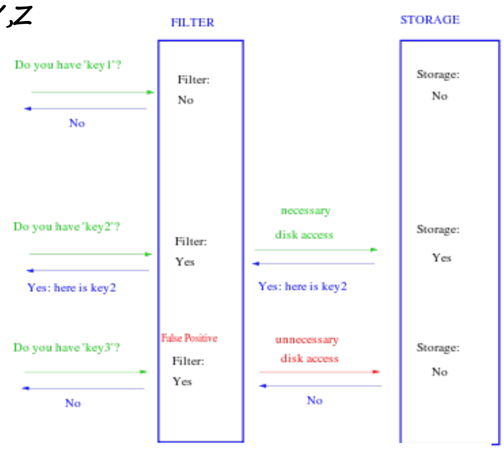
Bloom filters: (operates in Bits , actual data is stored in DB)

* Only trust it when returns 0 that means not in hash table/DB
* If returns 1 then it may not belong to out query element x because that i'th bit could have been flipped by some other element
* Deletion is not possible , lets say X flipped i'th bit and Y,Z also flipped same bit now if we delete X from DB then again we query X in bloom filter but hash(X) = i and i'th bit is still 1 which actually corresponds to Y,Z. If we flip i'th bit for X to 0 with that we will loose information about Y,Z



Bloom Filters: start with an empty bit array (all zeros), and k hash functions.

k1 = (13 - (x % 13))% 7,

k2 = (3 + 5x) % 7, etc.

0	1	2	3	4	5	6
0	0	0	0	0	0	0

Insert **129** → x=129

k1 = (13 - (x % 13))% 7
= (13 - (129 % 13)) % 7
= (13 -12) % 7
= 1 % 7
= 1

k2 = (3 + 5*x) % 7
= (3 + 5*129) % 7
= 648 % 7
= 4

0	1	2	3	4	5	6
0	1	0	0	1	0	0

k1 == 1, so we change bit 1 to a 1
k2 == 4, so we change bit 4 to a 1

Insert **479** : x= 479

k1 = (13 - (x % 13)) % 7
= (13 – (479 % 13)) % 7
= (13 – 11) % 7
= 2

k2 = (3 + 5x) % 7
= (3 + 5*479) % 7
= (2398) % 7
= 4

0	1	2	3	4	5	6
0	1	1	0	1	0	0

k1 == 2, so we change bit 2 to a 1
k2 == 4, so we would change bit 4 to a 1, but it is already a 1

- To check if 129 is in the table, just hash again using k1 and k2 and check the bits in the bit array.
- In this case, k1=1, k2=4: ***probably*** in the table!

0	1	2	3	4	5	6
0	1	1	0	1	0	0

- To check if 123 is in the table, hash and check the bits.

k1=0, k2=2: ***definitely cannot be*** in table because the 0 bit is still 0

0	1	2	3	4	5	6
0	1	1	0	1	0	0

- To check if 402 is in the table, hash and check the bits.

k1=1, k2=4: ***Probably*** in the table (but isn't! False positive!)

k1 = (13 - (x % 13))% 7
= (13 - (402 % 13)) % 7
= (13 -12) % 7
= 1 % 7
= 1

k2 = (3 + 5*x) % 7
= (3 + 5*402) % 7
= 2013 % 7
= 4

129 and 402 result in same bits

0	1	2	3	4	5	6
0	1	1	0	1	0	0

n is number of elements inserted ,
m is number of bits in array.

Because of this, we can say that the exact probability of false positives is

$$\sum_t \Pr(q=t)(1-t)^k \approx (1-E[q])^k = \left(1-\left[1-\frac{1}{m}\right]^{kn}\right)^k \approx \left(1-e^{-kn/m}\right)^k$$

The number ofoptimal hash functions k, a positive integer. for a given m and n, the value of k that minimizes the false positive probability is

$$k = \frac{m}{n}\ln 2.$$

O(k) is the time complexity for querying a bloom filter

links:

https://en.wikipedia.org/wiki/Bloom_filter

https://llimllib.github.io/bloomfilter-tutorial/

https://freecontent.manning.com/all-about-bloom-filters/

https://youtu.be/hkAONPOaC9w?t=1055