

Diving Deeper into Chaining and Linear Probing

COMP 480/580

17th Jan 2019

Announcements

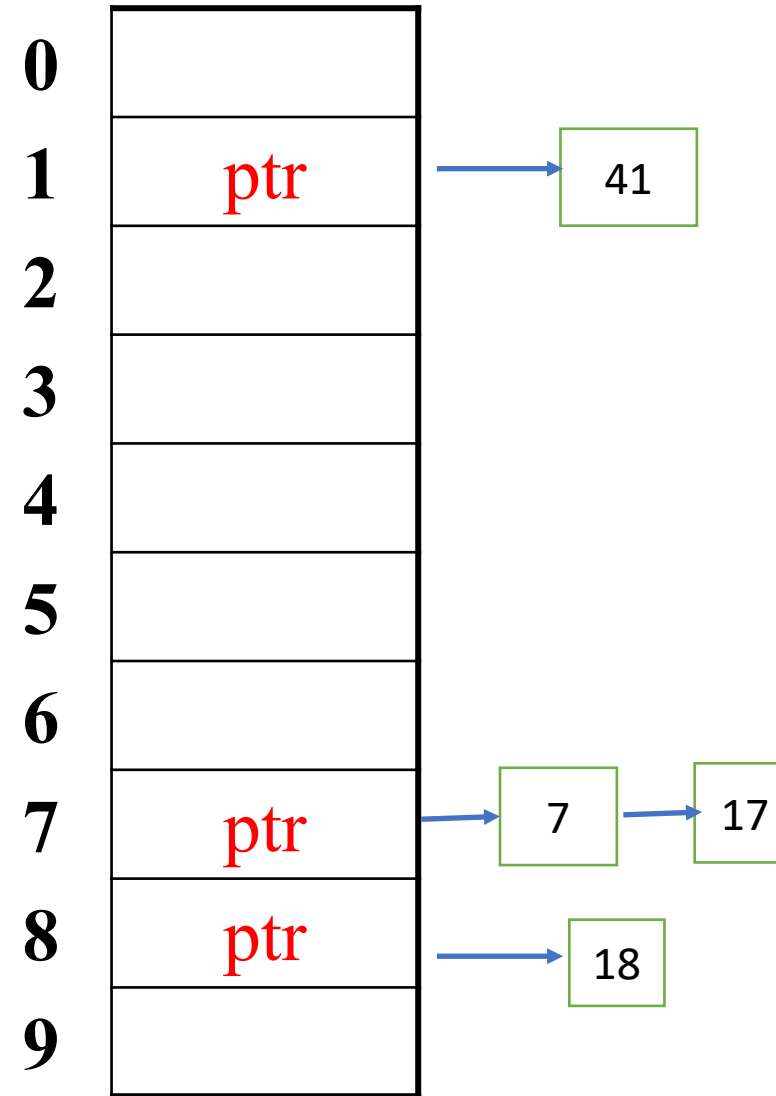
- Assignment 1 will be released today and is, due 2 weeks from now, 31st Jan. **Submission via Canvas.**
- Tomorrow is the deadline to pick project over final exams. No email from you means you are choosing final exam by default.
- **All deadlines are 11:59pm CT on the given day.**

Refresher

- k-universal hashing family H
 - A hash function is k-universal if for any set x_1, x_2, \dots, x_k
 - For h chosen uniformly from H , $h(x_1), h(x_2), \dots, h(x_k)$ are independent random variables.
 - **OR** $\Pr(h(x_1) = h(x_2) = \dots = h(x_k)) \leq \frac{1}{n^{k-1}}$
- We saw 2-universal family. Generally for k-independent family we need polynomial in k
 - $h(x) = (a_k x^k + a_{k-1} x^{k-1} + \dots + a_1 x + a_0) \bmod P \bmod R$
 - Higher independence is harder to achieve from both computation and memory perspective.

Separate Chaining.

- key space = integers
- TableSize = 10
- $h(K) = K \bmod 10$
- **Insert:** 7, 41, 18, 17



What we saw?

- m objects inserted into array of size n
- Expected length of chain $\leq 1 + \frac{m-1}{n}$
 - The quantity $\frac{m}{n}$ is termed as load factor denoted by α
 - Expected addition and search time $1 + \alpha$
- Worst Case: m
- Is this satisfactory? Are there better variants of chaining? How much better?

What is a good running time?

- $\text{Log}(n)$.
- **Natural question:** What is the probability that there exist a chain of size $\geq \log(n)$. More information than expected value!!

Counting and Approximations!

Theorem: For the special case with $m=n$, with probability at least $1-1/n$, the longest list is $O(\ln n / \ln \ln n)$.

Proof:

- Let $X_{i,k}$ = Indicator of {key i hash to slot k }, $\Pr(X_{i,k}=1) = 1/n$
- The probability that a particular slot k receives $>\kappa$ keys is (letting $m=n$)

(**assuming lot if independence**)
$$\binom{n}{\kappa} \frac{1}{n^\kappa} = \binom{n}{\kappa} \frac{1}{n^\kappa} < \frac{1}{\kappa!}$$

- If we choose $\kappa = 3 \ln n / \ln \ln n$, then $\kappa! > n^2$ and $1/\kappa! < 1/n^2$.
- Thus, the probability that any n slots receives $>\kappa$ keys is $< 1/n$.

Better?

- Can we do better?
- Use two hash functions, insert at the location with smaller chain.
- **Assignment:** Using $m=n$ slots, with probability at least $1-1/n$, the longest list is $O(\log \log n)$. Exponentially better!!

Linear Probing

$$F(i) = i$$

- Probe sequence:

0th probe = $h(k) \bmod \text{TableSize}$

1th probe = $(h(k) + 1) \bmod \text{TableSize}$

2th probe = $(h(k) + 2) \bmod \text{TableSize}$

...

i^{th} probe = $(h(k) + i) \bmod \text{TableSize}$

Probing or Open Addressing

0	8
1	109
2	10
3	
4	
5	
6	
7	
8	38
9	19

Insert:

38

19

8

109

10

- **Linear Probing**: after checking spot $h(k)$, try spot $h(k)+1$, if that is full, try $h(k)+2$, then $h(k)+3$, etc.

Major Milestones of Linear Probing

- In 1954, Gene Amdahl, Elaine McGraw, and Arthur Samuel invent linear probing as a subroutine for an assembler.
- In 1962, Don Knuth, in his first ever analysis of an algorithm, proves that linear probing takes expected time $O(1)$ for lookups if the hash function is truly random (n-wise independence).
- In 2006, Anna Pagh et al. proved that 5-independent hash functions give expected constant-time lookups.
- In 2007, Mitzenmacher and Vadhan proved that 2-independence will give expected $O(1)$ -time lookups, assuming there's some measure of randomness in the keys.

Linear Probing in Practice

- In practice, linear probing is one of the fastest general-purpose hashing strategies available.
- This is surprising – it was originally invented in 1954! It's pretty amazing that it still holds up so well.
- Why is this?
 - Low memory overhead: just need an array and a hash function.
 - **Excellent locality**: when collisions occur, we only search in adjacent locations.
 - Great cache performance: a combination of the above two factors

Analyze Expected cost of Linear Probing

- For simplicity, let's assume a load factor of $\alpha = \frac{m}{n} = \frac{1}{3}$.
 - What happens to linear probing of $\alpha \geq 1$
 - Contrast with chaining.
- A region of size m is a consecutive set of m locations in the hash table.
- An element q hashes to region R if $h(q) \in R$, though q may not be placed in R .
- On expectation, a region of size 2^s should have at most $\frac{1}{3} 2^s$ elements hash to it.
- It would be very unlucky if a region had twice as many elements in it as expected. A region of size 2^s is **overloaded** if at least $\frac{2}{3} 2^s$ elements hash to it.

A Provable Fact

- **Theorem:** The probability that the query element q ends up between 2^s and 2^{s+1} steps from its home location is upper-bounded by
 - $c \cdot \Pr[\text{the region of size } 2^s \text{ centered on } h(q) \text{ is overloaded}]$ for some fixed constant c independent of s .
- **Proof:** Set up some cleverly-chosen ranges over the hash table and use the pigeonhole principle. See Thorup's lecture notes.
 - <https://arxiv.org/abs/1509.04549>

Overall we can write the expectation as

- $E(\text{lookup time}) \leq$
 $O(1) \sum_1^{\log(n)} 2^s * \Pr[q \text{ is between } 2^s \text{ and } 2^{s+1} \text{ slot away from } h(q)]$
 $= O(1) \sum_1^{\log(n)} 2^s * \Pr[\text{the region of size } 2^s \text{ centered on } h(q) \text{ is overloaded}]$

If we can bound

$\Pr[\text{the region of size } 2^s \text{ centered on } h(q) \text{ is overloaded}]$

Recall

- A region is a contiguous span of table slots, and we've chosen $\alpha = 1/3$.
- An overloaded region has at least $2/3 \cdot 2^s$ elements in it.
- Let the random variable B_s represent the number of keys that hash into the block of size 2^s centered on $h(q)$ (q is our search query).
- We want to know $\Pr[B_s \geq 2/3 \cdot 2^s]$.
- Assuming our hash functions are at least 2-independent (why?), we have
 - $E[B_s] = 1/3 \cdot 2^s$.
- $\Pr[B_s \geq 2/3 \cdot 2^s]$ is equivalent to $\Pr[B_s \geq 2 \cdot E[B_s]]$.
- Thus, looking up an element takes, on expectation, at least
 - $O(1) \sum_1^{\log(n)} 2^s * \Pr[B_s \geq 2 \cdot E[B_s]]$

Apply Markov's

- No Assumptions!!
- $\Pr[B_s \geq 2 \cdot E[B_s]] \leq \frac{1}{2}$
- $O(1) \sum_1^{\log(n)} 2^s * \Pr[B_s \geq 2 \cdot E[B_s]]$
- $\leq O(1) \sum_1^{\log(n)} 2^{s-1}$
- $O(n)$. **BAAAAAAD**
- (we want $\Pr[B_s \geq 2 \cdot E[B_s]]$ to decay faster than 2^{-s} to get anything interesting)

Concentration Inequalities (Last Class)

- Markov's Inequality

- $\Pr[B_S - E[B_S] \geq k E[B_S]] \leq \frac{1}{k+1}$

- Chebyshev's

- $\Pr[|B_S - E[B_S]| \geq k] \leq \frac{\text{Var}[B_S]}{k^2}$

- Chernoff Bounds.

- $\Pr[B_S - E[B_S] \geq k E[B_S]] \leq 2e^{-\frac{(k^2 E[B_S])}{2+k}}$

A fair coin is tossed 200 times. How likely is it to observe at least 150 heads?

❖ Markov: ≤ 0.6666

❖ Chebyshev: ≤ 0.02

❖ Chernoff: ≤ 0.017

Concentration Inequalities (Last Class)

- Markov's Inequality

- $\Pr[B_s - E[B_s] \geq k E[B_s]] \leq \frac{1}{k+1}$

- Chebyshev's

- $\Pr[|B_s - E[B_s]| \geq k] \leq \frac{\text{Var}[B_s]}{k^2} \quad (\text{Var}[B_s] \text{ not far from } k \times E[B_s]^2)$

- Chernoff Bounds. (B_s is sum of i.i.d variables)

- $\Pr[B_s - E[B_s] \geq k E[B_s]] \leq 2e^{-\frac{(k^2 E[B_s])}{2+k}}$

More Independence → More Resources in Hash Functions

Can we bounds the variance?

- Define indicator variable:

$X_i = 1$ if i^{th} elements maps to block of size 2^s centered at $h(q)$

$$E[X_i] = \frac{2^s}{N}$$

$$B_s = \sum X_i ,$$

$$E[B_s] = E[\sum X_i] = \frac{1}{3} 2^s$$

Variance

- $\text{Var}[B_s] = E[B_s^2] - (E[B_s])^2$
- $E[(\sum X_i)^2] = E[\sum X_i^2 + \sum X_i X_j]$
- $= \sum E[X_i^2] + \sum E[X_i X_j]$
- $= \sum E[X_i] + \sum E[X_i]E[X_j]$
(assuming 3-independence!!)
- What does 3-independence buys us
 $\sum E[X_i]E[X_j] < (E[B_s])^2$ (why? ... do the math)
So $\text{Var}[B_s] \leq E[B_s]$

- $\Pr[|B_s - E[B_s]| \geq E[B_s]] \leq \frac{\text{Var}[B_s]}{E[B_s]^2} \leq \frac{1}{E[B_s]}$
- $\leq 3 \times 2^{-s}$
- $O(1) \sum_1^{\log(n)} 2^s * \Pr[B_s \geq 2 \cdot E[B_s]]$
- $\leq O(1) \sum_1^{\log(n)} 1 = O(\log(n))$

The bound is tight for 3-independence

Going Beyond: 4th Moment Bound

- $\Pr[|B_s - E[B_s]| \geq a] \leq \frac{4^{th} Moment[B_s]}{a^4}$
- **Assignment:** Prove that expected cost of linear probing is $O(1)$ assuming 5-independence using the above 4th moment bounds.

In practice 2-independence suffice?

- In 2007, Mitzenmacher and Vadhan proved that 2-independence will give expected $O(1)$ -time lookups, assuming there's some measure of randomness in the keys.
 - Assignment for Grads: Read this paper.

Cuckoo Hashing

- Worst case of both chaining and probing is $O(n)$. Expected is $O(1)$.
 - Includes both insertion and searching.
- Can we sacrifice insertions for worst case $O(1)$ searching?
 - YES
 - Cuckoo Hashing

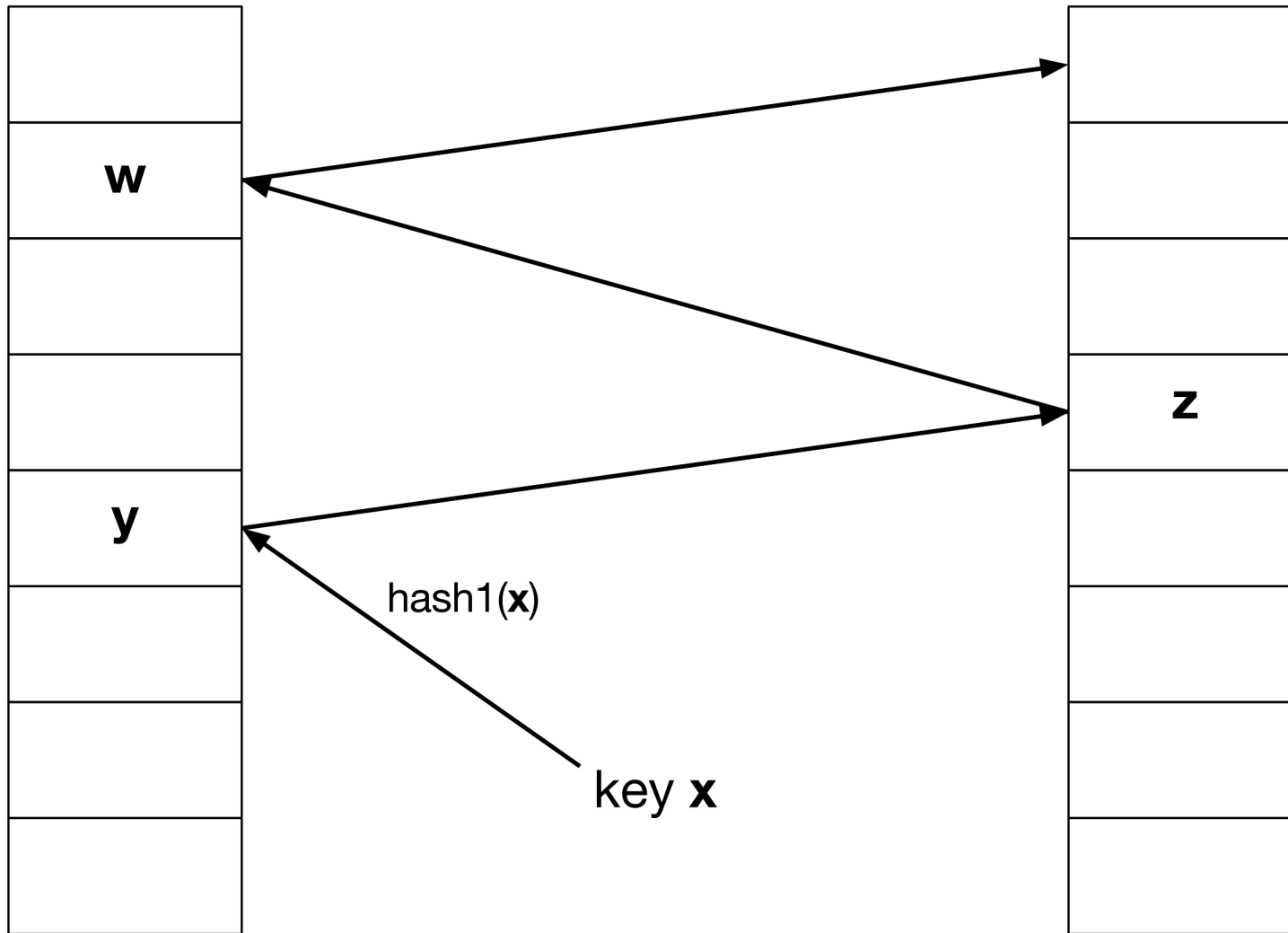


Figure 1. An example “cuckoo path”.

In practice

- In practice Cuckoo hashing is about 20-30% slower than linear probing which is the fastest of the common approaches.