Lecture 38: hashing

- Hashing is a tool to turn inputs into "nice", "spread out" random variables
- We'll set up model for hashing

Applications

- Hash tables are super-useful programming tools
- Load balancing in distributed systems
- Efficient approximation algorithms
Algorithm (e.g., bucket) takes \( t \) steps for each person in the same bucket as \( x \), for each \( k \) in \( 1, 2, \ldots \) steps.

\[
\begin{align*}
T &= 1 + 1 + 1 + \ldots + 1 = 13 \\
\sum_{k=1}^{3} t_k &= 1 + 2 + 2 + 1 + 3 + 3 + 3 = 17
\end{align*}
\]

\( T \) is an RV with \( T(s) = 10 \) and \( T(s_3) = 6 \)...

**Sample space** \( S \):

\[ S = \{ \text{Alice, Bob, } \ldots \} \]

**Random variable** \( X \):

\[ X = \{ \text{Alice, Bob, } \ldots \} \]

**Y** = buckets \( E_1, \ldots, E_m \)

**Function valued RV** \( H \):

\[ H : S \rightarrow [X \rightarrow Y] \]

\[ H(s_i) = \hat{s}_i \]

\[ H(x) = (H(s))(x) \]

\( H_x \) is a \( Y \)-valued RV.
what is expected # of values that hash to bucket 1?

8 names \( x \) \( |x| = 8 \)
26 buckets \( y \) \( |y| = 26 \)
expected # names in bucket \( 9/26 \)

let \( N_x = \# \) of \( x \)'s with \( H_x = y \)

let \( I_x = \) indicator variable for \( H_x = y \)
i.e., \( I_x(s) = \begin{cases} 1 & \text{if } H_x(s) = y \\ 0 & \text{otherwise} \end{cases} \)

\( N_x = I_{x_1} + I_{x_2} + I_{x_3} + \ldots + I_{x_8} \)

\( E(N_x) = \sum_{x} E(I_x) = \sum_{x} \frac{1}{26} = \frac{9}{26} \)

\[ 1 \cdot \Pr(H_x = 2) + 0 \cdot \Pr(H_x = 2) \]
\[ = \frac{9}{26} = \frac{1}{m} \]

H_{x_1} and H_{x_2} are indep.

Pr(H_{Alice} = 1 \mid H_{Andr} = 1) = \frac{1}{m}

Let \( S \) be a set of names.

Bad: \( H_{Mike} \) is always 1.

\[ \forall x, y, \Pr(H_x = y) = \frac{1}{|y|} \]

Bad: everyone hashes to same bucket.

satisfies (1)

let: \( S \) \( |S| = m \)

\( H_x(i) = \{ s_i \} \)

\( \forall x \neq x_2 \)

Pr(H_{Alice} = 1 \mid H_{Andr} = 1) = \frac{1}{m}