Linear probing with constant independence

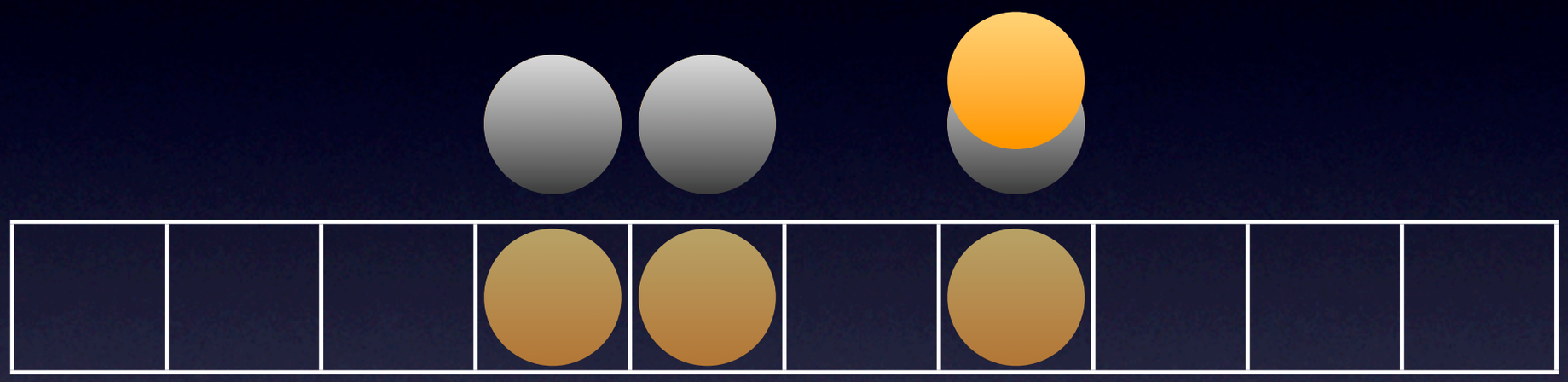
Anna Pagh, Rasmus Pagh, and Milan Ružić

IT University of Copenhagen

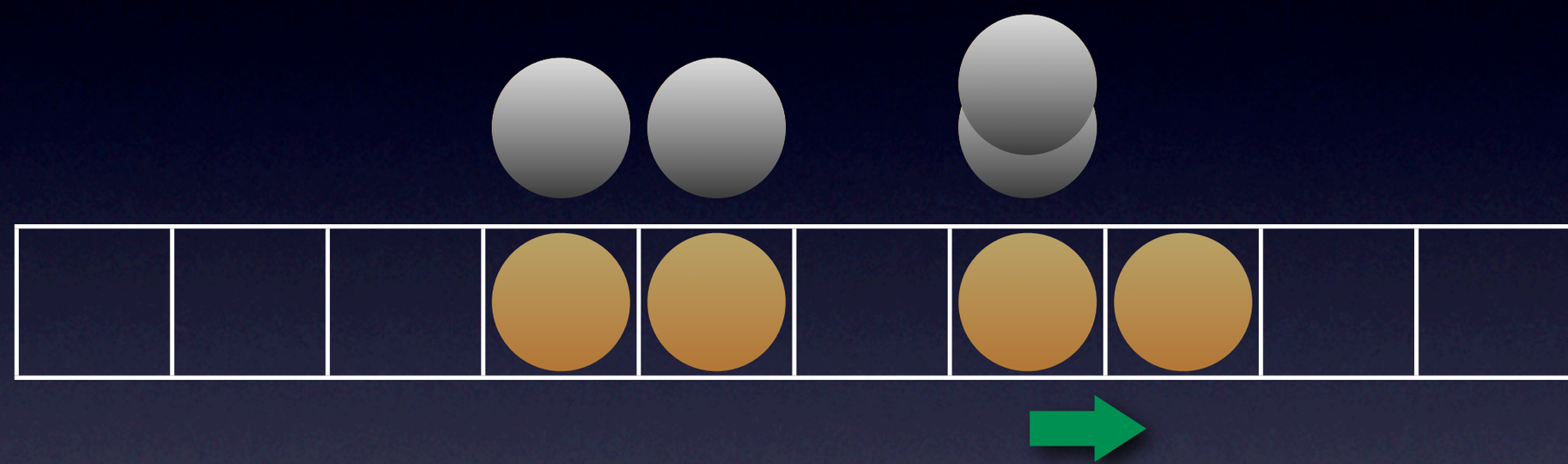
STOC 2007

Hashing with linear probing

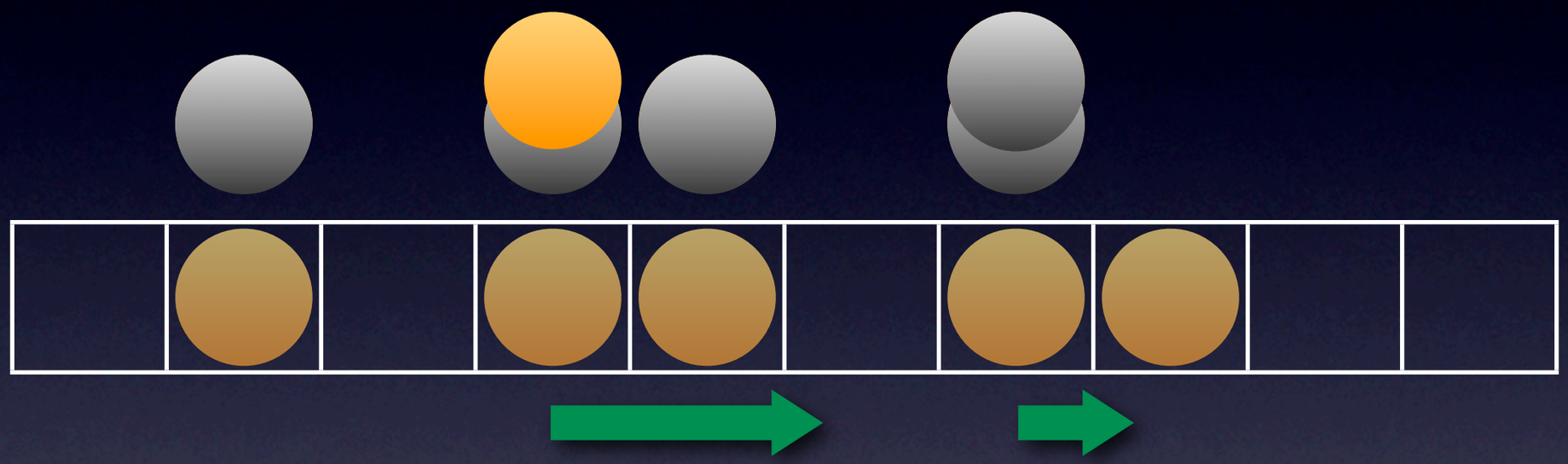
Hashing with linear probing



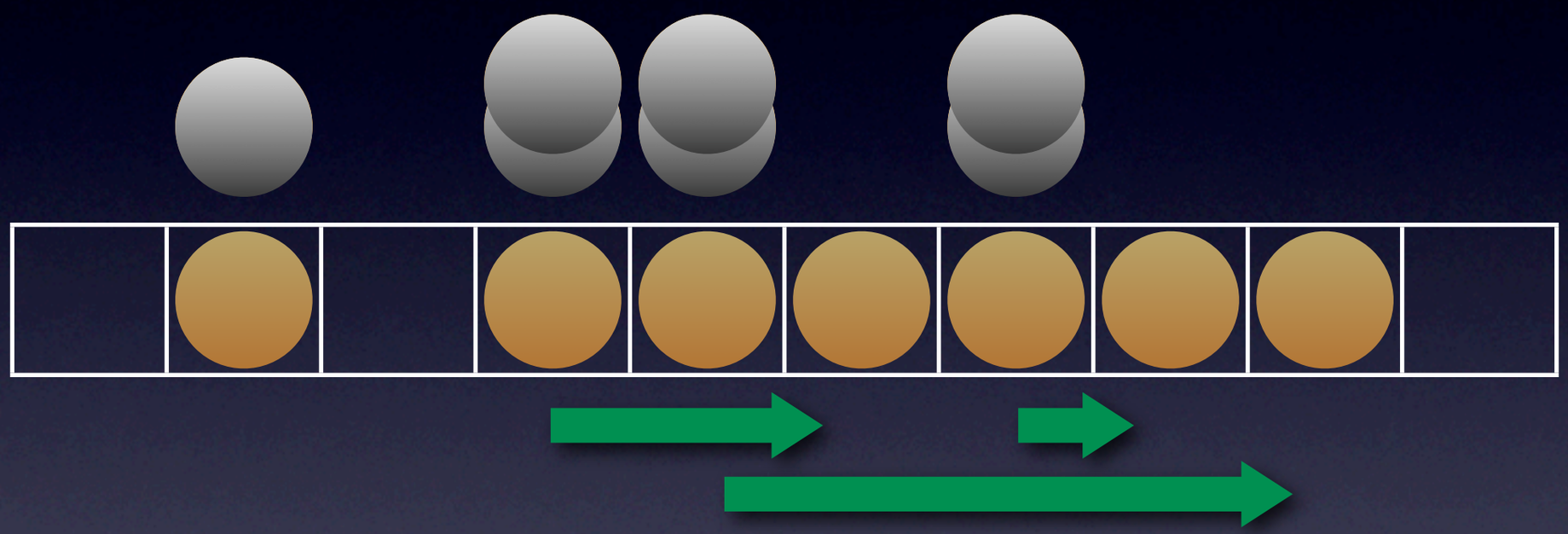
Hashing with linear probing



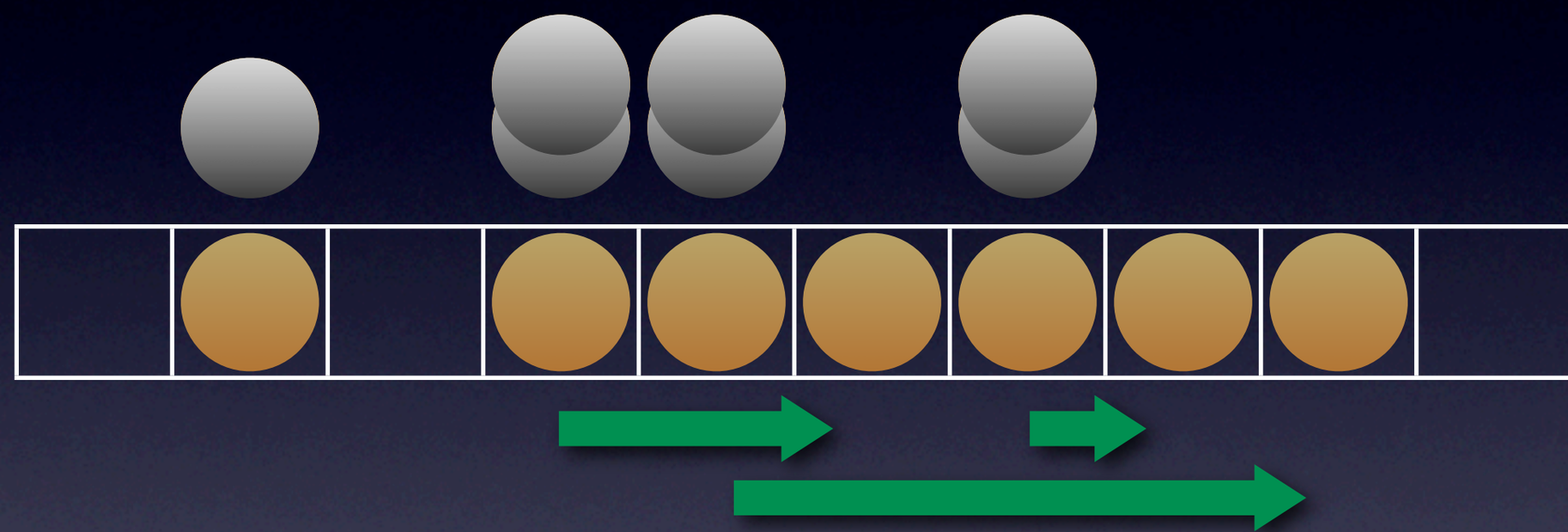
Hashing with linear probing



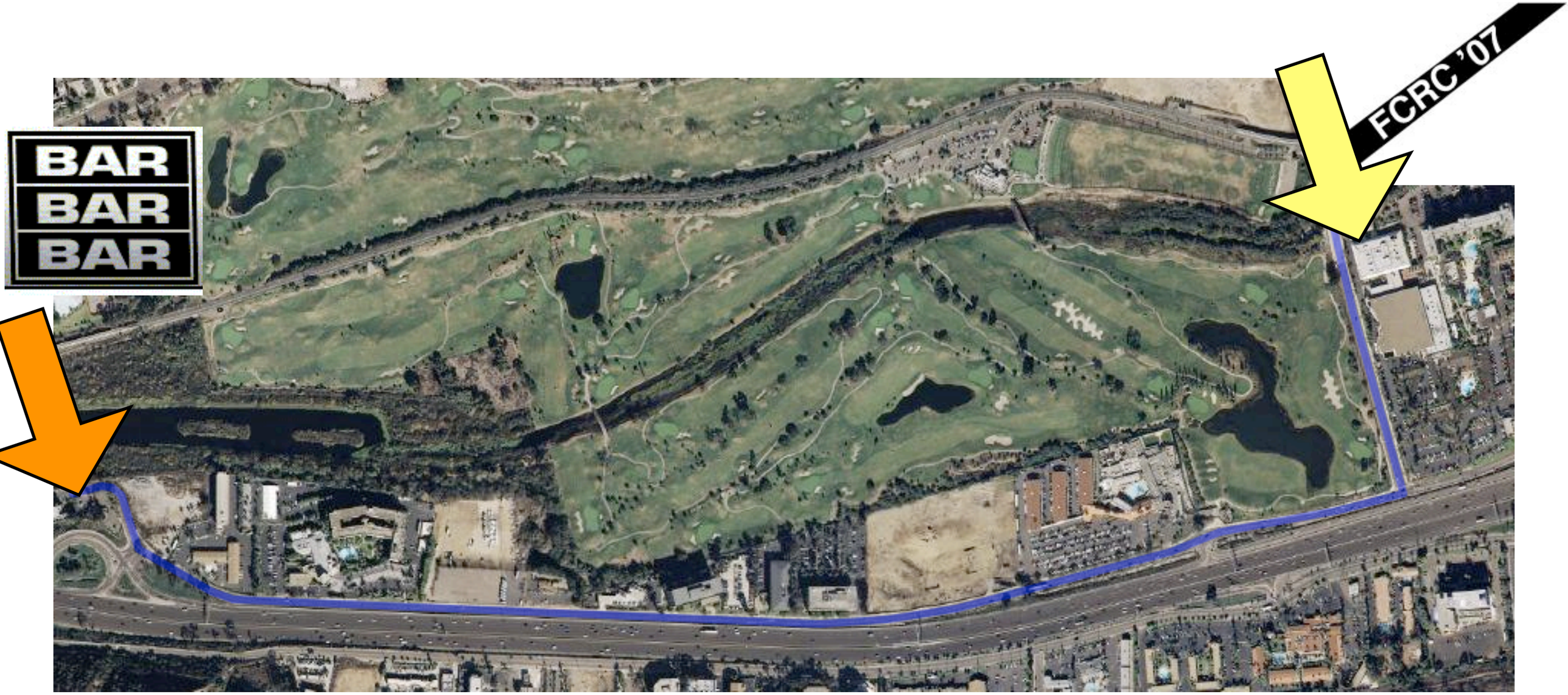
Hashing with linear probing



Hashing with linear probing

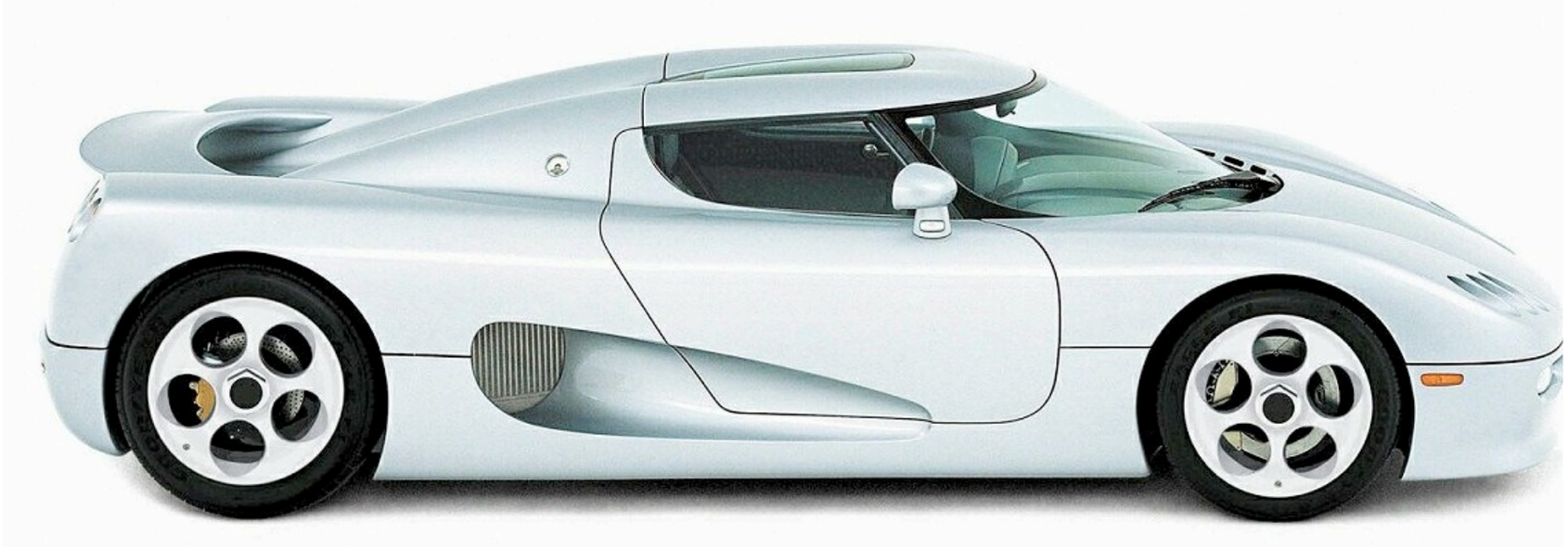


It was settled in the 60s that this is inferior to e.g. double hashing. So why care?

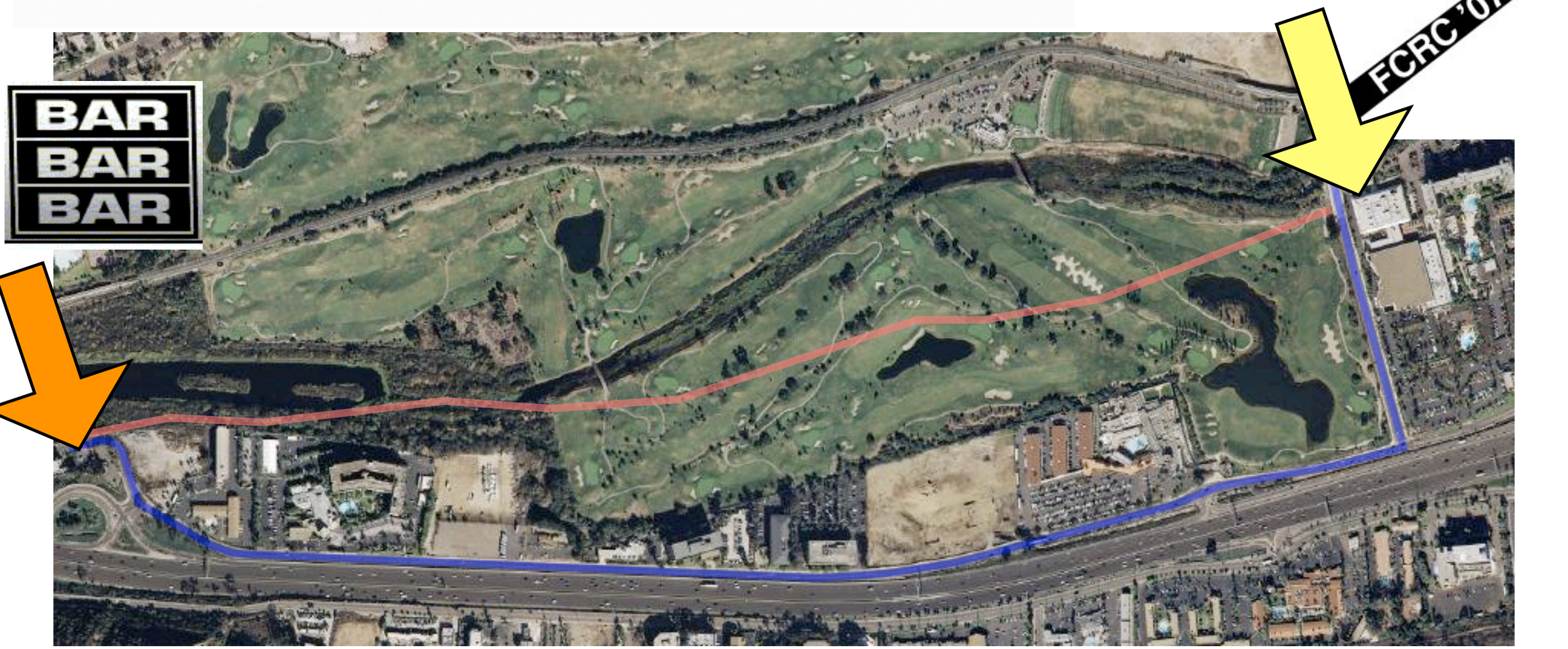

BAR
BAR
BAR
FCRC '07


BAR
BAR
BAR
FCRC '07

389 km/h



20 km/h



Race car vs golf car

- Linear probing uses a sequential scan and is thus *cache-friendly*.
- On my laptop: 24x speed difference between sequential and random access!
- Experimental studies have shown linear probing to be faster than other methods for load factor α in the range 30-70%.

For 4-byte words

For "small" keys

Race car vs golf car

- Linear probing uses a sequential scan and is thus *cache-friendly*.
- On my laptop: 24x speed difference between sequential and random access!

For 4-byte words

- Experimental studies have shown linear probing to be faster than other methods for load factor α in the range 30-70%.

For "small" keys

- **But**: No theory behind the hash functions used for linear probing in practice.

History of linear probing

- First described in 1954.
- Analyzed in 1962 by D. Knuth, aged 24. Assumes hash function h is truly random.

[My first analysis of an algorithm, originally done during summer 1962 in Madison.]

8354

NOTES ON "OPEN" ADDRESSING. D. Knuth 7/22/63

1. Introduction and Definitions. Open addressing is a widely-used technique for keeping "symbol tables." The method was first used in 1954 by Samuel, Amdahl, and Boehme in an assembly program for the IBM 701. An extensive discussion of the method was given by Peterson in 1957 [1], and frequent references have been made to it ever since (e.g. Schay and Spruth [2], Iverson [3]). However, the timing characteristics have apparently never been exactly established, and indeed the author has heard reports of several reputable mathematicians who failed to find the solution after some trial. Therefore it is the purpose of this note to indicate one way by which the solution can be obtained.

We will use the following abstract model to describe the method: N is a positive integer, and we have an array of N variables $x_1, x_2, \ldots, x_N$. At the beginning, $x_i = 0$, for $1 \leq i \leq N$.

To "enter the k-th item in the table," we mean that an integer a_k is calculated, $1 \leq a_k \leq N$, depending only on the item, and the following process is carried out:

History of linear probing

- First described in 1954.
- Analyzed in 1962 by D. Knuth, aged 24. Assumes hash function h is truly random.
- Over 30 papers using this assumption.
- Siegel and Schmidt (1990) showed that it suffices that h is *O(log n)-wise independent*.

History of linear probing

- First described in 1954.
- Analyzed in 1962 by D. Knuth, aged 24. Assumes hash function h is truly random.
- Over 30 papers using this assumption.
- Siegel and Schmidt (1990) showed that it suffices that h is *O(log n)-wise independent*.

<u>Our main result:</u>
It suffices that h is *5-wise independent*.

This talk

- Background and motivation
- Hash functions
- New analysis of linear probing
- Lower bound for 2-wise independence
- XOR probing

log(n)-wise independence

- Siegel (1989) showed time-space trade-offs for evaluation of a function from a log(n)-wise independent family:

	Time	Space
Lower bound	$\frac{\log(n)}{\log(s/\log n)}$	s
Upper bound 1*	$O(\log n)$	$O(\log n)$
Upper bound 2	$O(1)$	n^{ϵ}

- Upper bound 2 is theoretically appealing, but has a huge constant factor – and uses many random memory accesses!

log(n)-wise independence

- Siegel (1989) showed time-space trade-offs for evaluation of a function from a log(n)-wise independent family:

	Time	Space
Lower bound	$\frac{\log(n)}{\log(s/\log n)}$	s
Upper bound 1*	$O(\log n)$	$O(\log n)$
Upper bound 2	$O(1)$	n^{ϵ}

- Upper bound 2 is theoretically appealing, but has a huge constant factor – and uses many random memory accesses!

5-wise independence

- Polynomial hash function:

$$h(x) = \left(\sum_{i=0}^{4} a_i x^i \bmod p \right) \bmod r$$

Carter and Wegman (FOCS '79)

Already quite fast

- Tabulation-based hash function:

$$h(x_1, x_2) = T_1[x_1] \oplus T_2[x_2] \oplus T_3[x_1 + x_2]$$

Thorup and Zhang (SODA '04)

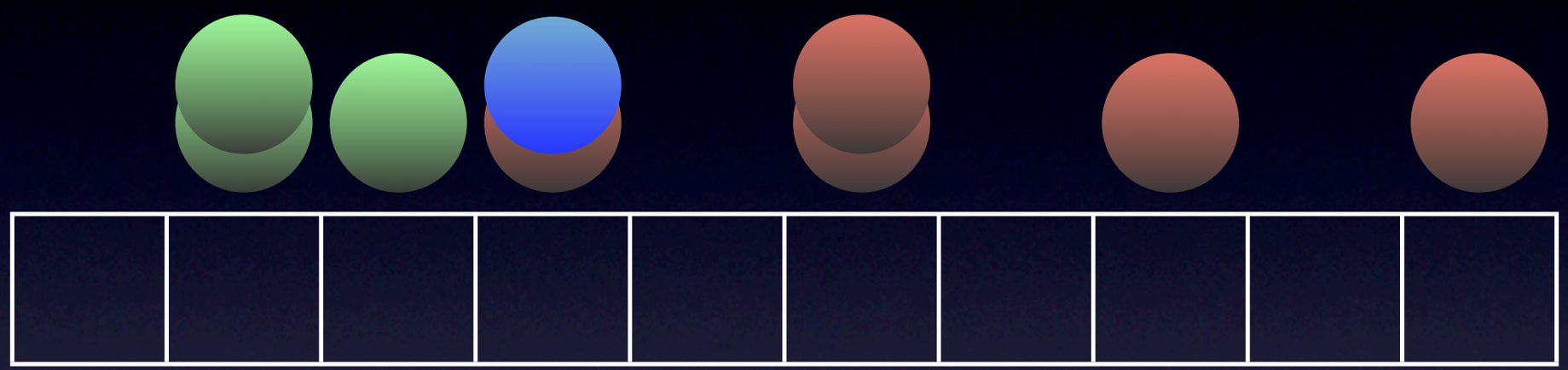
Within factor 2 of the fastest universal hash functions

This talk

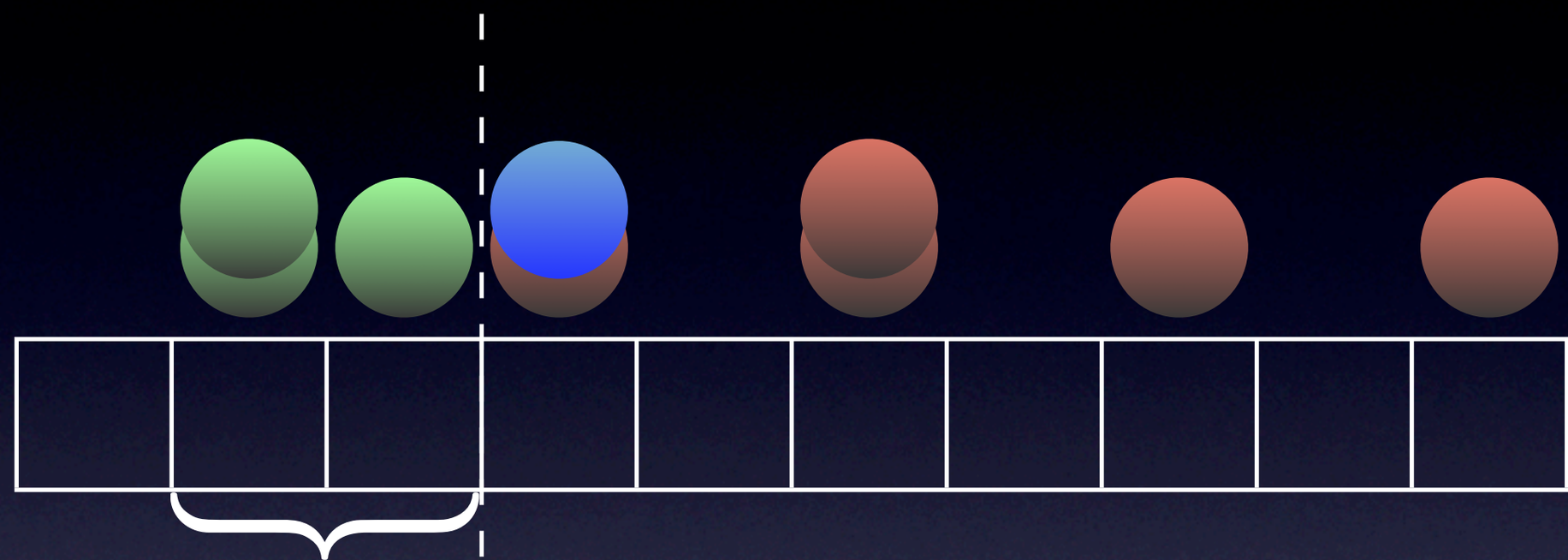
- Background and motivation
- Hash functions
- New analysis of linear probing
- Lower bound for 2-wise independence
- XOR probing

Insertion cost upper bound

Insertion cost upper bound

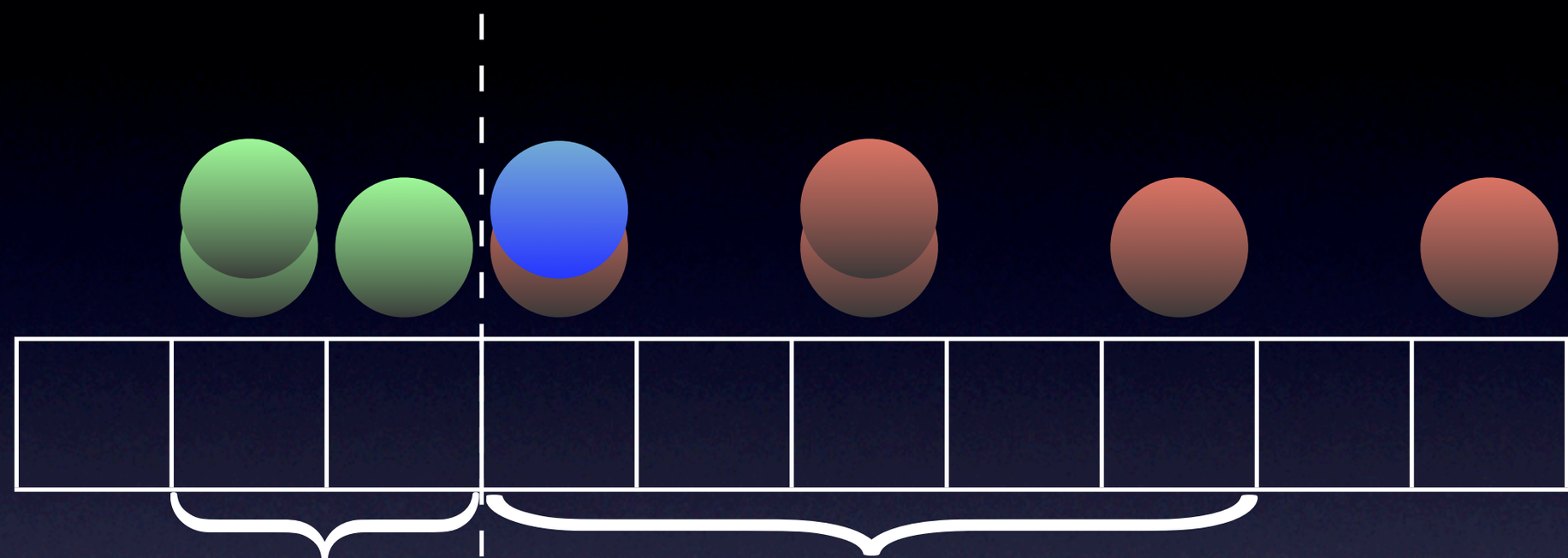


Insertion cost upper bound



1. Choose max t so B balls hash to B-t slots, for some B

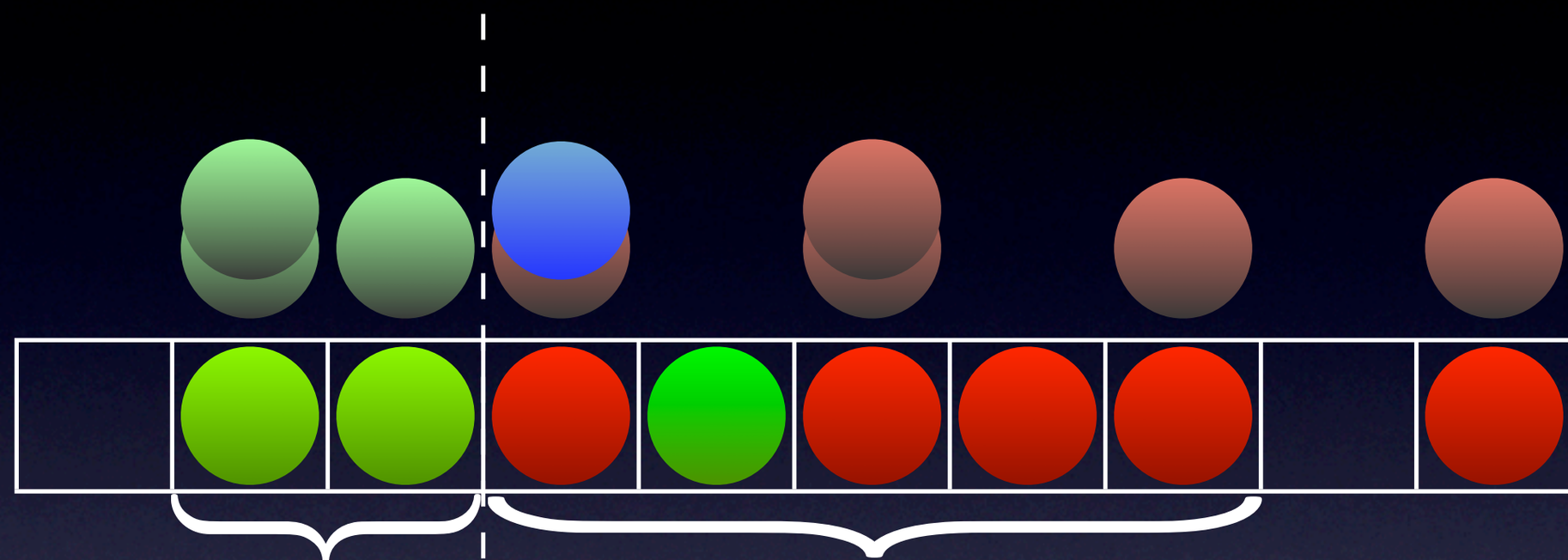
Insertion cost upper bound



1. Choose max t so B balls hash to B-t slots, for some B

2. Choose max C such that C balls hash to C+t slots

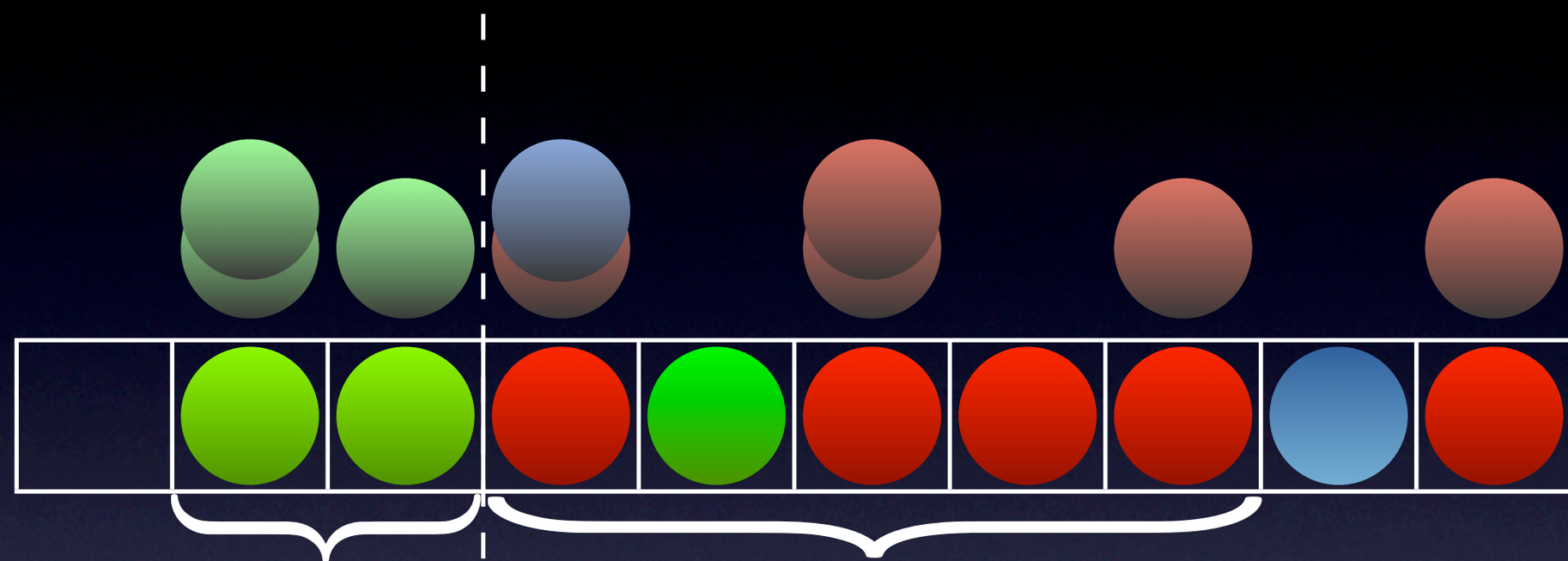
Insertion cost upper bound



1. Choose max t so B balls hash to B-t slots, for some B

2. Choose max C such that C balls hash to C+t slots

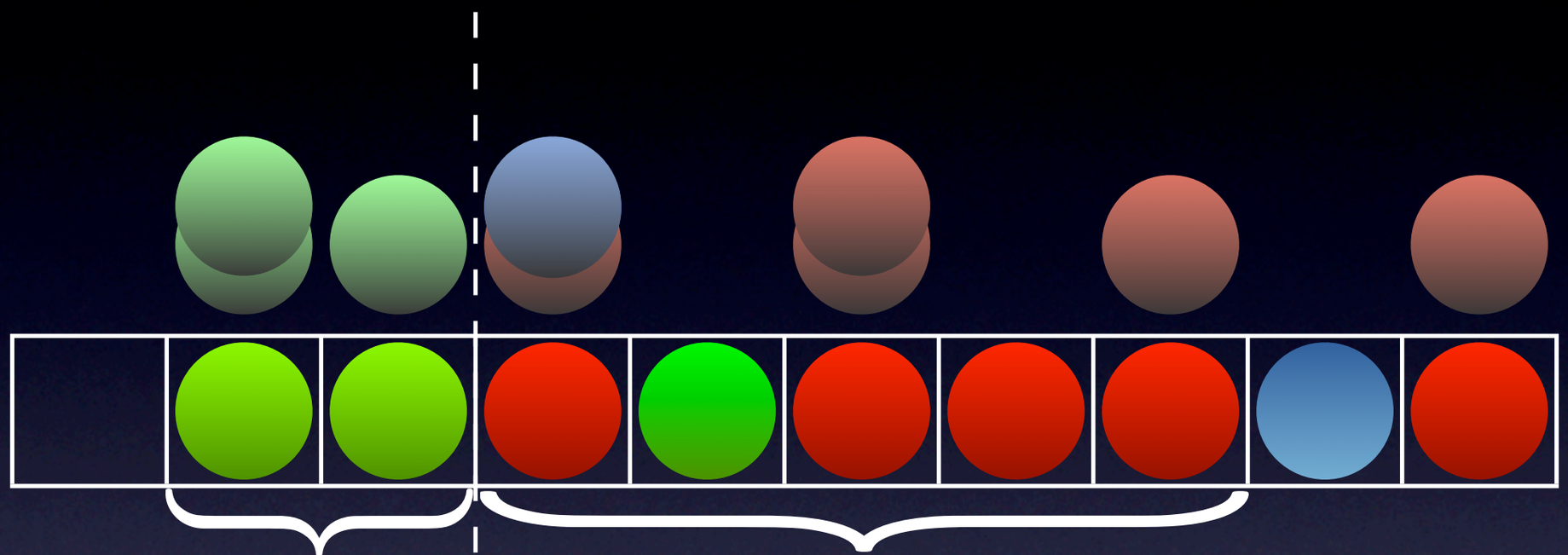
Insertion cost upper bound



1. Choose max t so B balls hash to B-t slots, for some B

2. Choose max C such that C balls hash to C+t slots

Insertion cost upper bound



1. Choose max t so B balls hash to B-t slots, for some B

2. Choose max C such that C balls hash to C+t slots

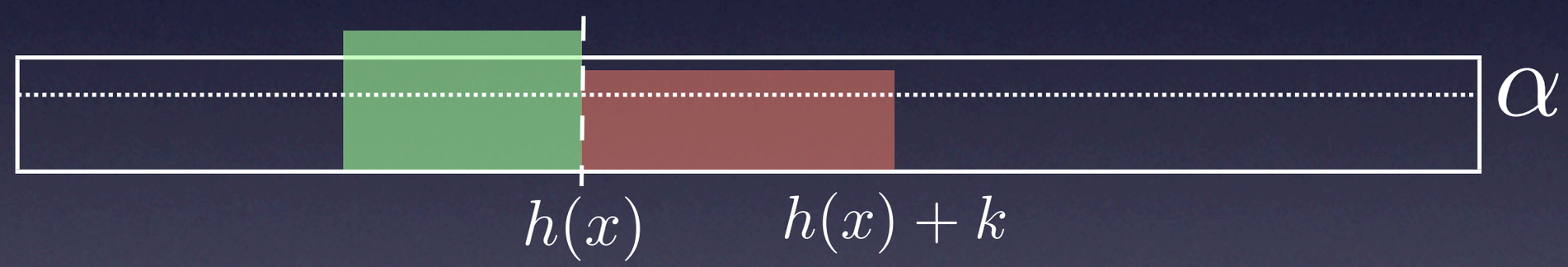
Lemma:

Cost(●)≤1+C+t

Proof idea

- <u>Lemma</u>: If operation on x goes on for more than k steps, then there are "unusually many" keys with hash values in either:

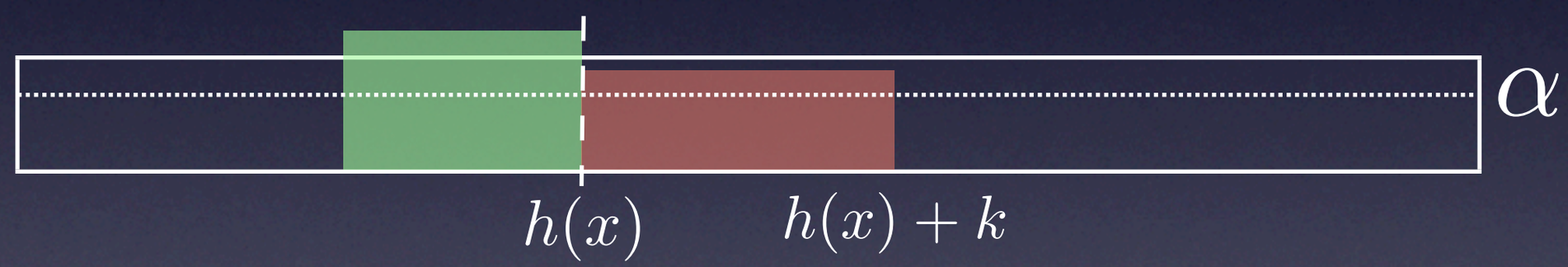
  1) Some interval with h(x) as right endpoint, or
  2) The interval [h(x),h(x)+k]



Proof idea

- <u>Lemma</u>: If operation on x goes on for more than k steps, then there are "unusually many" keys with hash values in either:

  1) Some interval with h(x) as right endpoint, or
  2) The interval [h(x),h(x)+k]



- To bound cost, upper bound probability of each event using tail bounds for sums of random variables with limited independence.

Our main result

Theorem 2 *Consider any sequence of insertions, deletions, and lookups in a linear probing hash table using a 5-wise independent hash function. Then the expected cost of any operation, performed at load factor α, is*

$$O(1 + (1 - \alpha)^{-3}) \ .$$

As a consequence, the expected average cost of successful lookups is $O(1 + (1 - \alpha)^{-2})$.

Our main result

Theorem 2 *Consider any sequence of insertions, deletions, and lookups in a linear probing hash table using a 5-wise independent hash function. Then the expected cost of any operation, performed at load factor α, is*

$$O(1 + (1 - \alpha)^{-3}) \ .$$

As a consequence, the expected average cost of successful lookups is $O(1 + (1 - \alpha)^{-2})$.

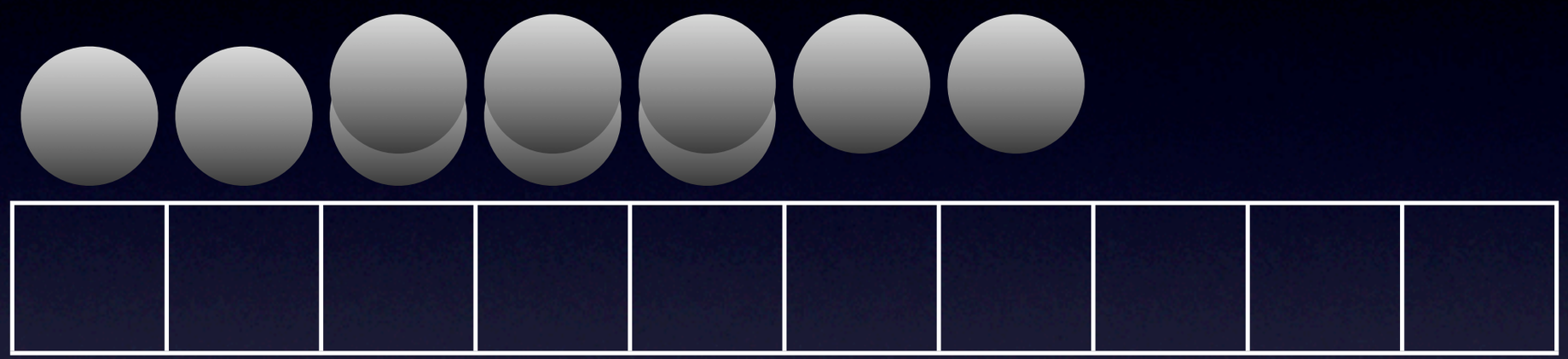
factor $(1 - \alpha)^{-1}$ from what can be proved using full independence

This talk

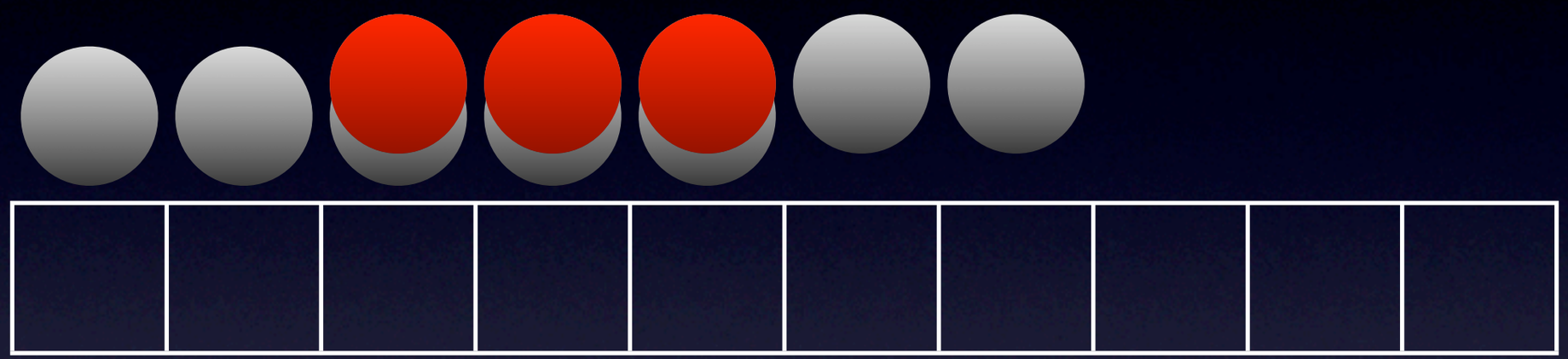
- Background and motivation
- Hash functions
- New analysis of linear probing
- ▸ Lower bound for 2-wise independence
- XOR probing

Cost lower bound

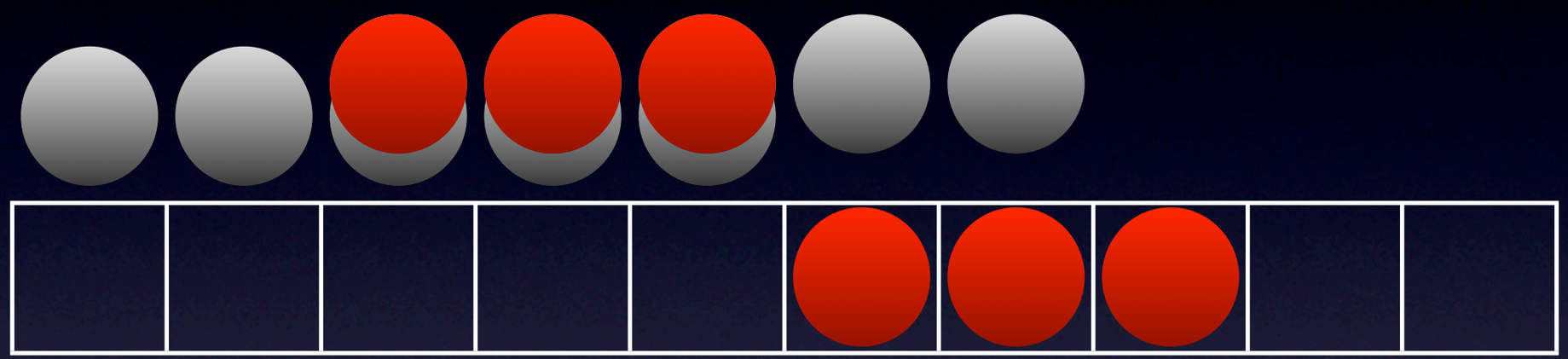
Cost lower bound



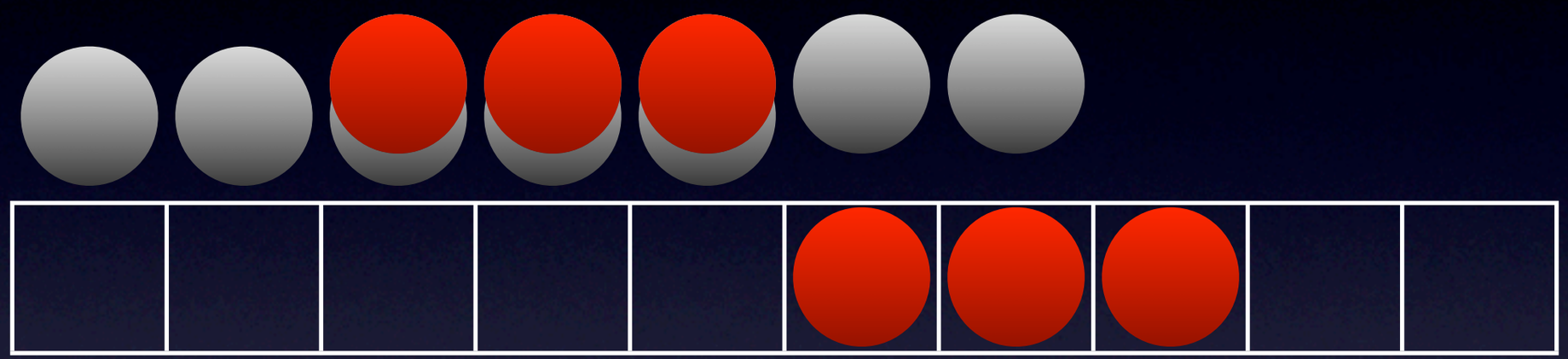
Cost lower bound



Cost lower bound



Cost lower bound



Lemma 2 *Suppose that the multiset of hash values for the keys is* $\bigcup_j I_j$*, where* $I_1, I_2, \ldots$ *are intervals. Then the total number of steps to perform the insertions is at least*

$$\sum_{j_1 < j_2} |I_{j_1} \cap I_{j_2}|^2 / 2 \ .$$

Bad example: "Linear hashing"

$$h(x) = (ax + b \bmod p) \bmod r$$

- First example of pairwise independence.

Bad example: "Linear hashing"

$$h(x) = (ax + b \bmod p) \bmod r$$

- First example of pairwise independence.
- Consider an *interval* $S_1=\{z+1,\ldots,z+n\}$.

Bad example: "Linear hashing"

$$h(x) = (ax + b \bmod p) \bmod r$$

- First example of pairwise independence.
- Consider an *interval* $S_1=\{z+1,\ldots,z+n\}$.
- <u>Observation</u>:
  Let $m=a^{-1} \pmod p$. Then $h(S_1)$ is the union of at most $m+1$ intervals (mod r).

Lower bound for n insertions

- <u>Idea:</u> Let S = union of two random intervals

  ⇒ Expect that the 2 times m+1 intervals have large overlap

  ⇒ Expected cost $\Omega\left(m\left(\frac{n}{m}\right)^2\right) = \Omega(n^2/m)$.

  ⇒ For random m, expected cost

  $$\Omega\left(\frac{1}{p}\sum_{m=1}^{p-1} n^2/m\right) = \Omega\left(\frac{n^2}{p}\log p\right) .$$

  ⇒ In the case p=O(n), Ω(n log n) cost!

XOR probing

$$\text{Linear probing:} \quad h(x), h(x)+1, h(x)+2, \ldots$$
$$\text{XOR probing:} \quad h(x), h(x) \oplus 1, h(x) \oplus 2, \ldots$$

- XOR probing: Probe sequence never leaves the (aligned) memory block before it has been fully traversed.
- For XOR probing, we can show the *same result as in the fully random case*, up to a constant factor, using 5-wise independence.

End remarks

- Theory and practice of linear probing now (seem) much closer.
- We can generalize to *variable key lengths*.

End remarks

- Theory and practice of linear probing now (seem) much closer.
- We can generalize to *variable key lengths*.
- <u>Open</u>:
  - Still many hashing schemes where theory does not provide satisfactory methods.
  - Tighter analysis, lower independence?

THE END

Why 5?

Why 5?

- For every key x, the hash values of the other keys are 4-wise independent with respect to h(x).
- 4-wise independence gives a tail bound that is sufficiently strong.
- 2-wise independence would give a tail bound that is too weak.