CS 5968/6968: Data Str & Alg Scalable Comp Spring 2023

Lecture NUMBER — DATE, 2017

Prof. Prashant Pandey *Scribe: Lin Jia*

1 Overview

In the last lecture, we covered four types of hash function:

- Totally Random
- Universal
- K-wise Independent
- Tabulation hashing

And two hash-table structures:

- Chaining
  - W.H.P. bound: $\Theta(\lg n / \lg \lg n)$
  - Amortize bound: $\Theta(1)$
- Perfect hashing
  - Query: Deterministic bound: $O(1)$, 2 probes
  - Drawbacks:
    - have dependent steps(the second hash value is depend on the first hash function's output)
    - do rehashing to prevent collision
    - need large space

In this lecture, we will mainly cover linear hashing, Cuckoo hashing, 2-choice hashing, and front-yard/backyard hashtable.

2 Linear Probing

Let x be the item to be inserted into a hashtable. To insert x, we compute $h(x)$ and place x at that location. If that location is occupied, we will move to the right next slot and check if it's empty or not. If this new location is empty, we can put x here. Otherwise, we need to keep searching for the next available slots by step size 1.

$$\tilde{h}(x, i) = (h(x) + i) \mod m$$

- need $m \geq (1+\epsilon)n$ not just $m = \Omega(n)$
- totally random h gives $1/\epsilon^2$ expected operations and $n(1+\epsilon)$ space.
- $O(\lg n)$-wise independent gives 1 expected
- 5-wise independent gives constant
- 4-wise and pairwise independent gives $\Omega(\lg n)$
- Simple tabulation gives $1/\epsilon^2$ expected
- drawback: size of a cluster is $O(1/\lg n)$, worse then chaining
- Some improvements upon linear probing.
  - Quadratic Probing: Instead of using step size 1, it uses larger step size (i) to avoid clustering.
  - Double Hashing: Double hashing is a collision resolution technique used in hash tables. It works by using two hash functions to compute two different hash values for a given key. The first hash function is used to compute the initial hash value, and the second hash function is used to compute the step size for the probing sequence.

$$\tilde{h}(x,i) = [(h_1(x) + i * h_2(k)] \mod m$$

  - Robin Hood Hashing

3 Cuckoo Hashing

The original Cuckoo Hashing uses two different hash functions to calculate hash values and store items in each table accordingly. Let A and B be the two hash tables and g be the hash function for A and h be the hash function for h. How Cuckoo hashing works like when we have an item x need to be inserted. We check if slot $g(x)$ in table A is occupied or not. If it's empty, we can directly insert x into A at slot $g(x)$. Otherwise, we need to kick out the existing item(let it be y) at $g(x)$ in table A. Then take the kicked-out item y and go to table B and check if slot $h(y)$ is empty or not. If that slot in table B is empty, we can put y into table B at $h(y)$. If it's not empty, we need to repeat the kick-out process and find a new slot for this new kicked-out item. We keep doing the above steps until we find a slot for the item(The item needs to be inserted or the item is kicked out) on our hand.

- 2 tables of size $(2+\epsilon)n$ space
- 2 hash functions $g \to A, h \to B$
- query(x): check $A[g(x)]$ or $B[h(x)]$
- insert(x): put in A or B, kick out y if necessary
  - if kicked y out of A, move y to B, keep repeating until find an empty spot
  - if stuck rehash entire structure

- $(2+\epsilon)n$ space
- 2 deterministic probes for query
- fully random or $\Theta(\lg n)$-wise independent gives $O(1)$ expected update time and $O(1/n)$ failure probability.
- Some improvements upon Cuckoo Hashing.
  - Sigle-table Cuckoo Hashing: Instead of using two separate tables, it still uses two hash functions but one single table to store the items. The load factor is 50%, which means when the table is half full, you will get the dead lock and you need to do resizing.
  - Wide-table Cuckoo hashing: The original Cuckoo hashing table only can store one item per entry. It can be improved by storing more items per entry, like b items. When $b = 4$, the load factor can be up to 96%

4 2-choice Hashing

2-choice Hashing uses two hash functions, adding the inserted item to the shortest list of two corresponding hash values. During queries, we do searching in both lists.

- The length of the longest chain: $\Theta(\lg \lg n)$ in expectation
- Generally, for d hash functions, the longest chain is length: $\Theta(\lg \lg n / \lg d)$

5 Frontyard/Backyard

Frontyard uses single hashing and one $n * b$ table, which can store b items per entry. Backyard will take care of the overflow. When some entry already has b items, the new item that falls into that entry will get inserted into to Backyard table. It usually implements as 2-choice hash table.

- Iceberg [1] implements Frontyard/Backyard with additional linked lists.
- Take-home question: How many items you can insert until you first time run out of b items of one entry? Prove it.

References

[1] Prashant Pandey, Michael A Bender, Alex Conway, Martín Farach-Colton, William Kuszmaul, Guido Tagliavini, and Rob Johnson. Iceberght: High performance pmem hash tables through stability and low associativity. *arXiv preprint arXiv:2210.04068*, 2022.