Cache-Oblivious Hashing

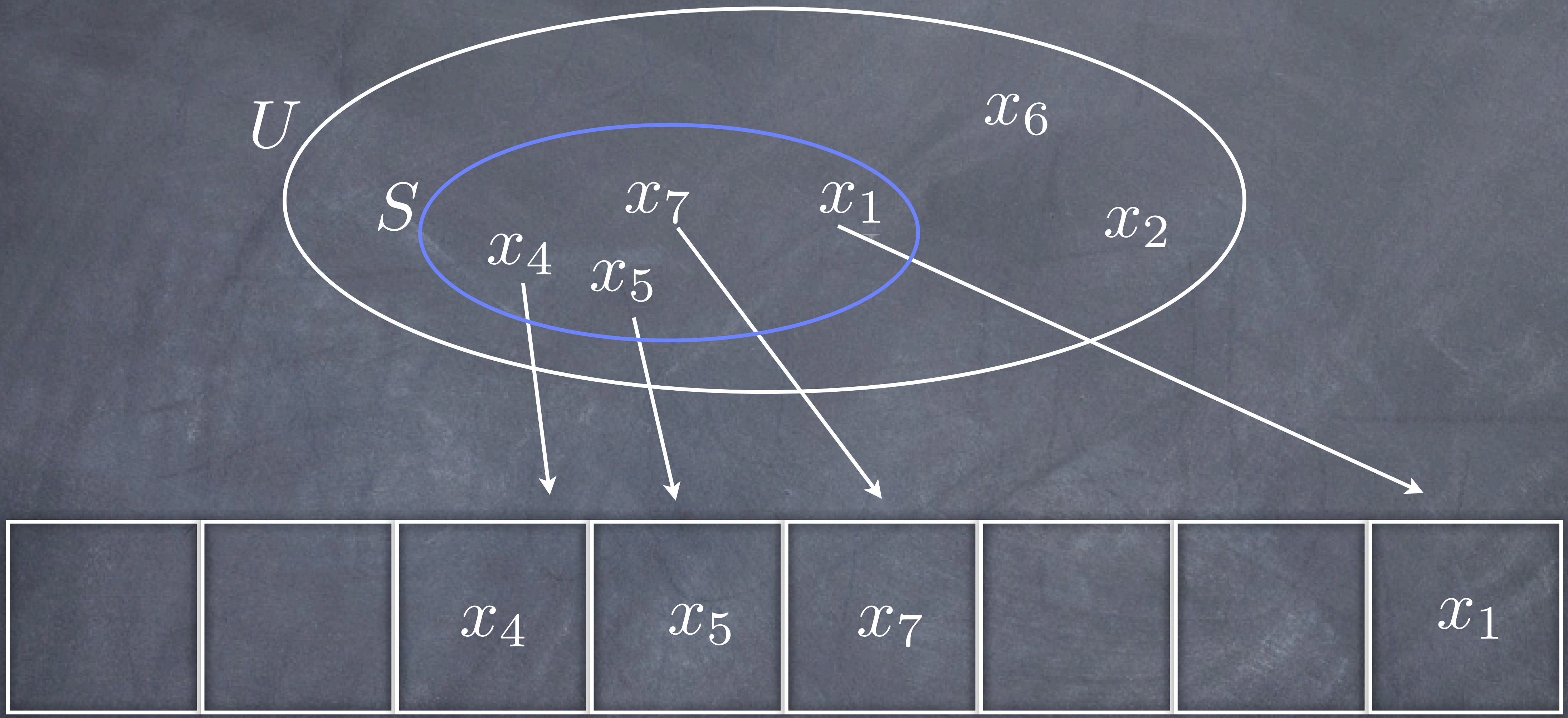
Zhewei Wei
Hong Kong University of Science & Technology

Joint work with Rasmus Pagh, Ke Yi
and Qin Zhang

Dictionary Problem

- Store a subset S of the Universe U.
- Lookup: Does x belong to S? If so, what is its associated data?
- Dynamic dictionary:
  - Insertion: Include x into the dictionary.
  - Deletion: Remove x from the dictionary.

Hashing

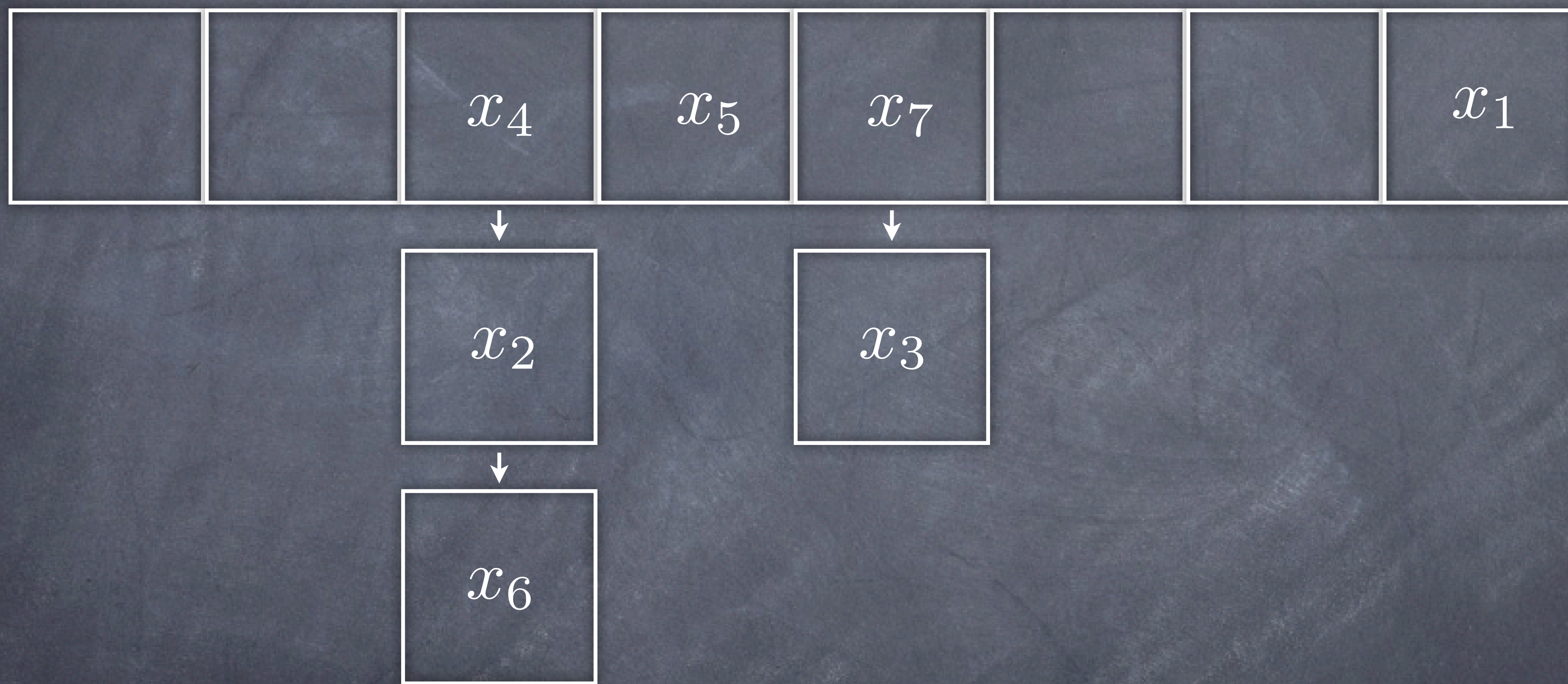


- Idea: Store the keys in random locations.
- Use a "hash function" h to generate and remember random locations.

Uniform Hashing Model

- Most analyses assume h to be a truly random hash function, i.e., h maps each key independently and uniformly to the hash table.
- Analyses match what happens on real-world data surprisingly well;
- Mitzenmacher and Vadhan (2008) shows that a simple hash function can be used to achieved the same performance as a truly random hash function does, under some mild assumption on the randomness of the data.

Hashing with Chaining

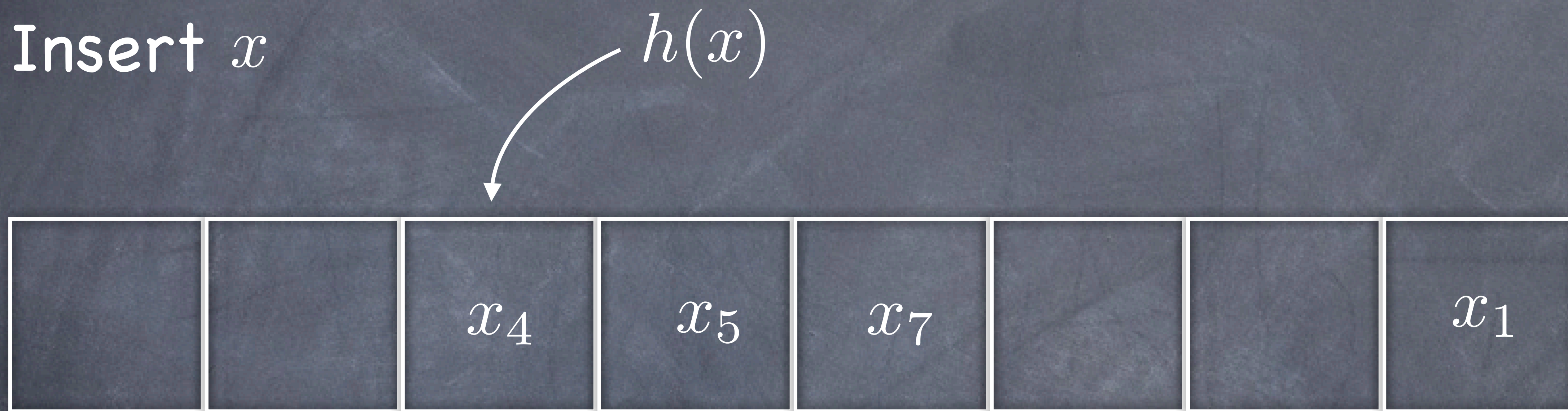


- Knuth: $C_n \approx 1 + \frac{\alpha}{2}$ $\qquad \alpha = n/r$

  $C'_n \approx 1 + \alpha$

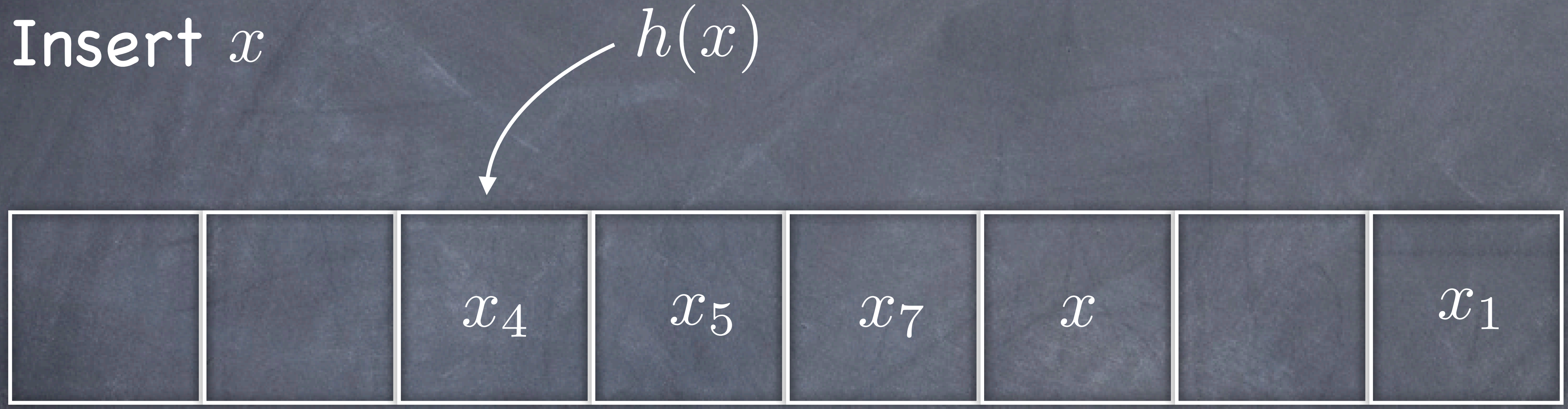
Hashing with Linear Probing

Insert x



Hashing with Linear Probing

Insert x

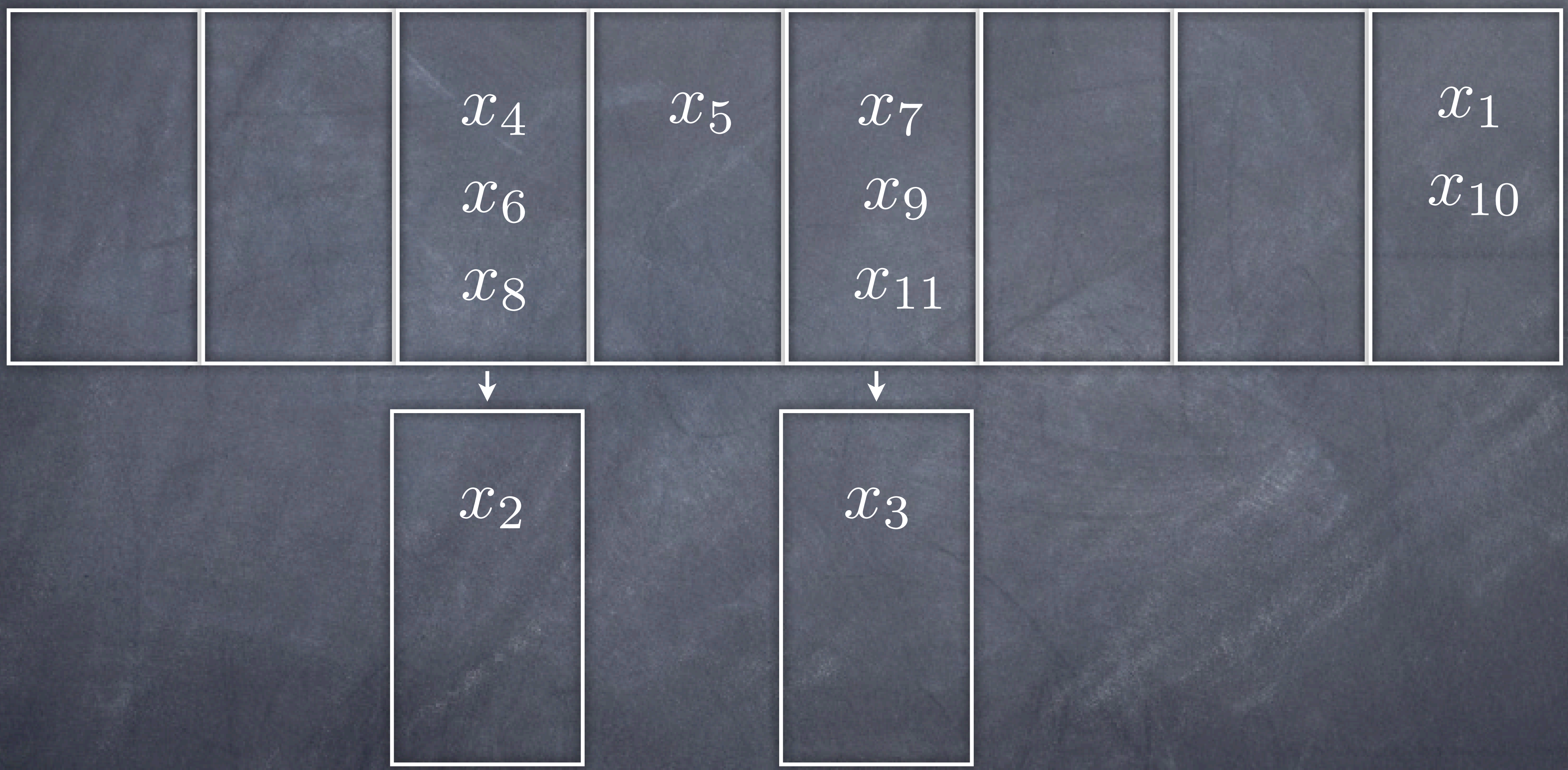


- Knuth:

$$C_n \approx \frac{1}{2} + \frac{1}{2(1-\alpha)}$$

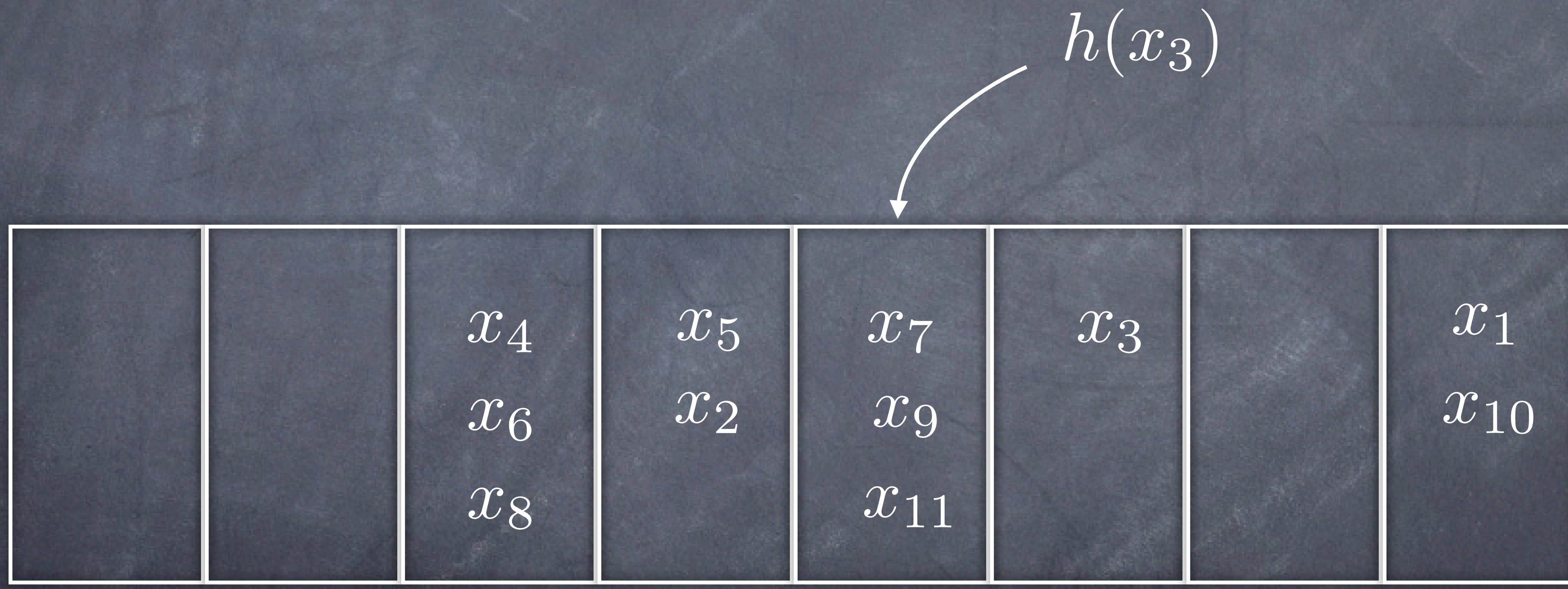
$$C'_n \approx \frac{1}{2} + \frac{1}{2(1-\alpha)^2}$$

External Hashing (Chaining)



- Block size $b = 3$

External Hashing (Linear Probing)



- Block size $b = 3$

External Hashing

- Each block can accommodate b keys.
- The cost of an operation (search, insertion) is the number of blocks accessed (I/Os).
- Knuth: expected I/O cost per operation:

$$1 + 2^{-\Omega((1-\alpha)^2 b)}$$

- For reasonable large b, the cost is very close to 1.

Cache-Oblivious Hashing

- Cache-Oblivious Model
  - Proposed by Frigo et al. (1999).
  - Similar to the I/O model, except that the algorithm does not know the memory size m and the block size b.
  - Algorithm must be optimized for all block sizes.
- Question: how to achieve the $1 + 2^{-\Omega(b)}$ bound without knowing b?

Our Results

- Linear probing ignoring the blocking is naturally cache-oblivious. However, analysis shows that its search cost is $1 + \Theta(\alpha/b)$ I/Os.

- Blocked probing (Pagh et al. 2007) achieves the desired $1 + 2^{-\Omega(b)}$ bound, under two assumptions:

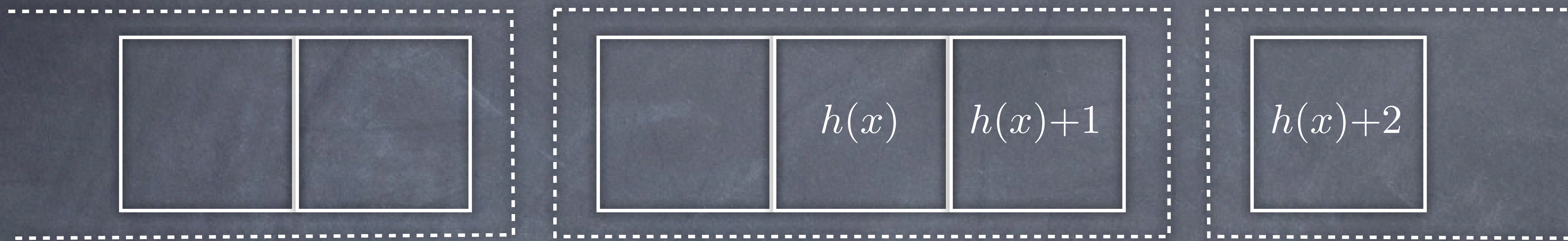
  - The block size b is a power of 2.

  - Every block starts at a memory address divisible by b.

Our Results

- A lower bound shows that both conditions is required to achieve the $1 + 2^{-\Omega(b)}$ bound;
- If one of the two conditions is dispersed, the best achievable bound is $1 + \Theta(\alpha/b)$.
- Neither of these two conditions is stated in the cache-oblivious model, but they indeed hold on all real machines.

Linear Probing ignoring the blocking

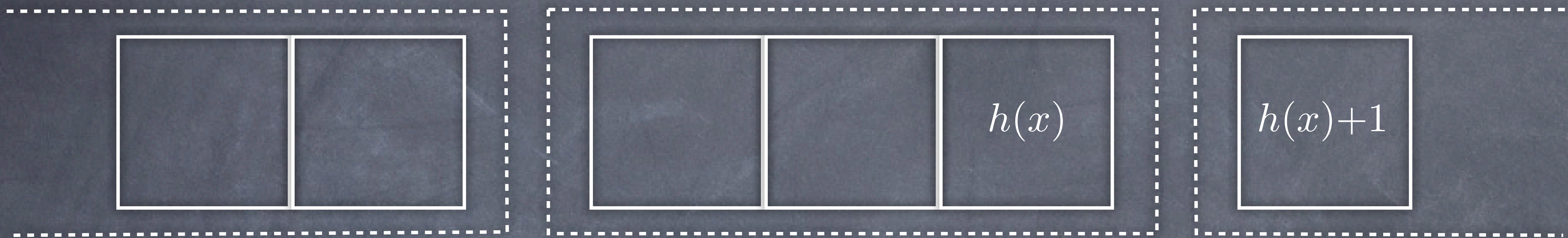


- Let CO_n and CO'_n denote the expected I/O cost for a successful and an unsuccessful search, respectively. For any block size b:

$$CO_n = 1 + (C_n - 1)/b$$
$$CO'_n = 1 + (C'_n - 1)/b$$

- The bound is $1 + \Theta(\alpha/b)$!

Linear Probing ignoring the blocking



- Intuition: consider a search for x. If $h(x)$ hits the last position of some block, and this position is occupied, then an extra I/O is needed.
- The probability that $h(x)$ hits the last position of some block is $1/b$.
- The probability that this position is occupied is $n/r = \alpha$.

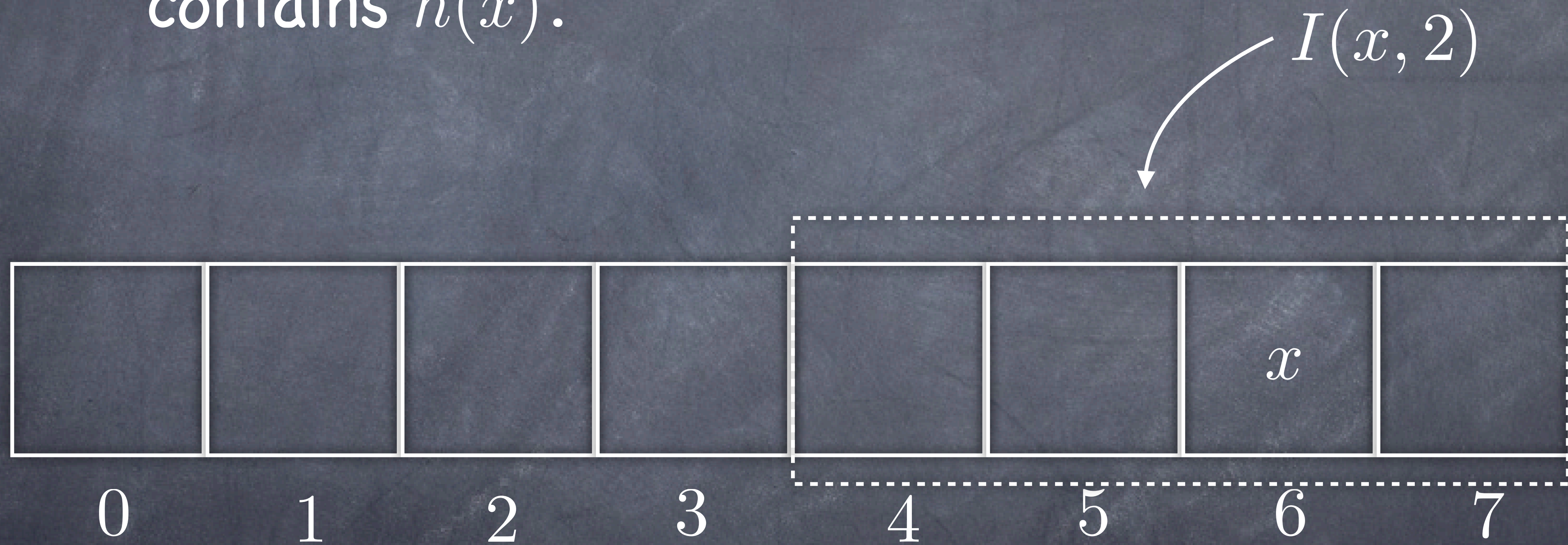
Blocked Probing

- Assuming r is a power of two.
- Suppose x is stored in location i_x.
- Define $d(x,i)$ to be equal to the position of the most significant bit in which $h(x)$ and i differ.

$d(x,i) = 0$ in case $i = h(x)$.

Blocked Probing

- Let $I(x,j) = \{i \mid d(x,i) \le j\}$.
- $I(x,j)$ is the aligned block of size 2^j that contains $h(x)$.



Blocked Probing

- Observation: under the two conditions, the block containing x is $I(x, \log b)$.
- Invariant 1: For $j = 0, \ldots,$ an operation on x will fully traverse $I(x, j)$ before moving to the next j.
- Invariant 2: each key is stored as close as possible to $h(x)$, i.e., If the number of keys with hash values in $I(x, j)$ is less than 2^j, then x is stored in $I(x, j)$.

Operations

- Insertion
  - For $j = 0, 1, 2, \ldots,$ search for an empty location in $I(x, j)$ and put x there;
  - If no empty location is found, search for a location $i_{x'}$ that contains a key x' with hash value $h(x') \notin I(x, j)$ (i.e., $d(x', i_{x'}) > j$). Swap x and x' and continue the insertion process with x'.
  - If both attempts fail, move to the next j.

Operations

- Search
  - For $j = 0, 1, 2, \ldots,$ inspect $I(x, j)$ until x is found.
  - Or an empty location is found.
  - Or a key x' with hash value $h(x') \notin I(x, j)$.
- Deletion
  - Find j such that $x \in I(x, j) \backslash I(x, j-1)$.
  - Check if there is a key in $I(x, j+1) \backslash I(x, j)$ that can be stored in $I(x, j)$.

Analysis of Blocked Probing

- Suppose we want to query key x.
- Let $X_j = |\{y \in S \mid h(y) \in I(x, j)\}|$.
- We will not visit any locations outside $I(x, j^*)$, where $j^* = \min\{j \mid X_j < 2^j\}$.
- $E[X_j] = \alpha 2^j$.
- By Chernoff bounds,

$$Pr[X_j \geq 2^j] \leq 2^{-(1-\alpha)^2(2^j-1)/2}$$

Analysis of Blocked Probing

- Assumption: b is a power of 2 and storage block are aligned to multiples of b.
- All locations in $I(x, \log b)$ can be accessed in 1 I/O.
- If the search goes on to step $j^* > \log b$, the number of I/Os required is $2^{j^*}/b$.

$$1 + \sum_{j=1+\log b}^{\infty} (2^j/b) 2^{-(1-\alpha)^2 (2^j - 1)/2} = 1 + 2^{-\Omega((1-\alpha)^2 b)}$$

Lower Bounds

- Neither of the two conditions is dispensable:
  - The block size b is a power of 2.
  - Every block starts at a memory address divisible by b.
- The best achievable bound is $1 + \Theta(\alpha/b)$ if
  - The hash table is required to work for all b.
  - Or the hash table is required to work for a single b, but an arbitrary shifting of the starting position is allowed.

The Model

- $U = [u]$: the universe.
- I_u: a random n-key sequence drawn from the universe randomly and independently.
- Assuming $u > n^3$, then w.h.p. all keys in I_u are distinct by the birthday paradox.
- Assume that all keys are stored in a table of size r on the external memory (not affecting the analysis).
- Assume $r = O(n)$.

The Block Layout

- Boundary-Oblivious Model
  - The hash table knows the block size b but not the block boundary;
  - A block spans from $ib - s$ to $(i+1)b - s - 1$.
- Block-Size-Oblivious Model
  - The blocks always start at positions that are multiples of b;
  - But the hash table is required to work for all $b = 1, \ldots, r$.

The Model

- The successful search for x is simulated by two functions:
  - $f(x)$ is the position where the algorithm makes its first probe;
  - $g(x)$ is the position of the last probe, where key x is stored.
- The description of f is stored in the internal memory.

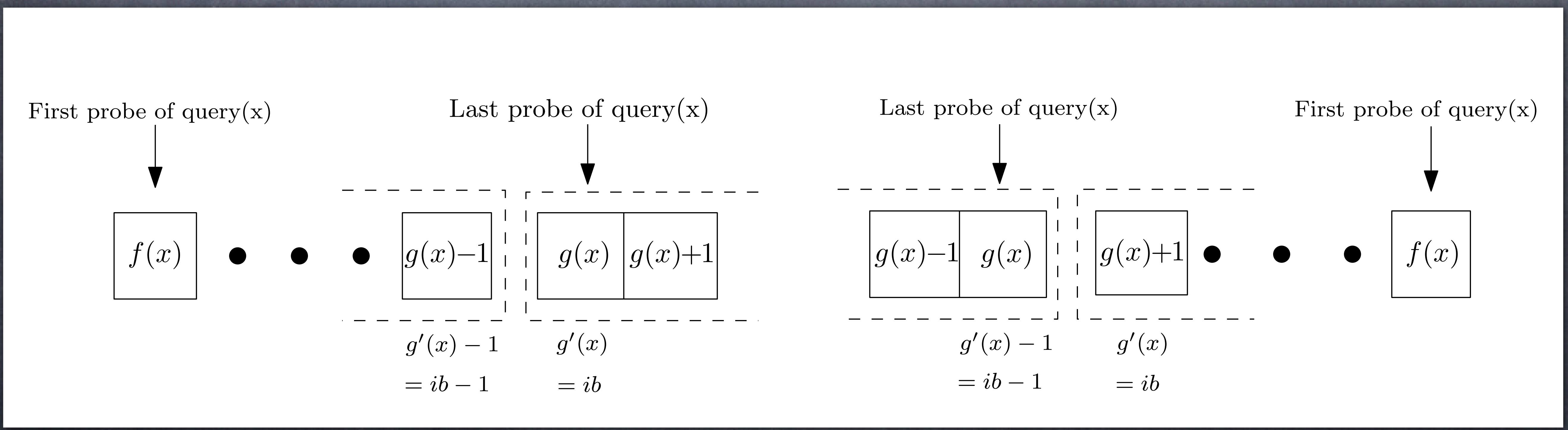
The Model

- Observation 1: The algorithm can employ a family of at most $2^{m \log u}$ $f's$.
- Observation 2: All $g(x)'s$ are distinct for the n keys.
- Observation 3: If $f(x)$ and $g(x)$ are on different blocks, the search for x will cost two I/Os.

The Model

- For $f(x) \neq g(x)$, let

$$g'(x) = \begin{cases} g(x) & \text{if } f(x) < g(x) \\ g(x)+1 & \text{if } f(x) > g(x) \end{cases}$$



- If $g'(x)$ is the first position of a block, at least two I/Os are needed.

Basic Idea

- For a random input, number of keys that need a second probe is large;
- For such a key x, its $f(x)$ and $g(x)$ are different, and thus $g'(x)$ is defined;
- Prove that at least one block layout will cause a large number of $g'(x)$'s to meet the starting position of some blocks, and these keys will need a second I/O to query.

A Bin-Ball Game

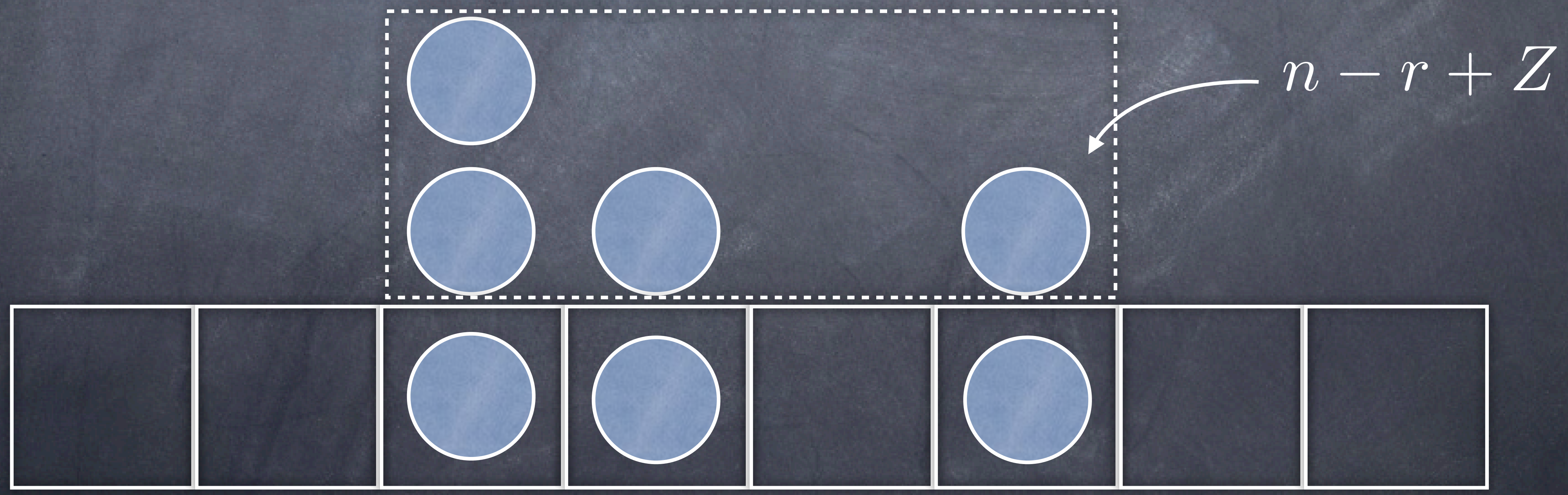
- Throw n balls into r bins independently.
- Each ball goes to the j-th bin w.p. β_j .
- $\beta = (\beta_1, \ldots, \beta_r)$ prefixed.
- Let Z denote the number of empty bins after n balls are thrown in.

$$Pr[Z \leq r - n + \frac{\alpha}{4} n] \leq e^{-\Omega(\alpha^2 n)}$$

A Bin-Ball Game

- The bin-ball game can be used to model the process that a prefixed hash function f maps a random input I_u.

- $n - r + Z$ is the number of keys that need a second probe, which is at least $\frac{\alpha}{4}n$ w.h.p..



A Bin-Ball Game

- Increasing the number of hash functions does not help, as long as n is unbounded from m and b.

- For a random input I_u, w.h.p. at least $\frac{\alpha}{4}n$ keys need a second probe.

Lower bound for the Boundary-Oblivious Model

- Number of $g'(x)$'s: at least $\frac{\alpha}{4}n$.
- For $s = 0, \ldots, b-1$, there exist one s such that $g'(x) = ib - s$.
- Sum up (on all s and all $g'(x)$) the number of times that a $g'(x)$ hits the first position of some block: at least $\frac{\alpha}{4}n$.
- By pigeon hole principle, there exist a s such that the number of $g'(x)$'s that hits the first position of some block is at least $\frac{\alpha n}{4b}$.

Lower bound for the Block-Size-Oblivious Model

- Number of $g'(x)$'s: at least $\frac{\alpha}{4}n$.
- Consider the set P which consists of all primes that are less than r.
- Fix a $b \in P$, number of keys need a second I/O is at least the number of $g'(x)$'s that are divisible by b.

Lower bound for the Block-Size-Oblivious Model

- Sum up (on all b and all $g'(x)$) the number of times that some $g'(x)$ is a multiple of some b is at least

$$\sum_{g'(x)} \mu(g'(x))$$

- $\mu(s)$: number of distinct prime factors of s.
- Lemma: at least $(1-o(1))$ fraction of $s \in [r]$ has $\mu(s) = \Omega(\log\log r)$.
- The set of all $g'(x)$'s has at least $\frac{\alpha}{8}n$ distinct values in $[r]$.

Lower bound for the Block-Size-Oblivious Model

- The summation
$$\sum_{g'(x)} \mu(g'(x)) = \Omega(r \log\log r)$$
- Lemma: P is the set of all primes less than r:
$$\sum_{b \in P} \frac{1}{b} = \log\log r + O(1)$$
- There exists a b, s.t. the number of $g'(x)$'s that are multiples of b is $\Omega(r) = \Omega(\frac{\alpha n}{b})$.

Open Questions

- Is the $1+2^{-\Omega(b)}$ bound optimal?
- If the internal memory size is $\Theta(n/b)$ bits, we can achieve *1* I/O worst-case query cost (perfect hashing).
- How about $m = O(b)$?