Lecture 2 Hashing

static perfect hashing via FKS, Bloom filters,

The Problem: Membership/Dictionary: maintain a set S of n items from a universe U under:
- query(x): $x \in S$? (+ information associated with x)
- insert(x) (dynamic)
- delete(x) (dynamic)

The Solution: A hash function $h : U \to [m]$ for some positive integer $m < |U|$.

- maintain a table $T[1 \dots m]$ of linked lists (chains)
- insert(x): add x to $T[h(x)]$.
- query(x): scan $T[h(x)]$.
- by pigeon hole principle $\forall h$ there exist $x \neq y$ s.t $h(x) = h(y) \Rightarrow$ our goal is short chains.

Perfect Hashing: every chains is of length $O(1)$. Constructed using hash functions that are good in expectation.

Theorem 1. *If $m > n$ and h is selected uniformly from all hash functions then insert/delete/query take $O(1)$ expected time.*

However, a random hash function requires $|U| \lg m$ bits to represent $\Rightarrow$ infeasible.

Universal Hashing:

weak universal hashing is enough to obtain $O(1)$ expected time per operation.

Definition 2. *A set $\mathcal{H}$ of hash functions is a weak universal family if for all $x, y \in U$, $x \neq y$,*

$$\Pr_{h \leftarrow \mathcal{H}}[h(x) = h(y)] = \frac{O(1)}{m}.$$

- Why is weak universal enough?
Pick m so that $\frac{n}{m} = O(1)$, and randomly pick $h \in \mathcal{H}$. For any $x \in U$ let $I_y = 1$ iff $h(x) = h(y)$.

$$E[\text{x's chain length}] = \underbrace{E\left[\sum_{y \in S} I_y\right] = \sum_{y \in S} E[I_y]}_{\text{linearity of expectation}} = 1 + \sum_{y \neq x} \Pr[h(x) = h(y)] \leq 1 + n \cdot \frac{O(1)}{m} = O(1)$$

Worst-case Guarantees in Static Hashing:

-Universal hashing gives good performance only in expectation $\Rightarrow$ vulnerable to an adversary.

Say $\mathcal{H}$ is a family of hash functions, and the expected length of the longest chain is $O(1)$.

$\Rightarrow$ We can construct a static hash table with $O(1)$ worst-case query time:

- pick a random $h \in \mathcal{H}$, hash every $x \in S$ (in $O(n)$ time).
- if longest-chain $\leq 2\cdot$expected-length then stop.
- otherwise, pick a new h and start over.

Pr(bad hash function) $\leq \frac{1}{2} \Rightarrow O(1)$ trials, $O(n)$ expected construction time.
Why? Markov: $Pr(X \geq a) \leq \frac{E(X)}{a}$

FKS - Static Hashing (Fredman, Komlós, Szemerédi [1])

- Construct static hash table with no collisions in expected $O(n)$ time, $O(n)$ worst-case space, and $O(1)$ worst-case query time.
- Requires a weak universal family $\mathcal{H}$
- Easy to implement.

First attempt: If $m = \Omega(n^2)$ and we randomly pick $h \in \mathcal{H}$ then

$$E[\text{number of collisions}] = \sum_{x,y \in S,\ x \neq y} \Pr[h(x) = h(y)] = \binom{n}{2} \cdot \frac{1}{m} \leq \frac{1}{2}$$

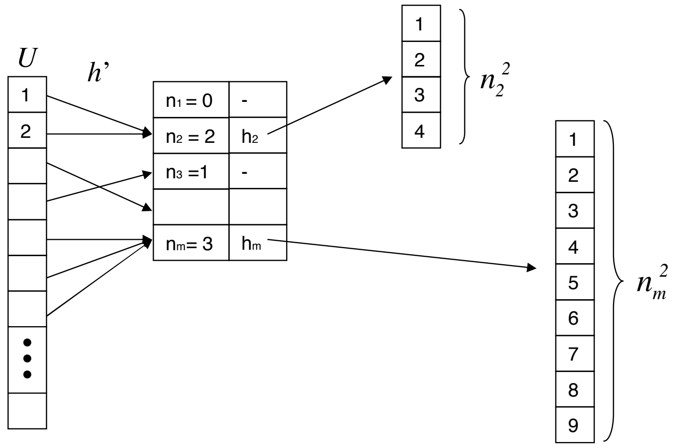
$\Rightarrow$ After expected $O(1)$ trials, we get a collision-free hash function (total time is $O(m) = O(n^2)$).

Second attempt: If $m = n$, the same calculation yields

$$E[\text{number of collisions}] = \binom{n}{2} \cdot \frac{1}{n} = O(n)$$

$\Rightarrow$ After expected $O(1)$ trials, we find a function h' that produces $O(n)$ collisions (total time is $O(n)$).

FKS: Use h' to hash into n buckets, then use h_i's to hash a bucket of size n_i to n_i^2 locations.



Let $n_i = |\{x \in S \mid h'(x) = i\}|$.

(I) The number of collisions is $\sum_{i\in[m]} \binom{n_i}{2} = O(n)$ because we choose h' so. Thus,

$$\sum_{i\in[m]} n_i^2 = O\left(\sum_{i\in[m]} \binom{n_i}{2}\right) = O(n).$$

(II) We can hash n_i elements into a table of size n_i^2 without any collisions in expected $O(n_i^2)$ time. $\Rightarrow$
- The construction takes $O(n) + O(n_1^2) + \ldots + O(n_m^2) = O(n)$ time in expectation
- Worst-case $O(n)$ space.
- Worst-case $O(1)$ query time (two hashes).

Bloom Filters (B. Bloom 1970 [7])

Suppose we want to store a set of strings $S = \{s_1, s_2, \ldots, s_n\}$ where each string requires N bits to store. We would like membership queries of the form "is $x \in S$?"

Use hashing! Hash the n strings into m slots using hash function h . (for every $j \in \{1, 2, \ldots, m\}$ $Pr[h(s_i) = j] = 1/m$

We could store in every slot a linked list of the strings that are hashed to it. The space complexity $O(mN)$.

Suppose that N is huge (i.e. string is an entire book or even a DNA sequence). Store a single bit for every slot. If there is one or more s_i that is hashed to j then we set the j^{th} bit to 1, otherwise set it to 0. When we want to test whether $s \in S$, we say yes if and only if the bit $h(s)$ is equal to 1.

If $s \in S$ then we always say $s \in S$. Why? If $s \notin S$ then we might (mistakenly) say $s \in S$ (false positive). Why? What is the probability that this happens?

$Pr[h(s) = 1] = 1 - Pr[h(s) = 0] = 1 - Pr[h(s_i) \neq h(s) \text{ for every } s_i] =$

$= 1 - Pr[h(s_1) \neq h(s)] \cdot Pr[h(s_2) \neq h(s)] \cdot \ldots \cdot Pr[h(s_n) \neq h(s)] = 1 - (1 - 1/m)^n$

What if we use k uniform and independent hash functions $h_1, h_2, \ldots, h_k$ s.t for every i, j $Pr[h_i(s_j) = 1/m]$? The false positive probability is then reduced to

$Pr[h_1(s) = 1] \cdot Pr[h_2(s) = 1] \cdots Pr[h_k(s) = 1] =$
$(1 - Pr[h_1(s) = 0]) \cdot (1 - Pr[h_2(s) = 0]) \cdots (1 - Pr[h_k(s) = 0]) =$
$(1 - Pr[h_1(s) = 0])^k =$
$\left(1 - (1 - 1/m)^{kn}\right)^k$

It is kn and not just n as before because every one of the n strings can not be hashed with any one of the k hash functions to $h_1(s)$.

Recall that $(1 - 1/x)^y$ is roughly equal to $e^{-y/x}$. Choosing $k = \ln 2 \cdot m/n$ minimizes prob. to be $0.6185^{m/n}$. Decreases when m (space) increases.

Cuckoo - Dynamic Hashing (Pagh and Rodler 2001 [5])

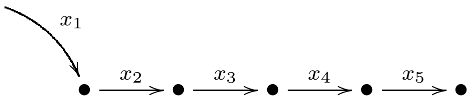
- $O(1)$ expected time for insert
- $O(1)$ worst-case time for queries/deletes.
- Requires two $O(\lg n)$-independent hash functions, h_1 and h_2. (OPEN: same bound using only $O(1)$-independent hash family)
- $m > 2n$ (we will use $m = 4n$).
- Invariant: x is either at $T[h_1(x)]$ or at $T[h_2(x)]$ $\Rightarrow$ query/delete takes worst-case two probes.

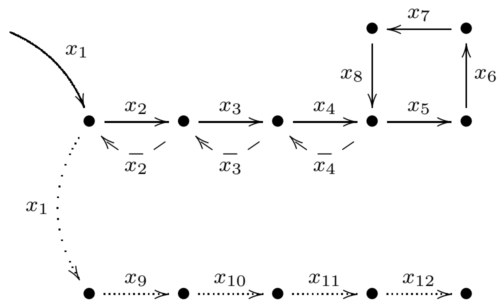
Insertion:

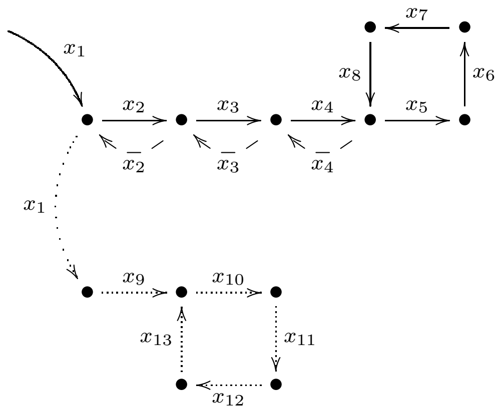
1. Compute $h_1(x)$,
2. If $T[h_1(x)]$ is empty, we put x there, and we are done.
   Otherwise, if $y \in T[h_1(x)]$, we evict y and put x in $T[h_1(x)]$.
3. We find a new spot for y by looking at $T[h_1(y)]$ or $T[h_2(y)]$ (the one that is not occupy by x).
4. Repeat this process. After $6 \lg n$ steps stop and rehash.

Let $x_1, x_2, \ldots, x_t$ be the items that are evicted during the process.
- Cuckoo graph $G = (V, E)$, where $V = [m]$ and $(h_1(x), h_2(x)) \in E$ for all $x \in U$. Insertion is one of three possible walks on G:

Key observation: our functions are $O(\lg n)$-independent so we can treat them as truly random functions.







- **No cycle:** $\Pr[1^{st} \text{ eviction}] = \Pr[T[h_1(x_1)] \text{ is occupied }] \leq$

$$\underbrace{\sum_{x \in S, x \neq x_1} (\Pr[h_1(x) = h_1(x_1)] + \Pr[h_2(x) = h_1(x_1)]) < n\frac{2}{m}}_{\text{union bound}} = \frac{2n}{4n} = \frac{1}{2}.$$

  By same reasoning, $\Pr[2^{nd} \text{ eviction}] \leq 2^{-2}$, and $\Pr[t^{th} \text{ eviction}] \leq 2^{-t} \Rightarrow$ the expected running time of this case is $\leq \sum_{t=1}^{\infty} t \cdot 2^{-t} = O(1)$.

  Also, $\Pr[\text{rehash}] \leq 2^{-6 \lg n} \leq \frac{1}{n^2}$ $(*)$

- **One cycle:** One of the path parts (solid, dashed or dotted) is at least $t/3$ long.
  $\Rightarrow$ the expected running time of this case is $\leq \sum_{t=1}^{\infty} t \cdot 2^{-t/3} = O(1)$.

  Also, $\Pr[\text{rehash}] \leq 2^{-(6 \lg n)/3} = \frac{1}{n^2}$ $(*)$

- **Two cycles:** Counting argument. How many two-cycle configurations are there?
  - The first item in the sequence is x_1.
  - At most n^{t-1} choices of other items in the sequence.

- At most t choices for where the first loop occurs, t choices for where this loop returns, and t choices for when the second loop occurs.
- We also have to pick $t-1$ hash values to associate with the items.

$\Rightarrow$ At most $t^3 n^{t-1}(4n)^{t-1}$ configurations.

The probability that a specific configuration occurs is $2^t(4n)^{-2t}$. Why?

$\Rightarrow$ The probability that some two-cycle configuration occurs is at most

$$\frac{t^3 n^{t-1}(4n)^{t-1}2^t}{(4n)^{2t}} = \frac{t^3}{4n^2 2^t}$$

$\Rightarrow$ The probability that a two-cycle occurs at all is at most

$$\sum_{t=2}^{\infty} \frac{t^3}{4n^2 2^t} = \frac{1}{4n^2}\sum_{t=2}^{\infty}\frac{t^3}{2^t} = \frac{1}{2n^2}\cdot O(1) = O\left(\frac{1}{n^2}\right) \ (*)$$

By $(*)$'s, Pr[insertion causes rehash] $\leq O(1/n^2)$.
$\Rightarrow$ Pr[n insertions cause rehash] $\leq O(1/n)$.
$\Rightarrow$ Rehashing (n insertions) succeeds with prob. $1 - O(1/n)$, so after constant number of trials.

- A trial takes $n \cdot O(1) + \underbrace{O(\lg n)}_{\text{last insertion}} = O(n)$ time in expectation.

$\Rightarrow$ Rehashing takes $O(n)$ time in expectation.
$\Rightarrow$ The expected running time of an insertion is $O(1) + O(1/n^2)\cdot O(n) = O(1) + O(1/n) = O(1)$.

References

[1] M. Fredman, J. Komlós, E. Szemerédi, *Storing a Sparse Table with $O(1)$ Worst Case Access Time*, Journal of the ACM, 31(3):538-544, 1984.

[2] G. Gonnet, *Expected Length of the Longest Probe Sequence in Hash Code Searching*, Journal of the ACM, 28(2):289-304, 1981.

[3] M. Mitzenmacher, *The Power of Two Choices in Randomized Load Balancing*, Ph.D. Thesis 1996.

[4] A. Ostlin, R. Pagh, *Uniform hashing in constant time and linear space*, 35th STOC, p. 622-628, 2003.

[5] R. Pagh, F. Rodler, *Cuckoo Hashing*, Journal of Algorithms, 51(2004), p. 122-144.

[6] A. Siegel, *On universal classes of fast hash functions, their time-space tradeoff, and their applications,* 30th FOCS, p. 20-25, Oct. 1989.

[7] B. Bloom *Space/Time Trade-offs in Hash Coding with Allowable Errors*, Communications of the ACM, 13(7):422–426, 1970.