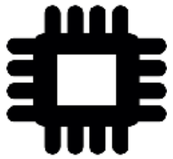
# The Memory Hierarchy & Bloom Filters

**Niv Dayan**

# Memory Hierarchy

CPU caches 

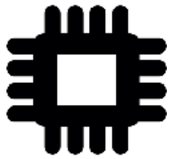
memory 

SSD 


disk 

# Memory Hierarchy

CPU caches



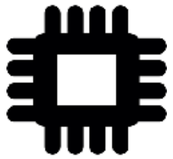
memory

SSD

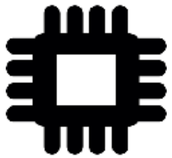

disk

**Fast & Expensive**

**Slow & Cheap**

| | | Latency |
|---|---|---|
| CPU caches |  | **10 ns** |
| memory |  | **100 ns** |
| SSD |  | **100 µs** |
| disk |  | **10 ms** |

# Latency

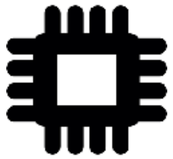


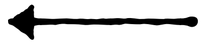
10 ns

100 ns

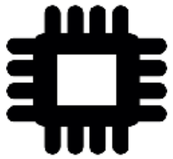

100 µs

10 ms



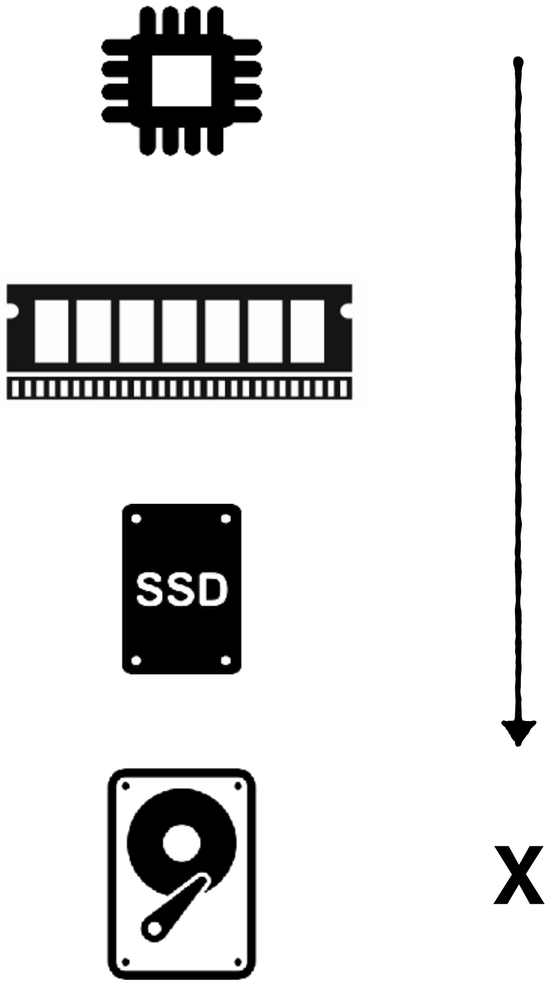
**Most data is here**




**Sometimes, we must fetch data from storage**

¯_(ツ)_/¯

**get key X**




Sometimes, we must fetch data from storage

¯_(ツ)_/¯

# But sometimes, user search for non-existing data

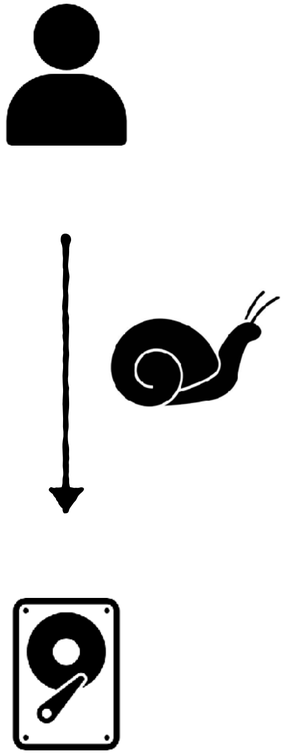
get key Q

{X, Y, Z}

# But sometimes, user search for non-existing data

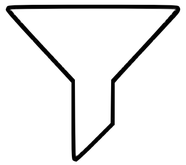
e.g., does key Q exist?

{X, Y, Z}



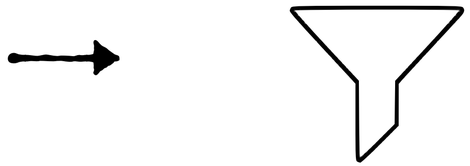
**Can we quickly tell if a key does not exist so that we don’t have to search storage?**

# What is a Filter?



# What is a Filter?

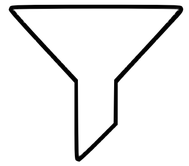
Does X exist?

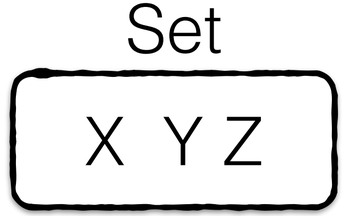


Set

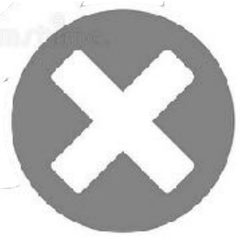
X Y Z

# What is a Filter?

Does X exist? →  →




**No false negatives**



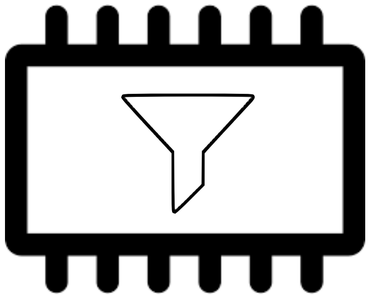
**false positives with tunable probability**

**Does key X exist**

Data

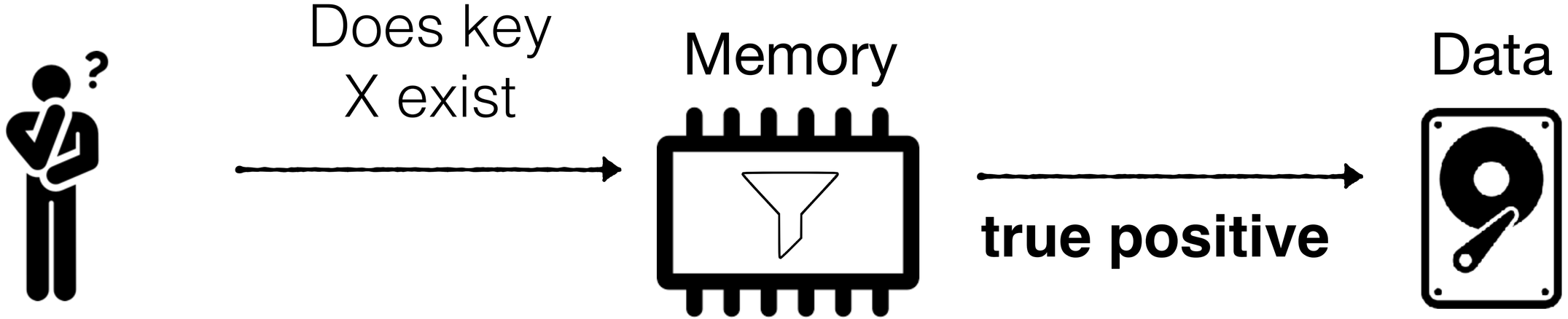
**Does key X exist**

**Memory**

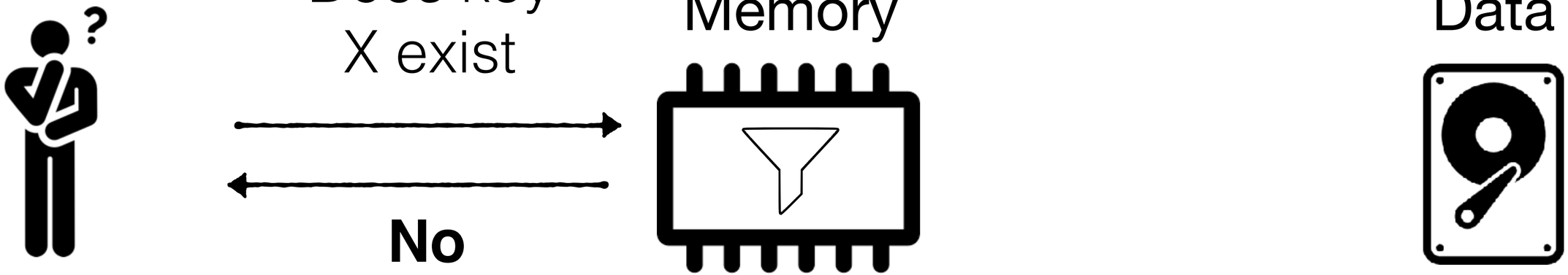


Data

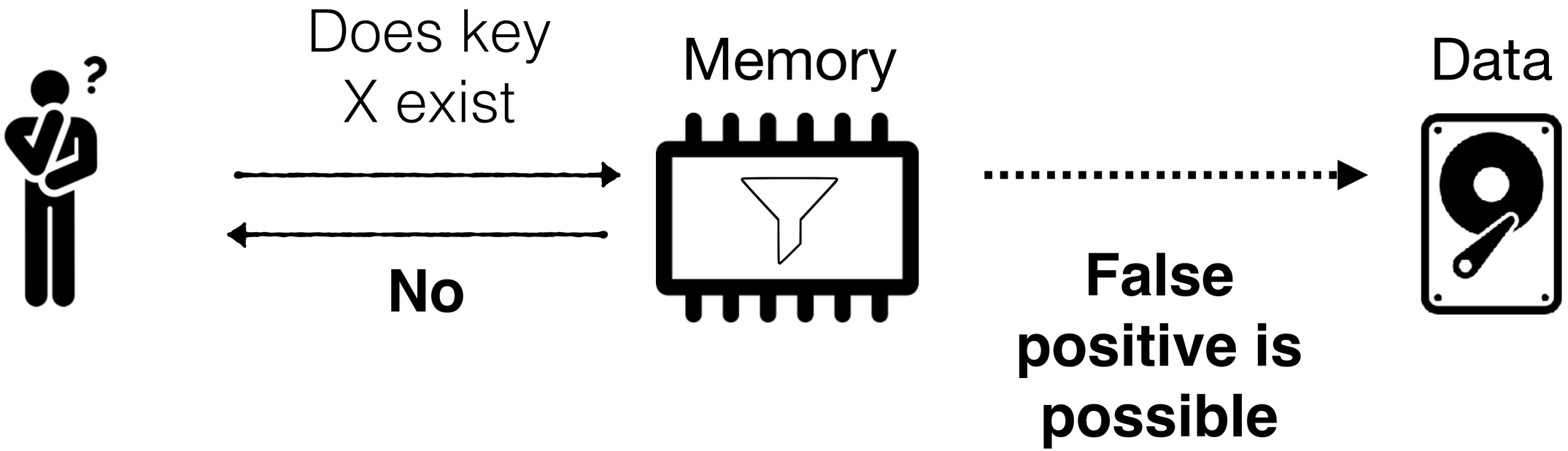
# If key X exists



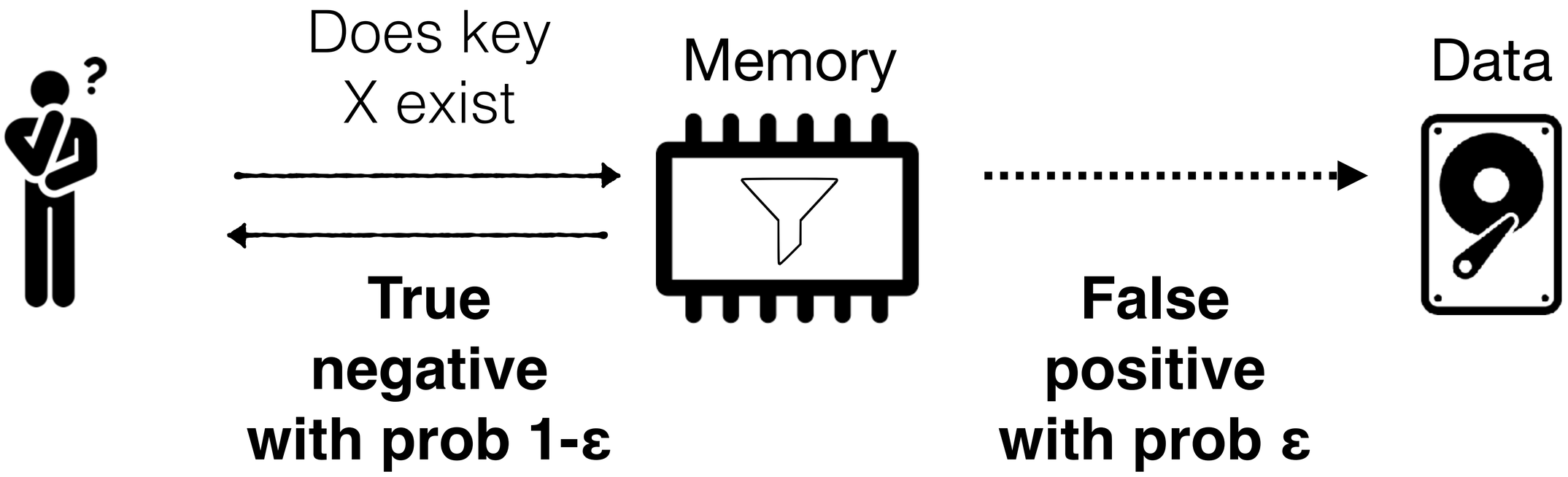
# If key X does not exist



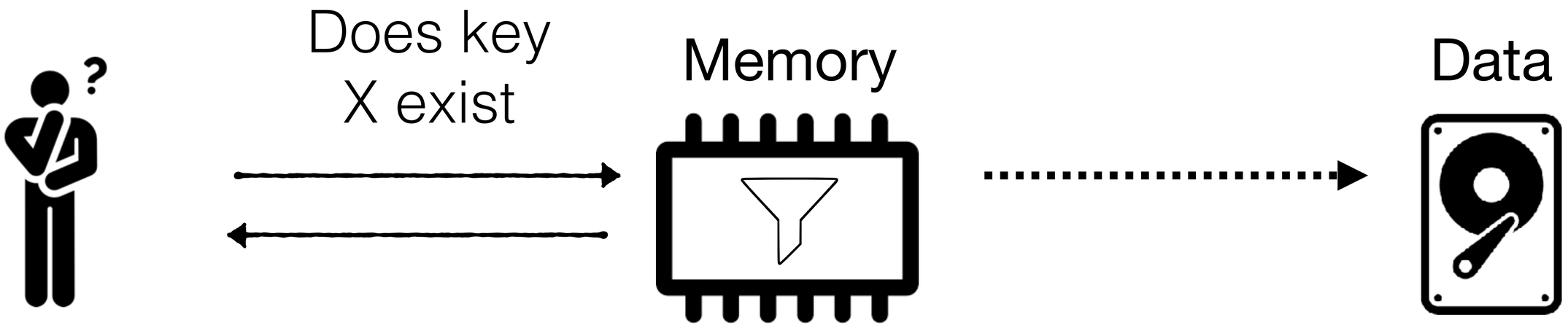
# If key X does not exist



# If key X does not exist



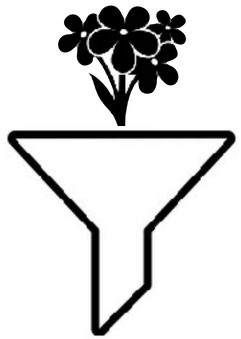
# If key X does not exist



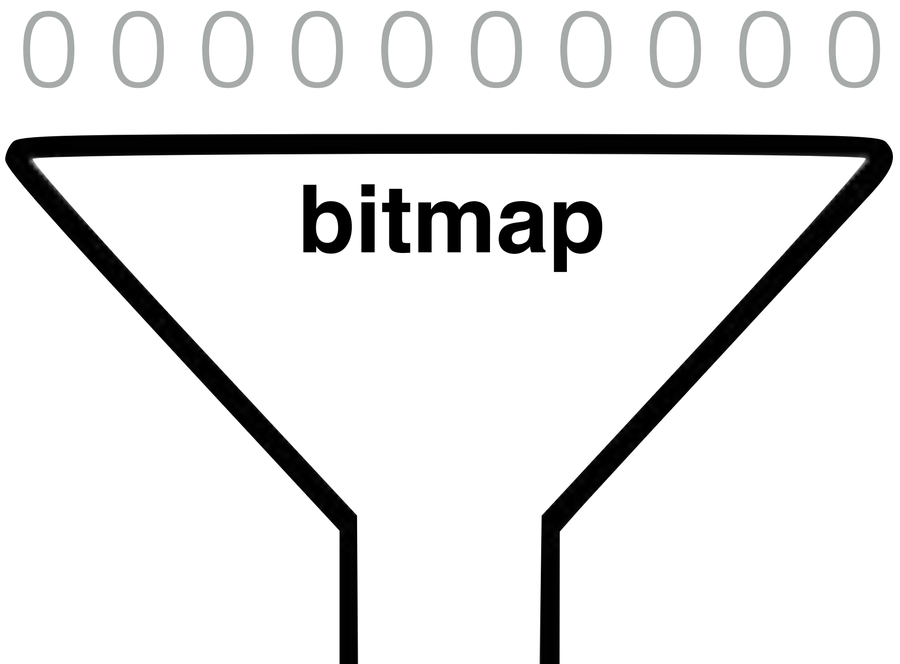

**ε - false positive rate - FPR**

# Bloom Filters

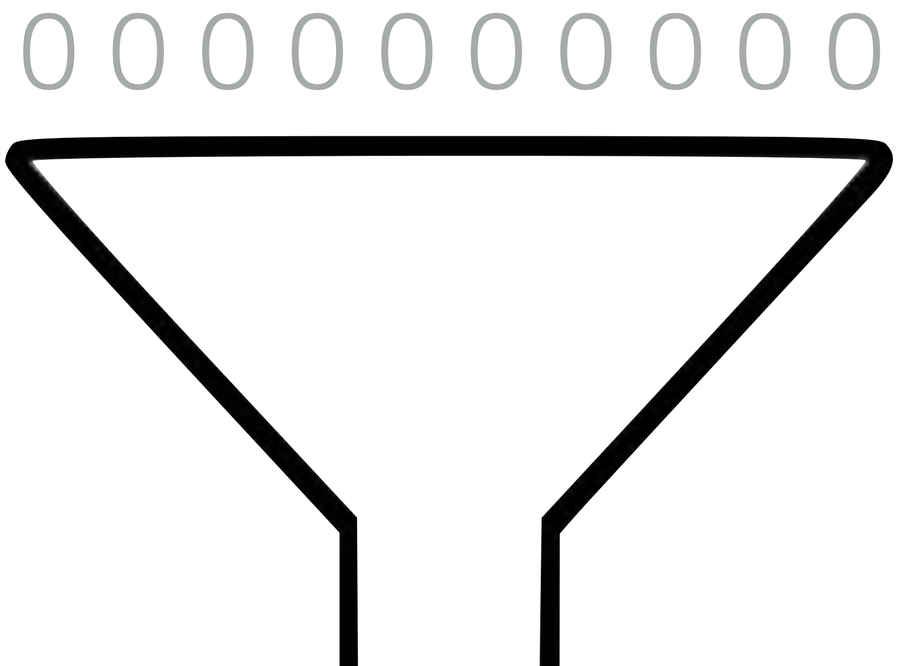
Space/time Trade-Offs in Hash Coding with Allowable Errors
Burton Howard Bloom. Communications of the ACM, 1970.



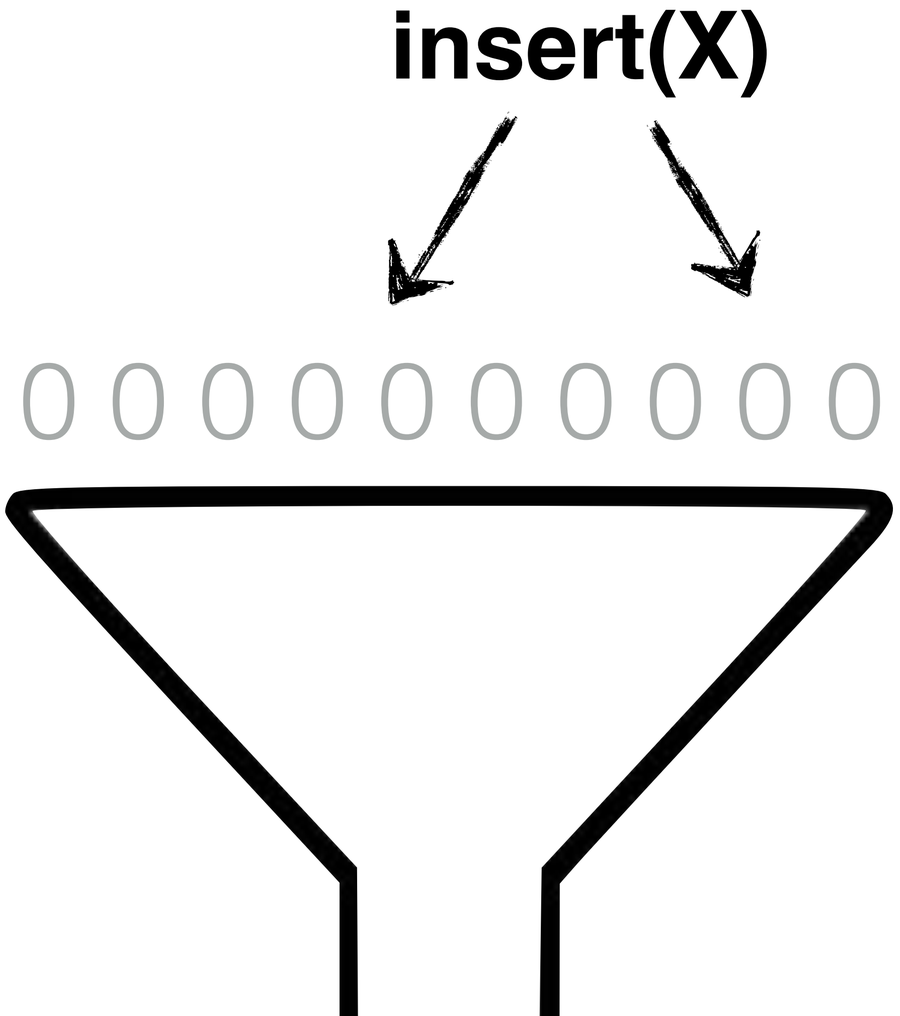
# *k* hash functions



# Inserts?

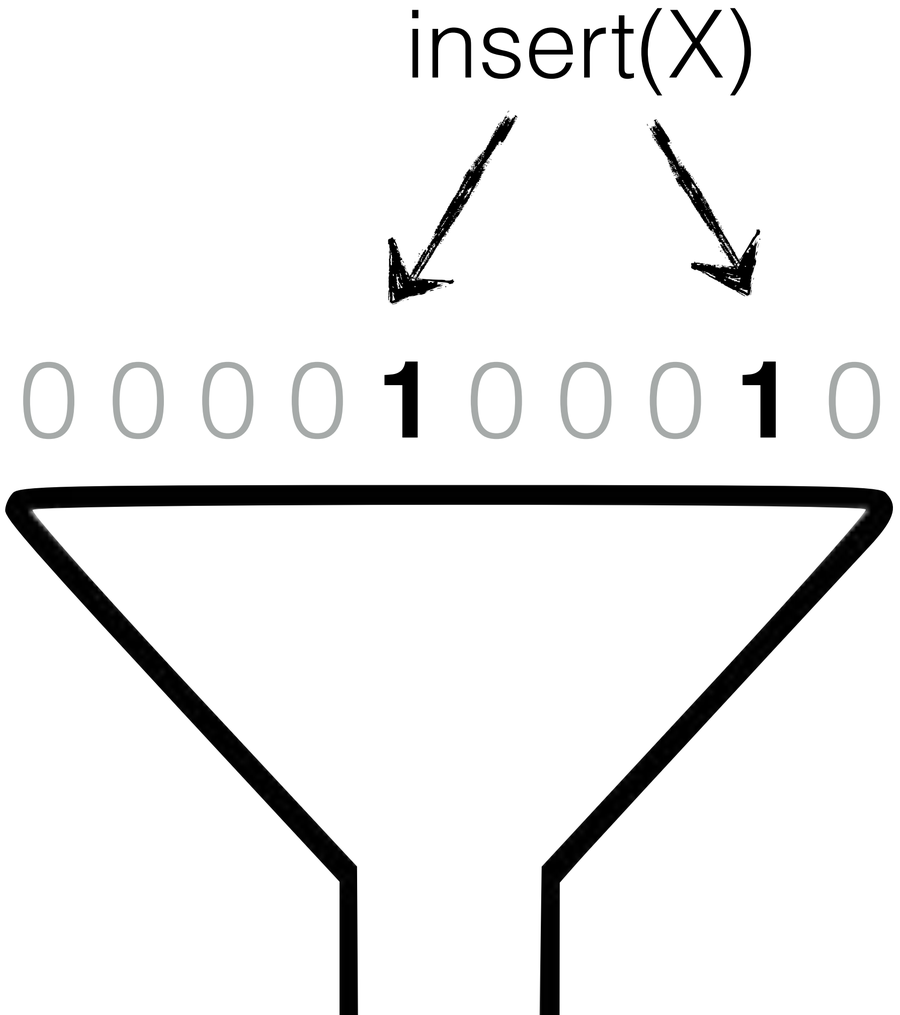


# Inserts: **hash key to k random bits**



# Inserts: hash key to k random bits

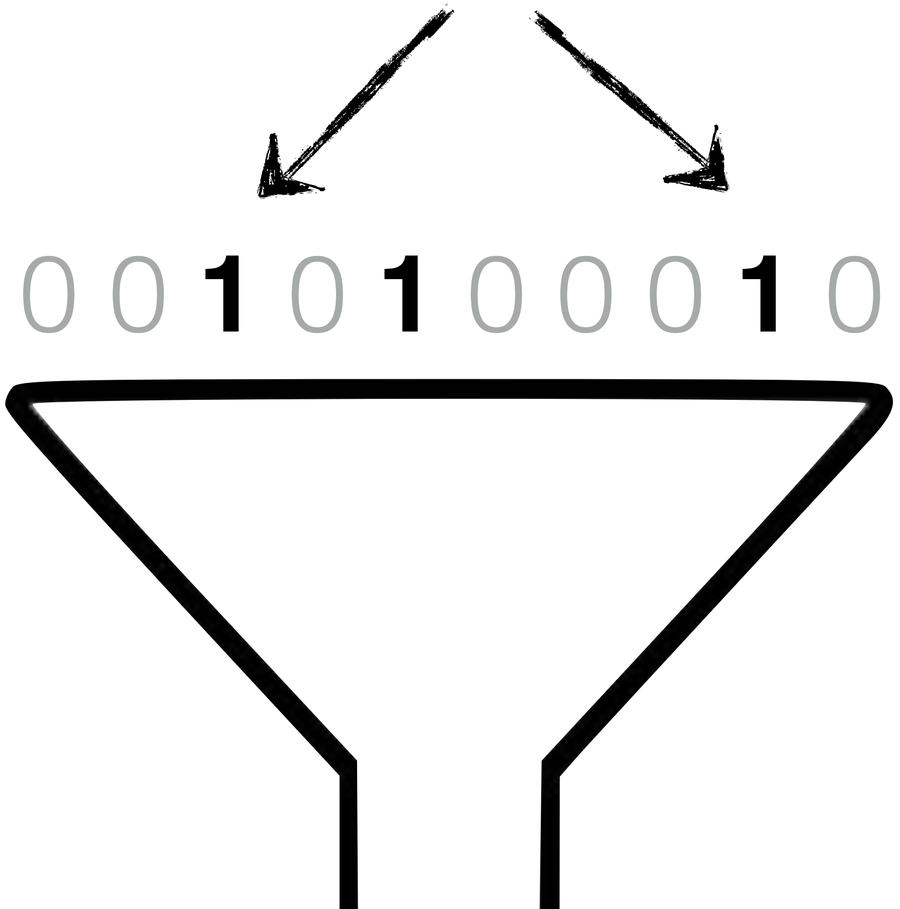
**Set from 0 to 1 or keep 1**



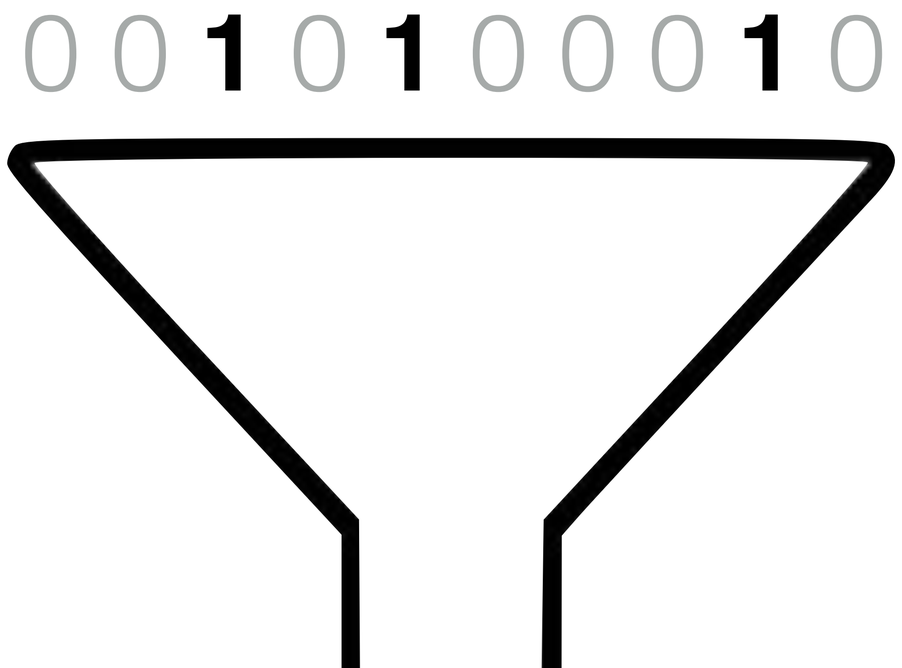
Inserts: hash key to k random bits

**Set from 0 to 1 or keep 1**

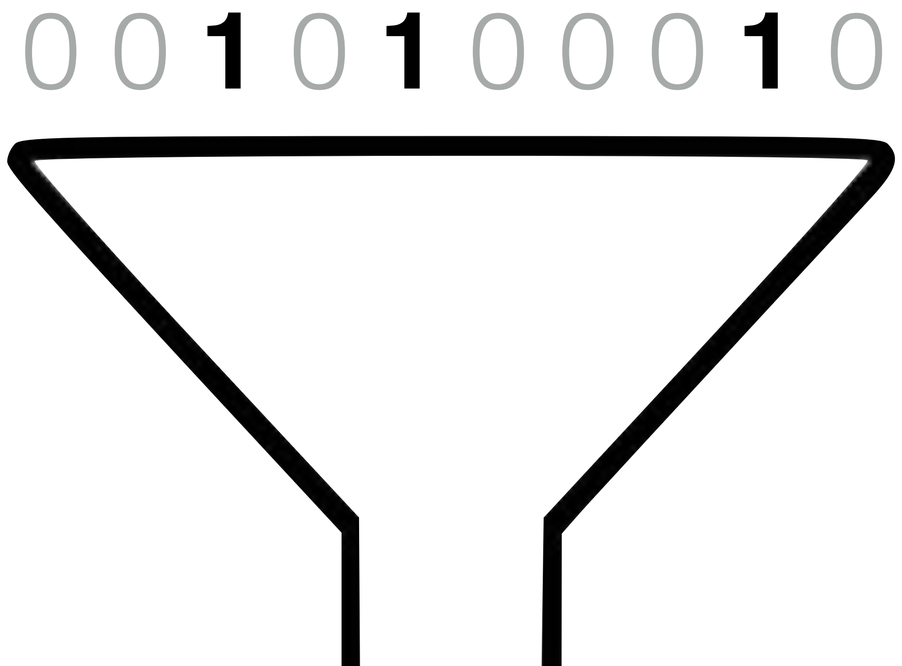
**insert(Y)**



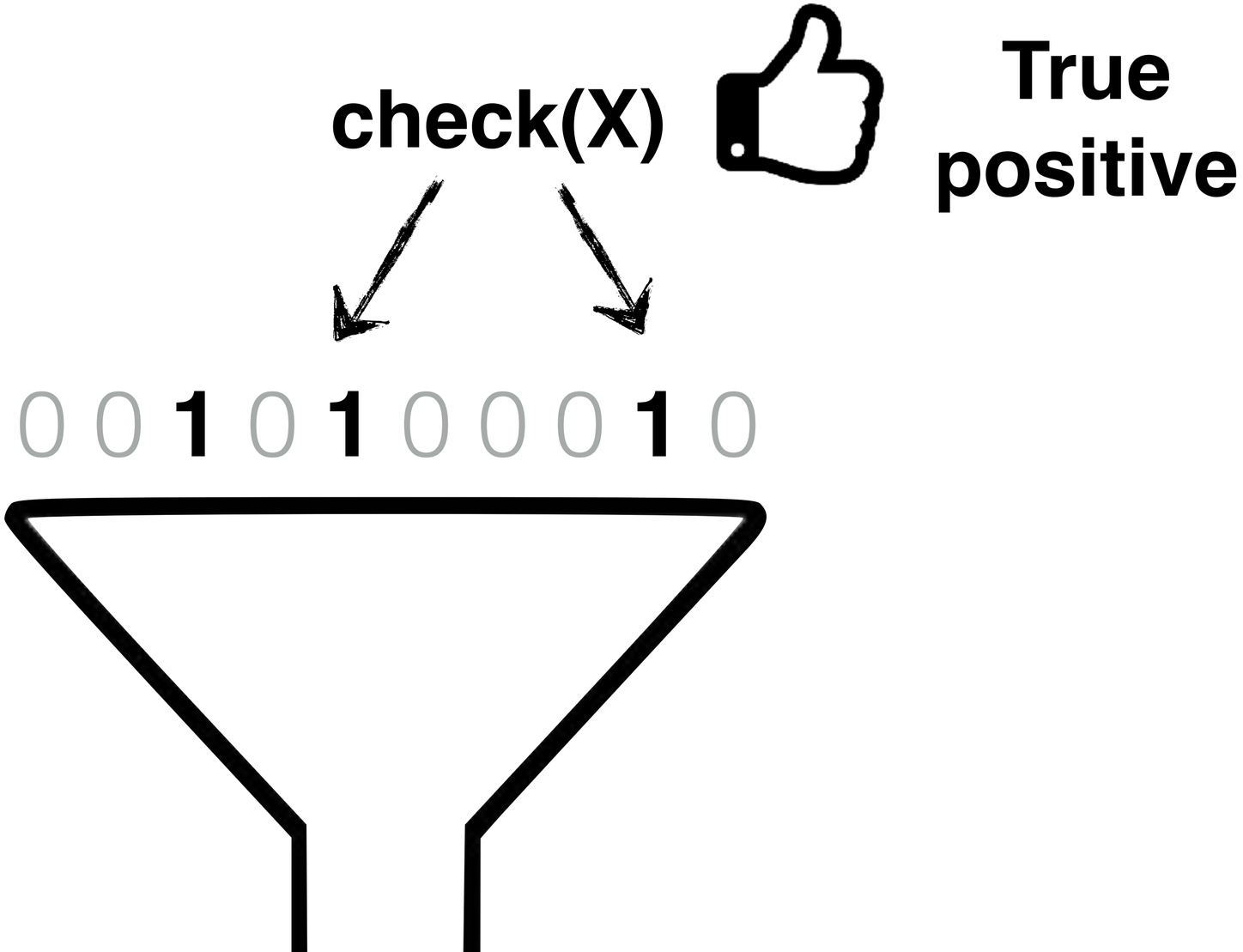
# Queries?



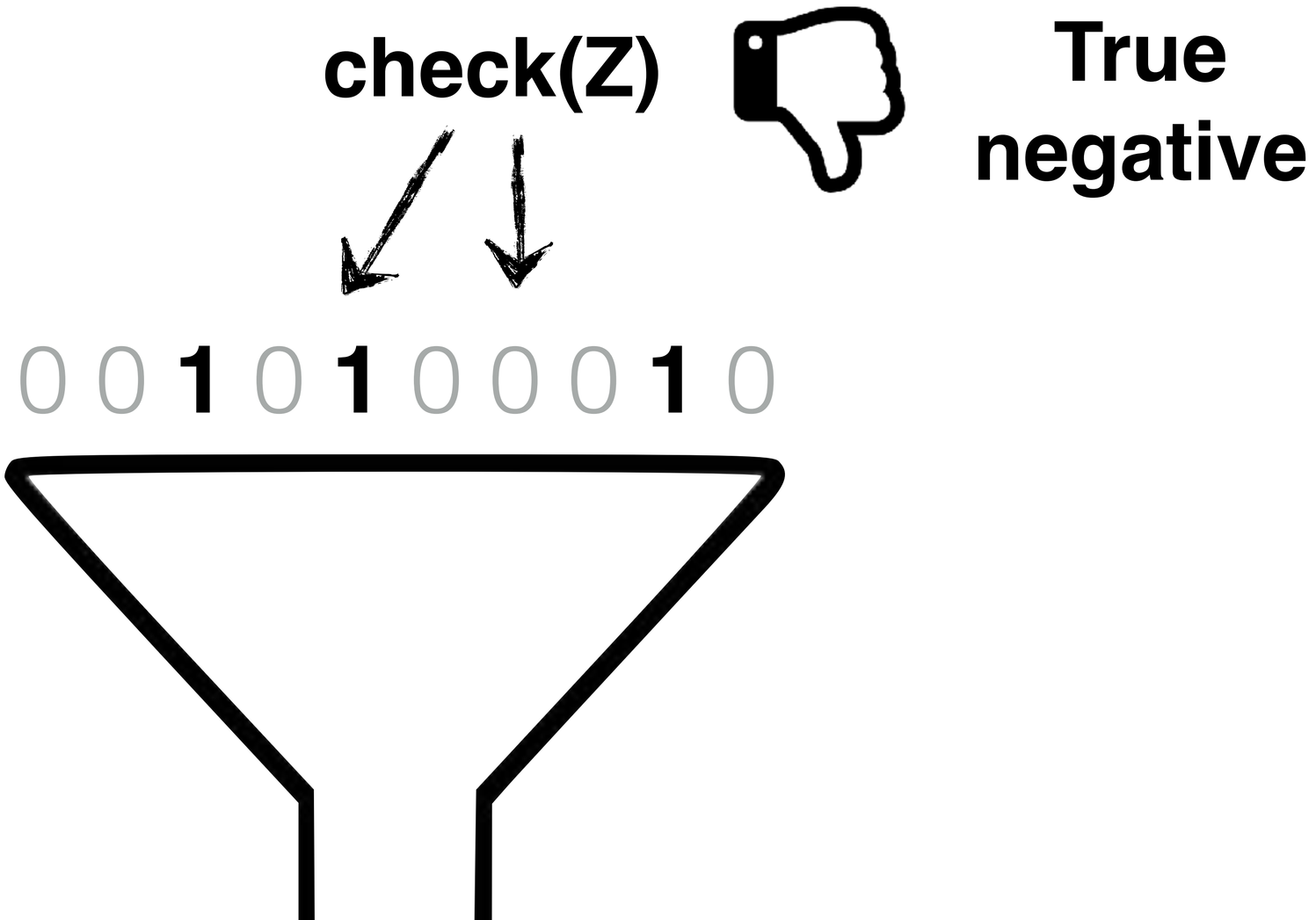
Queries: **return positive if all hashed bits are 1s**



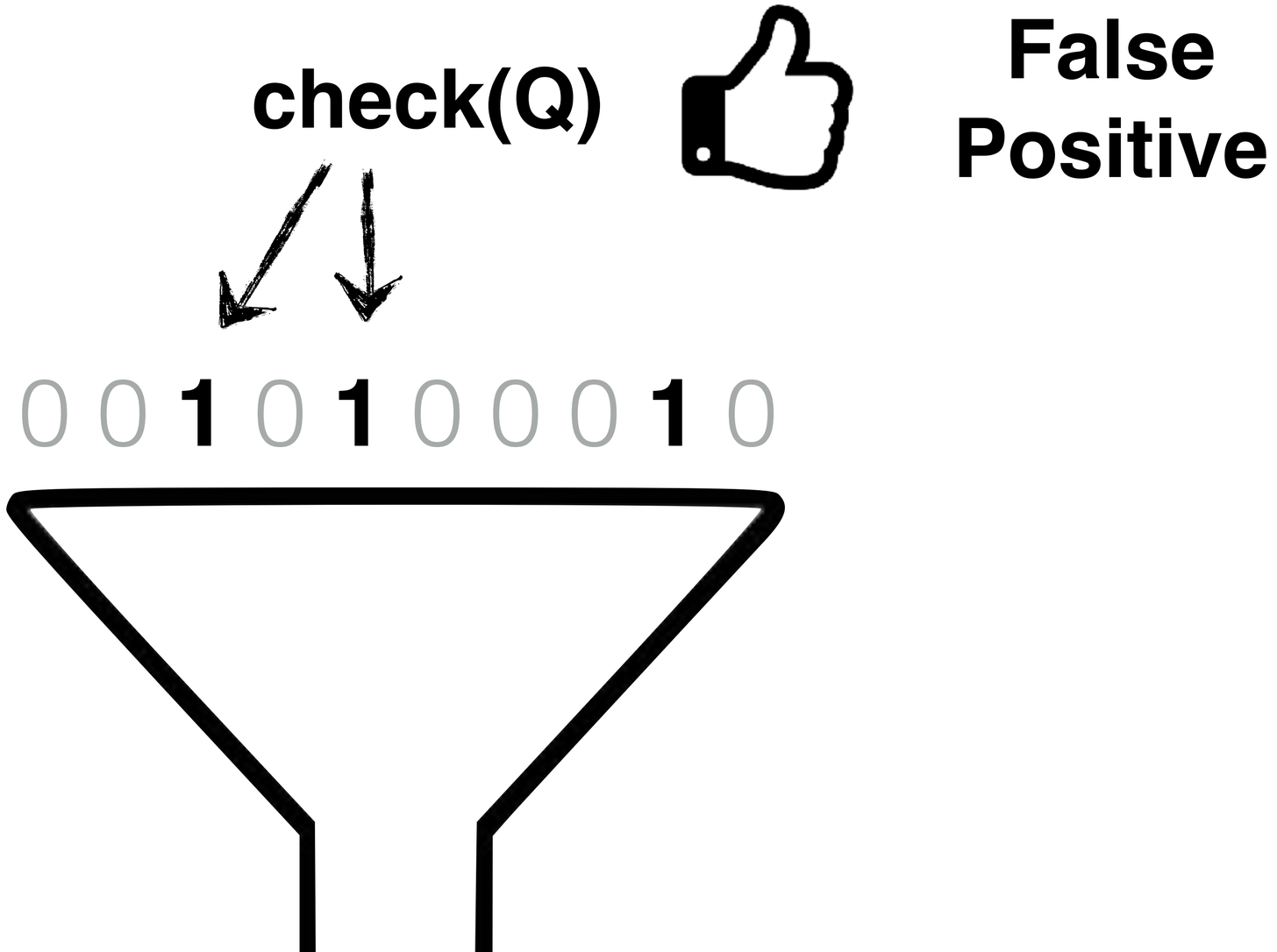
# Queries: return positive if all hashed bits are 1s



# Queries: return positive if all hashed bits are 1s

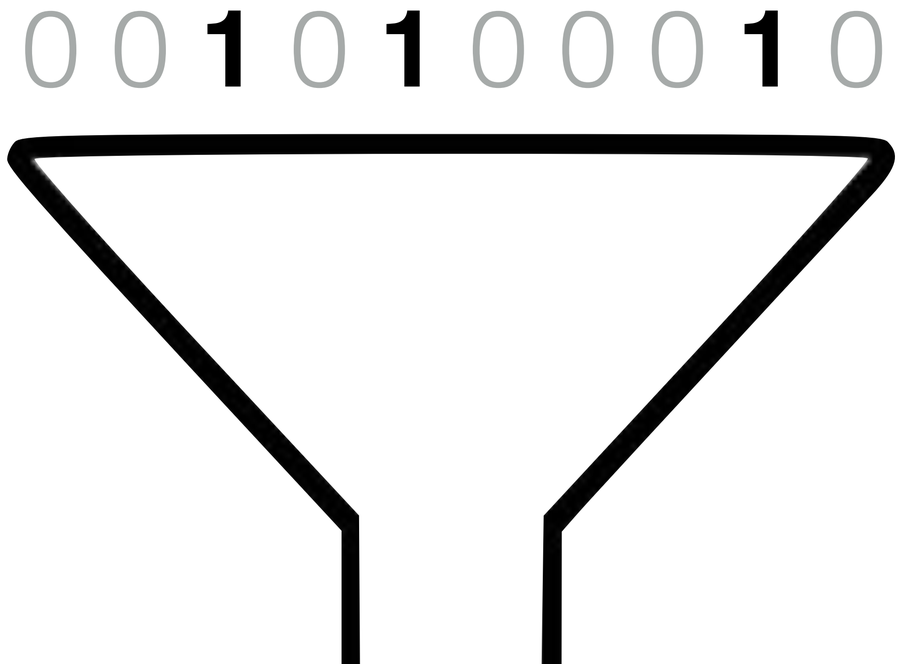


# Queries: return positive if all hashed bits are 1s



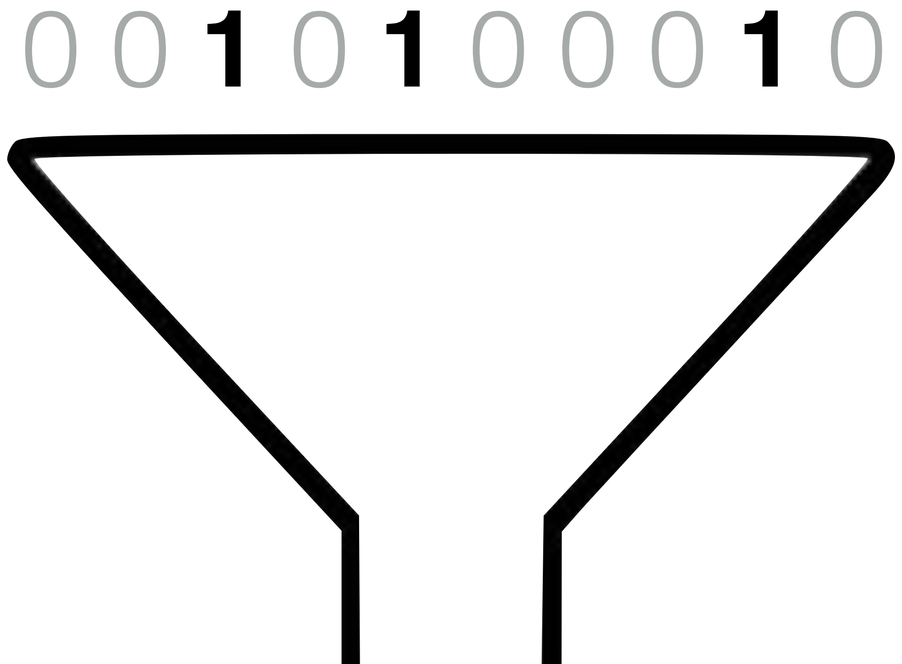
**Can the filter ever return a false negative?**

i.e., report that an existing key does not exist…



Can the filter ever return a false negative?

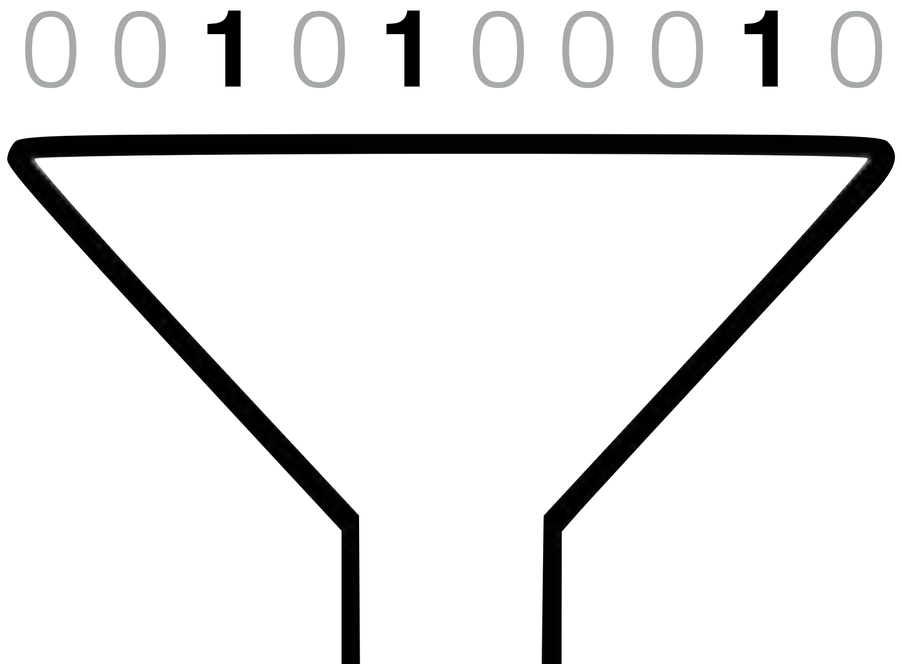
**No - the corresponding bits will always be 1s**



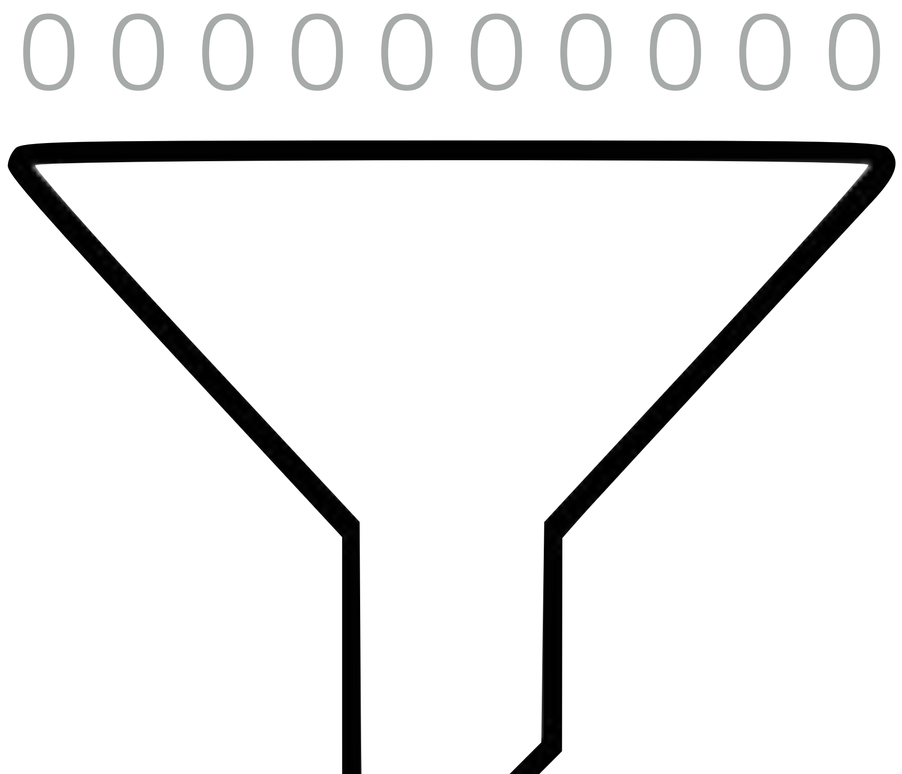
Can the filter ever return a false negative?

No - the corresponding bits will always be 1s

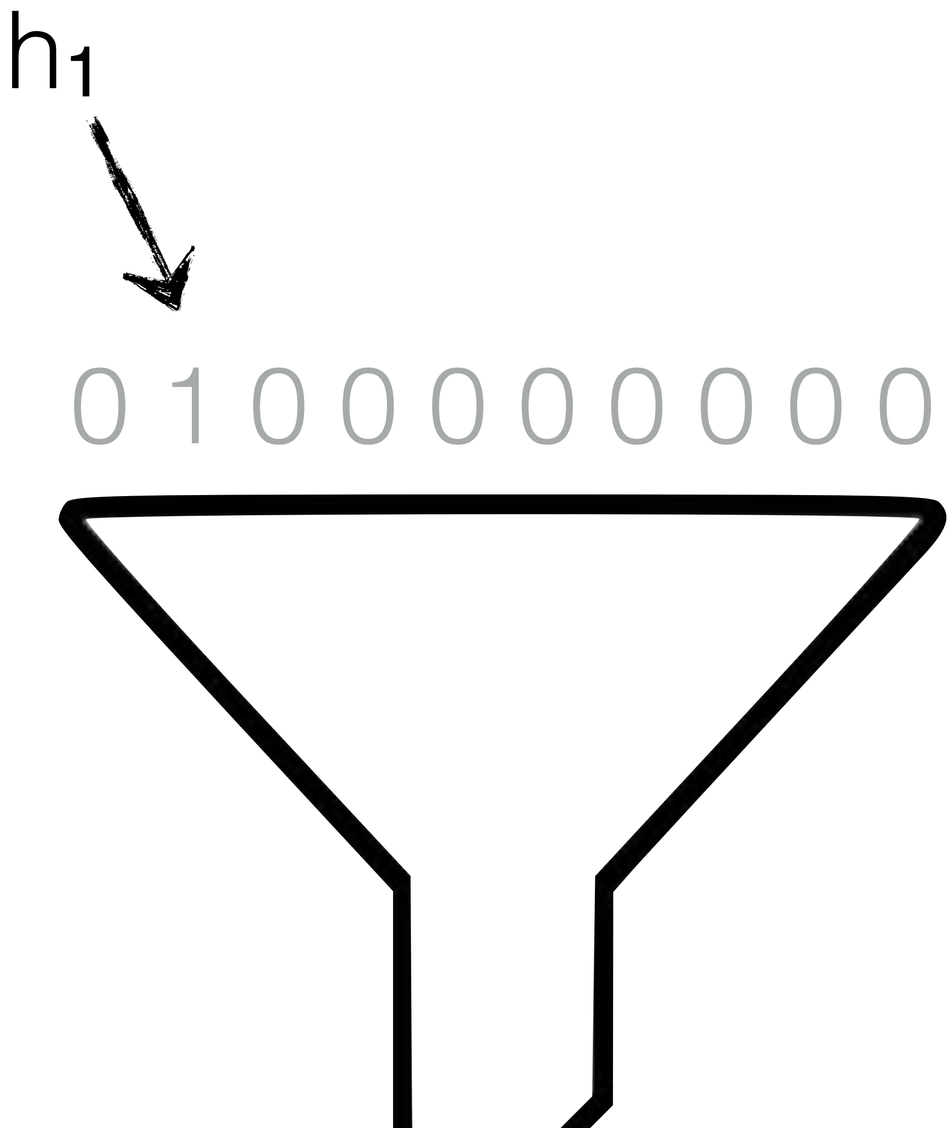
**False negatives are not allowed…**



# How many hash functions should we use?

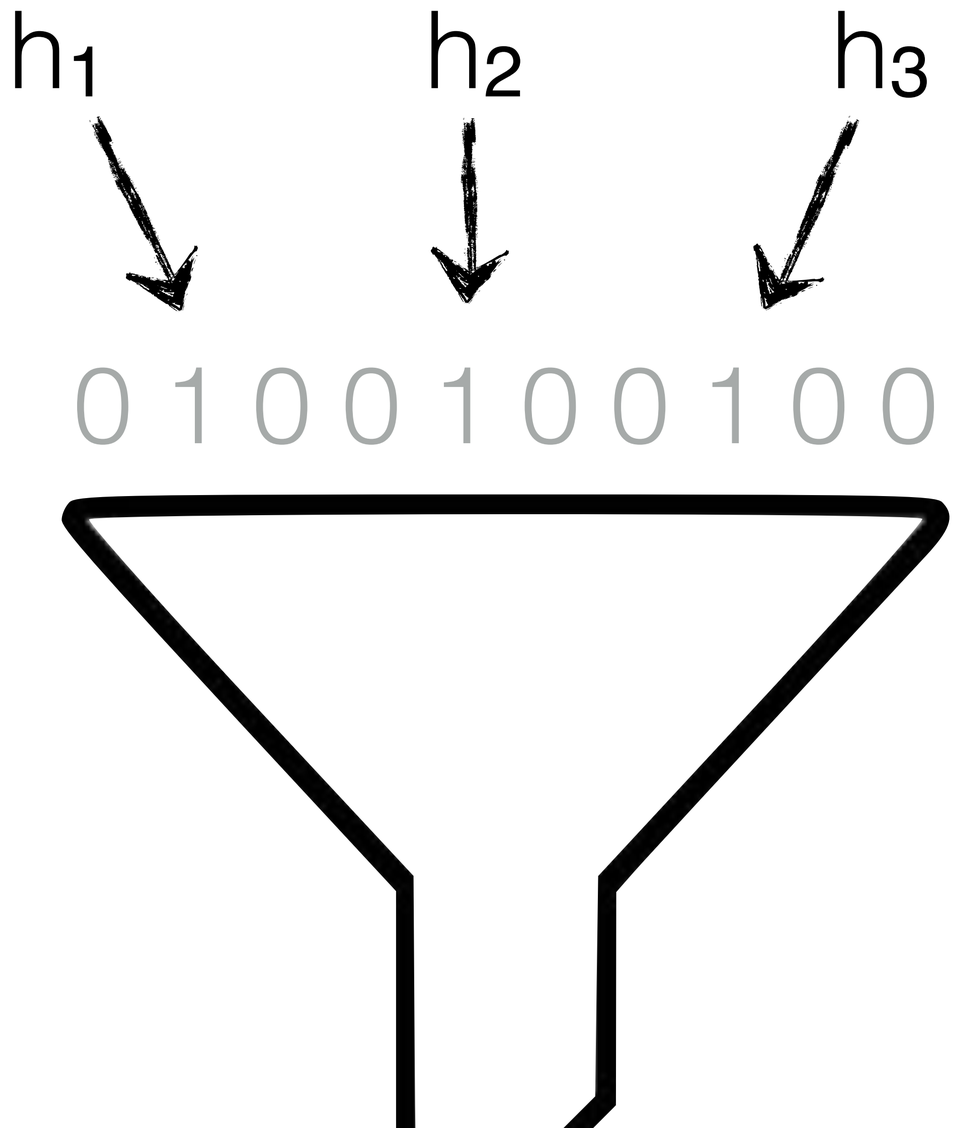


# How many hash functions should we use?




**One is too few: false positive occurs whenever we hit a 1**

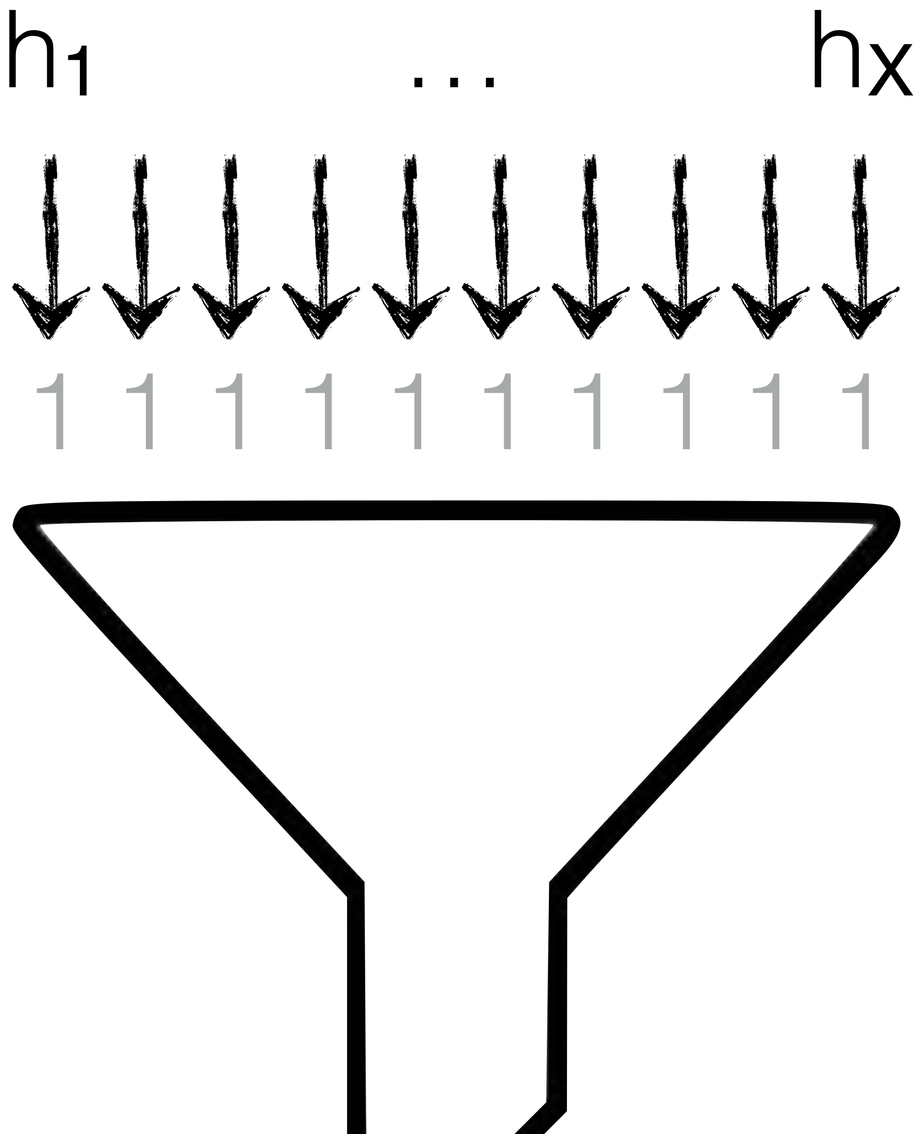
# How many hash functions should we use?




One is too few: false positive occurs whenever we hit a 1

**By adding hash functions, we initially decrease the false positive rate (FPR).**

# How many hash functions should we use?

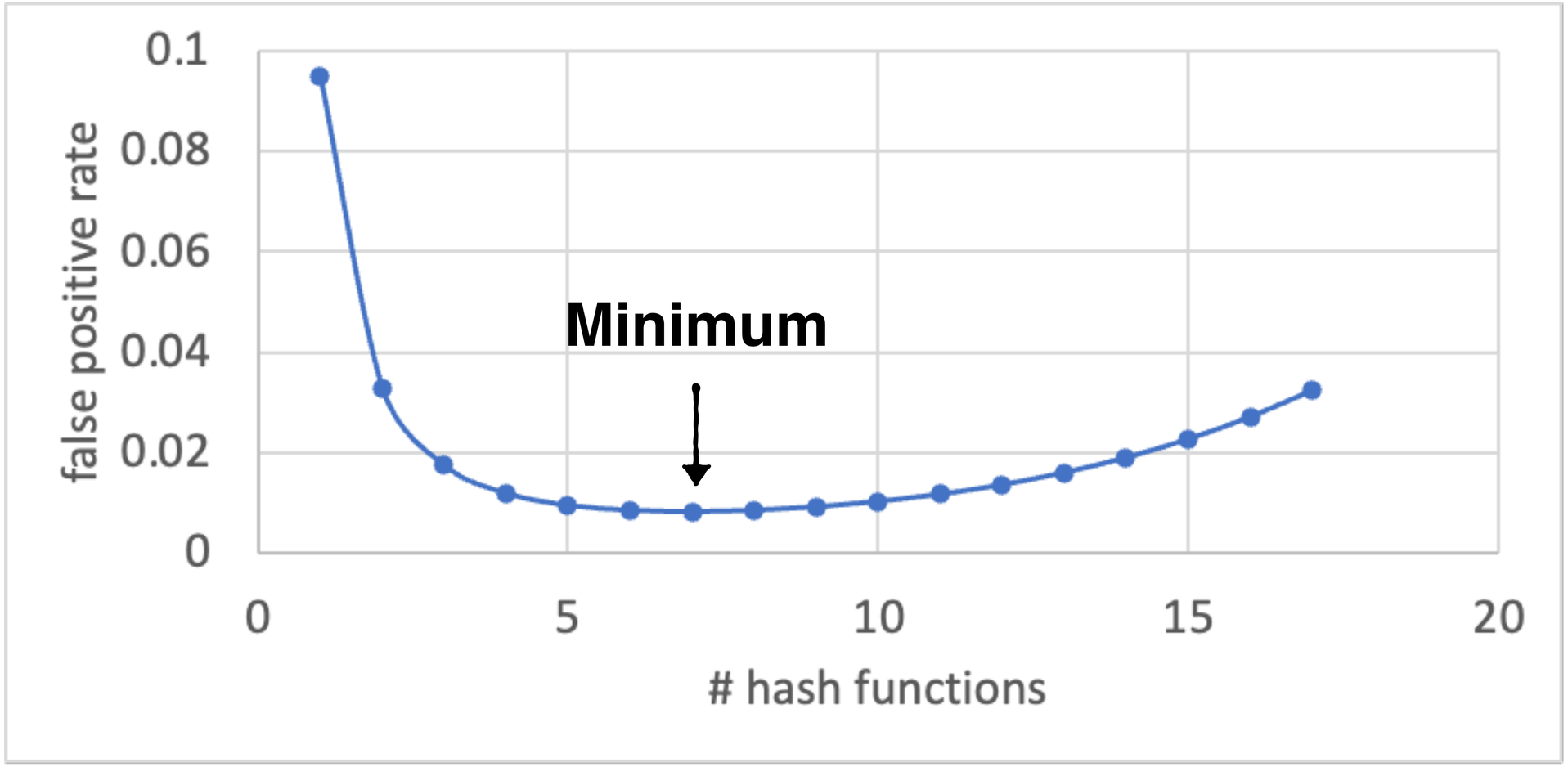



One is too few: false positive occurs whenever we hit a 1

By adding hash functions, we initially decrease the false positive rate (FPR).

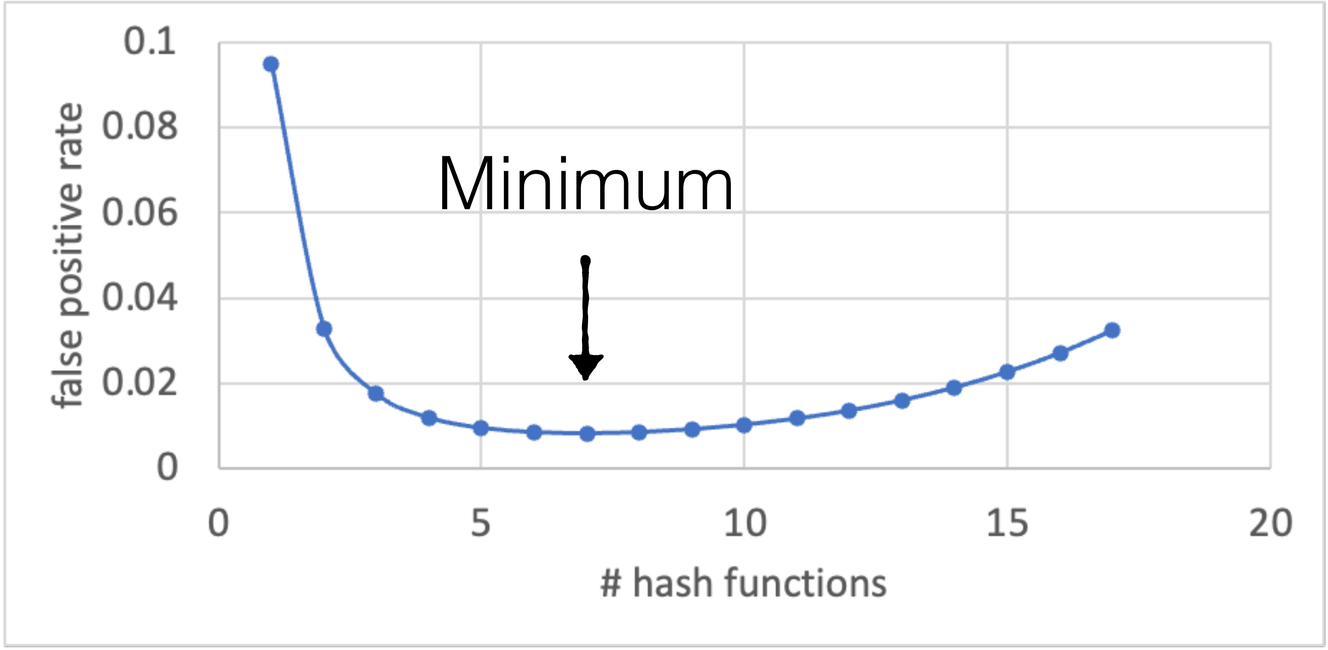
**But too many hash functions wind up increasing the FPR.**

# How many hash functions should we use?




(Drawn for a filter using 10 bits per entry)

# How many hash functions should we use?

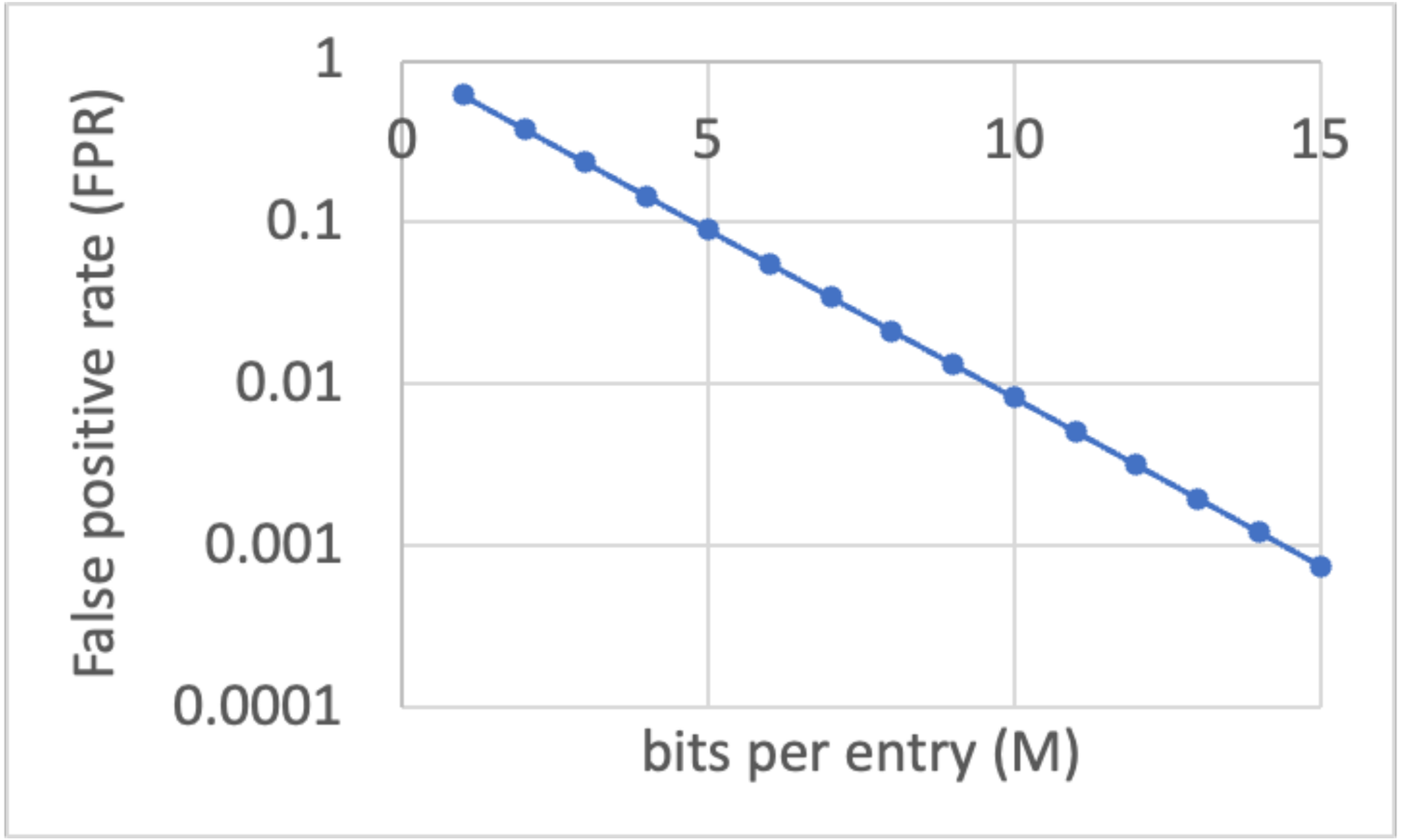



**Optimal # hash functions = ln(2) · *M***

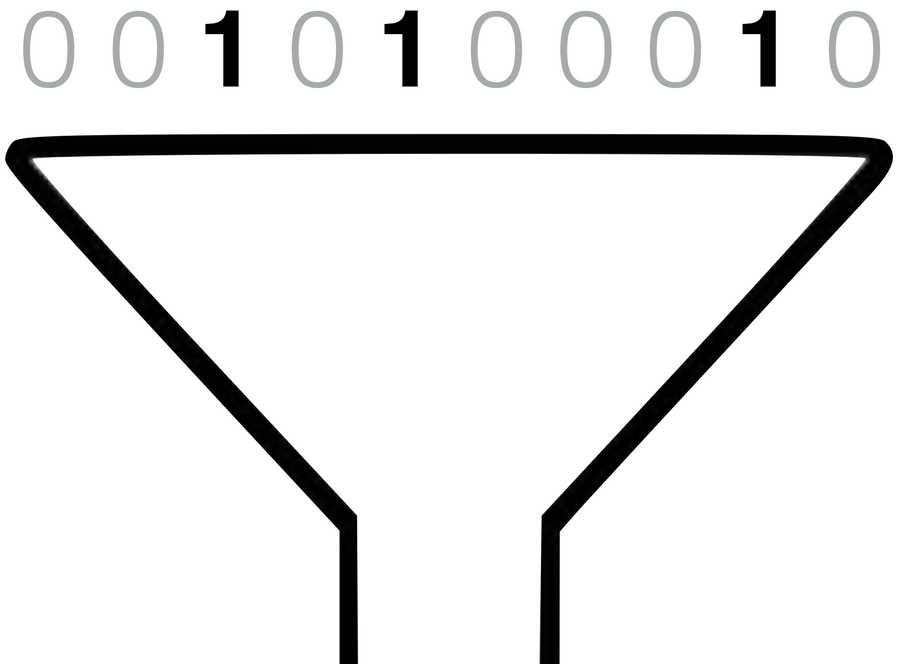
(*M* is the number of bits per entry)

assuming the optimal # hash functions,

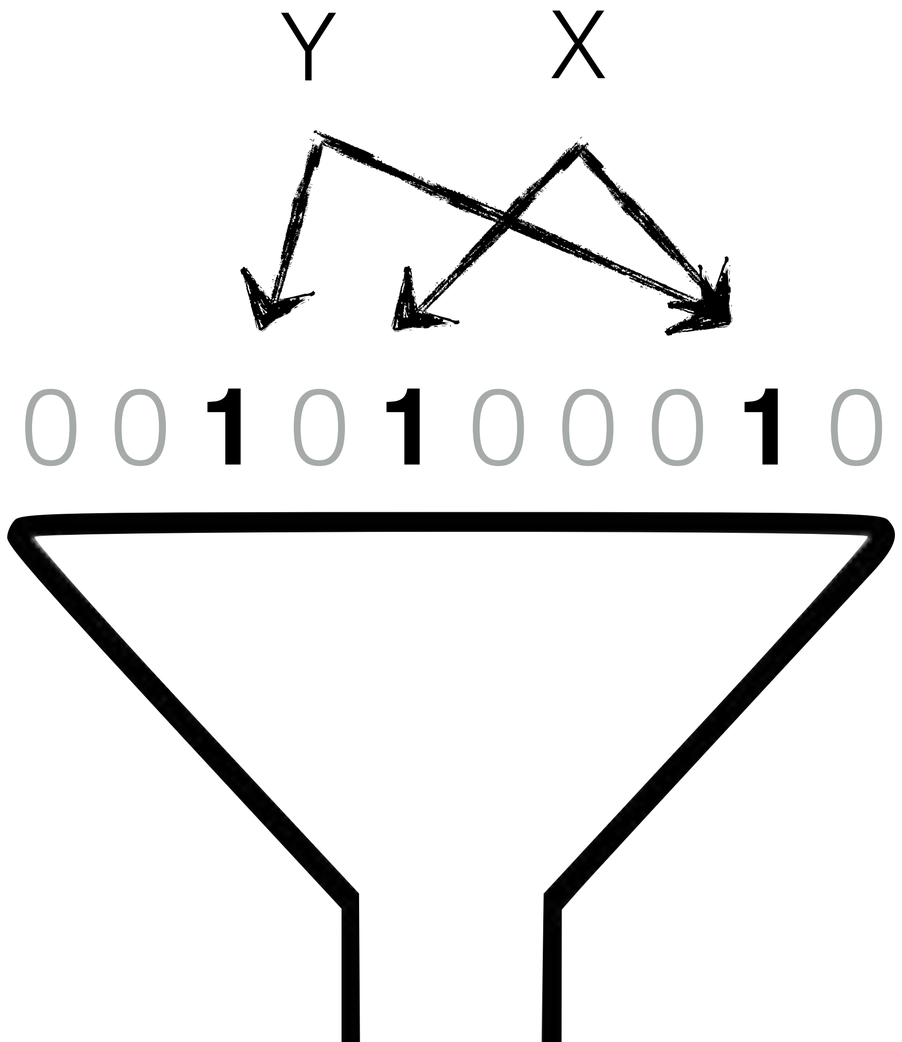
**false positive rate =** $2^{-M \cdot \ln(2)}$



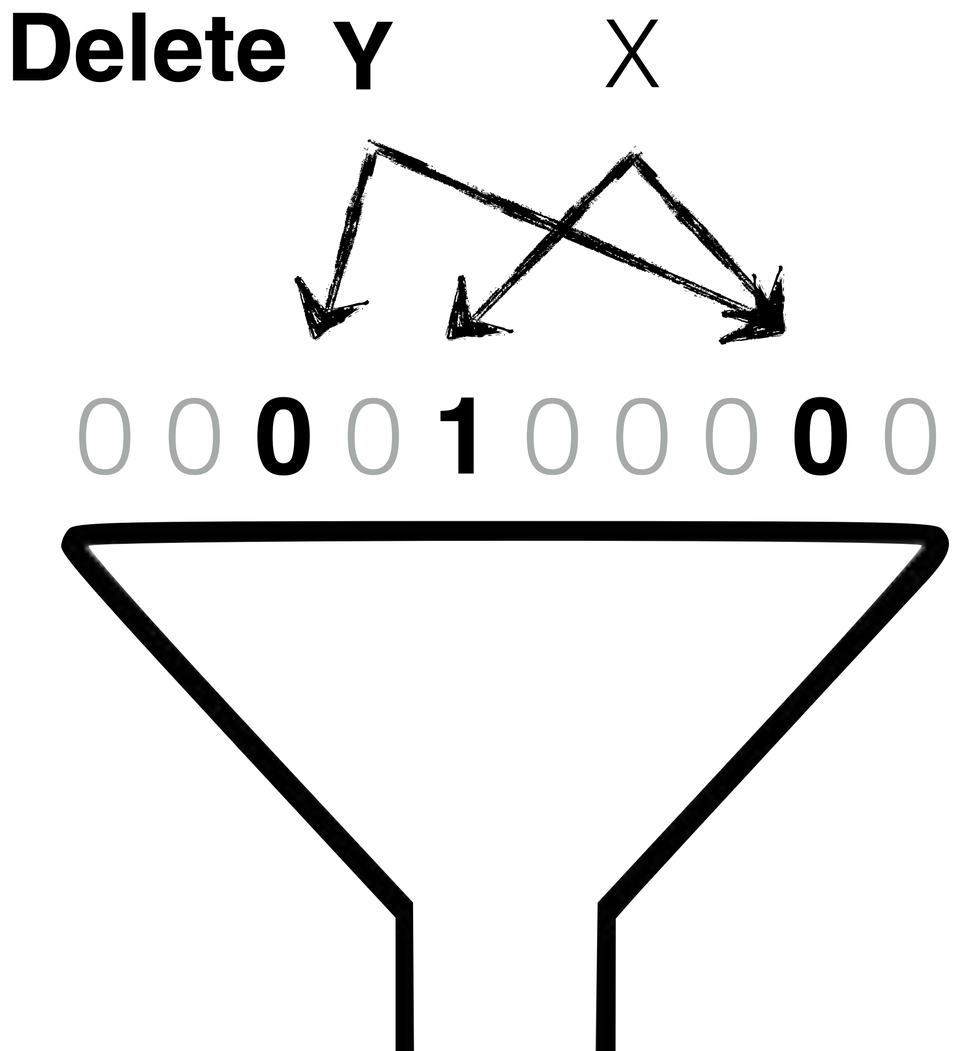
# deletes?



# deletes?

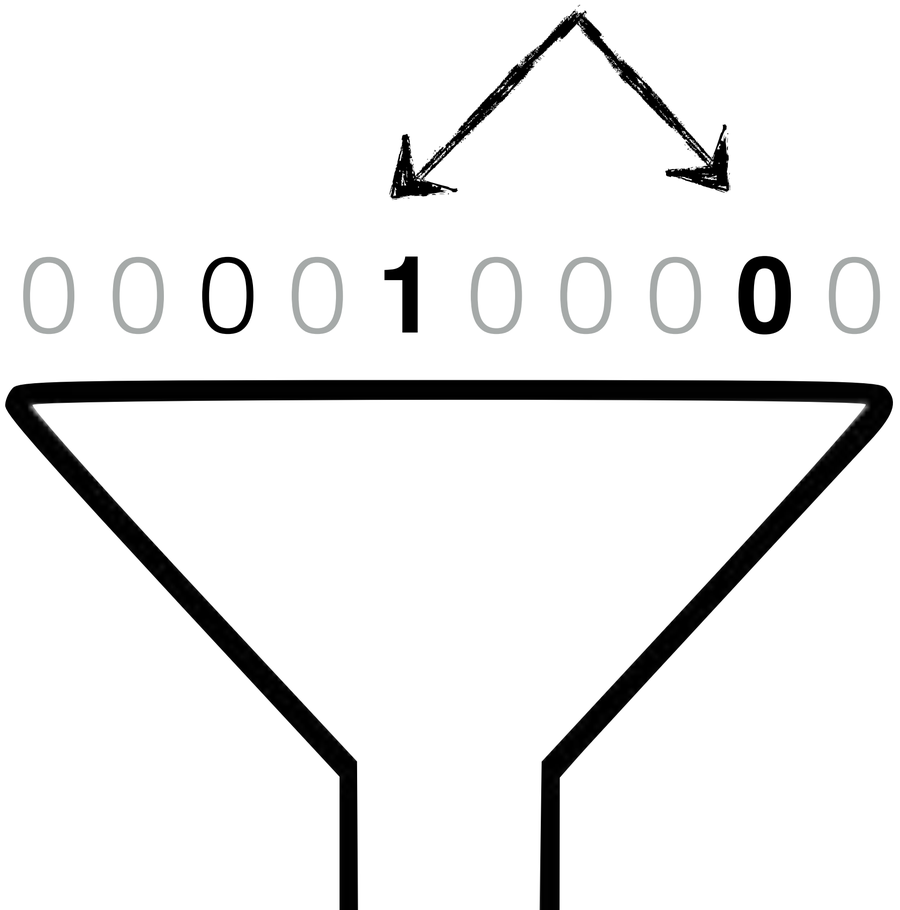


# deletes?



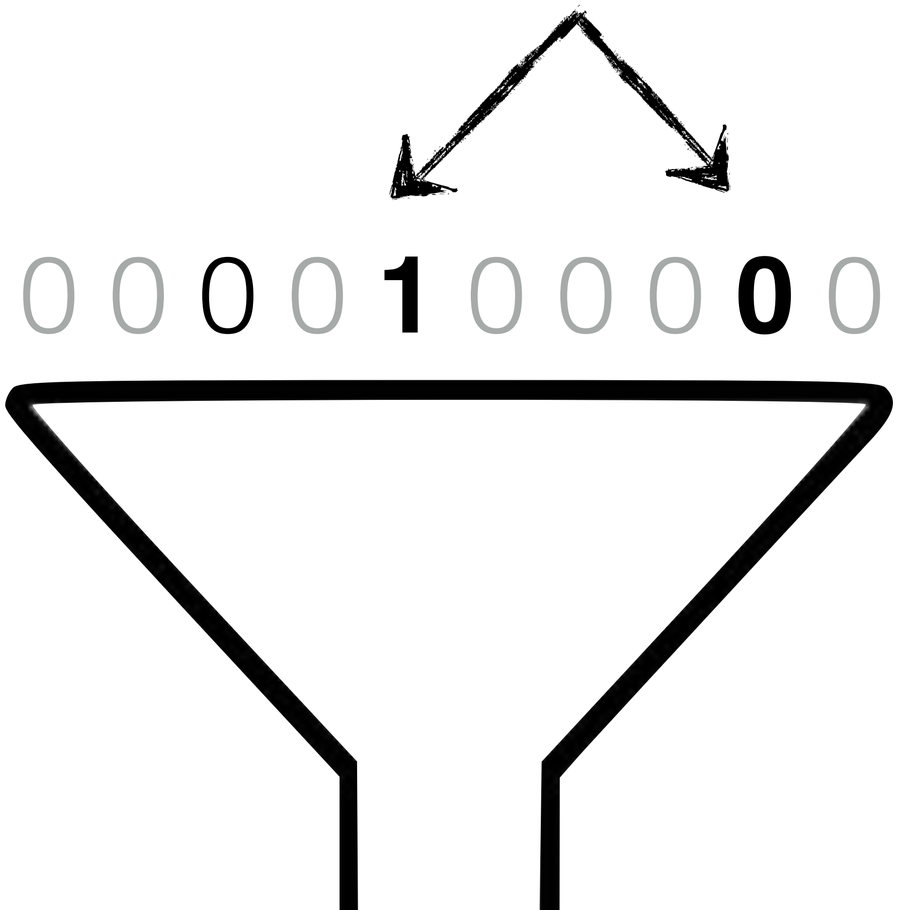
# deletes?

**Query x**



deletes?

Query X false negative



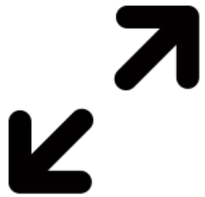
# Research Challenges

**deletes?**

# Research Challenges

deletes?

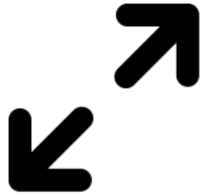
**Expansions?**



# Research Challenges

deletes?

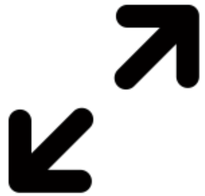
Expansions?



**Range?**

deletes?

Expansions?



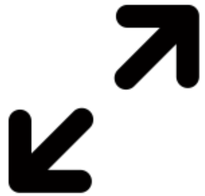
Range?

**All at the same time?**

# Subject to:

deletes?

Expansions?



Range?

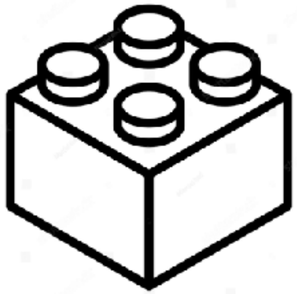
Subject to:

**High Performance**

**Small Size**

**Good Accuracy**

# Orca Lab



**CSC443: Database System Technology**

**CSC2525: Research topics in Database Management**