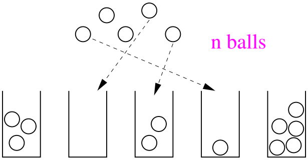
Balls and Bins: Hashing

November 3, 2019

Balls and Bins

Basic Model: Given n la distinguishable (labelled) balls we throw each one independently and uniformly into a set of the distinguishable (labelled) bins.

$$\mathbf{Pr}\left[\text{ball } i \ \rightarrow \ \text{bin } j\right] = \frac{1}{m}.$$



Probability space: $\Omega = \{(b_1, b_2, \ldots, b_n)\}$ where b_i denotes the index of the bin containing ball i-th. ball: $|\Omega| = m^n$.
For any $w \in \Omega, \mathbf{Pr}\left[w\right] = (\frac{1}{m})^n$

Balls and Bins as a model

Balls and Bins as a model, is very useful in different areas of problems in computer science. For ex.:

- The hashing data structure: keys are the balls and the slots in the array are the bins.
- Many situations in routing in nets: balls represent the connectivity requirements and the bins are the paths in the network
- The load balancing randomized algorithm, balls are the streaming jobs and the bins are the servers.

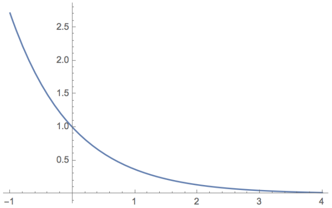
Recall as an application of Chernoff+UB, we proved that for n balls (jobs) and m bins (servers), under a uniform and independent distribution of jobs to servers, for $n >> m$, the probability the load of a server deviates from the expected load, was $1/m^2$.

General rules for the analysis of Balls & Bins

n balls to m bins.

- X_i a random variable counting the number of balls into bin-i. Then $X_i \in B(n, \frac{1}{m})$.
- As we know: $X_1, \dots X_m$ are not independent.
- The average load in a bin is $\mu = \mathbf{E}[X_i] = n/m$.
- Rule of thumb to do the analysis:
  - If $n >> m$, (μ large) use Chernoff bounds,
  - if $n = m$, ($\mu \in \Theta(1)$), use the Poisson approximation.

Recall that for small x, $e^{-x} \sim 1 - x$.



The Poisson Distribution

Recall that for $X \in B(n,p)$ if for large n and small p, we can have a good approximation: $\mathbf{Pr}[X = k] = \frac{e^{-\lambda}\lambda^k}{k!}$, where $\lambda = \mathbf{E}[X] = \mu = pn$.

For any $\lambda \in \mathbb{R}^+$, a r.v. X is said to have a Poisson $P(\lambda)$ distribution, if its PMF is $p_X(k) = \frac{e^{-\lambda}\lambda^k}{k!}$, for any $k = 0, 1, 2, 3, \ldots$

Notice p_X is a correct PMF, as:
$\sum_{k=0}^{\infty} \frac{e^{-\lambda}\lambda^k}{k!} = e^{-\lambda}(1 + \lambda + \frac{\lambda^2}{2!} + \cdots) = e^{-\lambda}e^{\lambda} = 1.$

Poisson is one of the most "natural" distributions: number of typos, number of rain drops in a square meter of roof, etc..

The Poisson Distribution: Basic Properties

Assume that $Y \in P(\lambda)$ approximates $X \in B(n,p)$, then as $\mathbf{E}[X] = np$ seems natural that $\mathbf{E}[Y] = np = \lambda$ and as $\mathbf{Var}[X] = np(1-p) = \lambda(1-p)$ and as p is small $\mathbf{Var}[X] \sim \lambda$ and $\mathbf{Var}[Y] = \lambda$. Formally, If $Y \in P(\lambda)$:

- $\mathbf{E}[Y] = \lambda$.

$$
\begin{aligned}
\mathbf{E}[Y] &= \sum_{k=0}^{\infty} k \frac{e^{-\lambda}\lambda^k}{k!} = e^{-\lambda}(\lambda + \frac{2\lambda^2}{2!} + \frac{3\lambda^2}{3!} \cdots) \\
&= e^{-\lambda}\lambda(1 + \lambda + \frac{2\lambda^2}{2!} + \frac{3\lambda^2}{3!} \cdots) = e^{-\lambda}\lambda e^{\lambda}
\end{aligned}
$$

Variance of Poisson r.v.

- $\mathbf{Var}[Y] = \lambda$.

To prove it, instead of computing $\mathbf{E}\left[X^2\right]$ we compute $\mathbf{E}[X(X-1)]$.
Notice $\mathbf{Var}[X] = \mathbf{E}\left[X^2\right] - \mathbf{E}[X]^2 = \mathbf{E}[X(X-1)] + \mathbf{E}[X] - \mathbf{E}[X]^2$.

$$\begin{aligned}
\mathbf{E}[X(X-1)] &= \sum_{x=0}^{\infty} x(x-1)\frac{\lambda^x e^{-\lambda}}{x!} = \sum_{x=2}^{\infty} \frac{\lambda^2 \lambda^{x-2} e^{-\lambda}}{(x-2)!} \\
&= e^{-\lambda}\lambda^2 \sum_{x=2}^{\infty} \frac{\lambda^{x-2}}{(x-2)!} \underbrace{=}_{y=x-2} e^{-\lambda}\lambda^2 \sum_{y=0}^{\infty} \frac{\lambda^y}{(y)!} \\
&= e^{-\lambda}\lambda^2 e^{\lambda}
\end{aligned}$$

So, $\mathbf{Var}[X] = \lambda^2 + \lambda - \lambda^2$

Sum of Poisson r. v.

Lemma If $Y \in P(\lambda)$ and $Z \in P(\lambda')$ are independent, then $Y + Z \in P(\lambda + \lambda')$.

Proof

$$\begin{aligned}
\mathbf{Pr}\left[Y+Z=j\right] &= \sum_{k=0}^{j} \mathbf{Pr}\left[(Y=k)\cap(Z=j-k)\right] = \sum_{k=0}^{j} \frac{e^{-\lambda}e^{-\lambda'}\lambda^k \lambda'^{j-k}}{k!(j-k)!} \\
&= \frac{e^{-(\lambda+\lambda')}}{j!} \sum_{k=0}^{j} \frac{j!}{k!(j-k)!}\lambda^k \lambda'^{j-k} = \frac{e^{-(\lambda+\lambda')}}{j!} \sum_{k=0}^{j} \binom{j}{k} \lambda^k (\lambda')^{j-k} \\
&= \frac{e^{-(\lambda+\lambda')} \times (\lambda+\lambda')^j}{j!} \Rightarrow (Y+Z) \in P(\lambda+\lambda') \qquad \square
\end{aligned}$$

Basic facts

Recall X_i counts the number of balls in i-th bin.

- Probability all n balls fell in the same bin: $(\frac{1}{m})^n$.
- Probability that bin i is empty:
  $\mathbf{Pr}[X_i = 0] = (1 - \frac{1}{m})^n \sim e^{-\frac{n}{m}} = e^{-\lambda}$.
- Let Y be number of empty bins, compute $\mathbf{E}[Y]$?.
  For $1 \leq i \leq m$, let Y_i be and i.r.v. such that $Y_i = 1$ iff bin $i = \emptyset$ Then,
  $\mathbf{E}[Y] = \sum_{i=1}^{m} \mathbf{E}[Y_i] = \sum_{i=1}^{m} \mathbf{Pr}[X_i = 0] = m(1 - 1/m)^n$. So

$$\mathbf{E}[Y] \sim me^{-\lambda}.$$

Probability the i-th. bin contains 1 ball

We can assume that m and n are large, (so $p = 1/m$ is small), $\lambda = n/m = \Theta(1)$

Exact computation: $\mathbf{Pr}[X_i = 1] = \binom{n}{1}(1/m)^1(1-1/m)^{n-1}$, where $\binom{n}{1}$ number choices exactly 1 ball goes into bin i,

$(1-1/m)^{n-1}$: remaining balls do not go to bin i.

$\mathbf{Pr}[X_i = 1] = \frac{n}{m}(1-1/m)^n(1-1/m)^{-1}$

Poisson approximation: Taking $\lambda = \frac{n}{m}$ and $(1-1/m)^n \sim e^{-\lambda}$ and noticing $(1-1/m) \to 1$:

$$\mathbf{Pr}[X_i = 1] = \lambda e^{-\lambda}.$$

For $n = 3000$ and $m = 1000$, $\lambda = 3$, the exact value of $\mathbf{Pr}[X_i = 1] = 0.149286$ and the Poisson approximation is 0.149361.

Probability the i-th. bin contains exactly r balls

We can assume that m and n are large, $n, m > r$,
Exact computation: $\mathbf{Pr}[X_i = r] = \binom{n}{r}(1/m)^r(1-1/m)^{n-r}$.

Poisson approximation:
$(1-1/m)^{n-r} = (1-1/m)^n(1-1/m)^{-r} = e^{-\lambda} \cdot 1^{-r}$

$$\binom{n}{r}(1/m)^r = \frac{1}{r!}\left(\frac{n}{m}\frac{n-1}{m}\cdots\frac{n-r+1}{m}\right)$$
$$= \frac{1}{r!}\lambda(1-\frac{1}{n})\cdots\lambda(1-\frac{r+1}{n}) = \lambda^r$$

$\therefore \mathbf{Pr}[X_i = r] \sim \frac{\lambda^r e^{-\lambda}}{r!}$

For $n = 4000$ and $m = 2000$, $\lambda = 2$, and $r = 100$, the exact value of $\mathbf{Pr}[X_i = r] = 5.54572 \times 10^{-130}$ and the approximation is 1.83826×10^{-130}

Probability at least one bin has a collision

$\mathbf{Pr}\left[\text{at least 1 bin } i \text{ has } X_i > 1\right] = 1 - \mathbf{Pr}\left[\text{every bin } i \text{ has } X_i \leq 1\right].$
If $k-1$ balls went to $k-1$ different bins. Then,
$\mathbf{Pr}\left[\text{The } k\text{th. ball goes into a non-empty bin}\right] = \frac{k-1}{m}$
$\mathbf{Pr}\left[\text{The } k\text{th. ball goes into an empty bin}\right] = (1 - \frac{k-1}{m})$

$$\begin{aligned}
\mathbf{Pr}\left[\text{every bin } i \text{ has } X_i \leq 1\right] &= \prod_{i=1}^{n-1}(1 - \frac{i-1}{m}) \sim \prod_{i=1}^{n-1} e^{-i/m} \\
&= e^{-\sum_{i=1}^{n-1} i/m} = e^{-\frac{1}{m}\sum_{i=1}^{n-1} i} = e^{-\frac{n(n-1)}{2m}} \\
&\sim e^{-\frac{n^2}{2m}}
\end{aligned}$$

Therefore, $\mathbf{Pr}\left[\text{at least 1 bin } i \text{ has } X_i > 1\right] \sim 1 - e^{-\frac{n^2}{2m}}.$

Birthday problem

How many students in a class, to have that with probability $> 1/2$ at least 2 have the same birthday

This is the same problem as above, with $m = 365$:

We need $e^{-\frac{n^2}{2m}} \leq \frac{1}{2} \Rightarrow \frac{n^2}{2m} \leq \ln 2$
$\Rightarrow n = \sqrt{2m \ln 2}$. If $m = 365$ then $n = 22.49$.

Therefore, If there are more than 23 students in a class, with probability greater than $1/2$, more than 2 students will have the same birthday

Coupon Collector's problem

A. de Moivre (VIIc.)

How many balls do we need to throw to assure that w.h.p. every bin contains ≥ 1 balls

- Let Y a r.v. counting the number of balls we have to throw until having no empty bins
- For $i \in [m]$, let $Y_i = \#$ balls between between $i-1$ bins are not empty and the bin i gets a ball.
- So $Y = \sum_{i=1}^{m} Y_i$
- For $i \in [m]$, define Z_i a r.v. $\#$ balls until first ball goes into $\to$ i-bin.
- Then $Y = Z_m$, and for $i \in [m]$, let $Y_1 = Z_1$, for $i \geq 2$, $Y_i = Z_i - Z_{i-1}$
- First we want $\mathbf{E}[Y] = \sum_{i=1}^{n} \mathbf{E}[Y_i]$. After we prove concentration

Coupon Collector's problem

$Y_i = \#$ of balls we have to throw to get a new non-empty bin (it will be the i-th. non-empty bin)

$\mathbf{Pr}\left[\text{a new ball going into non-empty bin}\right] = 1 - \frac{1}{m}$.
If $k = \#$ balls between $(i-1)$ and i:

$$\mathbf{Pr}\left[Y_i = k\right] = \left(\frac{i-1}{m}\right)^{k-1} \left(\underbrace{1 - \frac{i-1}{m}}_{p_i}\right).$$

Therefore $Y_i \in G(p_i)$ and $\mathbf{E}\left[Y_i\right] = \frac{m}{m+i+1}$.

$$\mathbf{E}\left[Y\right] = \sum_{i=1}^{m} \mathbf{E}\left[Y_i\right] = \sum_{i=1}^{n} \frac{m}{m-i+1} = n \sum_{j=1}^{n} \frac{1}{j} = n(\ln n + o(1)).$$

Coupon Collector's problem: Concentration

Let $\mathbf{E}[Y] = O(\ln m) \sim cm \ln m$ for constant $c > 1$

- For any ball i, define the event A_j^r: bin $j = \emptyset$ after the first r throws.
- Notice events $A_1^r, A_2^r, \ldots A_m^r$ are not independent.
- $\mathbf{Pr}\left[A_j^r\right] = (1 - \frac{1}{m})^r \sim e^{-r/n} \leq e^{-cm\ln m/m} = n^{-c}$.
- Let W be a r.v. counting the number of balls needed so every bin has load ≥ 1.

$$\mathbf{Pr}[W > cm \lg m] = \mathbf{Pr}\left[\cup_{i=1}^{n} A_j^{cm \ln m}\right] \underbrace{\leq}_{UB} \sum_{j=1}^{m} \mathbf{Pr}\left[A_j^{cm \ln m}\right]$$

$$\leq \sum_{j=1}^{m} n^{-c} = m^{1-c}.$$

$$\mathbf{Pr}[W > cm \lg m] \leq n^{1-c}.$$

Coupon Collector's problem: Concentration Bounds

- The previous bound using UB is more tight than the one using Chebyshev or Chernoff on random variable Y.
  (See homework)
- In Section 5.4.1 of MU book, there is a sharper bound for the Coupon collector's, using the Poisson approximation.

Maximum Load

This is a very similar problem to the job and servers, but with sharper bounds

Theorem If we throw n balls independently and uniformly into $m = n$ bins, then the maximum loaded of a bin as at most $\left(\frac{4 \lg n}{\lg \lg n}\right)$ balls, with probability $\leq 1 - \frac{1}{n}$, i.e. w.h.p.

Recall that if for any bin $1 \leq j \leq n$, $X_j =$ is a r.v. with its load.

We know $\{X_j\}$ are not independent and $\mathbf{E}[X_j] = n/n = 1$.

To show the above bound we use the following two inequalities:

$$(\frac{N}{K})^K \leq \binom{N}{K} \leq (\frac{Ne}{K})^K. \tag{1}$$

$$\text{Let } N > e. \text{ If } K \geq \frac{2 \ln N}{\ln \ln N} \text{ then } K^K \geq N. \tag{2}$$

Max-load: Proof Upper Bound

For $1 \leq k \leq n$, $\mathbf{Pr}\left[X_i \geq k\right] \leq \binom{n}{k}\frac{1}{n^k} \leq \left(\frac{ne}{k}\right)^k \frac{1}{n^k} \leq \left(\frac{e}{k}\right)^k$.

We want to prove that for $k \geq \frac{2 \ln n}{\ln \ln n} \Rightarrow \mathbf{Pr}\left[X_i \geq \frac{2 \ln n}{\ln \ln n}\right] \leq \frac{1}{n^2}$.

i.e. $\mathbf{Pr}\left[X_i \geq k\right] \leq \left(\frac{e}{k}\right)^k \leq \frac{1}{n^2} \Rightarrow \left(\frac{e}{k}\right)^{\frac{k}{e}} \geq n^{\frac{2}{e}}$

Taking ln: $\frac{k}{e} \geq \frac{2 \ln(n^{2/e})}{\ln \ln(n^{\frac{2}{e}})} = \frac{4 \ln n}{e \ln(\frac{2}{e} \ln n)} \Rightarrow k \geq \frac{4 \ln n}{\ln(\frac{2}{e} \ln n)}$

We proved that if $k \geq \frac{4 \ln(n)}{\ln(2/e) \ln \ln(n)}$ then $\mathbf{Pr}\left[X_i \geq k\right] \leq \frac{1}{n^2}$.

Then, using U-B
$\mathbf{Pr}\left[\exists i \in [n] \,|\, X_i \geq k\right] \leq \sum_{i=1}^{n} \mathbf{Pr}\left[X_i \geq k\right] \leq \frac{n}{n^2} = \frac{1}{n}$.

Further considerations on Max-load

1. The same proof could be extended to the case of n balls and m bins, with the constrain $n < m \ln m$.
2. We can obtain the same result by using Chernoff's bounds. (Nice exercise!)
3. In fact, the result could be extended to prove the Lower Bound: that w.h.p. the max-load is $\Omega(\frac{\ln n}{\ln \ln(n)})$ balls. One easy way to prove the lower bound is using Chebyshev's bound.
4. That result yields: Throwing n balls to n bins, w.h.p. we have a max-load of $\Theta(\frac{\ln n}{\ln \ln(n)})$.
5. We can obtain sharper bounds for max-load, using strong inequalities (Azuma-Hoeffding) or the Poisson approximation.

Poisson approximation

1. A difficulty with the exact (binomial) B & B model is that random variables could be dependent (for ex. bin's load).
2. We have seeing how to approximate the expressions arising from the exact computations by a Poisson, if p is small and n is large.
3. However, under the right conditions, we can approach the whole solution to the problem by using Poisson r.v. instead of Binomial. In the binomial case we have exactly n balls with probability $p = 1/m$, in the Poisson case we have an intensity $\lambda = n/m$, where n is the expected number of balls being used.
4. The Poisson case is to use directly independent Poisson random variables and it can be shown, under certain conditions give a good approximation to the solution. See for ex. section 5.4 in MU.

Dynamic Sets.

Given a universe $\mathcal{U}$ and a set of keys $\mathcal{S} \subset \mathcal{U}$, for any $k \in \mathcal{S}$ we can consider the following operations

- Search $(\mathcal{S}, k)$: decide if $k \in \mathcal{S}$
- Insert $(\mathcal{S}, k)$: $\mathcal{S} := \mathcal{S} \cup \{k\}$
- Delete $(\mathcal{S}, k)$: $\mathcal{S} := \mathcal{S} \backslash \{k\}$
- Minimum $(\mathcal{S})$: Returns element of $\mathcal{S}$ with smallest k
- Maximum $(\mathcal{S})$: Returns element of $\mathcal{S}$ with largest k
- Successor $(\mathcal{S}, k)$: Returns element of $\mathcal{S}$ with next larger key to k
- Predecessor $(\mathcal{S}, k)$: Returns element of $\mathcal{S}$ with next smaller key to k.

Recall Dynamic Data Structures

DICTIONARY

Data structure for maintaining $\mathcal{S} \subset \mathcal{U}$ together with operations:

- Search $(\mathcal{S}, k)$: decide if $k \in \mathcal{S}$
- Insert $(\mathcal{S}, k)$: $\mathcal{S} := \mathcal{S} \cup \{k\}$
- Delete $(\mathcal{S}, k)$: $\mathcal{S} := \mathcal{S} \backslash \{k\}$

PRIORITY QUEUE

Data structure for maintaining $\mathcal{S} \subset \mathcal{U}$ together with operations:

- Insert $(\mathcal{S}, k)$: $\mathcal{S} := \mathcal{S} \cup \{k\}$
- Maximum $(\mathcal{S})$: Returns element of $\mathcal{S}$ with largest k
- Extract-Maximum $(\mathcal{S})$: Returns and erase from $\mathcal{S}$ the element of $\mathcal{S}$ with largest k

Hashing functions

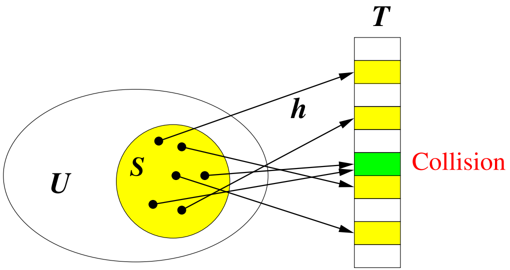
Data Structure that supports dictionary operations on an universe of numerical keys.

Notice the number of possible keys represented as 64-bit integers is $2^{64} = 18446744073709551616$.

Tradeoff time/space

Define a hashing table $T[0, \ldots, m-1]$

a hashing function $h : \mathcal{U} \to T[0, \ldots, m-1]$



Simple uniform hashing function.

A good hashing function must have the property that $\forall k \in \mathcal{U}$, $h(k)$ must have the same probability of ending in any $T[i]$.

Given a hashing table T with m slots, we want to stores $n = |\mathcal{S}|$ keys, as maximum.

Important measure: load factor $\alpha = n/m$, the average number of keys per slot.

The performance of hashing depends on how well h distributes the keys on the m slots: h is simple uniform if it hash any key *with equal probability* into any slot, independently of where other keys go.

How to choose h: The division method

Choose m prime and as far as possible from a power,

$$\boxed{h(k) = k \mod m}.$$

Fast ($\Theta(1)$) to compute in most languages ($k\%m$)!

Be aware: if $m = 2^r$ the hash does not depend on all the bits of K

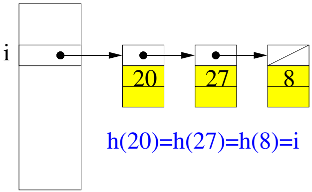
If $r = 6$ with $k = 1011000111\underbrace{011010}_{=h(k)}$

$(45530 \mod 64 = 858 \mod 64)$

Collision resolution: Separate chaining

For each table address, construct a linked list of the items whose keys hash to that address.

- Every key goes to the same slot
- Time to explore the list = length of the list



Cost of exploring the list

The cost of the dictionary operations:

- Insertion of a new key: $\Theta(1)$.
- Search of a key: $O($ length of the list$)$
- Deletion of a key: $O($ length of the list$)$.

Under the hypothesis that h is *simply uniform*, the expected number of keys falling into $T[i]$ is $\alpha = n/m$.

Therefore, the expected time to search the list at $T[i]$ is $O(1 + \alpha)$.

Theorem

Under the assumption of simple uniform hashing, in a hash table with chaining, an unsuccessful and successful search takes time $\Theta(1 + \frac{n}{m})$ *on the average.*

Bloom filter

Given a set of elements S, we want a Data structure for supporting insertions and querying about membership in S.

In particular we wish a DS s.t.

- *minimizes* the use of memory,
- *can check membership* as fast as possible.

Burton Bloom: The Bloom filter data structure. Comm. ACM, July 1970.

A hash data structure where each register in the table is one bit

Definition Bloom filter

Create a one bit hash table $T[0, \ldots, m-1]$, and a hash function h. Initially all m bits are set to 0.

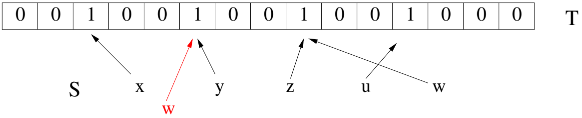
Giving a set $S = \{x_1, \ldots, x_n\}$ define a hashing function $h : S \to T$. For every $x_i \in S$, $h(x_i) \to T[j]$ and $T[j] := 1$. Given a set S a function $h()$ and a table $T[m]$:

Insert (x)
$h(x) \to i$
if $T[i] == 0$ **then**
  $T[i] = 1$
end if

inS(y)
$h(x) \to i$
if $T[i] == 1$ **then**
  return Yes
else
  return No
end if

Notice: once we have hashed S into T we can erase S.

False positives



Bloom filter needs $O(m)$ space and answers membership queries in $\Theta(1)$.

Inconvenience: Do not support removal and may have false positive.

In a query $y \in S$?, a Bloom filter always will report correctly if indeed $y \in S$ $(h(y) \to T[i]$ with $T[i] = 1)$,
but if $y \notin S$ it may be the case that $h(y) \to T[i]$ with $T[i] = 1$, which is called a False positive.

How large is the error of having a false positive?

Probability of having a false positives

Let $|S| = n$, we constructed a BF $(h, T[m])$ with all elements in S. If we query about $y \in S$?, with $y \notin S$, and $h(y) \to T[i]$, what is the probability that $T[i] = 1$?

After all the elements of S are hashed into the Bloom filter, the probability that a specific $T[i] = 0$ after hashing n elements is $(1 - \frac{1}{m})^n = e^{-n/m}$

Therefore, for a $y \notin S$, the probability of false positive π:

$$\pi = \mathbf{Pr}\left[h(y) \to T[i] \,|\, T[i] = 1\right] = 1 - (1 - \frac{1}{m})^n \sim 1 - e^{-n/m}.$$

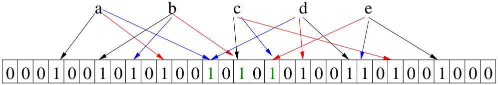
To minimize π, $1 - e^{-n/m}$ has to be small, $\Rightarrow 1/e^{n/m}$ small, i.e, $m >> n$.

For ex.: if $m = 100n, \pi = 0.0095$; If $m = n, \pi = 0.632$ and if $m = n/10, \pi = 0.9999$

Alternative: Amplify

Take k different functions $\{h_1, h_2, \ldots, h_k\}$ in the same 2-universal set of functions.

Ex. Bloom filter with 3 hash functions: h_1, h_2, h_3.



When making a query about if $y \in S$, compute $h_1(y), \ldots h_t(y)$, if one of them is 0 we certainty $y \notin S$, else (if all the k hashing go to bits with value 1) $y \in S$ with some probability.
After hashing the n elements k times to T, for an specific $T[i]$:

$$p = \mathbf{Pr}\left[T[i] = 0\right] = (1 - \frac{1}{m})^{kn} = e^{-kn/m}.$$

The probability f of a false positive:

$$f = \left(1 - e^{-kn/m}\right)^k = (1 - p)^k$$

Optimizing k

Given n and m we want to find the optimal value of k to minimize the probability of a false positive $f(k) = (1 - e^{-kn/m})^k$

Define $g(k) = \ln f(k) = k \ln(1 - e^{-kn/m})$. Minimizing f is equivalent to minimizing g.

To minimize the probability of having a false positive: $\frac{\mathrm{d}g(k)}{\mathrm{d}k} = 0$

$\Rightarrow \frac{\mathrm{d}g(k)}{\mathrm{d}k} = \ln(1 - e^{-kn/m}) + \frac{kne^{-kn/m}}{m(1-e^{-kn/m})} = 0,$
$\Rightarrow$ when n, m are given, to minimize f is $k_o = (\ln 2)\frac{m}{n}$.

In this case the false positive probability $f_o = 0.6185^{m/n}$.

Bloom filters allow a constant probability of false positive, $m = cn$ for small constant c, i.e. m grows linear wrt n.

For ex.: if $c = 2$ and $k = 6$ the false positive probability is around 2%.

Practical issues

For password checking:

If D has 100000 common words, each of 7 characters $\Rightarrow$ we need 700000 bytes

Use 5 tables of 160000 bits each$\Rightarrow$ need a total of 800000 bits $=$ 100000 bytes.

The probability of error is 0.02

On the other hand although the results shown before are asymptotic, there also work for practical values of n. Figure in the side table give the probability of false positive wrt to n