Filters :-

Filters represent a set approximately; trading of accuracy for space efficiency.

- → Bloom filter
- → quotient filter
- → cuckoo filter

} Dynamic

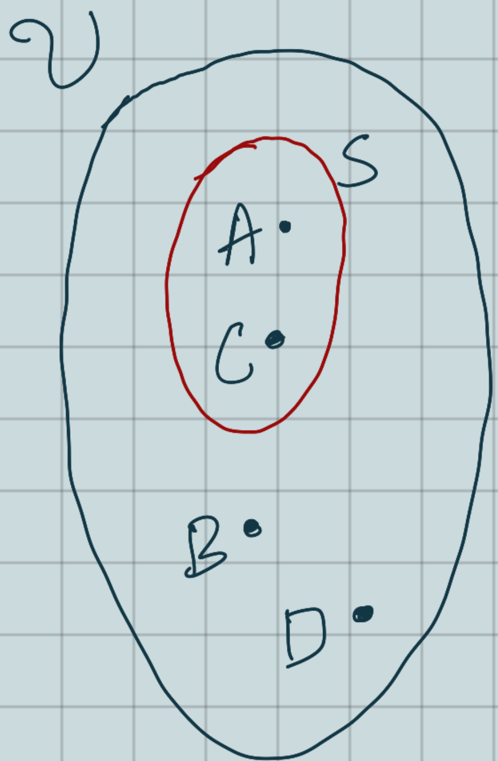
} deletes

- → XOR filter
- → Ribbon filter.

} Static filter

Static :- the set of items is known in advance

Dynamic :- the set of items is not known in advance.



A : ✓

B : ✗

C : ✓

D : ✓ false positive

A filter guarantees a false-positive rate ϵ

if $q \in S$, return ✓ with prob. 1

if $q \notin S$, return $\begin{cases} \times \text{ with prob.} > 1-\epsilon \\ \checkmark \text{ with prob.} \leq \epsilon \end{cases}$

Filter have one-sided errors.
(No False Negatives)

Space usage:

Filter

$\geq n \lg \frac{1}{\epsilon}$ bits

Dictionary
(hash-table)

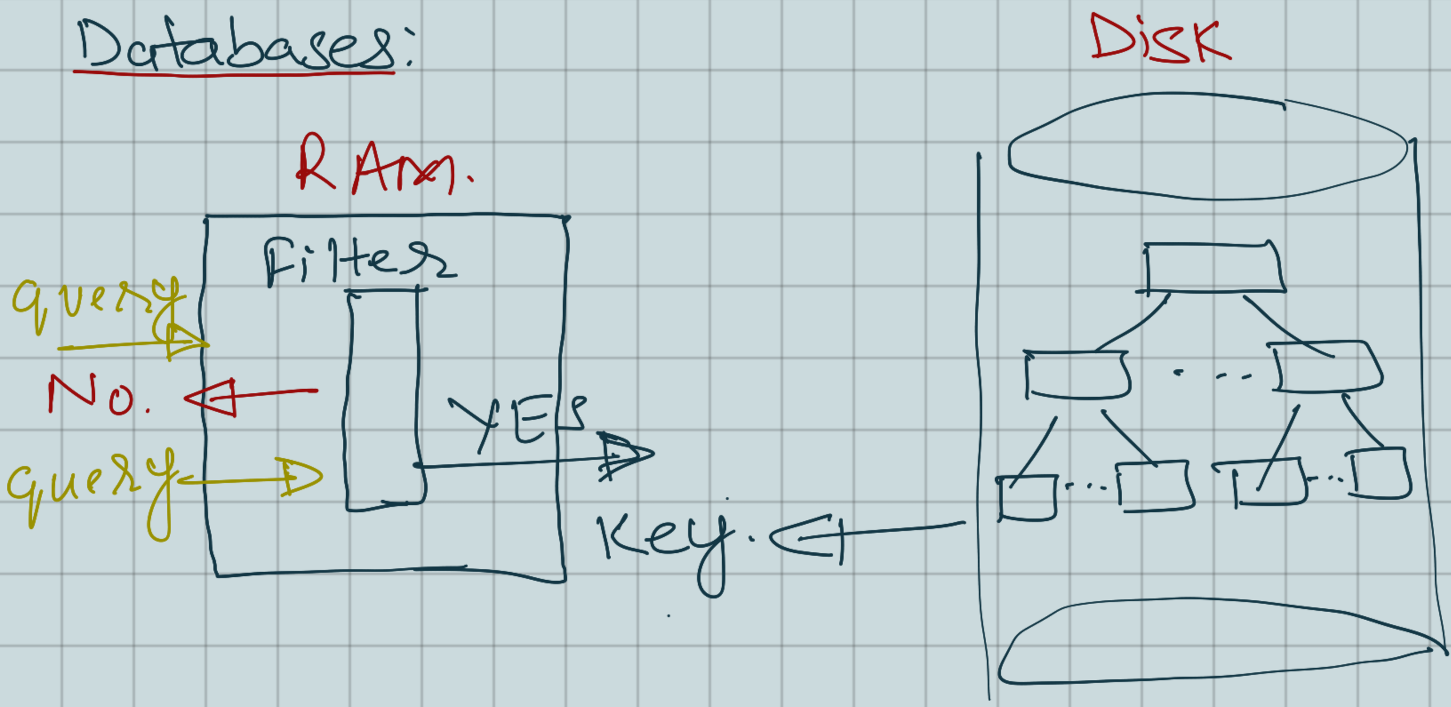
$\Omega(n \cdot \lg |U|)$ bits

for most practical purposes:

$\epsilon = 2\%$, a filter requires ~ 8 bits/item

Filter use case:-

Databases:

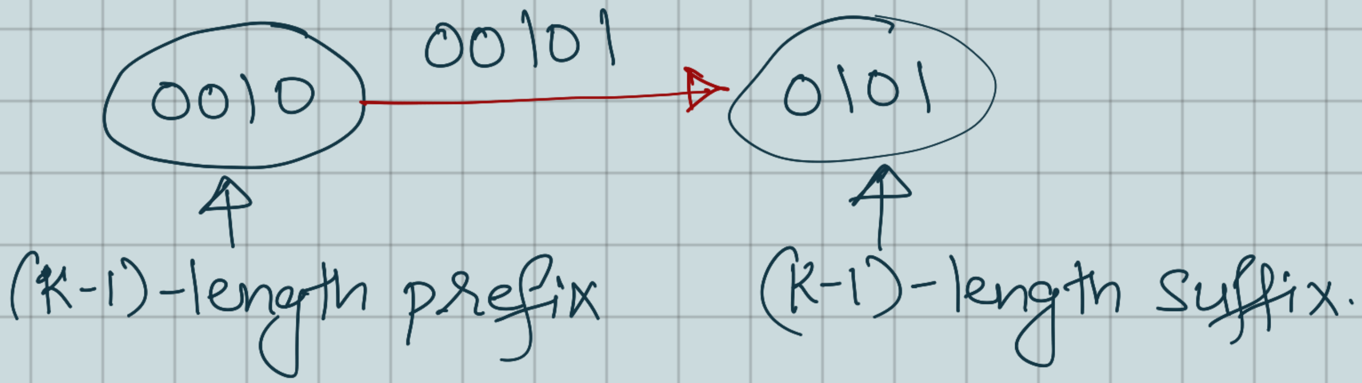


→ In a database, disk accesses are costly and dominate the performance.

→ RAM is much smaller in size than Disk.
→ filter represent the key set approx. in RAM
→ filters can help avoid going to disk for negative queries.

→ For use in databases, filters are often required to support insert, queries, deletes.

de Bruijn graph:



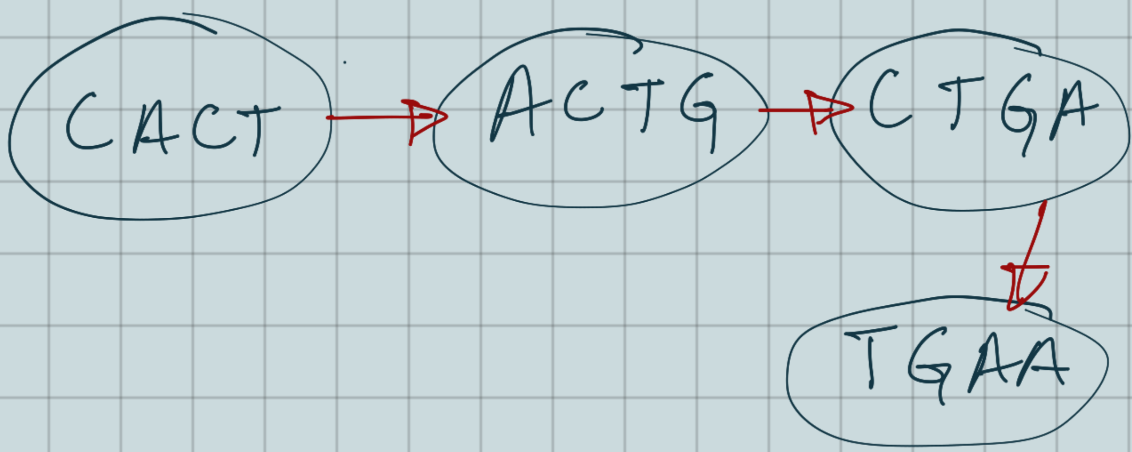
An edge is a length-K string connecting its two (K-1) substrings.

In genomics:-

CACTGAA → Read.

K-mers (K=4):

CACT
ACTG
CTGA
TGAA

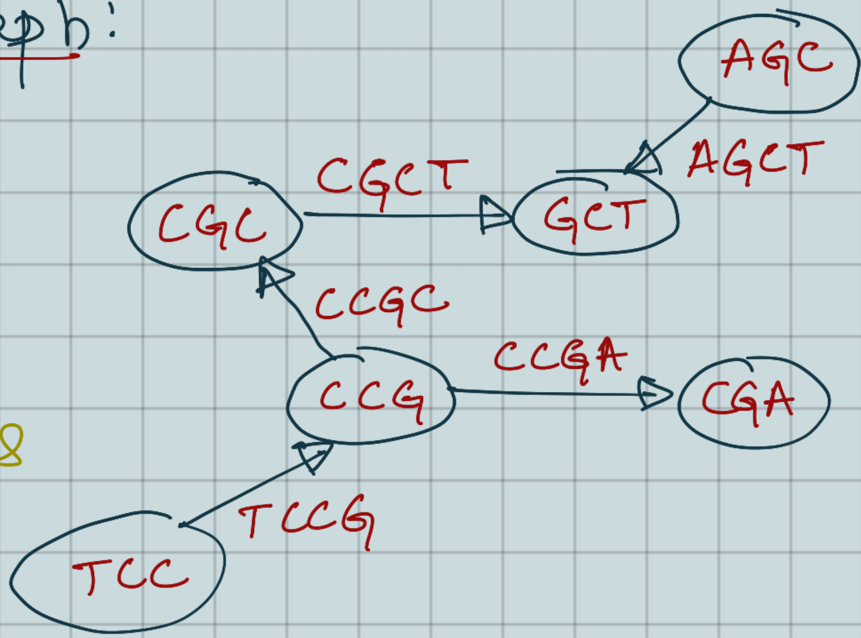


de Bruijn graph:

Set:

T C C G
C C G C
C C G A
C G C T
A G C T

Edges



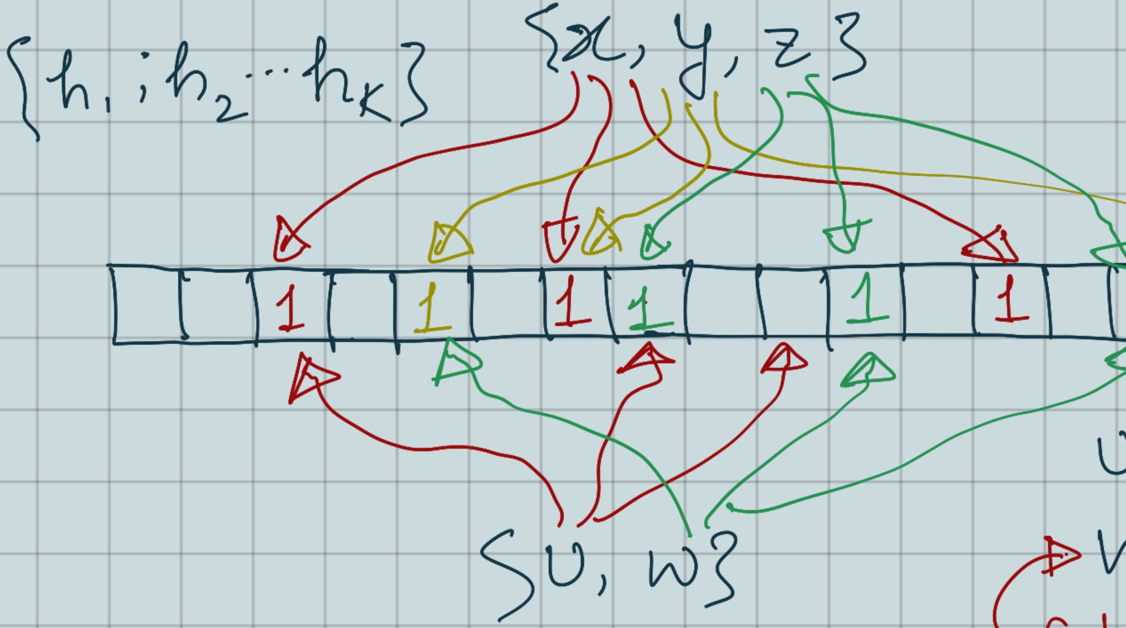
A filter can be used to represent a de Bruijn graph approximately.

→ Using a filter to traverse the de Bruijn graph causes a small number of topological errors.

Question:- How can you remove the topological errors?

Bloom filter (BF):

A BF consists of a bit vector of size m and K hash functions.



$m = 16$
$K = 3$

u : ✗
w : ✓ ← false positive.

Space : $\simeq 1.44\, n \lg\left(\frac{1}{\epsilon}\right)$ bits.

→ Bloom filters do not support deletes.

→ Q: how can you delete in Bloom filter?

→ Q: Can you merge two Bloom filters?

False positive analysis:

Two parameters:- m, K

$$Pr\left(\begin{array}{l}\text{a certain bit is not set 1}\\ \text{by a certain hash ftn.}\end{array}\right) = 1 - \frac{1}{m}$$

$$Pr\left(\begin{array}{l}\text{a certain bit is not set 1}\\ \text{by any hash ftn.}\end{array}\right) = \left(1 - \frac{1}{m}\right)^K$$

$$Pr\left(\begin{array}{l}\text{a certain bit is not set 1}\\ \text{after n insertions}\end{array}\right) = \left(1 - \frac{1}{m}\right)^{Kn}$$

$$\boxed{\lim_{m \to \infty} \left(1 - \frac{1}{m}\right)^m = \frac{1}{e}}$$

$$= e^{-Kn/m}$$

$$Pr\left(\begin{array}{l}\text{a certain bit is 1}\\ \text{after n insertions}\end{array}\right) = 1 - e^{-Kn/m}$$

$$Pr\left(\begin{array}{l}\text{all K bits are 1 during}\\ \text{a query of item not in}\\ \text{the set}\end{array}\right) = \left(1 - e^{-\frac{Kn}{m}}\right)^K$$

$$\parallel$$

False positive rate

- Using more space reduces false positive rate
- Adding more items increases false positive rate

- For a given value of m, n, there is a sweet spot for the number of hash functions.

$$\boxed{K = \frac{m}{n} \ln 2}$$

$$\boxed{\epsilon = 2^{-K}}$$

Q:- We have 6 TB of database of 512B keys (≈ 12.88 Billion keys)
We want a false positive rate ~ 1.56%

We need:-
6 37 bit hash functions

≈ 1 Byte / Key, Total: ~ 12 GB

Single-hashing filters:-

- Use a hash function to hash items to a p-bit fingerprint
- Store fingerprints compactly in a hash table.

$$\underbrace{\longleftrightarrow}_{\lg |U| \text{ bits}}^{x} \longrightarrow \underbrace{\longleftrightarrow}_{P\text{-bits}}^{h(x)}$$

- Only source of false positives:
  - Two distinct elements x and y, where $h(x) = h(y)$
  - if x is stored and y isn't, query(y) gives a false positive

$$\boxed{Pr(x \,\&\, y \text{ collide}) = \frac{1}{2^P}}$$

How to store fingerprints compactly.

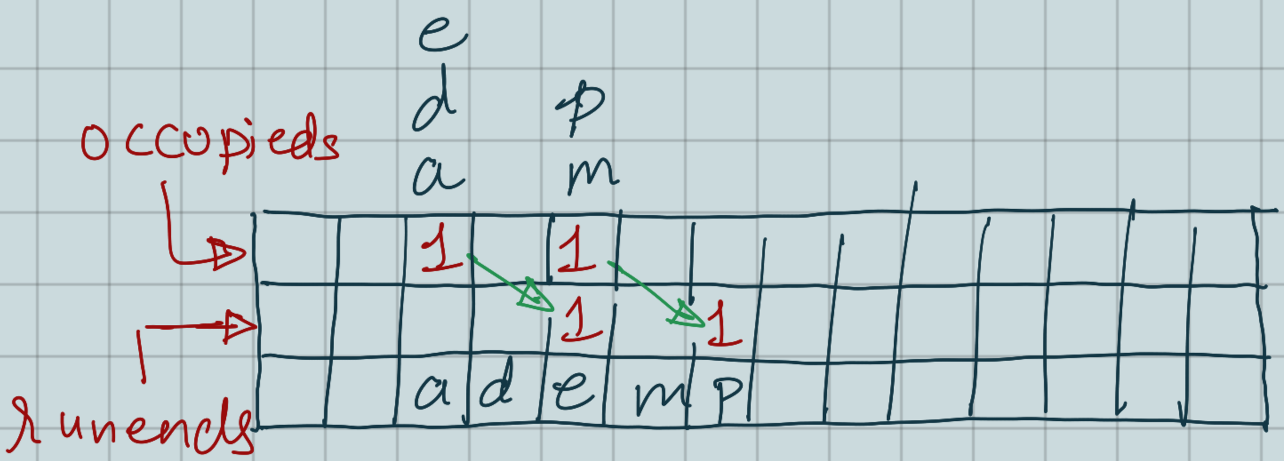
Quotienting:-

$$\underbrace{h(x)}_{p} \longrightarrow \boxed{\underbrace{b(x)}_{q} \,\big|\, \underbrace{t(x)}_{r}}$$

b(x): location in the table
t(x): " stored in the table

Q: how handle collisions?

↳ Linear probing
↳ Robin hood hashing.

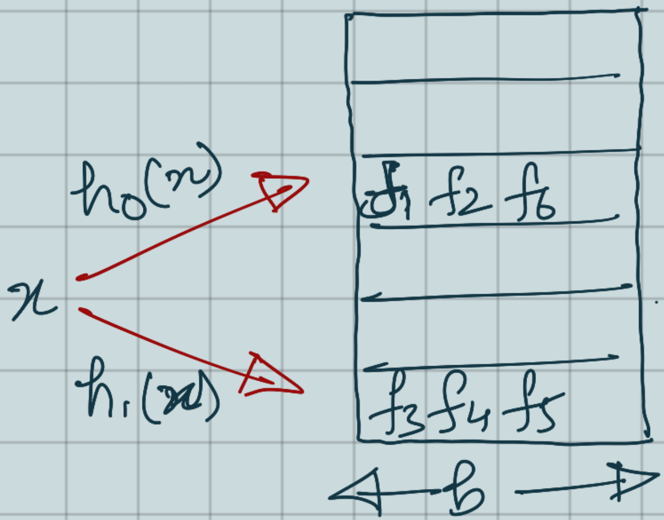


$$\boxed{\text{Space} : \approx n \lg\left(\frac{1}{\epsilon}\right) + 2.125\, n}$$

$$\boxed{\text{false positive rate} : \frac{1}{2^r}}$$

Cuckoo hashing :-

Typically, $b = 4$.



1. Compute $h_0(x)$ & $h_1(x)$
2. Insert $f(x)$ into emptier block
3. Kick an item if needed.

Note :- $h_0(x)$ & $h_1(x)$ need to be dependent to support kicking

Space : $\approx n \lg\left(\frac{1}{\epsilon}\right) + 3n$

false positive rate: $\approx \dfrac{2b}{2^f}$