optimal binary search tree

CIS 315
also known as OBST

build a BST on given words (or values)

\( v_1 \ v_2 \ \ldots \ v_n \) each with probabilities

\( p_1 \ p_2 \ \ldots \ p_n \)

probabilities: \( p_1 + p_2 + \ldots + p_n = 1 \) (not strictly necessary)

goal:
optimize expected search time

\[ \sum p_i \cdot \text{depth}(v_i) \]
example

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.3</td>
<td>0.2</td>
<td>0.3</td>
<td>0.1</td>
</tr>
</tbody>
</table>

expected depth:
0.1 \cdot 1 + 0.3 \cdot 0 + 0.2 \cdot 2 + 0.3 \cdot 3 + 0.1 \cdot 1 = 1.5

eXpected depth:
0.1 \cdot 2 + 0.3 \cdot 1 + 0.2 \cdot 2 + 0.3 \cdot 0 + 0.1 \cdot 1 = 1.0
usual approach

try all possible $v_k$ as root

get optimal left and right subtrees recursively

important note: if

is optimal for $abc$, then the **extra** contribution of that subtree in

is $p_1 + p_2 + p_3$ (because their depth has increased by one)
subproblem/recurrence

\[ m[i,j] \] is the expected depth of a node in the OBST for nodes \( v_i, v_{i+1}, \ldots, v_j \)

\[
\begin{align*}
m[i,i-1] &= 0 \quad \text{-- no nodes} \\
m[i,i] &= 0 \quad \text{-- one node} \\
m[i,j] &= \min_{i \leq k \leq j} (m[i,k-1] + m[k+1,j] + (p_i + \ldots + p_{k-1}) + (p_k + \ldots + p_j))
\end{align*}
\]
partially worked example

<table>
<thead>
<tr>
<th></th>
<th>$p_1$</th>
<th>$p_2$</th>
<th>$p_3$</th>
<th>$p_4$</th>
<th>$p_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.1</td>
<td>0.3</td>
<td>0.2</td>
<td>0.3</td>
<td>0.1</td>
</tr>
</tbody>
</table>

$m[i,j]$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
<td>0.3</td>
<td>0.8</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
<td>0.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0.2</td>
<td>0.3</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0.1</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

$r[i,j]$ is the minimum $k$ used by $m[i,j]$. 

r[i,j] is the minimum k used by m[i,j]
before we write code

\[ m[i,j] = \text{MIN}_{i \leq k \leq j}( m[i,k-1] + m[k+1,j] + (p_i + \ldots + p_{k-1}) + (p_{k+1} + \ldots + p_j) ) \]

may be hard to calculate because of the sum within the MIN

let \( s[i,j] = p_i + p_{i+1} + \ldots + p_j \)

\( (\text{maybe pre-compute it}) \)

now

\[ m[i,j] = \text{MIN}_{i \leq k \leq j}( m[i,k-1] + m[k+1,j] + s[i,j] - p_k ) \]
code – version 1

for $i = 1$ to $n$
    
    $m[i,i]=m[i,i-1]=0$
    $s[i,i]=p_i$

for $d=1$ to $n-1$
    for $i=1$ to $n-d$
        
        $j=i+d$
        $s[i,j]=s[i,j-1]+p_j$

    $m[i,j]=\text{MIN}_{i\leq k\leq j}(m[i,k-1]+m[k+1,j]+s[i,j]-p_k)$
fuller code

for i = 1 to n
  m[i,i]=m[i,i-1]=0
  s[i,i]=p_i
for d=1 to n-1
  for i=1 to n-d
    j=i+d
    s[i,j]=s[i,j-1]+p_j
    m[i,j]=m[i,i-1]+m[i+1,j]+s[i,j]-p_i
    r[i,j]=i
  for k = i+1 to j
    exp=m[i,k-1]+m[i+1,j]+s[i,j]-p_k
    if exp<m[i,j] then
      m[i,j]=exp
      r[i,j]=k
return m[1,n]

looks like O(n^3) time O(n^2) space
r is used to reconstruct tree