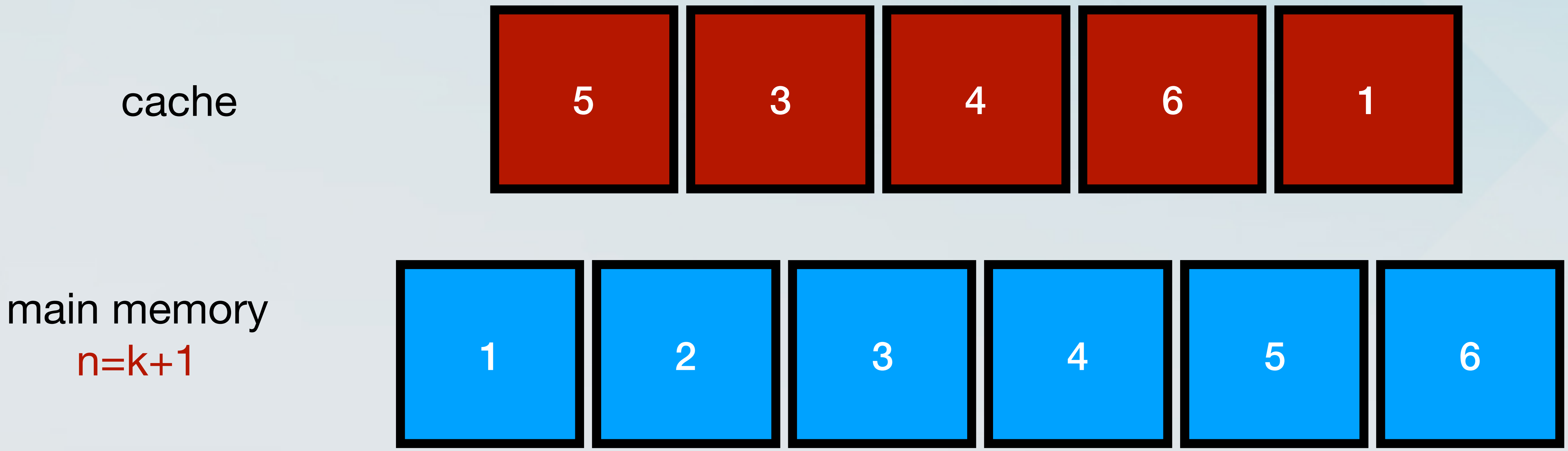
Randomised Marking



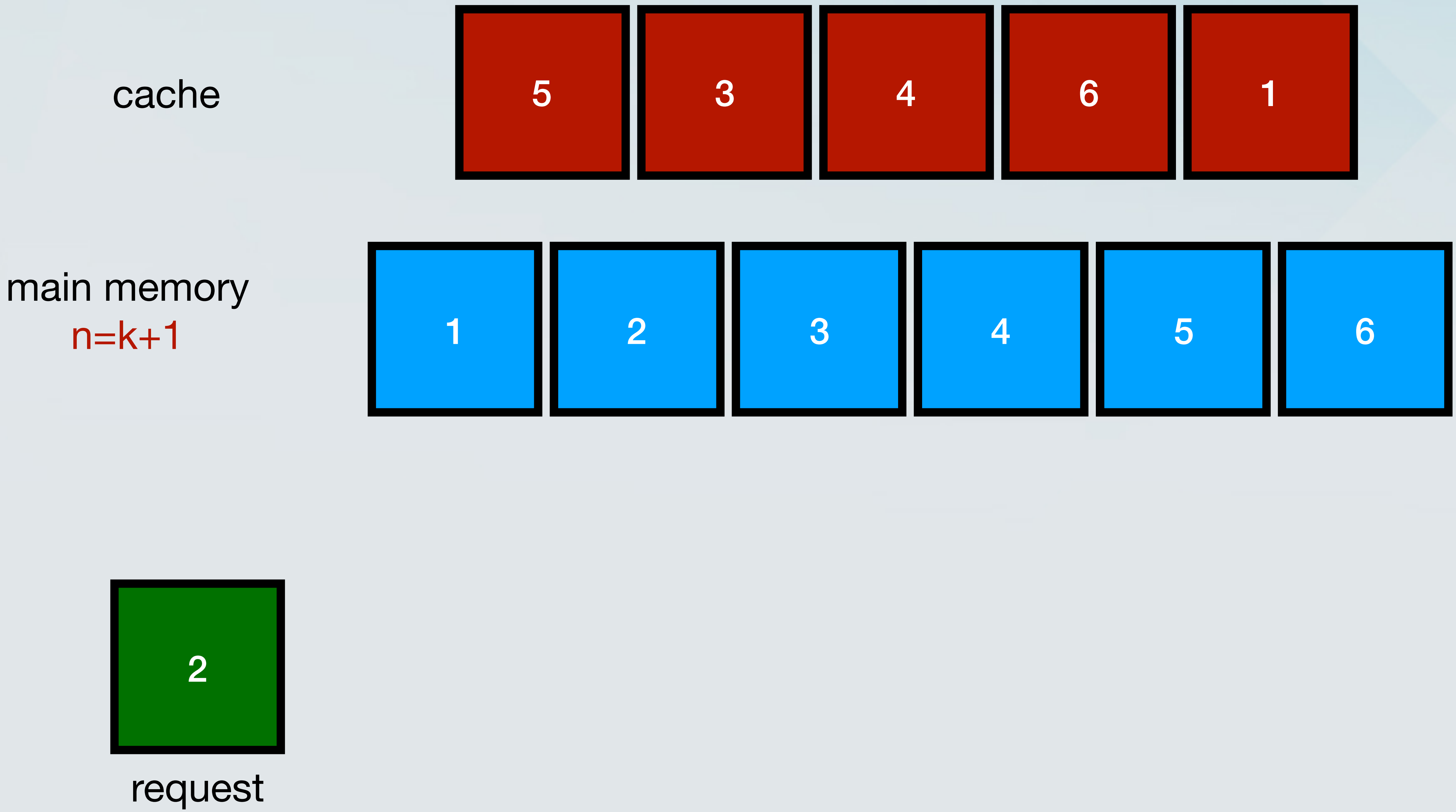
Randomised Marking



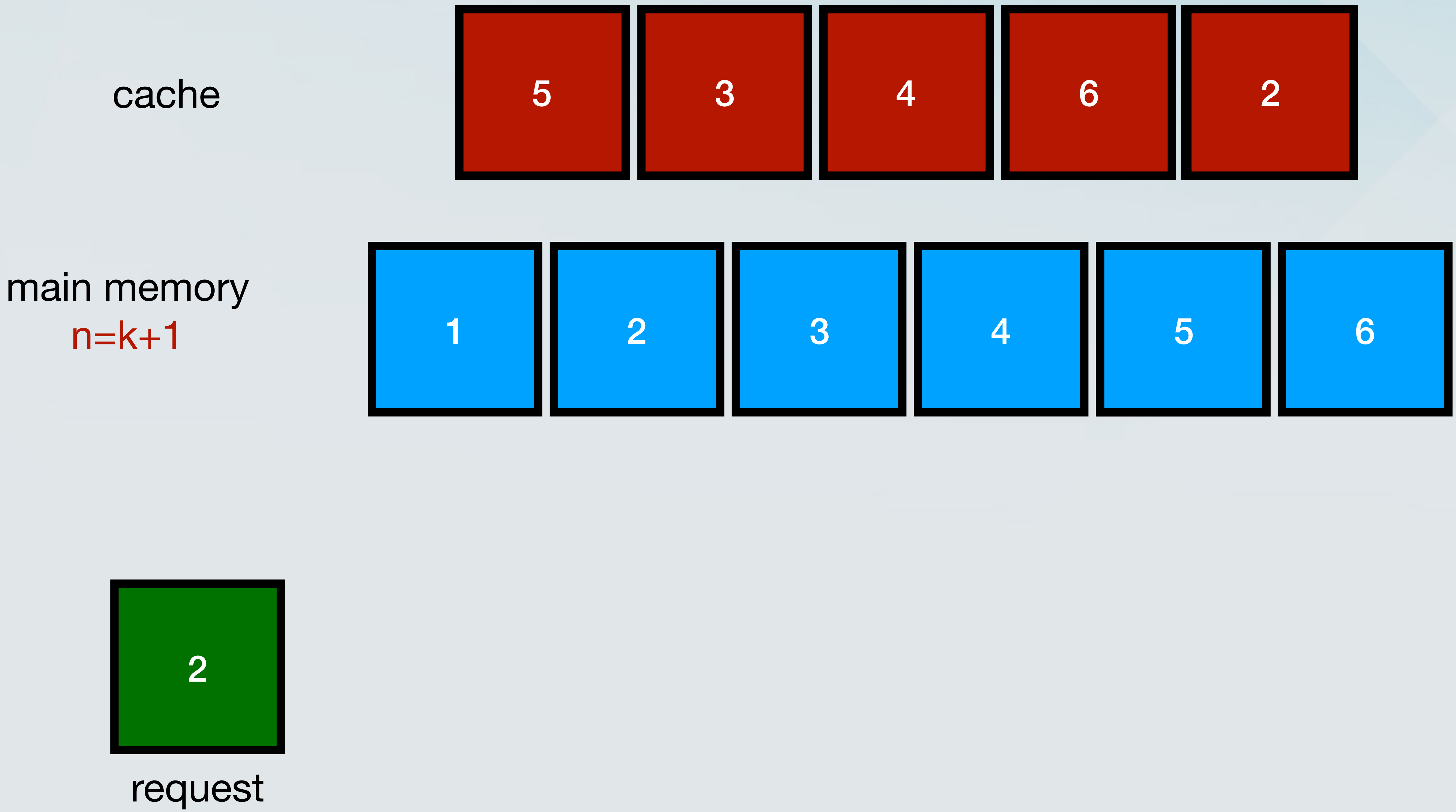
Randomised Marking



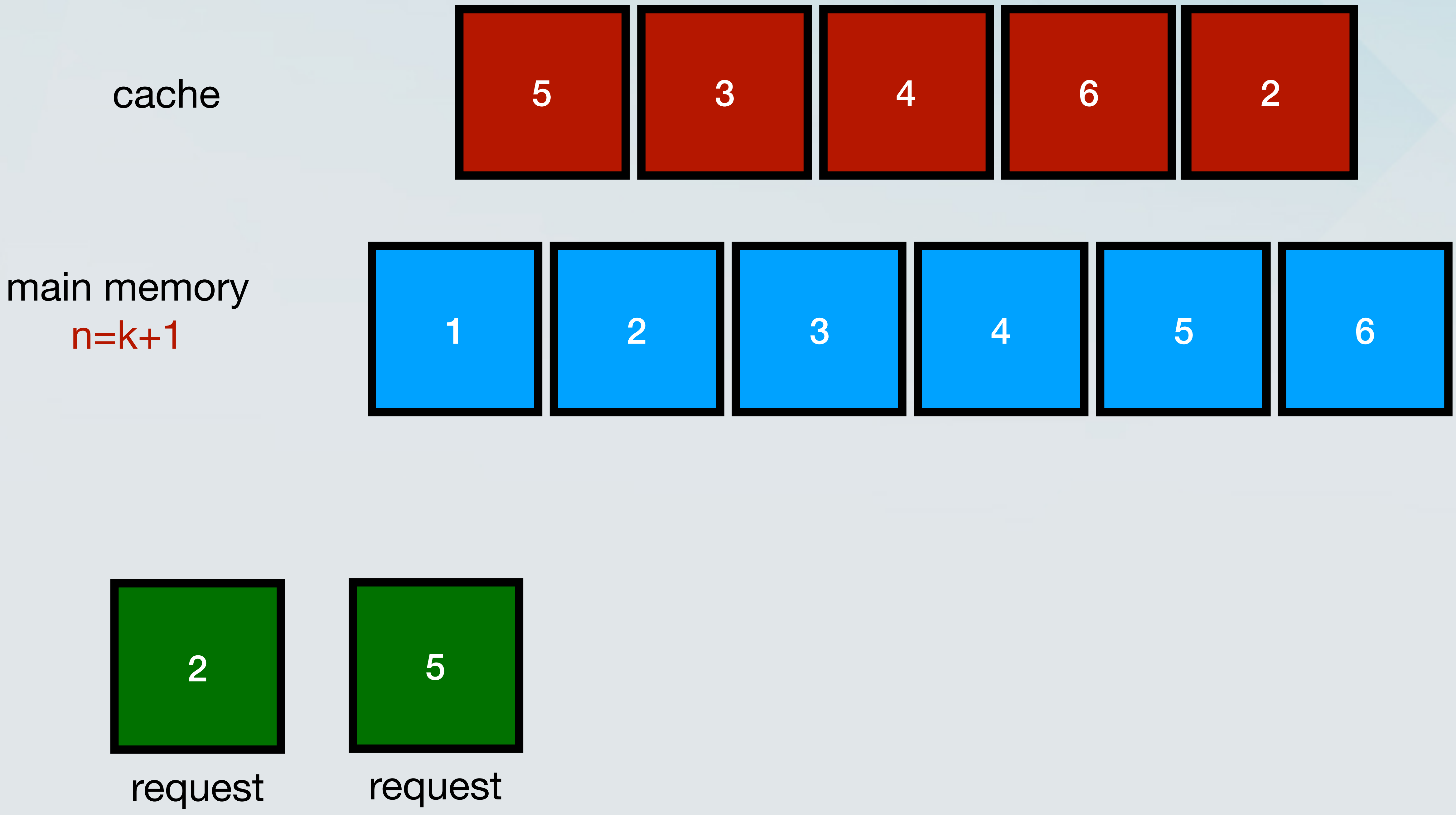
Randomised Marking



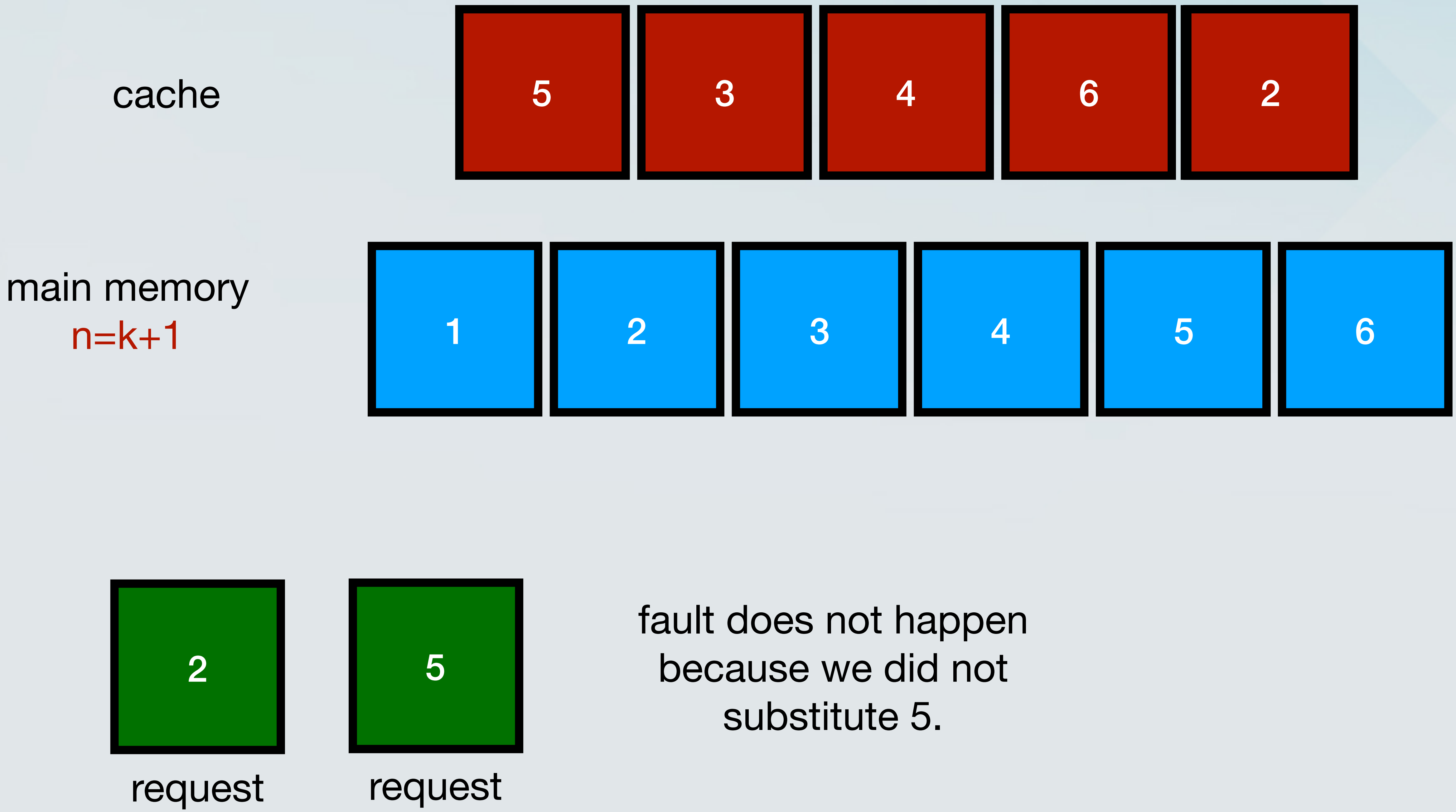
Randomised Marking



Randomised Marking



Randomised Marking



fault does not happen because we did not substitute 5.

The proof

- Assume without loss of generality that RMA makes a “*fault*” on the first request.
- Consider phase i,
  - let m_i be the number of “*new*” pages in the phase, i.e., pages which were not requested in phase i-1.
  - call the remaining $k-m_i$ pages “*old*”.
- Every time a “*new*” page is requested, we have a “*fault*”.
- Every time an “*old*” page is requested, we may have “*fault*”.
  - The “*fault*” happens if we replaced the “*old*” page with a “*new*” one.

The proof

- Assume (*wlog*) that the m_i requests for "*new*" pages come first and the $k-m_i$ requests for "*old*" pages follow.

The proof

- Assume (*wlog*) that the m_i requests for “*new*” pages come first and the $k\text{-}m_i$ requests for “*old*” pages follow.
  - What is the probability that on the first “*old*” page request, we make a “*fault*”?

The proof

- Assume (*wlog*) that the m_i requests for "*new*" pages come first and the $k-m_i$ requests for "*old*" pages follow.

  - What is the probability that on the first "*old*" page request, we make a "*fault*"?

  - It is the probability that the "*old*" page was replaced on the first request, or the second request or, …, or the the m_i*'th* request.

The proof

- Assume (*wlog*) that the m_i requests for "*new*" pages come first and the $k-m_i$ requests for "*old*" pages follow.
  - What is the probability that on the first "*old*" page request, we make a "*fault*"?
  - It is the probability that the "*old*" page was replaced on the first request, or the second request or, …, or the the m_i*'th* request.
  - This is m_i/k.

The proof

- The "*fault*" happens if we replaced the "*old*" page with a "*new*" one.
  - What is the probability of that happening?
- Assume (*wlog*) that the m_i requests for "*new*" pages come first and the $k-m_i$ requests for "*old*" pages follow.

The proof

- The "*fault*" happens if we replaced the "*old*" page with a "*new*" one.
  - What is the probability of that happening?
- Assume (*wlog*) that the m_i requests for "*new*" pages come first and the $k-m_i$ requests for "*old*" pages follow.
  - What is the probability that on the second "*old*" page request, we make a "*fault*", given that we made a "*fault*" on the first "*old*" page request?

The proof

- The "*fault*" happens if we replaced the "*old*" page with a "*new*" one.
  - What is the probability of that happening?
- Assume (*wlog*) that the m_i requests for "*new*" pages come first and the $k-m_i$ requests for "*old*" pages follow.
  - What is the probability that on the second "*old*" page request, we make a "*fault*", given that we made a "*fault*" on the first "*old*" page request?
  - This is $m_i/(k-1)$.

The proof

- The expected number of "*faults*" of our algorithm in phase i is

  $m_i + m_i/k + m_i/(k-1) + \ldots + m_i/(k-(k-m_i)+1 \leq m_i H_k$

- Summing up over all the phases, we have that:

$$\mathbb{E}[\textbf{cost of RMA}] \leq H_k \sum_{i=1}^{n} m_i$$

Arguing about the OPT

- What is the number of "*faults*" that the optimal offline algorithm makes?
- Let's look at two *consecutive* phases i-1 and i.
- Let n_{i-1} and n_i be the number of "*faults*" of OPT on those phases.
- By the definition of a phase, there are m_i new pages in phases i-1 and i.
    - It holds that $n_{i-1} + n_i \leq m_i$
    - Summing up, we get: $OPT \geq \frac{1}{2}\sum_{i=1}^{n} m_i$

Combining

$$\mathbb{E}[\textbf{cost of RMA}] \leq H_k \sum_{i=1}^{n} m_i \qquad OPT \geq \frac{1}{2} \sum_{i=1}^{n} m_i$$

The competitive ratio of RMA is $2H_k$