CS 7280: Data Str & Alg Scalable Comp Spring 2025

Lecture 9 — Feb 10, 2025

Prof. Prashant Pandey *Scribe: Yibo Zhao*

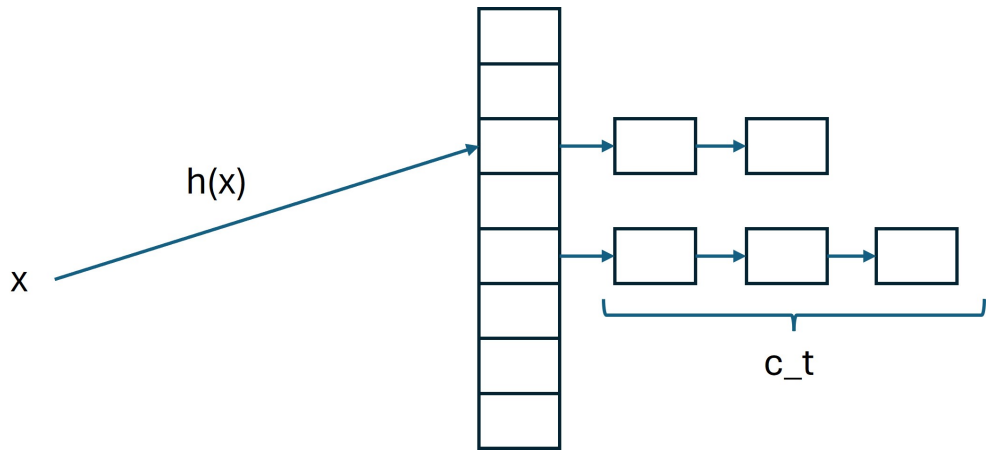
1 Overview

In the last lecture, we went over universal hashing, pair-wise independent hashing, and k-wise independent hashing. We also talked about two kinds of hash tables: chaining and linear probing. They have their problems: chaining hash table has a W.H.P chain length of $\mathcal{O}\left(\frac{\log n}{\log \log n}\right)$. Linear probing, on the other hand, may suffer from clustering. In practice, linear probing is usually preferred because it requires less data transfer between different layers of memory. This method of evaluating the cost is referred as the **IO Model**.

In this lecture, we will cover different algorithms for hash tables:

- perfecting hashing
- Cuckoo Hahing
- 2 Choice Hashing
- Iceberg Hashing

2 Perfect Hashing



To construct a hash table using perfect hashing, we use the 2-level hashing technique. We store each chain c_t as a hash table of size $\theta({c_t}^2)$. After picking a hash function for the first level, we don't want any collisions in the second level of hash tables. We can achieve this by trying hash functions from a universal family of hash functions for the second level. This method only work

for a static set of data. It is guaranteed that a set satisfactory set of hash functions can be found after a constant number of trials. Here is why,

$$\boxed{\begin{array}{c}\textbf{Markov's Inequality}\\ P(X \geq \alpha \mathbb{E}[X]) \leq \frac{1}{\alpha}, \quad \text{for } \alpha > 0.\end{array}} \tag{1}$$

$$\boxed{\begin{array}{c}\textbf{Chernoff Bound}\\ P(c_t > c\mu) \leq \frac{e^{(c-1)M}}{(cM)^{cM}}\end{array}} \tag{2}$$

Let $\boxed{c = 1 + \epsilon}$, this provides exponentially decreasing bounds on the probability of deviation of a random variable from its expected value.

Expected trails to build a collision free hash table,

$$\mathbb{P}\left[\phi \text{ collision in } c_t \text{ tables}\right] \geq \frac{1}{2} = \mathcal{O}(1) \tag{3}$$

$\boxed{M = \mathcal{O}(n)}$, $\mathbb{E}[Space] = \mathcal{O}(n)$

3 Linear Probing

insert(n): put x at first available slot $[h(n) + i] \mod m$ for $i \to 1, 2, 3 \ldots$

For m slots, n items,

- $m \geq (1 + \epsilon)n$, not just $m = \Omega(n)$
- when totally random hashing, $\mathcal{O}\left(\frac{1}{\epsilon^2}\right)$ expected separation, using $(1 + \epsilon)n$ space

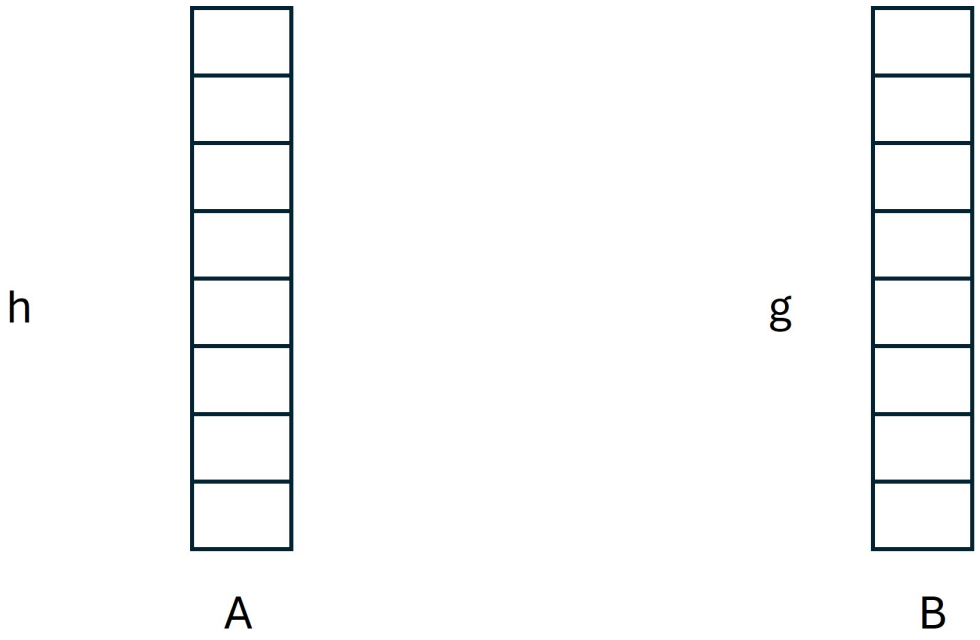
3.1 Quadratic Probing

insert(n): put x at $[h(n) + i^2] \mod m$ for $i \to 1, 2, 3 \ldots$, do quadratic jumps when searching for next available slot

3.2 Double Hashing

Use two hash functions $h_1(k), h_2(k)$, put x at $\tilde{h}(k, i) = h_1(k) + ih_2(k) \mod m$

4 Cuckoo Hashing

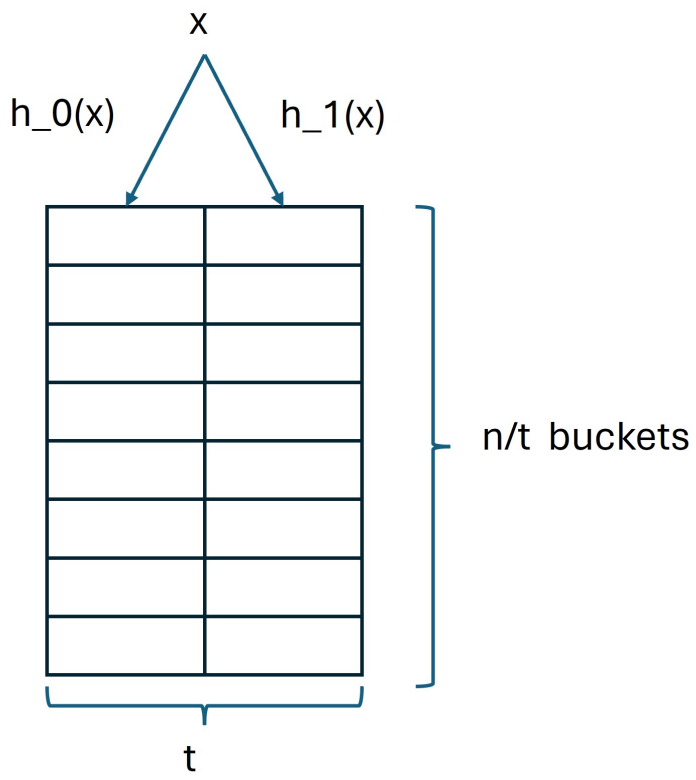


- Use 2 tables of size $(2+\epsilon)n$ space
- Use 2 hash functions h, g
- query(x), query table A using hash function h, if not found, query table B using hash function g. Only 2 deterministic probe.
- insert(x): put in $A[h(x)]$ or $B[g(x)]$. If y is kicked out from $A[h(y)]$, move it to $B[g(y)]$. Repeat if $B[g(y)]$ is also occupied, until you find an empty slot. If encounter a cycle, the hash table is considered full and no more keys can be inserted.
- Using $\mathcal{O}(\log n)$ - wise independent hash functions, will have $\mathcal{O}(1)$ expected updates, $\mathcal{O}\left(\frac{1}{n}\right)$ failure probability when constructing on keys.
- Using 6-wise independent hash function, fail W.H.P even if $m = n^{1+\epsilon}$
- Using simple tabulation hasing, fail with probability $\theta(\frac{1}{n^{1/3}})$

Define $\boxed{\text{load factor} = \frac{\#\text{ of slot used}}{\text{all slots}}}$, putting more keys in each slot of a cuckoo hashing table can increase its load factor

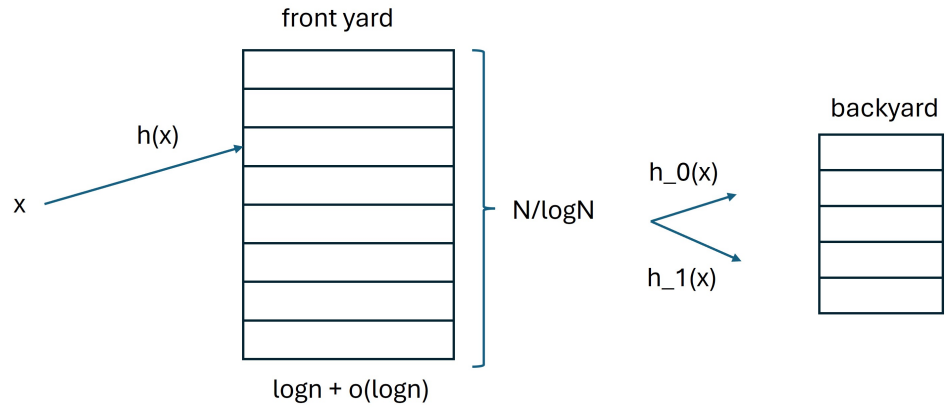
- 1 key per slot $\rightarrow 50\%$
- 2 key per slot $\rightarrow 75\%$
- 4 key per slot $\rightarrow 96\%$

5 2 Choice Hashing



- Use two hash function, adding the inserted item to the less loaded bucket.
- During queries, we search in both buckets
- The load in the fullest bucket $\mathcal{O}(\log \log n)$, smaller than using single hash which is $\mathcal{O}\left(\frac{\log n}{\log \log n}\right)$ W.H.P
- If uses α hashes, the load in the fullest bucket is $\theta(\frac{\log \log n}{\log \alpha})$

6 Iceberg Hashing



Use single hashing for the front yard, put overflow of frontyard to the backyard using 2 choice hashing.

Iceberg Theorem: If you throw N balls in $\frac{N}{\log N}$ bins of size $\log N + o(logN)$, the number of overflows will be $\mathcal{O}\left(\frac{N}{\log N}\right)$

Two Choice Theorem: If you throw N balls into $\frac{N}{\log N}$ bins using two choice hashing, the fullest bin will have $\log N + \log\log N + \mathcal{O}(1)$ balls.

- This does not hold when we delete items.
- Theorem does hold with deletion of average load in $\mathcal{O}(1)$

References

[1] Michael A. Bender, Alex Conway, Martín Farach-Colton, William Kuszmaul, and Guido Tagliavini. Iceberg Hashing: Optimizing Many Hash-Table Criteria at Once. *J. ACM*, 70(6), Article 40 (December 2023), 51 pages. `https://doi.org/10.1145/3625817`.