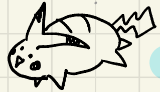
Bloom Filters I



False Positive: We report $x \in X$, but it is not

False Negative: We report $x \notin X$, but it is

We will **tolerate false positives** (very few) but **no false negatives**

Examples:

$X = \{$weak passwords$\}$

→ Screen passwords for safety

- Allow **no** weak passwords
- May flag a few good passwords as being weak.

$X = \{$URL's of malicious web sites$\}$

→ Allow no malicious sites

- May flag a few safe ones

Bloom Filter:

- 1970 by Burton Howard Bloom
- Can store very large sets X
- Answers queries in **$O(1)$ time**
- Uses **$O(n)$ bits** - $n = |X|$

  (May be smaller than space needed to store all keys!)

Filtering:

- Given a large set X of keys answer **membership queries** is $x \in X$
- **Objectives:**
  - Fast! $O(1)$ time
  - Yes/No: No values/Just keys
  - Errors allowed

Bloom Filter:

Let $X \subseteq U$ (universe)

$|X| = n$ large!

Parameters: k, m - TBD

Bit Vector: $B[0..m-1]$

k Hash Functions: $h_1, \dots, h_k$

$h_i : U \to \{0, \dots, m-1\}$

[Think: h_i maps keys to **random** locations in bit vector]

```
find(Key x) {
    for (i = 1, 2, ..., k) {
        if (B[h_i(x)] == 0) return false
    return true
```

find("b") $h_i(b) = [2,3,7] \Rightarrow$ true

find("c") $h_i(c) = [2,5,9] \Rightarrow$ false

find("d") $h_i(d) = [0,4,7] \Rightarrow$ true??

Example:

	0	1	2	3	4	5	6	7	8	9
B	1	0	1	1	1	0	0	1	0	0

insert("a") $h_i(a) = [0,7,4]$

insert("b") $h_i(b) = [2,3,7]$

(all other entries 0)

Initially: $B[i] = 0$, $i = 0, \dots, m-1$

```
insert(Key x) {
    for (i = 1, 2, ..., k)
        B[h_i(x)] ← 1
}
```

Bloom Filters II

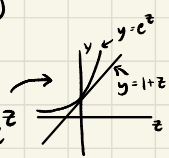
Controlling False Positives:

Obs:

m - Bigger is better (but more storage)

k - Trickier to balance

Math Facts:



① If $|z|$ is small, $1+z \approx e^z$

② If an event occurs w. probability p, the prob. of ℓ independent occurrences is p^ℓ.

Analysis:

Assume: $n=|X|$, $m=|B|$, $k=$ no. hashes

- Prob of hitting an arbitrary loc. of B, $B[j]$, with any hash is $1/m$.
- Prob of missing is $1-1/m$. So:

$$\Pr(h_i(x) \neq j) = 1 - 1/m$$

- After inserting all n keys, each with k hashes, prob of missing $B[j]$ is:

$$\Pr(B[j]=0) = (1-1/m)^{k\cdot n}$$

- Assuming m large $\Rightarrow |1/m|$ small $\Rightarrow$

$$\Pr(B[j]=0) \approx (e^{-1/m})^{k\cdot n} = e^{-kn/m}$$

Partial Correctness:

- If $x \in X$, all hash locations $B[h_i(x)]$ set to 1 $\Rightarrow$ true
- If $x \notin X$, if any hash loc. $B[h_i(x)]$ is 0 $\Rightarrow$ false

$\rightarrow$ But by coincidence, all may be set to 1 by other keys $\Rightarrow$ true (false positive)

Define: $p = e^{-kn/m}$

(Later, we'll show best $p = 1/2$)

- Prob. that any entry $B[j]=0$ is p $\Rightarrow$ Expected num. of 0's is mp
- Useful fact from prob. theory (Concentration about mean)
- If m large - actual num. of 0's is very nearly $m \cdot p$

In summary: $\Pr[FP] = \left(\frac{1}{2}\right)^{(\ln 2)\frac{m}{n}}$

To achieve a false pos. rate of $\delta > 0$ set

$$m = \frac{\lg 1/\delta}{\ln 2} \cdot n$$

Equiv: Num of bits per key is

$$\frac{m}{n} \approx \frac{\lg 1/\delta}{\ln 2} = O(\log 1/\delta)$$

False Positive Probability:

- False positive (FP) occurs if all $h_i(x)$ set to 1 by other keys
- Since prob $B[j]=0$ is p, we have

$$\Pr[FP] = \Pr[\{B[h_i(x)]=1\}_{i=1}^{k}] = (1-p)^k$$

- To simplify - Take $\ln$

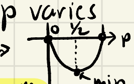
$$\ln(\Pr[FP]) = k \cdot \ln(1-p)$$

- By def. of p, we have:

$$\ln p = -kn/m \iff k = -\frac{m}{n} \ln p$$

$$\Rightarrow \ln(\Pr[FP]) = -\frac{m}{n} \ln p \cdot \ln(1-p)$$

- How to set k to minimize this?
  - Assume m, n fixed, but p varies
  - $\ln p \cdot \ln(1-p) \rightsquigarrow$



  - Set $p = 1/2$

$$\Rightarrow \Pr[FP] = (1-p)^k = \left(\tfrac{1}{2}\right)^{(\ln 2)\frac{m}{n}} \approx (0.618)^{m/n}$$