CS420+500: Advanced Algorithm Design and Analysis

Lectures: April 3rd and April 5th, 2017

Prof. Will Evans *Scribe: Alexander Julian Martinez*

In this lecture we:

- Discussed Universal Hash Functions;
- Cuckoo Hashing;
- And Cuckoo Graphs.

Reading: Rasmus Pagh's Cuckoo Hashing for Undergraduates

1 Universal Hash Functions

Theorem 1. *Let T be a table of size m. Then for key k,*

$$E[n_{h(k)}] \le \begin{cases} \alpha + 1 & k \in T \\ \alpha & otherwise \end{cases}$$

where there are n items in table T, n_i denotes the number of items in bucket i, and $\alpha = \frac{n}{m}$ is the load factor.

Proof. Let Y_k be the number of keys that are not k which hash to the same slot as k. Then

$$Y_k = \sum_{l \neq k,\ l \in T} X_{kl}$$

where

$$X_{kl} = \begin{cases} 1 & h(k) = h(l) \\ 0 & otherwise \end{cases}$$

Then

$$E\big[Y_k\big] = E\Big[\sum_{l \neq k,\ l \in T} X_{kl}\Big] = \sum_{l \neq k,\ l \in T} E\big[X_{kl}\big] = \sum_{l \neq k,\ l \in T} \frac{1}{m}$$

If $k \notin T$, then $n_{h(k)} = Y_k$ and

$$\big|\{l \in T \mid l \neq k\}\big| = n \implies E[Y_k] = \frac{n}{m}$$

If $k \in T$, then $n_{h(k)} = Y_k + 1$ and

$$\big|\{l \in T \mid l \neq k\}\big| = n - 1 \implies E[Y_k] = \frac{n-1}{m}$$

□

Example: Of a universal set of hash functions

$$h_{a,b}(k) = \big((ak + b) \ mod \ p\big) \ mod \ m$$

where p is a prime bigger than any fixed key. Choose $a \in \{1, 2, \dots, p-1\}$ and $b \in \{0, 1, \dots, p-1\}$ to select a hash function from the set.

2 Cuckoo Hashing

- Find operation takes $O(1)$
- Delete operation takes $O(1)$
- Insert operation takes $O(1)$ amortized

We use two hash functions h_1 and h_2. A key k will either be stored in $h_1(k)$ or in $h_2(k)$. Each slot in table contains just one key. Below is psuedocode for the insert operation:

```
insert(k)
    pos = h_1(k)
    repeat n times {
            b = T[pos]
            T[pos] = k
            if b == NULL return
            if pos = h_1(b) then
                    pos = h_2(b)
            else pos = h_1(b)
            k = b
   }
   rehash();
   insert(k);
```

3 Cuckoo Graphs

Definition 2. *(Cuckoo Graph)*
A ***cuckoo graph*** *for n items is a graph $G = (V, E)$ where*

$$V = \{0, 1, \dots, m-1\}$$
$$E = \{\big(h_1(k), h_2(k)\big) \mid k \in T\}$$

We assume that $h_1(k)$ and $h_2(k)$ are uniform and independent random edges.

Note: An insertion will succeed if there is no cycle in the cuckoo graph.

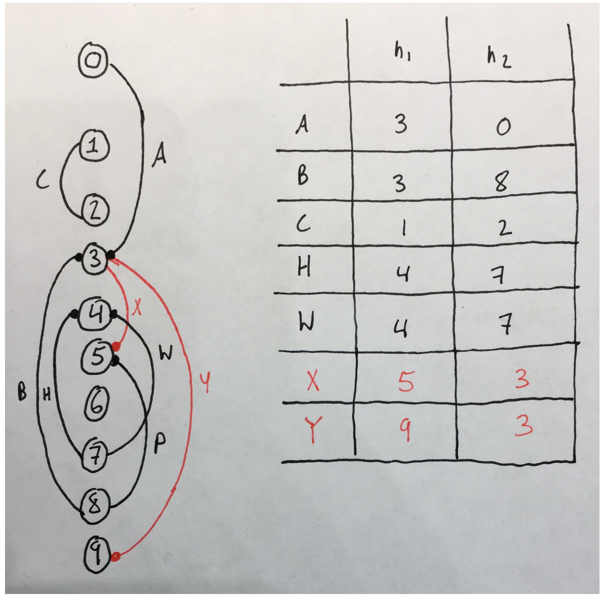

Figure 1: Cuckoo Graph

First we conside paths,

Lemma 3. *For some constant $c > 1$ and $m \geq 2cn$, then*

$$Pr\Big[cuckoo\ graph\ has\ path\ of\ length\ l\ from\ i\ to\ j\Big] \leq \frac{1}{mc^l}$$

Proof. We proceed by induction on l.
<u>Base Case:</u> Let $l = 1$. Then edge (i, j) exists with probability $\leq n \cdot \frac{1}{m^2} = n \cdot \frac{2}{2cn \cdot m} = \frac{1}{c^1 \cdot m}$.
For $l > 1$, the shortest path from i to j has length l if and only if there exists p and

1. there exists a shortest path from i to p of length $l - 1$.
   <u>*note*:</u> occurs with probability $\leq \frac{1}{m \cdot c^{l-1}}$ by induction
2. there exists the edge (p, j)
   <u>*note:*</u> occurs with probability $\leq \frac{1}{m \cdot c}$

Together we obtains

$$probability\ \leq \frac{1}{m^2 \cdot c^l}$$

By summing over possible nodes p we get

$$probability\ \leq \frac{m \cdot 1}{m^2 \cdot c^l} = \frac{1}{m \cdot c^l}$$

Probability that k and k' hash to the same path ("bucket") of cuckoo graph is probability of a path from $h_1(k)$ or $h_2(k)$ to $h_1(k')$ or $h_2(k')$ which is

$$\leq 4 \cdot \sum_{l=1}^{\infty} \frac{1}{mc^l} = \frac{4}{m} \cdot \frac{1}{c-1} = O\Big(\frac{1}{m}\Big)$$

□

Rehash: means choose new hash function and rehash all keys. Probability that a rehash occurs is

$$\begin{aligned}
&\leq Pr\left[hashing\ creates\ cuckoo\ graph\ with\ a\ cycle\right] \\
&\leq \sum_{i=1}^{m} Pr\left[\underline{cycle}\ involving\ i\right] \\
&\leq \sum_{i=1}^{m} \sum_{l=1}^{\infty} \frac{1}{mc^l} \\
&\leq \frac{1}{2}\ for\ c \geq 3
\end{aligned}$$

where a cycle is a pth from i to i. Then the expected number of rehashes is less than or equal to 2.