Final Stretch
- list all topics from exams 1, 2
- list all topics not tested in exams 1, 2
- spend little time on most days to go over some topic.

Open Addressing.
- each element is stored in the table.
- n element
- m = |T|
- \( \alpha \) : load factor \( \leq 1 \).
  - keys \( \rightarrow \) slot \( m \) \( T \).
  - \( h : U \times \{0,1,...,m-1\} \rightarrow \{0,1,...,m-1\} \)
  - \( \langle h(k,0), h(k,1), ..., h(k,m-1) \rangle \) is a
permutation of \{0, 1, \ldots, m-1\}.

**Uniform Hashing assumption**: All m!
permutations of \{0, 1, \ldots, m-1\} are equally likely to be chosen as a
probe seq.

**Linear Probing**

**Insert (T, k)**

if T is full: err

probe ← h(k), offset ← 1

while T[probe] is occupied do

probe ← (probe + offset) mod m
\[ T[\text{probe}] \leftarrow k \]

generate m probe sequences: (\[
\begin{array}{c}
8 \\
1 \\
2 \\
3 \\
\vdots \\
4 \\
5 \\
6 \\
\vdots
\end{array}
\])

Double Hashing:

\[ h(k, i) = (h_1(k) + i h_2(k)) \mod m \]

\[ \rightarrow \text{Insert} \ (T, k) \]

if T is full: error

probe \leftarrow h(k), \ offset \leftarrow h_2(k)

while T[probe] is occupied do

probe \leftarrow (probe + offset) \mod m

T[probe] \leftarrow k

generate m^2 probe sequences: \[ h_1(k), \ h_2(k) \mod m \]
\[ h_2(k) = 2 + (k \mod 7) \]

\[ h_3(k) = k \mod 13 \]

86 51 45 96 75 81 23 \[ \boxed{19} \] 36 48

0 1 2 3 4 5 6 7 8 9 10 11 12

| 19 | 23 | 81 | 96 | 45 | 86 | 36 | 75 | 57 |

---

**Analysis**

Thus: In an open address hashtable containing \( m \) elements and load factor \( n/m \), the
Supposed no. of probes in an unsuccessful search is \( \leq \frac{1}{1-x} \), assuming uniform hashing assumption.

**Proof:** Let \( X \) be the r.v. denoting the # probes in an unsuccessful search. We want to find \( E(X) \).

\[
E[X] = \sum_{i=1}^{\infty} P_r[X \geq i]
\]

\( A_i \) : event that the \( i \)th probe is unsuccessful.

\[
P_r[X \geq i] = P_r[A_1 \cap A_2 \cap A_3 \cap \ldots \cap A_{i-1}]
\]

\[
= P_r[A_1] \cdot P_r[A_2 | A_1] \cdot P_r[A_3 | A_1 \cap A_2] \cdot \ldots
\]

\[
= \frac{n}{m} \cdot \frac{n-1}{m-1} \cdot \frac{n-2}{m-2} \cdot \ldots \cdot \frac{n-i+2}{m-i+2}
\]
\[ \frac{n}{m} \cdot \frac{n}{m} \cdot \frac{n}{m} \cdots \left( \frac{n-k}{m-k} \leq \frac{n}{m} \right) \]

\[ \left( \frac{n}{m} \right)^{i-1} = \alpha^{i-1} \]

Eqn 0 known

\[ E[X] = \sum_{i=1}^{\infty} \alpha^{i-1} = \alpha^0 + \alpha^1 + \alpha^2 + \cdots \]

\[ = \frac{1}{1-\alpha} \]

Alt: X: geometric r.v. with success prob \( 1-\alpha \cdot E[X] \leq \frac{1}{1-\alpha} \leftarrow \text{uniform hashy assumption??} \)

Thus: In open addressing, the expected
# probes in a successful search is
given by \( \frac{1}{\log(\frac{1}{x})} \); assume SUHA.

**Proof:** let \( x_1, x_2, \ldots, x_n \) be the order
in which elements are inserted into \( T \).

Search time = Insert time.

Insert first \( \frac{n}{2} \) elements \((x_1, x_2, \ldots, x_{n/2})\)

- fraction of empty slots during each
of the insertions \( \geq \frac{1}{2} \).

- exp \( \# \) probes for each insert \( \leq 2 \).

- total \( \# \) probes to insert all
  elements \( \leq \frac{M}{2} \cdot 2 = \lceil \frac{M}{2} \rceil \).

Next \( \frac{M}{11} \)

Insert first \( \frac{n}{2} \) elements \((x_{n/2+1}, x_{n/2+2}, \ldots, x_{n/2+\frac{n}{2}})\)

- fraction of empty slots during each
If the inputs $\geq \frac{1}{2}$: \[ \frac{1}{14} \]
- exp # probes for each input $\leq 2$.
- total # probes to insert all elements $\leq \frac{M}{2} \cdot 2 = \lfloor M \rfloor$.

Insert $\left\lceil \frac{M}{8} \right\rceil$ elements.
- fraction of empty slots among each of the inputs $\geq \frac{1}{2}$: \[ \frac{1}{8} \]
- exp # probes for each input $\leq 2$.
- total # probes to insert all elements $\leq \frac{M}{2} \cdot 2 = \lfloor M \rfloor$.

The total no of probes needed is
Insert first \( \frac{m}{2} \), then the next \( \frac{m}{4} \), \( \frac{m}{8} \), etc.

is given by

\[
\underbrace{m + m + \cdots + m}_i = \left\lfloor \frac{m}{i} \right\rfloor.
\]

After inserting the first \( \frac{m}{2} \) elements,

\[
\begin{align*}
\text{next } & \frac{m}{4} \\
\vdots \\
\text{and } & \frac{m}{2^i} 
\end{align*}
\]

the fraction of empty slots = \( \frac{1}{2^i} \).

Summary: To obtain a table in which \( \frac{1}{2^i} \) fraction of slots are empty, the expected number of probes \( \leq m_i \).
We have inserted $x_1, x_2, \ldots, x_n$. The fraction of empty slots is $1 - \alpha$.

Expected # probes needed?

\[
\frac{1}{\frac{i}{2}} = 1 - \alpha
\]

\[
\therefore 2^i = \frac{1}{1 - \alpha}
\]

\[
i = \log \left( \frac{1}{1 - \alpha} \right)
\]

Thus, to get $1 - \alpha$ fraction of table empty, the exp. # probes

\[
\leq m \cdot \log \left( \frac{1}{1 - \alpha} \right)
\]

# probes per insert (search)
\[
\frac{m}{n} \cdot \log \left( \frac{1}{1-x} \right)
\]
\[
\frac{1}{\eta/m} \cdot \log \left( \frac{1}{1-x} \right)
\]
\[
\frac{1}{\lambda} \cdot \log \left( \frac{1}{1-x} \right)
\]

**Tries:** "Retrieval".

A data structure that is used to preprocess a text to support queries efficiently.

**Scenario:**

\( S \): set of strings over alphabet \( \Sigma \).
$\mathcal{S} : |S|$

$$n = \sum_{\mathcal{S}} |S|$$

**Definition:** A standard trie is any full prefix-free tree with the following properties:

- The tree has $n$ leaves, each leaf corresponding to a string in $S$.
- Each internal node has at most $|\Sigma|$ children.
- Each edge/node is labeled with a character from $\Sigma$. The ordered list of edges gives the canonical ordering of the
labels.

\[ S = \{ \text{bell, boy, boat, been, call, cat, copy} \} \]

No string can be a prefix of another string.

Always fix this by adding a $\$$ at the end of each string.

Properly...
hit of the trie: length of the longest
    stay in S.

# nodes in the trie: \( O(n) \).

\[ \leq n + 1 \]

Searching for a stay of length \( m \).

\( O(m) \)  \( \sigma \)

\( O(m | \Sigma |) \)  \( \approx \) const, in practice

PATRICIA Trie (Compressed Trie)
Compressed trie is "better than" full binary tree. In a full binary tree
# leaves ≈ # int nodes.

\[ O(s) \]
$T: \text{ ababaabaab }$

Suffix tree:

```
$  \checkmark$
$b  \checkmark$
$ab  \checkmark$
$aab  \checkmark$
$aaab  \checkmark$
$baaab$. \checkmark$
```