Today: 10 pm - 11 pm.

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:
- every element is stored in the table.

\[
| n \leq m \Rightarrow \alpha = \frac{n}{m} \leq 1.
\]

\[
h : U \times \{0, 1, \ldots, m-1\} \rightarrow \{0, 1, 2, \ldots, m-1\}
\]

\[
\langle h(k,0), h(k,1), h(k,2), \ldots, h(k,m-1) \rangle
\]

is a permutation of \(\{0, 1, \ldots, m-1\}\).

Uniform hashing assumption: a probe seq...
is equally likely to be any of the \( m! \) permutations of \( \{0, 1, \ldots, m-1\} \).

**Linear Probing:**

**Insut** \( (T, k) \)

\[
\text{if } T \text{ is full then} \quad \text{o/p error}
\]

\[
\text{probe } \leftarrow h(k), \quad \text{offset } \leftarrow 1
\]

\[
\text{while } T[\text{probe}] \text{ is occupied do}
\]

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

\[
T[\text{probe}] \leftarrow k
\]

# probe say : m.

**Double Hashing**

\[
h(k, i) = (h_1(k) + i \cdot h_2(k)) \mod m,
\]
\textbf{Insert} \((T, k)\)

\textbf{if} \(T\) is full \textbf{then}

\textbf{o/p} error

\[ h_1(k) \]

\textbf{probe} \leftarrow h(k), \; \text{offset} \leftarrow h_2(k) \]

\textbf{while} \(T[\text{probe}]\) is occupied \textbf{do}

\textbf{probe} \leftarrow (\text{probe} + \text{offset}) \mod m

\[
T[\text{probe}] \leftarrow k
\]

\[ h_1(k) \mod m \quad \# \text{probe set size } m^2. \]

\[ h_2(k) \mod m. \]

\[ h_1(k) = k \mod 13 \]

\[ h_2(k) = 2 + (k \mod 7) \]
18 25 36 41 54 6 81 96 75

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>96</td>
<td>41</td>
<td>81</td>
<td>75</td>
<td>18</td>
<td>6</td>
<td>54</td>
<td>36</td>
<td>125</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Analysis.**

**Unsuccessful Search.**

Thus: Given an open addressing hash table with load factor $\alpha = \frac{n}{m}$, the expected no. of probes is $\frac{1}{1-\alpha}$, under the uniform hashing assumption.
Proof: Let $X$ be the r.v. denoting the number of probes needed in an unsuccessful search.

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

$A_i$: event that $i$th probe was unsuccessful.

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

$$= \Pr[A_1] \cdot \Pr[A_2 | A_1] \cdot \Pr[A_3 | A_1 \cap A_2] \ldots$$

$$\leq \frac{n}{m} \cdot \frac{n}{m} \cdot \frac{n}{m} \cdot \ldots \cdot \frac{n - (i-2)}{m - (i-2)}$$

$$\leq \left( \frac{n}{m} \right)^{i-1} \lambda^{i-1}$$
Eqn (1) becomes

\[ E(X) = \sum_{i=0}^{\infty} x^i p^i = 1 + x + x^2 + \cdots = \frac{1}{1 - x} \]

**Alt:** \( X \): geometric r.v. with 

\[ \begin{align*}
\text{success prob} & \quad (1 - \alpha) \\
\text{empty slot} & \quad \frac{1}{1 - \alpha}.
\end{align*} \]

\[ \rightarrow \text{Simple Uniform Hashing Assumption.} \]

**Successful Search.**

Simple Uniform Hashing Assumption.

Thus: In open addressing in which 

\[ \alpha = \frac{n}{m}, \] under the simple uniform
Hashing assumption, the exp. # probes in a successful search \( \leq \frac{1}{a} \log \left( \frac{1}{1-a} \right) \).

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

Insert first \( m/2 \) elements.

- at least \( 1/2 \) the table is empty during each of the insertions.
- Exp. # probes to find an empty slot during each insertion \( \leq 2 \).
- Total # probes \( \leq \frac{m}{2} - 2 = \lfloor m/2 \rfloor \).

Insert the next \( m/2 \) elements.
- at least $\frac{1}{2}$ the table is empty during each of the insertions.

- Exp. # probes to find an empty slot during each insertion $\leq 2$.

- Total # probes $\leq \frac{m}{2} \cdot 2 = \lceil \frac{m}{2} \rceil$.

Insert the next $\frac{m}{8}$ elements

- at least $\frac{1}{8}$ the table is empty during each of the insertions.

- Exp. # probes to find an empty slot during each insertion $\leq 8$.

- Total # probes $\leq \frac{m}{8} \cdot 8 = \lceil \frac{m}{8} \rceil$.
The total \# probes needed in expectation to insert the first \( \frac{m}{2} \) & then the next \( \frac{m}{4} \) & \( \ldots \) \& \( \frac{m}{2^i} \) is given by

\[
\leq m + m + \ldots + m = \sum_{i=0}^{\infty} \left\lfloor \frac{m}{2^i} \right\rfloor
\]

At the end of inserting \( \frac{m}{2} + \frac{m}{4} + \frac{m}{8} + \ldots + \frac{m}{2^i} \) elements, fraction of table that is empty \( = \frac{1}{2^i} \)

After inserting \( x_1, x_2, \ldots, x_n \), we know that the fraction of table that is empty \( = 1 - \alpha \).

We want to find the \# probes it
tools to get \((1-x)\) fraction of balls to be empty.

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

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

\[
i = \lg \left( \frac{1}{1-x} \right)
\]

\[
\therefore \text{Exp \# probes needed to insert (search)} \quad x_1, \ldots, x_m \leq m \cdot 1\lg \left( \frac{1}{1-x} \right)
\]

\[
\text{Exp \# probes needed to insert m elements} \geq \frac{m \cdot 1\lg \left( \frac{1}{1-x} \right)}{n}
\]

\[
= \frac{1}{1} \lg \left( \frac{1}{1-x} \right)
\]
Trie:

data structure that is used to preprocess text to answer queries efficiently.

**Def.** A standard trie is an ordered tree $T$ with the full property:

**Scenario.** $S$: set of steps

$\&: |S|$

$n = \sum_{\& \in S} |\&|$
\( \Sigma : \) alphabet

1. Each node (edge) except the root is labeled with a char in \( \Sigma \).
2. \( T \) has \( 8 \) leaves and each leaf corresponds to a string in \( S \).

\[ S = \{ \text{cat, cold, cot, cop, bell, boy, bery} \} \]

Assumption: no string is a prefix of another string.

- We can take care of this by
Putting a $ at the end of each string.

Properties:

Let $T$ be the tree; length of the longest string in $S$.

$\# \text{ nodes in the tree: } O(n)$

Search for a string of length $m$.

$O(m) \cdot \text{cobb}$

Precisely: $O(m \cdot |\Sigma|)$.

$log$ small.

Wasting of space: A lot of $d + 1$ nodes.

Coalesce all edges of $d + 1$ nodes.
m to 1 edge.

\[
\begin{array}{c}
\text{b} \\
\text{a} \\
\text{ball} \\
\text{e}
\end{array}
\]

\[
\begin{array}{c}
\text{e} \\
\text{e} \\
\text{e} \\
\end{array}
\]

\[
\begin{array}{c}
\text{b} \\
\text{c} \\
\text{a} \\
\text{t}
\end{array}
\]

\[
\begin{array}{c}
\text{t} \\
\text{p} \\
\text{p}
\end{array}
\]

\[
\begin{array}{c}
\text{Compressed trie.}
\end{array}
\]

\[
\begin{array}{c}
(3,2,2)
\end{array}
\]

\[
\begin{array}{c}
\text{(4,1,2)}
\end{array}
\]

\# nodes in the trie: \( O(8) \)
at least as good as a full binary tree.

Suffix tree:

\[ \Gamma: \text{abaabaaas} \]

\[ S: \{ \text{all suffixes of } \Gamma \} \]

\[ S = \{ a\$, aa\$, aaa\$, baaa\$, \ldots \} \]