15:853: Algorithms in the Real World
Fall 2019

Lecture 15

October 24 2019

Lecturer: Rashmi Vinayak *Scribe: Damla Senol*

15.1 Recap From Last Class

A hash table is a dictionary, that allows inserts, deletes, and lookups. It uses a random hash function to map elements of a large universe U into the table, for quick lookups. We want the hash function to have as few collisions as possible. A hash function h from a *universal* hash family H gives us the following property:

Definition 1. *A family H of hash functions from U to $[M]$ is* universal *if for every hash function $h \in H$, and any $x \neq y \in U$,*

$$P[h(x) = h(y)] \leq \frac{1}{M}$$

Universal hash functions help us avoid collisions, since they distribute the values very well across the M slots in the hash table. However, every hash table must handle collisions. To know how good a hash table, we must study how many collisions it can have, and how it handles them.

[slide 17] Let $C(x)$ be the number of elements other than x that map to the same slot as x. Let N be the number of elements that we map into the hash table. $\mathbb{E}[C(x)] = \frac{N-1}{M}$ and $\mathbb{E}[C] = \binom{N}{2}\frac{1}{M}$

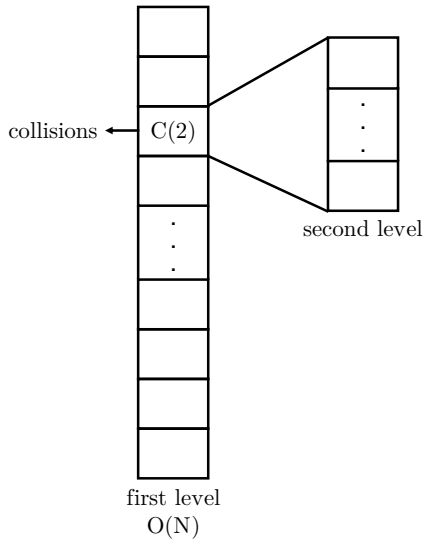
if $M = O(N)$ then expected lookup time is $O(1)$

[slide 18] If $M = O(N^2)$, we can get collision-free hash table after a few iterations. But, N^2 is a huge space requirement.

We now study ways to handle collisions.

15.2 Perfect Hashing

- Two-level hashing, where each slot in the first level does collision-free hashing in the second level.
- C(i) = number of elements mapped to location i in the first level
- For the second level table to be collision-free, it should have $C(i)^2$ (e.g., $C(2)^2$ in the above example) elements so that we can find a collision-free function for it (from Slide 18).



- How much space does this take (Slide 20)? Only $O(N)$!

15.3 k-wise Independence

Definition: A family of H hash functions mapping $U \to [M]$ is k-wise independent if for any k distinct keys, $x_1, x_2, ..., x_k$, and for any k non-necessarily distinct values $\alpha_1, \alpha_2, ..., \alpha_k$, we have $P(h(x_1) = \alpha_1 \cap h(x_2) = \alpha_2 \cap ... \cap h(x_k) = \alpha_k) \leq \frac{1}{M^k}$.

Some useful properties of k-wise independence are on slide 22. Universal is like 1-wise dependence. The larger the k, the stronger the property.

15.4 Some Constructions of 2-wise Independent Families

15.4.1 Construction-1

- $|U| = 2^u$ and $M = 2^m$ and x is a u-binary vector.
- $h(x) = Ax + b$ where A is a $m \times u$ matrix and b is a m-length random vector.
- There are $um + m = (u + 1)m$ random bits $\to O(um)$ random bits
- There are $2^{(u+1)m}$ random hash functions

15.4.2 Construction-2

- Reducing random bits
- $h(x) = Ax + b$ but A is not going to be a matrix where each element is independent
- First row and first column will be independent, but rest will be $A_{i,j} = A_{i-1,j-1}$

- $\begin{bmatrix} A_{11} & A_{12} & A_{13} & ... & ... & ... \\ A_{21} & A_{11} & A_{12} & A_{13} & ... & ... \\ A_{31} & A_{21} & A_{11} & A_{12} & A_{13} & ... \\ ... & ... & ... & ... & ... & ... \end{bmatrix}$

15.4.3 Construction-3

- Using finite fields, description on Slide 25.
- We want to show that H is 2-wise independent.

  Proof:

  $P(h(x_1) = \alpha_1 \cap h(x_2) = \alpha_2) \leq \frac{1}{M^2}$

  $P(h(x_1) = \alpha_1 \cap h(x_2) = \alpha_2) \leq \frac{1}{|U|^2}$ since $h : (U \to U)$

  $ax_1 + b = \alpha_1$

  $ax_2 + b = \alpha_2$

  $P\left[\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \end{pmatrix}^{-1} \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix}\right] = \frac{1}{|U|^2} \to$ truncate to m bits.

15.4.4 Construction-4: k-wise independence

- k-wise independence using finite fields
- Pick k random elements $a_0, a_1, ..., a_{k-1}$
- $h(x) := a_0 + a_1x + a_2x^2 + ... + a_{k-1}x^{k-1}$ over $GF(2^u)$
- Similar proof as before

15.5 Other Hashing Schemes

15.5.1 Simple Tabulation Hashing

- $U = [k]^u$
- key $x = x_1x_2x_3...x_u$
- T is a 2-dimensional array; $u \times k$.
- Each element of T is an m-bit random binary vector (3^{rd} dimension; depth is of length m)
- $h(x) = T[1, x_1] \oplus T[2, x_2] \oplus ... \oplus T[u, x_u]$ ($\oplus$ is XOR)
- **Theorem:** (a) is 3-wise independent, but (b) not 4-wise independent.

15.5.2 Open Addressing

- **Linear probing:** If h(x) is occupied, try the consecutive space (i.e., h(x)+1) until you find an unoccupied one.

  Deletion is problematic.

  Clustering occurs.

  Insertions in O(1) (i.e., constant time) in expectation.

- Other probing options: quadratic, step-size.