HASHING

Hashing with Chaining

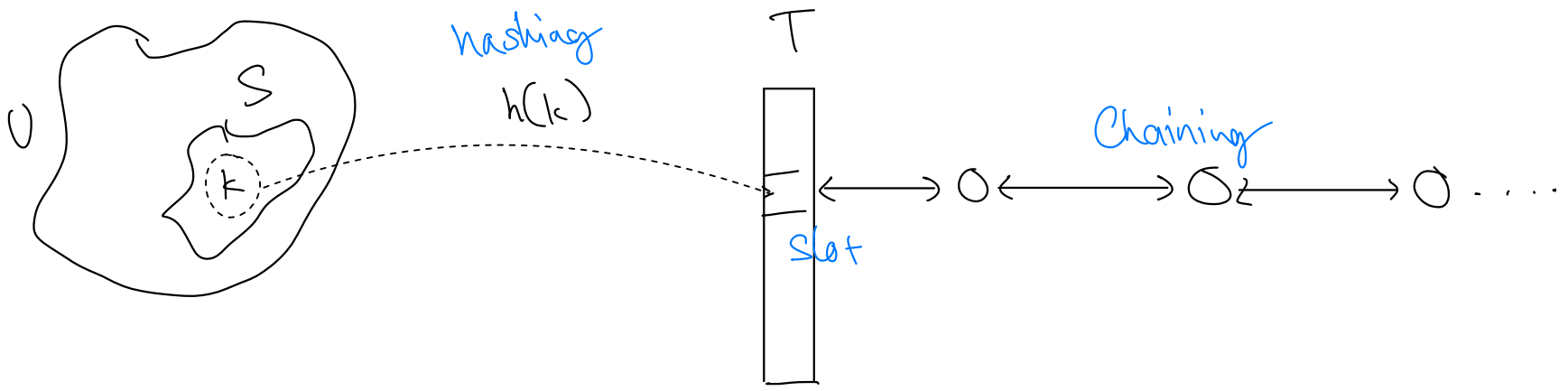
Universal set : $U = \{ \dots \dots \dots \}^{\infty}$

Key set $S \subseteq U$, stored in a hash table $T[0 \dots m-1]$

Hash func : $h(k) : U \rightarrow \{0, \dots, m-1\}$

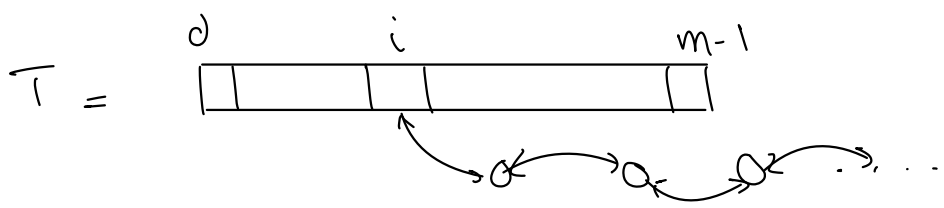
where $k \in S$

$\Rightarrow$ key k is hashed into slot $h(k)$



Collision: Many keys k in S hashed into the same slot

↳ **Solution:** (Doubly) linked-list



Load factor: average number of keys per slot

Complexity:

- Insert and Delete : $\Theta(1)$
- Search : $\Theta(n)$ if chain has size $\Theta(n)$

Challenge: Can we design a good hashing strategy that distributes keys evenly across the slots ?

↳ No. If you hash a set of n keys, assuming $h : U \rightarrow \{0 \dots m-1\}$ and $|U| \geq m \cdot n$, then exist n keys that can be hashed to the same slot

Simple Uniform Hashing

Simple Uniform Hashing Properties

- Any element in U is equally likely to hash into any m slots in T

$\Leftrightarrow \quad P(h(k) = q) = \frac{1}{m} \quad \forall$ key $k \in S \subseteq U$, $\forall$ slot q

Lemma

Hashing $S \subseteq U$, $|S| = n$, onto hashtable $T[0 \dots m-1]$ using function $h: U \to \{0 \dots m-1\}$ with Simple Uniform Hashing

Let L_k be the number of keys in slot $h(k)$ in T

$$E[L_k] < \alpha + 1 \qquad : \alpha = n \cdot 1/m$$

Proof:

Let X be number of keys in slot $h(k)$ that is not k.

- Case 1: k is not in T

$\Leftrightarrow X = L_k$
Since $L_k \sim Bin(n, \frac{1}{m})$ $\Rightarrow X \sim Bin(n, \frac{1}{m})$

$\Rightarrow E[X] = n \cdot \frac{1}{m} = \alpha$
$< \alpha + 1$

- Case 2: k is in T

$\Leftrightarrow L_k = X + 1$
Since $L_k \sim Bin(n, \frac{1}{m})$ $\Rightarrow X \sim Bin(n-1, \frac{1}{m})$

$\Rightarrow E[X] = (n-1) \cdot \frac{1}{m} = \alpha - 1$
$< \alpha + 1$

$\Rightarrow$ Proven

Collary:

The expected time to search for a key under Simple Uniform Hashing is: $\Theta(1 + \alpha) \qquad \alpha = n/m$

Problem: In practice, we cant implement Simple Uniform Hashing

↳ Solution:

Universal Hashing, provably good expected search time for hashing with chaining

Universal Hashing

Randomized Algorithm : an extra call to Random(a, b), which will return a number in range [a, b] with (almost) equal probability

$$\Rightarrow \quad E[\text{running\_time}] = \frac{1}{n} \sum^{n} \text{running-time}$$

n: outcomes generated by Random(a, b)

Definition (Universal Hashing)

Let $\mathcal{H} = \{h_1, h_2, \dots h_r\}$ be a set of hash functions size r.
If we can constract set $\mathcal{H}$ s.t. :

$$\left| \{ h_i : \underbrace{h_i(x) = h_i(y)}_{\text{collision}} \} \right| \leq r/m \quad \forall \text{ distinct } x, y$$

Then : $P[h_i(x) = h_i(y)] \leq r/m \cdot 1/r = \frac{1}{m}$

↳ In words : If we carefully craft a set of $\mathcal{H}$ hash functions s.t. the number of functions resulted in collision is lesser than r/m, then the probability of collision is $1/m$ for any $h \in \mathcal{H}$

Lemma :

Let h be random hash function from universal family $\mathcal{H}$ be used to hash n keys into table $T[0 \dots m-1]$
Let L_k be number of keys in slot $h(k)$ in T, $\forall k \in U$:

$$E[L_k] < \alpha + 1$$

Proof: (that $E[L_k] < \alpha + 1 \quad \forall \quad h \in \mathcal{H}$)

Let X_{kl} be i.r.v. s.t. $X_{kl} = 1$ if $h(k) = h(l)$ (collision)
$X_{kl} = 0$ otherwise ($h(k) \neq h(l)$)

$Y = \sum_{l \neq k} X_{kl}$: number of keys in slot $h(k)$ in T that isn't k

- Case 1: k not in T

$\Rightarrow Y = L_k$

$\Rightarrow E[L_k] = E[Y] = E\left[\sum_{l \neq k} X_{kl}\right]$

$= \sum_{l \neq k} E[X_{kl}]$ (linearity of expectation)

$= \sum_{l \neq k} \left(0 \cdot P(X_{kl} = 0) + 1 \cdot P(X_{kl} = 1) \right)$

$= \sum_{l \neq k} P(X_{kl} = 1)$ (collision probability)

$\leq \sum_{l \neq k} \frac{1}{m} = n \cdot \frac{1}{m}$

$= \alpha$

- Case 2: k is in T

$\Rightarrow L_k = Y + 1$

$\Rightarrow E[L_k] = E[Y] + 1$

Since $E[Y] \leq (n-1)\frac{1}{m}$ (similar reasoning to Case 1)

$< \alpha$

$\Rightarrow E[L_k] < \alpha + 1$

$\Rightarrow$ Proven

Universal Family $\mathcal{H}_{p,m}$

How do we actually construct such set $\mathcal{H}$ satisfies the properties of Universal Hashing ?

Example 1: Using prime number

- Choose a prime p s.t. $|U| = p$ (or $p > |U|$)
- Define hash function $h_{a,b}$ as:

$$h_{a,b}(k) = ((a \cdot k + b) \bmod p) \bmod m \quad (1)$$

value between $[1, p]$

where $a, b \in \mathbb{N}$, $1 \leq a < p$, $0 \leq b < p$

$\Rightarrow$ Set $\mathcal{H}_{p,m}$ is collection of all hash function defined by equation (1)

$|\mathcal{H}_{p,m}| = p \cdot (p-1)$ $\rightarrow$ very large

Proof: $H_{p,m}$ is Universal Family

For any distinct keys k, l.

let $H^* = \{ h_{a,b} : h_{a,b}(k) = h_{a,b}(l) \}$

Show:

$$|H^*| \leq \frac{p(p-1)}{m} \rightarrow r = p(p-1) = |H_{p,m}|$$

$$\forall \; 1 \leq a < p, \quad 0 \leq b < p$$

Fix a pair (a, b)

$$\text{let } \begin{cases} r = (ak + b) \bmod p \\ s = (al + b) \bmod p \end{cases} \Rightarrow \begin{cases} h_{a,b}(k) = r \bmod m \\ h_{a,b}(l) = s \bmod m \end{cases}$$

(a) Show bijection between pairs (a, b) and (r, s)

By definition, for distinct keys k and l:

$\forall$ pair (r, s), $\exists$ a pair (a, b) that maps to it through the function $\begin{cases} r = (ak + b) \bmod p \\ s = (al + b) \bmod p \end{cases}$

(b) Using the bijection to show "Number of pairs (r, s) for which $h_{a,b}(k) = h_{a,b}(l)$ is at most $\frac{p(p-1)}{m}$"

By definition, we know:

$$\begin{cases} h_{a,b}(k) = r \bmod m \\ h_{a,b}(l) = s \bmod m \end{cases}$$

$$\Rightarrow P[h_{a,b}(k) = h_{a,b}(l)] = P[r \bmod m = s \bmod m]$$

linearity $\Rightarrow$ $$\sum_{H} P[h_{a,b}(k) = h_{a,b}(l)] = \sum_{H} P[r \bmod m = s \bmod m]$$

$$= \frac{\#(r, s)}{p}$$

$$x \bmod m = i \bmod m$$

$$(x < p) \qquad (0 \leq i < m)$$

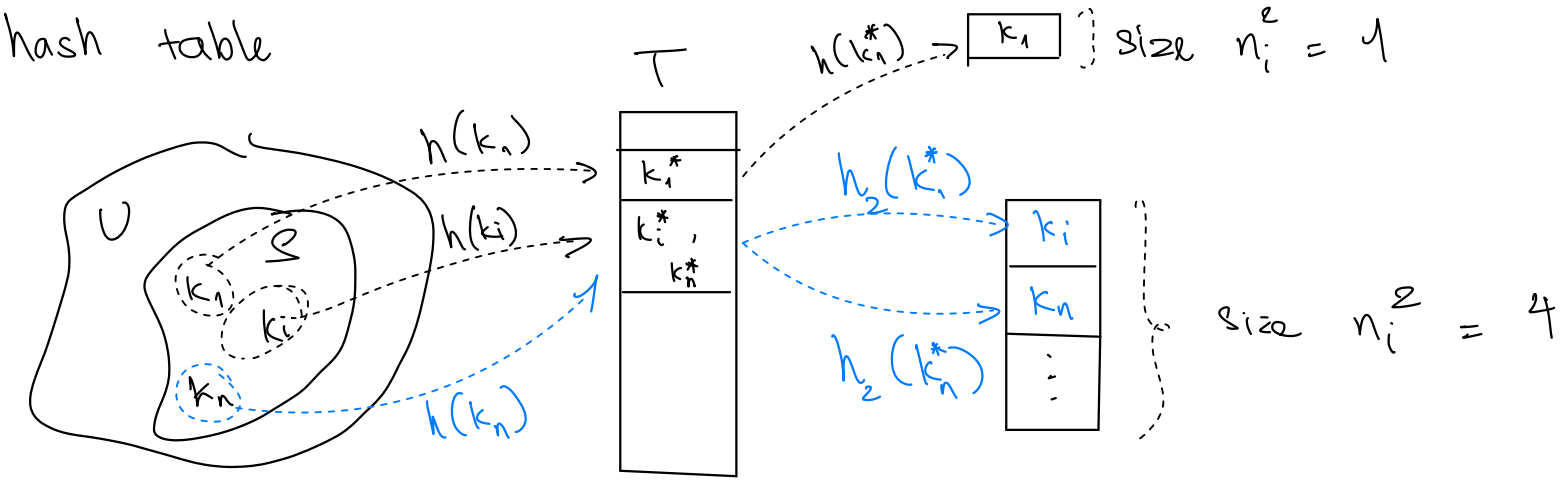
$$\Rightarrow x \in X : |X| \leq p/m$$

Summary: Universal Hashing

- Formulates hashing as a randomized algorithm
  - a) Makes one initial call to Random(1, r) to pick a random hash function h from the universal family
  - b) Hashes the set S using hash function h
- Achieves constant expected time for search (using $m = \theta(n)$)
  - Insert and Delete take $O(1)$

Perfect Hashing

main idea : instead of chaining when collision happens, use another hash table



Pseudo:

Step 1: Pick random hash function h from $H_{p,n}$ to map keys in S to primary table $T[0 \dots n-1]$

Step 2: Let n_i be number of keys hashed into i^{th} slot.
If $\sum_{i}^{n} n_i^2 > 4n$, repeat steps 1-2 with new random hash function until the condition holds.

Step 3: For $i := 0 \rightarrow n-1$:

- Pick random hash function h_i from universal class H_{p,n_i^2}
- Use h_i hash n_i keys already in slot i^{th} of the primary table, into a secondary table of size n_i^2
- If there is collision, repeat Step 3 for i with new random hash function h_i until the keys are hashed into the i^{th} secondary hash table with no collisions.

Claim 1: Expected number of collisions

If n keys are hashed onto table of size m :

$$E[\text{total collisions}] = C(n,2) \cdot \frac{1}{m}$$

Proof:

Let { X be r.v. the number of collisions
{ X_{ij} be i.r.v. event that keys i^{th} and j^{th} collide

Then:

$$E[X] = E\left[\sum_{j=i+1}^{r} \sum_{i=1}^{r} X_{ij}\right] = \sum_{j=i+1}^{r} \sum_{i=1}^{r} E[X_{ij}]$$

$$\leq C(n,2) \cdot 1/m$$

Claim 2: Expected total # of keys squared (primary table)

Let n_i be the keys to be hashed by random function h_i

$$E\left[\sum_{i=0}^{n-1} n_i^2\right] < 2n \quad \text{and} \quad P\left[\sum_{i=0}^{n-1} n_i^2 > 4n\right] < \frac{1}{2}$$

↳ **Proof:**

- Using this fact: $a^2 = a + 2 \cdot C(a,2) \quad \forall\ a \geq 0,\ a \in \mathbb{N}$

$$\left(C(a,2) = \frac{a \cdot (a-1)}{2} \right)$$

- $$E\left[\sum_{i=0}^{n-1} n_i^2\right] = E\left[\sum_{i=0}^{n-1} \left(n_i + 2 \cdot C(n_i,2) \right)\right] \quad \text{(above fact)}$$

$$= E\left[\sum^{n} n_i\right] + 2\, E\left[\sum^{n} C(n_i,2)\right] \quad \text{(linearity)}$$

$$= n + 2 \cdot E[\text{total collisions}]$$

→ # collisions in i^{th} slot

$$\leq n + 2 \cdot C(n,2) \cdot \frac{1}{n}$$

$$\leq n + 2 \cdot \frac{n(n-1)}{2} \cdot \frac{1}{n}$$

$$\leq n + n - 1$$

$$< 2n$$

- $$P\left[\sum^{n} n_i^2 > 4n\right] \leq \frac{1}{4n} E\left[\sum^{n} n_i^2\right] \quad \text{(Markov inequality)}$$

$$< \frac{1}{4n} 2n \quad \text{(prev. above)}$$

$$= \frac{1}{2}$$

⇒ **Recall:** In Step 2, to hash n_i keys into slot i^{th} of primary table, we need this condition to hold $\sum^{n} n_i^2 \leq 4n$.

From Claim 2, $P\left[\sum^{n} n_i^2 > 4n\right] < 1/2$

⇒ Number of iterations taken to run steps 1-2 before we get set of keys n_i that satisfy the condition is 2

Claim 3: Probability collision among n_i keys (secondary table)

If n_i keys are hashed onto secondary table of size n_i^2, then:

$$P[\text{any collision among } n_i \text{ keys}] < \frac{1}{2}$$

Proof

- Let X be number of collisions
- By Claim 1, we are hashing n_i keys into n_i^2 slots:

$$E[X] \leq C(n_i, 2) \cdot \frac{1}{n_i^2} < \frac{1}{2}$$

- By Markov inequality

$$P[X \geq 1] < E[X] < \frac{1}{2}$$

$\Rightarrow$ Proven

$\Rightarrow$ Recall: In step 3, we repeat the iteration with different hash function h_i' picked from H_{p,n_i^2} until there is no collision in the secondary table.

Claim 3 allow us to say that "Expected number of iterations before we picks a "good" function h_i is 2."

Open Addressing (Closed Hashing)

Another hashing technique that uses "probing" to find an empty slot to store the colliding key

- Pros: - efficient when load factor is low
  - space efficiency (does not use 2-level hashing scheme like Perfect Hashing)

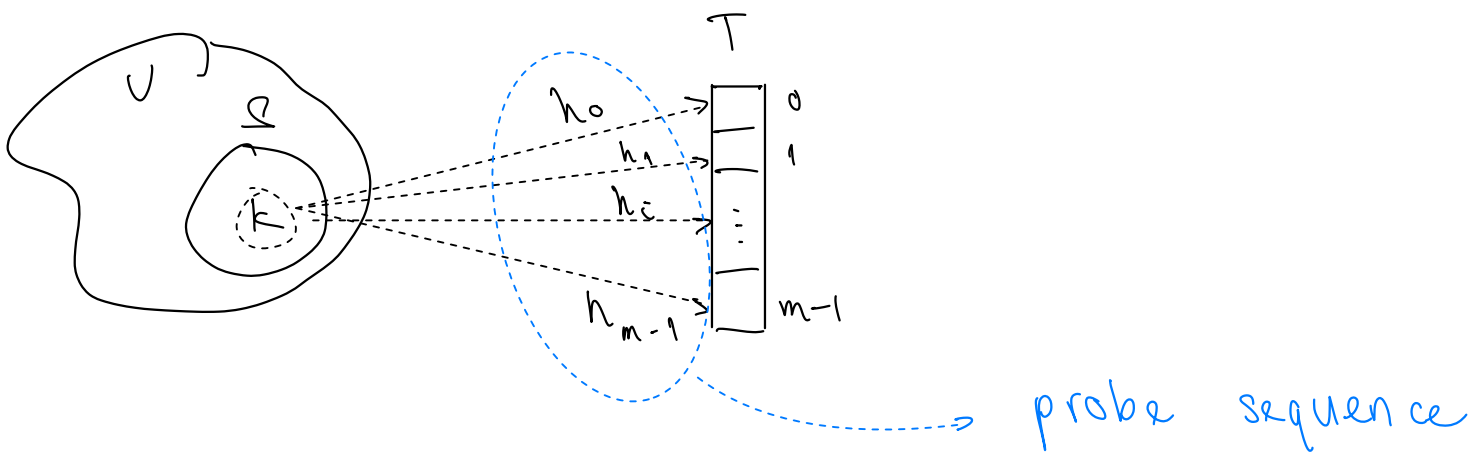
Main idea:

Sequence of hash functions $[h_0, h_1, h_2, \ldots, h_{m-1}]$

Probe sequence $[h_0(k), h_1(k), h_2(k), \ldots, h_{m-1}]$

↳ is a permutation of $\{0, 1, 2, \ldots, m-1\}$

⇒ Different hash functions in the sequence maps key k to different slot in table T



Pseudo: (SEARCH_KEY, INSERT_KEY)

```
SEARCH_KEY (k)
for i : 0 → m-1 :
    if T[h_i(k)] == k :
        return h_i(k)
    else if T[h_i(k)] == ∅ :
        return "absent"
return "full"
```

```
INSERT_KEY (k):
for i : 0 → m-1 :
    if T[h_i(k)] == k :
        return h_i(k)
    else if T[h_i(k)] == ∅ :
        T[h_i(k)] ← k
return "full"
```

Uniform Assumption:

Just like Chaining, we assume the probing sequence is equally likely to be any of the permutations created from set $\{0, 1, 2, \ldots m-1\}$

⇒ Consequences of Uniform Assumption.

Suppose there are currently n elements in the hash table T size m

- Uniformity of hash value $h_i(k)$:

  Fix key k and index i, $h_i(k)$ is equally likely to be hashed in any of the slot in table T

  ⇒ Probability of a slot being occupied is n/m :

  $$P[h_0(k) \neq \emptyset] = n/m \quad (1)$$

- Independence of sub-probe sequence

  Ignore the first hash function $h_0(k)$, the remaining probe sequence $[h_1(k), h_2(k), \dots h_{m-1}(k)]$ is equally likely to be any permutation created from set $\{0, 1, 2, \dots, m-1\} \setminus \{h_0(k)\}$

  ⇒ If a slot is occupied, recursively perform the same operation on the sub-probe sequence $\{0, 1, 2, \dots, m-1\} \setminus \{h_0(k)\}$ (2)

(1) & (2) ⇒ The recurrence relation for expected number of probes (# of attempt to find empty slot)

Recurrence for expected number of probes

or "Expected number of attempts to find an empty slot"

$$E[T(m, n)] = 1 + \frac{n}{m} \cdot E[T(m-1, n-1)]$$

Proof: $E[T(m,n)] \leq \frac{1}{1-\alpha}$ $\alpha = n/m$

- Base case: $n = 0$

  $E[T(m, 0)] = 1$ Because guarantee to find an empty slot in the beginning

- Induction: Assume $E[T(m-1, n-1)] = 1 + \frac{n-1}{m-1} \cdot E[T(m-2, n-2)]$

$$\begin{aligned} E[T(m,n)] &= 1 + \frac{n}{m} E[T(m-1, n-1)] \\ &\leq 1 + n/m \cdot {}^{m-1}/_{m-n} \\ &< 1 + n/m \cdot m/(m-n) \\ &= \frac{m}{m-n} \quad \text{(true)} \end{aligned}$$

Rewrite $\frac{m}{m-n} = \frac{1}{1-\alpha}$ where $\alpha = \frac{n}{m}$ is load factor

⇒ Expected time for an unsuccessful search is $O(1)$ $\left(\frac{1}{1-\alpha}\right)$

In practice, we cant generate the ideal (uniform) probe sequences, so we must rely on simpler probing scheme to resolve collision

Linear Probing

The simplest probing scheme

$$h_i(k) := (h(k) + i) \mod m \quad \forall \quad 0 \leq i \leq m-1$$

- Pros:
  - Better cache performance (since it visits consecutive entries in the hash table)
  - If load factor is strictly < 1 $\Rightarrow$ the expected length of any probe sequence provably constant
- Cons:
  - Number of probes grow quickly as load factor approach 1 (since occupied cells in the table tend to stick together)

Example:

1. Initial hash: key k is hashed into slot $h(k)$
2. If slot $h(k)$ is occupied, trigger probing:
   $$h_0(k) := (h(k) + 0) \mod m$$
3. Continue with $i := 1 \rightarrow m-1$ until find empty slot

Note: Wrap around, if $h_i(k)$ reaches end of table, it wraps around to the beginning of the table due to modulo

Binary Probing

Let hash table size $m = 2^l$ $(2, 4, 8, 16, \ldots)$

Unlike linear probing, binary probing use bitwise XOR operation to determin next slot

$$h_i(k) := h(k) \oplus i \quad \forall \quad 0 \leq i \leq m-1$$

- Pros:
  - Less cache misses (since the pattern of binary probing values tend to check current cache before moving to the next cache line)

Example: Consider a hash table of size 16 with cache line of 8.

- Initial hash: $h(k) = 5$
- First probe: $h_1(k) = 5 \oplus 1 = 4$
- Second probe: $h_2(k) = 5 \oplus 2 = 7$
- Third probe: $h_3(k) = 5 \oplus 3 = 6$

$\vdots$

$h_4(k) = 5 \oplus 4 = 1$

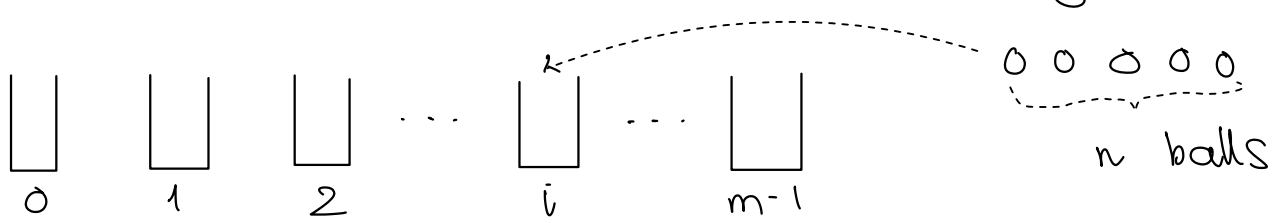
$h_5(k) = 5 \oplus 5 = 0$

$h_6(k) = 5 \oplus 6 = 3$

- Seventh probe: $h_7(k) = 5 \oplus 7 = 2$ checked all of cache line
- Eight probe: $h_8(k) = 5 \oplus 8 = 13$ start checking out side cache line

Balls and Bins as Hashing

Throwing n balls into m bins $\sim$ Hashing n keys into m-size table



Each ball is thrown independently and uniformly: $P(\text{ball } i \to \text{bin } j) = \frac{1}{m}$

Let $\begin{cases} X_j \text{ be r.v. of number of balls in bin-}j \Rightarrow X_j \sim Bin(n, \frac{1}{m}) \\ \alpha = E[X_j] = \frac{n}{m} \text{ be the load factor} \end{cases}$

Rule of thumb for analysis

- If $n \gg m$ (large α), use Chernoff bounds
- If $n \approx m$ ($\alpha = \Theta(1)$), use Poisson approximation

Case 2: $n \approx m$

Recall that for r.v. $X \sim Bin(n, p)$ with large n and small p, we can say that $X \sim Pois(\lambda)$ with PMF:

$$P(X = k) = \frac{e^{-\lambda}\lambda^k}{k!}, \quad k = 0, 1, 2, \dots \qquad \lambda = n \cdot p = \frac{n}{m}$$

Recall: basic properties of Poisson distribution

$E[X] = \lambda$

$Var[X] = \lambda$

$X + Y \sim Pois(\lambda + \lambda')$ for $X \sim Pois(\lambda)$ and $Y \sim Pois(\lambda')$

S_0 :

Since $\begin{cases} X_j \sim Bin(n, \frac{1}{m}) \\ n \approx m \end{cases}$

$\Rightarrow X_j \sim Pois(\frac{n}{m})$

Facts

. Probability of n balls in same bin: $(\frac{1}{m})^n$

. Probability that bin j is empty: $(1-\frac{1}{m})^n \approx e^{-\frac{n}{m}}$

. Let Y be number of empty bins, $E[Y] = ?$

Denote Y_j be i.r.v of if bin-j is empty

$\Rightarrow Y_j = \begin{cases} 1 & \text{if bin-}j\text{ is empty} \\ 0 & \text{otherwise} \end{cases}$

Then: $E[Y] = \sum_{j=0}^{m-1} E[Y_j]$ (linearity of expectations)

$= \sum_{j=0}^{m-1} E[X_j = 0]$

$= m\,(1-\frac{1}{m})^n$

$\approx m \cdot e^{-n/m}$

. Probability j^{th} bin contains 1 ball

. Exact computation:

$P[X_j = 1] = C(n,1) \cdot (\frac{1}{m})^1 \cdot (1-\frac{1}{m})^{n-1}$

$= \frac{n}{m}(1-\frac{1}{m})^n \cdot (1-\frac{1}{m})^{-1}$

. Poisson approx:

$P[X_j = 1] = e^{-\lambda} \cdot \lambda = e^{-n/m} \cdot \frac{n}{m}$

. Probability j^{th} bin contains exactly r balls

. Exact computation:

$P[X_j = r] = C(n,r) \cdot (\frac{1}{m})^r \cdot (1-\frac{1}{m})^{n-r}$

$= \frac{1}{r!}(\frac{n}{m} \cdot \frac{n-1}{m} \cdot \frac{n-2}{m} \cdots \frac{n-r+1}{m}) \cdot (1-\frac{1}{m})^n (1-\frac{1}{m})^{-r}$

$= \lambda^r \cdot e^{-\lambda} \cdot 1^{-r}$

. Poisson approx:

$P[X_j = r] = \frac{e^{-\lambda} \cdot \lambda^r}{r!}$

- Probability at least one bin has a collision

$$P[\text{at least 1 bin has more than 1 balls}] = 1 - P[\text{every bin-}j \text{ has } X_j \leq 1]$$

Fix a ball k^{th}:

$$P[\text{ball } k \to \text{non-empty bin}] = \frac{k-1}{m}$$

$$P[\text{ball } k \to \text{empty bin}] = 1 - \frac{k-1}{m}$$

Therefore:

$$P[\text{every bin-}j \text{ has } X_j \leq 1] = \prod_{i=1}^{n-1} \left(1 - \frac{i-1}{m}\right)$$

$$\approx \prod_{i=1}^{n-1} e^{-i/m}$$

$$\Rightarrow P[\text{at least 1 bin has collision}] = 1 - \prod_{i=1}^{n-1} e^{-i/m}$$

Coupon Collector's problem: How many balls we have to throw to assure that every bin contains ≥ 1 balls

Let
- Y be number of balls to throw until no empty bins left
- Y_i be number of balls to throw until hits an empty bin, given that $i-1$ bins have been filled

$$\Rightarrow Y_0 = 1 \quad \text{and} \quad Y = \sum_{i=0}^{m-1} Y_i$$

$$= 1 + Y_1 + Y_2 + \ldots + Y_{m-1}$$

$$\Rightarrow \begin{cases} P[\text{ball } i \to \text{non-empty bin}] = \frac{i-1}{m} \\ P[\text{ball } i \to \text{empty bin}] = 1 - \frac{i-1}{m} \end{cases}$$

$$\Rightarrow P[Y_i = k] = \left(\frac{i-1}{m}\right)^{k-1} \left(1 - \frac{i-1}{m}\right)$$

$$\Rightarrow Y_i \sim \text{Geometric}\left(p = 1 - \frac{i-1}{m}\right)$$

$$\Rightarrow E[Y_i] = \frac{1}{p} = \frac{m}{m+i+1}$$

By linearity of expectation:

$$E[Y] = \sum_{i=0}^{m-1} E[Y_i] = \sum_{i=0}^{m-1} \frac{m}{m+i+1}$$

$$= m \sum_{i=0}^{m-1} \frac{1}{i} \approx m\left(\ln(m) + O(1)\right)$$

Maximum load theorem

If we throw n balls into m bins independently and uniformly, where $m \approx n$, then:

Max load of a bin $\leq \frac{4 \lg n}{\lg \lg n}$ with probability $\leq 1 - \frac{1}{n}$