CompSci 161
Winter 2023 Lecture 14:
Dynamic Programming V:
Optimal [Offline] Binary Search Trees
In ICS 46, you saw “online” search trees
  • Additions happened one at a time
  • Resolve addition before next request
  • Had to maintain “balance”
  • Did not know probability distribution of requests.

Today we will look at “offline” search trees
  • Know full set of keys at beginning
  • Know probability distribution of requests
  • Want to minimize expected lookup time
  • Even if that means bad lookup for some
Examples of Binary Search Trees

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>( i )</td>
<td>.13</td>
<td>.21</td>
<td>.11</td>
<td>.01</td>
<td>.22</td>
<td>.08</td>
<td>.24</td>
</tr>
</tbody>
</table>

E[lookup] = 2.69

E[lookup] = 2.12
Problem Statement

- Input: $n$ probabilities, $p_1 \ldots p_n$
- $p_i$ is probability of looking up $i$th key.
- Goal: build binary search tree.
  - Minimize expected lookup cost.

Check for understanding

- Suppose we have $d_i$ (depth of each node)
- Root has $d_i = 1$, its children have $d_i = 2$, etc.
- What is the expected lookup cost of this tree?

$E_{\sum\, p \cdot d_i}$
Creating the Dyn Prog Algorithm

Define Tree(i, j): cost of opt tree keys i through j

Base cases:

if i = j, return P_i
if i > j, return 0

Which key(s) can be the root of a binary search tree consisting of keys i through j?

Cost of BST, rooted at r, has keys i through j?

\[
\begin{align*}
\text{Tree}(i, r-1) + \sum_{k=i}^{r-1} P_k + P_r + \text{Tree}(r+1, j) + \sum_{k=r+1}^{j} P_k
\end{align*}
\]
First make recursive solution

Tree\((i,j)\) :

\[
\begin{align*}
\text{if } j &< i \text{ then} \\
\quad \text{return } 0 \\
\text{else if } j = i \text{ then} \\
\quad \text{return } p_i \\
\text{else} \\
\quad r = i \\
\quad \min = \text{Tree}(i, r-1) + \text{Tree}(r+1, j) + \sum p_k
\end{align*}
\]

\[
\text{for } r = i+1 \ldots j \\
\quad \text{Cost} = \text{Tree}(i, r-1) + \text{Tree}(r+1, j) + \sum
\]

Ea case \(O(n)\), 
\[
\exists O(n^2) \text{ cases, Total: } O(n^3)
\]
Iterative Version: Topological Order

- Caution: some recursive calls to higher values.
- We can’t iterate increasing $i$ and $j$ together.
- $\text{Tree}[i,j]$ will make calls to:
  - $\text{Tree}[i, r - 1]$ for $i \leq r \leq j$
  - $\text{Tree}[r + 1, j]$ for $i \leq r \leq j$

- For example, $\text{Tree}[2, 5]$ will call:
  - $\text{Tree}[2, 1]$ and $\text{Tree}[3, 5]$ ($r = 2$)
  - $\text{Tree}[2, 2]$ and $\text{Tree}[4, 5]$ ($r = 3$)
  - $\text{Tree}[2, 3]$ and $\text{Tree}[5, 5]$ ($r = 4$)
  - $\text{Tree}[2, 4]$ and $\text{Tree}[6, 5]$ ($r = 5$)
Table looks like

<table>
<thead>
<tr>
<th>$k_1$</th>
<th>$k_2$</th>
<th>$k_3$</th>
<th>$k_4$</th>
<th>$k_5$</th>
<th>$k_6$</th>
<th>$k_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_1$</td>
<td>.13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$k_2$</td>
<td></td>
<td>.21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$k_3$</td>
<td></td>
<td></td>
<td>.11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$k_4$</td>
<td></td>
<td></td>
<td></td>
<td>.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$k_5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.22</td>
<td></td>
</tr>
<tr>
<td>$k_6$</td>
<td>Fill in Tree$(i,j)$ where $j = i + 8$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$k_7$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.24</td>
</tr>
</tbody>
</table>
Iterative Version: Memoize the Data

\[ \text{for } i \leftarrow 1 \ldots n \text{ do} \]
\[ \text{Tree}[i, i - 1] \leftarrow 0 \]
\[ \text{Tree}[i, i] \leftarrow p_i \]

\[ \text{for } s = 1 \ldots n - 1 \]
\[ \text{for } i = 1 \ldots n - s \]
\[ j = i + s \]
\[ \text{// fill in Tree}(i, j) \]
How to get the tree itself?

<table>
<thead>
<tr>
<th></th>
<th>$k_1$</th>
<th>$k_2$</th>
<th>$k_3$</th>
<th>$k_4$</th>
<th>$k_5$</th>
<th>$k_6$</th>
<th>$k_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_1$</td>
<td>0.13</td>
<td>0.47</td>
<td>0.69</td>
<td>0.72</td>
<td>1.28</td>
<td>1.52</td>
<td>2.12</td>
</tr>
<tr>
<td>$k_2$</td>
<td></td>
<td>0.21</td>
<td>0.43</td>
<td>0.46</td>
<td>1</td>
<td>1.17</td>
<td>1.73</td>
</tr>
<tr>
<td>$k_3$</td>
<td></td>
<td></td>
<td>0.11</td>
<td>0.13</td>
<td>0.47</td>
<td>0.63</td>
<td>1.19</td>
</tr>
<tr>
<td>$k_4$</td>
<td></td>
<td></td>
<td></td>
<td>0.01</td>
<td>0.24</td>
<td>0.4</td>
<td>0.95</td>
</tr>
<tr>
<td>$k_5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.22</td>
<td>0.38</td>
<td>0.92</td>
</tr>
<tr>
<td>$k_6$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.08</td>
<td>0.4</td>
</tr>
<tr>
<td>$k_7$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.24</td>
</tr>
</tbody>
</table>

To be continued...