1 Finding the optimal binary search tree

We are given \( n \) keys \( k_1, k_2, \ldots, k_n \) and a probability \( p_i \) that a key \( k_i \) is queried. We assume that \( \sum_{i=1}^{n} p_i = 1 \) that is, we only allow queries in the set of keys (alternately, we assume we have listed all possible queries). Given a particular binary search tree \( T \), we compute the cost of the tree as

\[
\sum_{1 \leq i \leq n} \text{depth}_T(k_i) p_i,
\]

which is the average number of queries needed to find a key: we ask for key \( k_i \) with probability \( k_i \) and finding it will take \( \text{depth}_T(k_i) \) queries. (The root of the tree has depth 1.)

How do we find the optimal binary search tree? Suppose \( k_r \) is in the root of the tree, then \( k_1, \ldots, k_{r-1} \) are to the left of the root and \( k_{r+1}, \ldots, k_n \) are to the right of the root. Call the trees they form \( T_{1r-1} \) and \( T_{r+1n} \), respectively. Then both of these trees are optimal binary search trees. So we can solve the problem by trying all possibilities for \( k_r \) and then computing the optimal search trees on both sides recursively; or, actually, using dynamic programming.

Let \( T_{ij} \) be the optimal binary search tree for keys \( k_i, \ldots, k_j \), and let \( c_{ij} \) be the cost of \( T_{ij} \). Let \( p_{ij} \) be the probability that we ask for a key in \( T_{ij} \), in other words:

\[
p_{ij} = \sum_{k=i}^{j} p_k.
\]

Then

\[
c_{ij} = \min_{i \leq r \leq j} \left[ (c_{i \ r-1} + p_{i \ r-1}) + (c_{r+1 \ j} + p_{r+1 \ j}) + 1 \ast p_r \right],
\]
because if \( k_r \) is at the root of the tree, the left tree has cost \( c_{ir-1} \) to which we must add \( 1 \times p_{ir-1} \), because we will ask for a key in that tree with probability \( p_{ir-1} \), increasing the average height of that tree by 1 with that probability to get \( (c_{ir-1}+p_{ir-1}) \). The same reasoning applies to the right tree, and the root will cost us 1 query with a probability of \( p_r \). Now,

\[
c_{ij} = \min_{i \leq r \leq j} [(c_{i \cdot r-1} + p_{i \cdot r-1}) + (c_{r+1 \cdot j} + p_{r+1 \cdot j}) + 1 \times p_r]
\]

\[
= p_{ij} + \min_{i \leq r \leq j} [c_{i \cdot r-1} + c_{r+1 \cdot j}]
\]

Since \( p_{i \cdot r-1} + p_{r+1 \cdot j} + p_r = p_{ij} \), by definition.

Here is a small example:

\[
\begin{array}{cccccc}
  & 1 & 2 & 3 & 4 & 5 \\
k & 0.2 & 0.1 & 0.15 & 0.25 & 0.3 \\
p & 0.2 & 0.3 & 0.45 & 0.7 & 1 \\
  & 0.1 & 0.25 & 0.5 & 0.8 \\
  & 0.15 & 0.4 & 0.7 \\
  & 0.25 & 0.55 \\
  & 0.3 \\
\end{array}
\]

We first precompute the \( p_{ij} \), using dynamic programming (details left to the reader ...).

Now, \( c_{ii} = p_i \), since there is only one key in the tree. So we start the matrix of \( c_{ij} \) as

\[
\begin{array}{cccccc}
  & 0.2 & * & * & * & * \\
  & 0.1 & * & * & * \\
  & 0.15 & * & * \\
  & 0.25 & * \\
  & 0.3 \\
\end{array}
\]

Let us compute \( c_{12} \). There are two possibilities: \( k_1 \) is on top, or \( k_2 \) is on top. In the first case, the cost of the tree is \( c_{11} + (c_{22} + p_{22}) = 0.4 \), in the second case, the tree costs \( c_{22} + (c_{11} + p_{11}) = 0.5 \); let us double-check with the formula we derived earlier:

\[
c_{12} = p_{12} + \min(c_{11}, c_{22}),
\]
that is, \( c_{12} = 0.3 + \min(0.2, 0.1) = 0.4 \), which checks with our earlier com-
putation. So putting \( k_1 \) on top is the cheaper choice for keys \( k_1, k_2 \):

\[
\begin{array}{cccccc}
0.2 & 0.4 & * & * & * \\
0.1 & * & * & * \\
0.15 & * & * \\
0.25 & * \\
0.3
\end{array}
\]

Similarly, \( c_{23} = \min(0.35, 0.4) = 0.35 \).

\[
\begin{array}{cccccc}
0.2 & 0.4 & * & * & * \\
0.1 & 0.35 & * & * \\
0.15 & * & * \\
0.25 & * \\
0.3
\end{array}
\]

Next, we can compute \( c_{34} = 0.55 \) and \( c_{45} = 0.8 \):

\[
\begin{array}{cccccc}
0.2 & 0.4 & * & * & * \\
0.1 & 0.35 & * & * \\
0.15 & 0.55 & * \\
0.25 & 0.8 \\
0.3
\end{array}
\]

As a final example, let us compute \( c_{13} \). Now there are three possibilities:
\( k_1, k_2, \) or \( k_3 \) on top. The first possibility costs \( p_{13} + c_{23} = 0.45 + 0.35 = 0.8 \),
the second \( p_{13} + c_{11} + c_{33} = 0.45 + 0.2 + 0.15 = 0.8 \) and the third \( p_{13} + c_{12} = 
0.45 + 0.4 = 0.85 \), so we go with either the first or the second choice (they
are equally good) for a cost of 0.8:

\[
\begin{array}{cccccc}
0.2 & 0.4 & 0.8 & * & * \\
0.1 & 0.35 & * & * \\
0.15 & 0.7 & * \\
0.25 & 0.8 \\
0.3
\end{array}
\]