CSCB63 – Design and Analysis of Data Structures

Akshay Arun Bapat

Username problem

- Registering a new user into a system
- Asking for their username (has to be unique)

- Search through all available ones
- Complexity
  - Linear if we decide to search through all of them one by one
  - Binary search possible if we store them in sorted order (balanced tree or simply sorted array)
  - B-Tree to get a better time bound (index)

Can we do better?

Bloom filter

- Not an index, just a filter

- Answers YES or NO
  - Yes - key in the structure
  - No - key not in the structure

- Isn't always right (we can control this rate though)
- Guarantees that NO is always right

Recap: Binary classification errors

- False positive - System says YES (positive), but actually isn't (false, actually is NO)

- False negative - System says NO (negative), but actually isn't (false, actually is YES)

Bloom filter

- False positive - can happen
- False negative - cannot happen

NO means no
YES can be maybe

Bloom filter

- Collection of m bits
- Collection of k hash functions
- All hash functions hash to one of the m bits

0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0	1	0	1	0	0	0	1

- It is an auxiliary structure, assists another data structure

Bloom filter insert

- For all k hash functions, compute $h_i(key) = j$
- Turn bit j to 1

```
insert(B, key)
  for i in 0 to k-1:
    B[B.h_i(key)] = 1
```

Complexity? $\mathcal{O}(k)$

Insert in main structure after

Bloom filter search

- For all k hash functions, compute $h_i(key) = j$
- Yes, if all j bits are 1
- No, otherwise

```
0. search(B, key):
1.   for i in 0 to k-1:
2.     if B[B.h_i(key)] = 0:
3.       return False
4.   return True
```

Complexity? $\mathcal{O}(k)$

Search in main structure after (only if YES)

Bloom filter delete

- Re-hash everything again (same for growing number of bits, same as increasing buckets in hash table)

- Deleting from bloom filters is not possible
- Advanced filters (out of scope)

- Just ignore deletes, affects false positive probability

Delete in main structure after

False positive probability

- Assume simple uniform hashing for all hash functions
- Assume hash functions are independent of each other
- Pr(certain bit is set by a certain hash function)

$$\frac{1}{m}$$

- Pr(certain bit is not set by a certain hash function)

$$1 - \frac{1}{m}$$

- Pr(certain bit is not set by k hash functions)

$$\left(1 - \frac{1}{m}\right)^k$$

False positive probability

- n items have been inserted so far
- Pr(certain bit is not set after n insertions)

$$\left(1 - \frac{1}{m}\right)^{nk}$$

- Pr(certain bit is set after n insertions)

$$1 - \left(1 - \frac{1}{m}\right)^{nk}$$

- Pr(false positive) = Pr(all k hash functions hash to a set bit)

$$\left(1 - \left(1 - \frac{1}{m}\right)^{nk}\right)^{k}$$

Advantages of bloom filter

- Fast - constant for number of hash functions
- Space efficient - 10 bits per key for $< 1\%$ false positive probability