Optimal Binary Search Tree

We solved the Optimal Binary search tree three ways
(See http://www.cs.usu.edu/~allanv/cs5050/cs5050.html Problem 1)
(1) Greedily
(2) Using exhaustive recursion
(3) Using Memoizing
(4) A logical extension is to solve it via dynamic programming.

Things to add:
1. Interact with the user so they can pick problem size or frequency values
2. Allow the user to pick a greedy strategy
   a. Select largest frequency
   b. Select node nearest the middle of the keys (to get a balanced tree)
   c. Other strategies? Perhaps build the tree from the bottom up – picking a sequence whose total frequency was smallest.
3. Show how you use dynamic programming to not only find the cost of the optimal binary search tree, but build it.

<table>
<thead>
<tr>
<th>i</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_i$</td>
<td>0.24</td>
<td>0.22</td>
<td>0.23</td>
<td>0.3</td>
<td>0.01</td>
</tr>
</tbody>
</table>

Backtrack the Optimal BST

- Recording every $r$ during filling.
4. Show the tree you create graphically. See the demo at http://webpages.ull.es/users/jriera/Docencia/AVL/AVL%20tree%20applet.htm for ideas.

Perhaps you would show the nodes with a size proportional to its frequency.

5. Demonstrate the time required to use the various methods. An animation found at http://www.cs.auckland.ac.nz/software/AlgAnim/opt_bin.html may serve as inspiration.

6. Allow the user to "pull" on the nodes of a tree to suggest a better tree. When a user wants a node to "move up" – this could be implemented with a rotation (as in splay trees).