CSEP 521: Applied Algorithms
Lecture 7 Hashing

Richard Anderson
January 25, 2021

Announcements

- Homework 4 is available
  - Three problems
  - Program – evaluate "two choice" hashing
- Thursday, Cuckoo Hashing
  - Reading + Video link

Randomness so far

- Average case QuickSelect
- MinCut Analysis
- Binary Space Partition
- Average Case for Stable Marriage
- Primality Testing

- A random world is more predictable than a deterministic one
  - Law of large numbers

Data structures

- Keeping track of stuff
- Supporting algorithms

```
Heapsort( A, n)
       H = new Heap()
       for j = 1 to n-1
              Heap.Insert(A[j])
       for j = 1 to n-1
              A[i] = Heap.DeleteMin()
```

- Sometimes they matter and sometimes they don't

Data structure trade offs

- Operation Time
- Space
- Accuracy
- Implementation complexity

Hashing


Hans Peter Luhn

- Tracking information associated with keys
  - Set
  - Search tree
  - Arrays if the keys can be an index

- Key idea – map from key space S to table T, $|S| = n$, $|T| = m$
  - Hash function h, store data at location h(x)
  - Collision if $h(x) = h(y)$ for $x \neq y$

- In practice, O(1) access

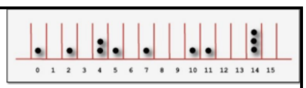
Hash functions

- Start by assuming h is completely random
  - Universe U, |U| = d, table size m
  - Ω : set of all mappings from 1..d to 1..m
- Lots of work in
  - Creating practical hash functions
  - Identifying weaker assumptions than "completely random"
- For some applications, "random" hash functions are important
- Useful class of hash functions, H = { $H^p_{a,b}$ | p prime, a, b in [1 .. p-1]}
  - $H^p_{a,b}(x)$ = (a x + b) mod p

Collision resolution (review)

- Method 1 – Chaining (Closed addressing, open hashing)
- Method 2 – Table based (Open addressing, closed hashing)
- Load factor α, ratio of stored elements to table size
  - For Chaining, want 0.5 <= α <= 1
  - For Table based, need α < 1, α <= 0.75 recommended
- Common approach is to increase table size (e.g., by a factor of 2) and rehash when load factor exceeds a bound

Balls and boxes



- N boxes, repeatedly assign balls to random boxes
- Coupon collecting – expected number of balls until every box is occupied
- How about if we assign K balls at random to N boxes
  - How many cells are occupied?
  - What is the expected number of balls in the first box?
  - What is the expected maximum for the number of balls assigned to any cell?
- Balls and boxes basis for the theory of hashing

N balls in N boxes
What is the maximum number of balls in any box?

- Definition w.h.p.
  - For any j, with appropriate choice of constants, probability of failure is $O(n^{-j})$
- Maximum number of balls in a box is O(log n / log log n)
- log n / log log n analysis
  - Compute the probability that a given bin has more k items
  - Show that this is less than 1 / k!
  - Choose k = c log n / log log n, so that 1/k! < $1/n^2$
  - Probability that any bin has more than k items is less than 1/n

The Math

$$\mathbf{Pr}[\text{bin}_i \text{ gets more than } k \text{ elements}] \le \binom{n}{k} \cdot \frac{1}{n^k} \le \frac{1}{k!}$$

By Stirling's formula,

$$k! \sim \sqrt{2nk}\left(\frac{k}{e}\right)^k$$

If we choose $k = O(\frac{\log n}{\log \log n})$, we can let $\frac{1}{k!} \le \frac{1}{n^2}$. Then

$$\mathbf{Pr}[\exists \text{ a bin} \ge k \text{ balls}] \le n \cdot \frac{1}{n^2} = \frac{1}{n}$$

So with probability larger than $1 - \frac{1}{n}$,

$$\text{max load} \le O\left(\frac{\log n}{\log \log n}\right)$$

Power of hashing twice
Load balancing

- Let h_1 and h_2 by random hash functions
- When element x is inserted, it goes to the cell $h_1(x)$ or $h_2(x)$ with least number of elements elements
- Find must check cells $h_1(x)$ and $h_2(x)$
- The maximum number of elements assigned to any cell is O(loglog n) with high probability

Proof (Intuition)

- Ball has height k when it is placed in a bin with k-1 balls
- Expect <= n/2 bins with 2 balls
- Expect <= $n/2^2$ bins with 3 balls
- Expect <= $n/2^4$ bins with 4 balls
- Expect <= $n/2^8$ bins with 5 balls
- Expect <= $n/2^{16}$ bins with 6 balls
- Expect <= $n/2^{32}$ bins with 7 balls

Tracking keys without data

- If the key domain is [1..n] a bit vector is ideal
- What if you hash into a bit vector?
- What type of errors occur

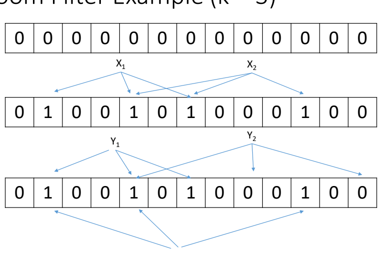
Bloom Filter

- Basic idea – k-hash functions
- Bits are set at $h_1(x), h_2(x), \ldots, h_k(x)$
- Lookup is done by reading $h_1(x), h_2(x), \ldots, h_k(x)$
- Can we get a false negative
- Can we get a false positive

Bloom Filter

- Alternative data structures: List, Hash Table
- Critical reason for using Bloom Filter – limited storage
  - Lots of data
    - Devices with limited memory (e.g., network routers)
    - Need for main memory versus going to disk
  - Don't need to remember the actual data (in the data structure)
- Measure of interest – number of bits per data element
- Bloom filters have been left out of computer science curriculum

Bloom Filter Example (k = 3)



Some Bloom Filter Math

- Table size m, data items n
- After all members of S have been hashed probability of specific element being zero is
- False probability rate
- Express rate as a function of probability

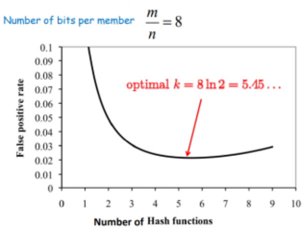
$$p' = (1 - \frac{1}{m})^{kn} \approx e^{-kn/m} = p$$

$$(1 - p')^k \approx (1 - p)^k$$

$$f' = (1 - p')^k = (1 - (1 - \frac{1}{m})^{kn})^k$$

$$f = (1 - p)^k = (1 - e^{-kn/m})^k$$

False positive rate vs. k
Find optimal with calculus



Rewrite f as

$$f = \exp(\ln(1-e^{-kn/m})^k) = \exp(k \ln(1-e^{-kn/m}))$$

Let $g = k \ln(1-e^{-kn/m})$

Minimizing g will minimize $f = \exp(g)$

$$\frac{\partial g}{\partial k} = \ln(1-e^{-kn/m}) + \frac{k}{1-e^{-kn/m}} \frac{\partial(1-e^{-kn/m})}{\partial k}$$

$$= \ln(1-e^{-kn/m}) + \frac{k}{1-e^{-kn/m}} \frac{n}{m} e^{-kn/m} = -\ln(2) + \ln(2) = 0$$

if we plug $k = (m/n)\ln 2$ which is optimal

(It is in fact a global optimum)

Bloom Filter Applications

- Dictionary to detect speling mistakes
  - All good words let through, some mistakes will happen
- List of malicious URLs in browser
- List of keys needed for a database join
- Akamai web caching, avoid caching data only requested once
  - List of requests put into a Bloom filter, store data on the second request

Bloom filter deletes

- Why do Bloom filters fail for deletes?
- Counting Bloom Filters
- Each cell is a counter (4 bits considered sufficient)
- Insert, add one to each target cell
- Delete, delete one from each target cell
- Find, test if target cells non-zero

- On overflow, leave counter at maximum value

Bloom Filter Deletes (k = 3)