Chapter 5, Dynamic Programming:
- Introduction and sample problems (last week Friday)
- Principles of DP (Monday)
- DP algorithms, solutions to sample problems (Wednesday)
- Optimal BSTs (**Today**)  
- Finish up optimal BSTs, review for test 2 (next week Monday)
- **Test 2**, next week Wed Apr 6, *not* covering DP

Today (and Monday):
- Optimal BST definition
- The Optimal-BST-building problem
- The dynamic programming solution
Why this problem?

- It connects dynamic programming with the quest for a better map.
- Its hardness is in the right places (building the table—hard; reconstructing solution—trivial).
- It is a representative of a bigger concept: What if we had more information—how would that change the problem.

Game plan:

- Understand the problem itself
- Understand the recursive characterization
- Understand the table-building algorithm
The **optimal binary search tree** problem:

- Assume we know all the keys $k_0, k_1, \ldots k_{n-1}$ ahead of time.
- Assume further that we know the probabilities $p_0, p_1, \ldots p_{n-1}$ of each key’s lookup.
- Assume even further that we know the “miss probabilities” $q_0, q_1, \ldots q_n$ where $q_i$ is the probability that an *extraneous key* falling between $k_{i-1}$ and $k_i$ will be looked up.
- We want to build a tree to minimize the expected cost of a look up, which is the total weighted depth of the tree:

$$
\sum_{i=0}^{n-1} p_i \text{ depth}(k_i) + \sum_{i=0}^{n} q_i \text{ depth}(m_i)
$$

where $\text{depth}(x)$ is the number of nodes to be inspected on the route from the root to node $x$, $k_i$ stands for the node containing key $k_i$ [notational abuse], and $m_i$ is the dummy node between keys $k_{i-1}$ and and $k_i$.

- Note that the rules of probability require $\sum_{i=0}^{n-1} p_i + \sum_{i=0}^{n} q_i = 1$
<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>84</td>
<td>eat</td>
<td>24</td>
<td>ham</td>
<td>10</td>
<td>fox</td>
</tr>
<tr>
<td>not</td>
<td>83</td>
<td>will</td>
<td>21</td>
<td>there</td>
<td>9</td>
<td>on</td>
</tr>
<tr>
<td>them</td>
<td>61</td>
<td>sam</td>
<td>19</td>
<td>train</td>
<td>9</td>
<td>tree</td>
</tr>
<tr>
<td>a</td>
<td>59</td>
<td>with</td>
<td>19</td>
<td>anywhere</td>
<td>8</td>
<td>say</td>
</tr>
<tr>
<td>like</td>
<td>44</td>
<td>am</td>
<td>16</td>
<td>house</td>
<td>8</td>
<td>so</td>
</tr>
<tr>
<td>in</td>
<td>40</td>
<td>could</td>
<td>14</td>
<td>mouse</td>
<td>8</td>
<td>be</td>
</tr>
<tr>
<td>do</td>
<td>36</td>
<td>here</td>
<td>11</td>
<td>or</td>
<td>8</td>
<td>goat</td>
</tr>
<tr>
<td>you</td>
<td>34</td>
<td>the</td>
<td>11</td>
<td>box</td>
<td>7</td>
<td>let</td>
</tr>
<tr>
<td>would</td>
<td>26</td>
<td>eggs</td>
<td>10</td>
<td>car</td>
<td>7</td>
<td>may</td>
</tr>
<tr>
<td>and</td>
<td>24</td>
<td>green</td>
<td>10</td>
<td>dark</td>
<td>7</td>
<td>me</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Key or miss event combined frequency

```
{}                      0
a                       59
{ am and anywhere are be boat box car could dark }       92
do                      36
{ eat eggs fox goat good green ham here house }          86
i                       84
{ if let }                5
in                      40
{}                      0
like                    44
{ may me mouse }        16
not                    83
{ on or rain same say see so thank that the }           65
then                   61
{ there they train tree try will with would }          99
you                   34
{}                      0
```
\( k_i \)
\( a \) \( do \) \( i \) \( in \) \( like \) \( not \) \( then \) \( you \)
\( p_i \)
\( .073 \) \( .045 \) \( .104 \) \( .05 \) \( .055 \) \( .103 \) \( .076 \) \( .042 \)
\( q_i \)
\( .001 \) \( .113 \) \( .107 \) \( .006 \) \( .001 \) \( .02 \) \( .081 \) \( .122 \) \( .001 \)
Total weighted depth for a given tree (expected lookup cost):

\[\sum_{i=0}^{n-1} p_i \cdot \text{depth}(k_i) + \sum_{i=0}^{n} q_i \cdot \text{depth}(m_i)\]

Let \(\text{depth}_{k_a}(k_i)\) be the depth of the node with \(k_i\) in the subtree rooted at node with \(k_1\). For example, if \(k_r\) is the root of the entire tree and \(k_a\) is a child of the root, then

\[\text{depth}_{k_r}(k_i) = \text{depth}_{k_a}(k_i) + 1\]

Then we can rewrite the total weighted depth as

\[\sum_{i=0}^{r-1} p_i \cdot \text{depth}_{k_r}(k_i) + \sum_{i=0}^{r} q_i \cdot \text{depth}_{k_r}(m_i) + \sum_{i=r+1}^{n-1} p_i \cdot \text{depth}_{k_r}(k_i) + \sum_{i=r+1}^{n} q_i \cdot \text{depth}_{k_r}(m_i)\]

left subtree total weighted depth (absolute)  
right subtree total weighted depth (absolute)
Again, let $k_r$ be the root of the entire tree and $k_a$ and $k_b$ be the root’s children. Then

\[
\begin{align*}
& \sum_{i=0}^{r-1} p_i \left( \text{depth}_{k_a}(k_i) + 1 \right) + \sum_{i=0}^{r} q_i \left( \text{depth}_{k_a}(m_i) + 1 \right) + p_r + \sum_{i=r+1}^{n-1} p_i \left( \text{depth}_{k_b}(k_i) + 1 \right) \quad (\text{left subtree total weighted depth (absolute)}) \\
& \quad + \sum_{i=r+1}^{n} q_i \left( \text{depth}_{k_b}(m_i) + 1 \right) \quad (\text{right subtree total weighted depth (absolute)})
\end{align*}
\]

Convert to “relative depth”:

\[
\begin{align*}
& \sum_{i=0}^{n-1} p_i + \sum_{i=0}^{n} q_i + \sum_{i=0}^{r-1} p_i \text{ depth}_{k_a}(k_i) + \sum_{i=0}^{r} q_i \text{ depth}_{k_a}(m_i) + \sum_{i=r+1}^{n-1} p_i \text{ depth}_{k_b}(k_i) + \sum_{i=r+1}^{n} q_i \text{ depth}_{k_b}(m_i) \\
& \quad (\text{total probability}) \quad (\text{left subtree total weighted depth (relative)}) \quad (\text{right subtree total weighted depth (relative)})
\end{align*}
\]

Let $TWD(k)$ be the total weighted depth of the tree rooted at $k$ (relative to $k$) and $TP(k)$ be the total probability of the tree rooted at $k$. Then

\[TWD(k_r) = TP(k_r) + TWD(k_a) + TWD(k_b)\]
Let \( P[i][j] \) be the total probabilities of the keys and misses in the range \([i, j]\):

\[
P[i][j] = \sum_{k=i}^{j} p_k + \sum_{k=i}^{j+1} q_k
\]

Let \( C[i][j] \) be the least total weighted depth of any BST composed from keys in the range \([i, j]\). The recursive characterization is

\[
C[i][j] = \begin{cases} 
2q_i + p_i + 2q_{i+1} & \text{if } i = j \\
P[i][j] + \min \left\{ \begin{array}{l} q_i + C[i + 1][j] \\ C[i][r - 1] + C[r + 1][j] \text{ for } r \in (i, j) \\ C[i][j - 1] + q_{j+1} \end{array} \right\} & \text{if } i < j
\end{cases}
\]
\[
C[i][j] = \begin{cases} 
2q_i + p_i + 2q_{i+1} & \text{if } i = j \\
P[i][j] + \min \left\{ \begin{array}{l}
q_i + C[i + 1][j] \\
C[i][r - 1] + C[r + 1][j] \text{ for } r \in (i, j) \\
C[i][j - 1] + q_{j+1}
\end{array} \right\} & \text{if } i < j
\end{cases}
\]
\[ C[i][j] = \begin{cases} 
2q_i + p_i + 2q_{i+1} & \text{if } i = j \\
P[i][j] + \min \left\{ \begin{array}{l} 
q_i + C[i + 1][j] \\
C[i][r - 1] + C[r + 1][j] \quad \text{for} \quad r \in (i, j) \\
C[i][j - 1] + q_{j+1} \end{array} \right\} & \text{if } i < j 
\end{cases} \]
\[
C[i][j] = \begin{cases} 
2q_i + p_i + 2q_{i+1} & \text{if } i = j \\
P[i][j] + \min \left\{ q_i + C[i+1][j], C[i][r-1] + C[r+1][j] \text{ for } r \in (i, j) \right\} & \text{if } i < j 
\end{cases}
\]

\[
P[i][j] = \begin{cases} 
q_i + p_i + q_{i+1} & \text{if } i = j \\
\begin{cases} 
q_i + p_i + P[i+1][j] \\
or \quad P[i][r-1] + p_r + P[r+1][j] \text{ for } r \in (i, j) \\
or \quad P[i][j-1] + p_j + q_{j+1} \end{cases} & \text{if } i < j 
\end{cases}
\]
<table>
<thead>
<tr>
<th>i</th>
<th>0</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>0</td>
<td>.073</td>
<td>.045</td>
<td>.104</td>
<td>.05</td>
<td>.055</td>
<td>.103</td>
<td>.076</td>
<td>.042</td>
</tr>
<tr>
<td>1</td>
<td>.001</td>
<td>.113</td>
<td>.107</td>
<td>.006</td>
<td>.001</td>
<td>.02</td>
<td>.081</td>
<td>.122</td>
</tr>
</tbody>
</table>

```
2.916/2
2.125/5
.818/1
.691/2
1.458/6
1.873/5
1.783/2
1.538/2
1.24/1
1.202/2
1.247/2
1.212/5
1.227/6
1.108/2
.975/2
.666/2
.613/5
1.018/6
1.038/6
.748/1
.818/1
.439/2
.216/4
.438/5
.829/6
.691/6
.301
.485
.33
.064
.097
.305
.482
.288
```
Coming up:

Catch up on projects . . .

Due Fri, Apr 1 (end of day)
Do Project 6.1.b as a practice problem
Take quiz (on Section 6.4)

Due Mon, Apr 4 (end of day—though a skim is recommended for Apr 1)
Read Section 6.5
(No quiz on Section 6.5)

(See Schoology for practice problems for Test 2)

Do Optimal BST project (suggested by Friday, April 9)