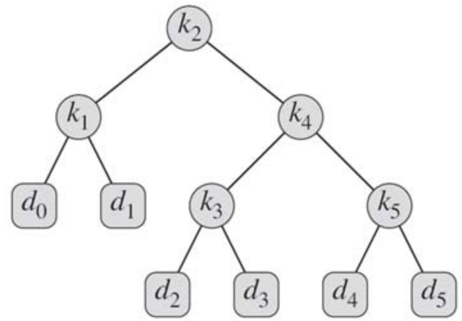
Optimal Binary Search Trees (1/2)

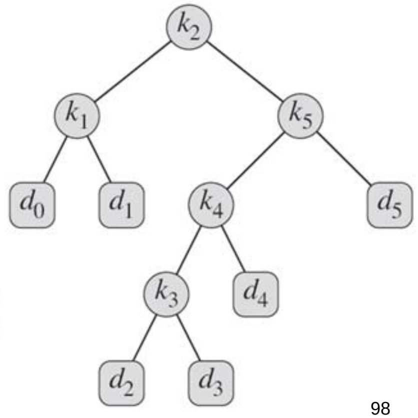
- Optimal Binary Search Trees: For a given set of probabilities, we wish to construct a binary search tree whose expected search cost is smallest.
- Is an optimal binary search tree always a tree whose overall height is smallest??
- Should we always put the key with the greatest probability at the root to construct an optimal binary search tree??

Optimal Binary Search Trees (2/2)

i	0	1	2	3	4	5
p_i		0.15	0.10	0.05	0.10	0.20
q_i	0.05	0.10	0.05	0.05	0.05	0.10

What is the expected cost?





E[search cost in T] = 2.80

Step 1: The Structure of an Optimal Binary Search Tree (1/2)

- If an optimal binary search tree T has a subtree T' containing keys k_i, …, k_j, then this subtree T' must be optimal as well for the subproblem with keys k_i, …, k_j and dummy keys d_{i-1}, …, d_j.
- Given keys k_i, …, k_j, one of these keys, say k_r ($i \le r \le j$), is the root of an optimal subtree containing these keys.
  - The left subtree of the root k_r contains the keys k_i, …, k_{r-1} (and dummy keys d_{i-1}, …, d_{r-1}), and the right subtree contains the keys k_{r+1}, …, k_j (and dummy keys d_r, …, d_j).

Step 1: The Structure of an Optimal Binary Search Tree (2/2)

- As long as we examine all candidate roots k_r, where $i \le r \le j$, and we determine all optimal binary search trees containing k_i, …, k_{r-1} and those containing k_{r+1}, …, k_j, we are guaranteed that we will find an optimal binary search tree.
- Suppose that in a subtree with keys k_i, …, k_j, we select k_i (k_j) as the root. We interpret k_i's left subtree (k_j's right subtree) that contains the keys k_i, …, k_{i-1} (k_{j+1}, …, k_j) as containing no keys.

Step 2: A Recursive Solution (1/2)

- Let us define $e[i, j]$ as the expected cost of searching an optimal binary search tree containing the keys k_i, …, k_j.
  - Ultimately, we wish to compute $e[1, n]$.
- For a subtree with keys k_i, …, k_j, let us denote this sum of probabilities as

$$w(i, j) = \sum_{l=i}^{j} p_l + \sum_{l=i-1}^{j} q_l \ .$$

$$e[i, j] = \begin{cases} q_{i-1} & \text{if } j = i - 1, \\ \min\limits_{i \le r \le j} \{e[i, r-1] + e[r+1, j] + w(i, j)\} & \text{if } i \le j \ . \end{cases}$$

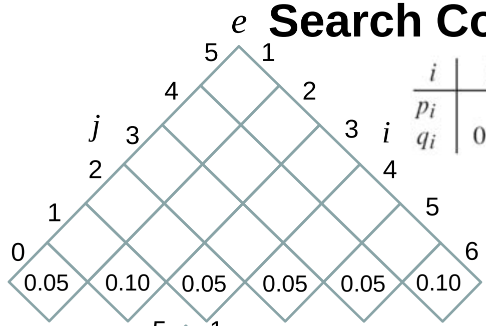
Step 2: A Recursive Solution (2/2)

- We define $root[i, j]$, for $1 \le i \le j \le n$, to be the index r for which k_r is the root of an optimal binary search tree containing keys k_i, …, k_j.

Step 3: Computing the Expected Search Cost (1/7)

```
OPTIMAL-BST(p, q, n)
let e[1..n+1, 0..n], w[1..n+1, 0..n],
        and root[1..n, 1..n] be new tables
for i = 1 to n + 1
    e[i, i-1] = q_{i-1}
    w[i, i-1] = q_{i-1}
for l = 1 to n
    for i = 1 to n - l + 1
        j = i + l - 1
        e[i, j] = ∞
        w[i, j] = w[i, j-1] + p_j + q_j
        for r = i to j
            t = e[i, r-1] + e[r+1, j] + w[i, j]
            if t < e[i, j]
                e[i, j] = t
                root[i, j] = r
return e and root
```


Step 3: Computing the Expected Search Cost (2/7)



i	0	1	2	3	4	5
p_i		0.15	0.10	0.05	0.10	0.20
q_i	0.05	0.10	0.05	0.05	0.05	0.10