COMP 480/580 — Probabilistic Algorithms and Data Structure Jan 28, 2020

Lecture 5

Lecturer: Anshumali Shrivastava *Scribe By: Takahiro Mollenkamp*

1 Separate Chaining

Separate chaining is a way of hashing so that each hash table entry points to a linked list of all the items that hash to that entry. Expected chain length is $1 + \frac{m-1}{n}$. The load factor α is $\frac{m-1}{n}$. Expected addition and search time is $1 + \alpha$ but the worst case is m.

Now, we do a probabilistic analysis to improve search time.

Theorem: In the special case m = n, with $p = 1 - \frac{1}{n}$, longest list is $O(\frac{ln(n)}{ln(ln(n))})$

Proof: Let $X_{i,j} = 1$ if key i in hash slot j, 0 otherwise. Then, $p(X_{i,k}) = \frac{1}{n}$. Assuming a lot of independence, probability that a slot j receives $> k$ keys is

$$\binom{n}{k}\frac{1}{m^k} = \binom{n}{k}\frac{1}{m^k} < \frac{1}{k!}$$

.

Choosing $k = 3\frac{ln(k)}{ln(ln(k))}$, $k! > n^2$ so $1/k! < 1/n^2$. Thus, the probability that any n slots receives $> k$ keys is $< 1/n$

If we use two hash functions and choose to insert at the location with shorter chain, with the condition m = n and with probability 1 - 1/n, the longest chain is O(log(log(n))

2 Linear Probing

Linear probing is a hashing scheme where collisions are resolved by continuing to hash cells h(k)+1, h(k)+2 until an empty cell if cell h(k) is occupied during insertions and searches. In practice, linear probing is one of the fastest since it has a low memory overhead and an excellent locality in collisions.

Analysis: assume load factor $\alpha = 1/3$. Let a region R of size m denote a consecutive set of m locations. An element hashes to R if $h(q) \in R$.

E(number of hashes in R of size 2^s) = $\frac{1}{3}2^s$

A region is overloaded if at least $\frac{2}{3}2^s$ elements hash to R.

Theorem: p[element q ends up between 2^s and 2^{s+1} from h(q)] is bounded by

c * p[region of size 2^S centered on h(q) is overloaded]

for some constant c.

Proof: `https://arxiv.org/abs/1509.04549`

$$E[\text{lookup time}] \leq O(1) \sum_{s=1}^{log(n)} 2^s p[\text{q is between } 2^s \text{and} 2^{s+1} \text{slots from h(q)}]$$
$$= O(1) \Sigma_{s=1}^{log(n)} c * p[\text{region of size } 2^s \text{ centered at h(q) is overloaded}]$$

Let B_s represent the number of keys that hash into a block of size 2^s centered on h(q).

E(B_s) = $\frac{1}{3}2^s$ so

$$E(\text{lookup time}) = O(1) \sum_{s=1}^{log(n)} 2^s * P[B_s \geq 2 * E(B_s)]$$

.

Using Markov's inequality, $P[B_S \geq 2 * E(B_S)] \leq 1/2$. As a result,

$$E(\text{lookup time}) = O(1) \sum_{S=1}^{log(n)} 2^{S-1}, \text{ which is } O(n).$$

Now, we aim to bound the variance so that we can use Chebyshev's inequality. Define $X_i = 1$ if i-th element maps to region R of size 2^m, $X_i = 0$ otherwise.

$$E(X_i) = 2^s/N; E(B_s) = E(\Sigma X_i) = 2^s/3$$
$$Var[B_S] = E[B_S^2] - (E[B_S])^2$$

$$E[(\sum X_i)^2] = E[\sum X_i^2 + \sum X_i X_j]$$
$$= \sum E[X_i] + \sum E[X_i] * E[X_j] \text{ assuming independence}$$

Since $\sum E[X_i]E[X_j] < E[B_S]^2$, $Var[B_s] \leq E[B_s]$

Using this result, we use Chebyshev's inequality, letting $a = E[B_s]$. Then,

$$P[|B_s - E[B_s]| \geq E[B_s] \leq \frac{Var[B_s]}{(E[B_s]^2)}$$
$$\leq \frac{1}{(E[B_s])}$$
$$\leq 3 * 2^{-s}$$

Finally,

$$E(\text{lookup time}) = O(1) \sum_{s=1}^{log(n)} 2^s * P[B_s \geq 2 * E(B_s)]$$
$$= O(1) \sum_{S=1}^{log(n)} 1$$
$$= O(log(n))$$

3 Cuckoo Hashing

Cuckoo hashing utilizes two hash tables and corresponding two hash functions to achieve worst case O(1) lookup time. When a new key is inserted, if one of two cells corresponding to each hash function's result is open, the new key goes there. Other wise, we replace the occupying key with new key and move the displaced key to its other hashed cell, continuing the pattern if necessary.

This pattern has the potential of infinite loop if a displaced key eventually displaces a key that was inserted earlier. However, we only need to check two cells for our item so it has a O(1) search time.

In practice, it is 20-30% slower than linear probing.