طراحی و تحلیل الگوریتم ها

دکتر امیر لکی زاده

استادیار گروه مهندسی کامپیوتر دانشگاه قم

4. Optimal binary search trees

Suppose that we are designing a program to translate text from English to French.

For each occurrence of each English word in the text, we need to look up its French equivalent. One way to perform these lookup operations is to build a binary search tree with n English words as keys and French equivalents as satellite data. Because we will search the tree for each individual word in the text, we want the total time spent searching to be as low as possible.

4. Optimal binary search trees

What we need is known as an ***optimal binary search tree***. Formally, we are given a sequence $K = \langle k_1, k_2, \ldots, k_n \rangle$ of n distinct keys in sorted order (so that $k_1 < k_2 < \cdots < k_n$), and we wish to build a binary search tree from these keys.

For each key k_i, we have a probability p_i that a search will be for k_i. Some searches may be for values not in K, and so we also have $n + 1$ "dummy keys" $d_0, d_1, d_2, \ldots, d_n$ representing values not in K. In particular, d_0 represents all values less than k_1, d_n represents all values greater than k_n, and for $i = 1, 2, \ldots, n-1$, the dummy key d_i represents all values between k_i and k_{i+1}. For each dummy key d_i, we have a probability q_i that a search will correspond to d_i.

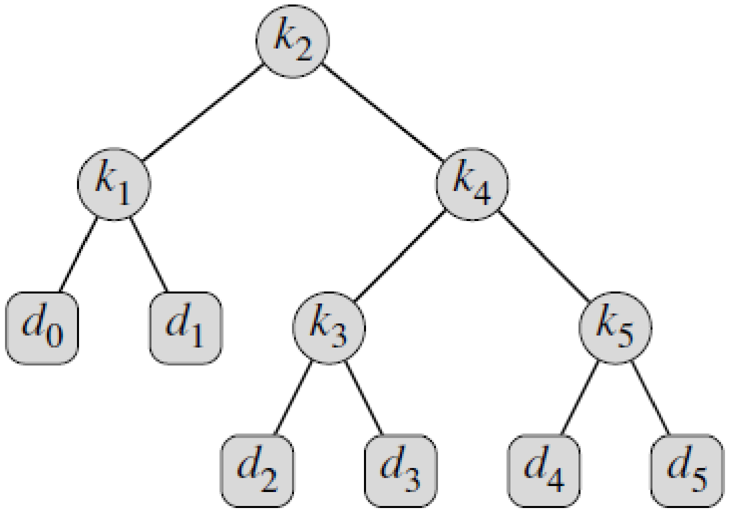
4. Optimal binary search trees

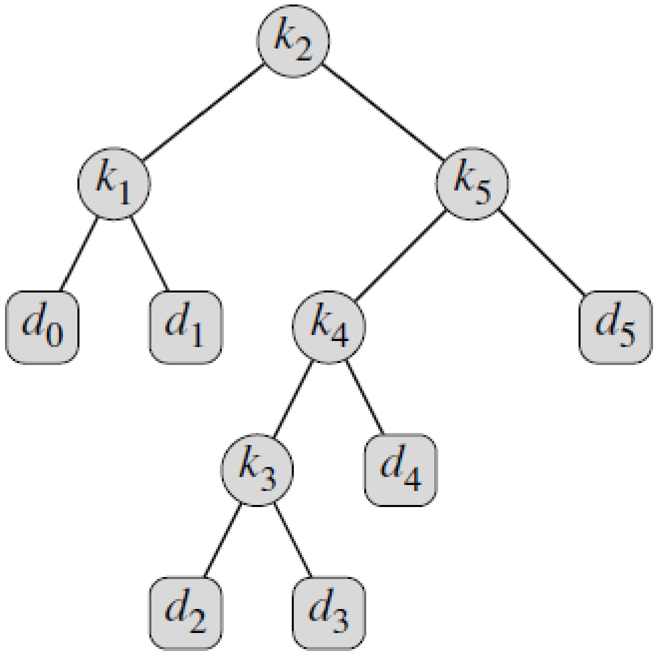
Every search is either successful (finding
some key k_i) or unsuccessful (finding some dummy key d_i), and so we have

$$\sum_{i=1}^{n} p_i + \sum_{i=0}^{n} q_i = 1 \, .$$

4. Optimal binary search trees

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





4. Optimal binary search trees

$$
\begin{aligned}
\mathrm{E}\left[\text{search cost in } T\right] &= \sum_{i=1}^{n} (\operatorname{depth}_T(k_i) + 1) \cdot p_i + \sum_{i=0}^{n} (\operatorname{depth}_T(d_i) + 1) \cdot q_i \\
&= 1 + \sum_{i=1}^{n} \operatorname{depth}_T(k_i) \cdot p_i + \sum_{i=0}^{n} \operatorname{depth}_T(d_i) \cdot q_i \,, \qquad (15.16)
\end{aligned}
$$

نکاتی درباره مثال اخیر:

4. Optimal binary search trees

$k_i, \ldots, k_j,$

k_r $(i \leq r \leq j)$, will be the root

The left subtree of the root k_r will contain the keys $k_i, \ldots, k_{r-1}$ (and dummy keys $d_{i-1}, \ldots, d_{r-1}$), and the right subtree will contain the keys $k_{r+1}, \ldots, k_j$ (and dummy keys $d_r, \ldots, d_j$).

4. Optimal binary search trees

Let us define $e[i, j]$ as
the expected cost of searching an optimal binary search tree containing the keys $k_i, \ldots, k_j$. Ultimately, we wish to compute $e[1, n]$.

4. Optimal binary search trees

the expected cost of searching an optimal binary search tree containing the keys $k_i, \ldots, k_j$. Ultimately, we wish to compute $e[1, n]$.

The easy case occurs when $j = i - 1$. Then we have just the dummy key d_{i-1}. The expected search cost is $e[i, i - 1] = q_{i-1}$.

4. Optimal binary search trees

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

$$e[i, j] = p_r + (e[i, r-1] + w(i, r-1)) + (e[r+1, j] + w(r+1, j)) \ .$$

Noting that

$$w(i, j) = w(i, r-1) + p_r + w(r+1, j) \ ,$$

we rewrite $e[i, j]$ as

$$e[i, j] = e[i, r-1] + e[r+1, j] + w(i, j) \ .$$

4. Optimal binary search trees

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

$root[i, j]$

$e[1 \,..\, n+1, 0 \,..\, n]$

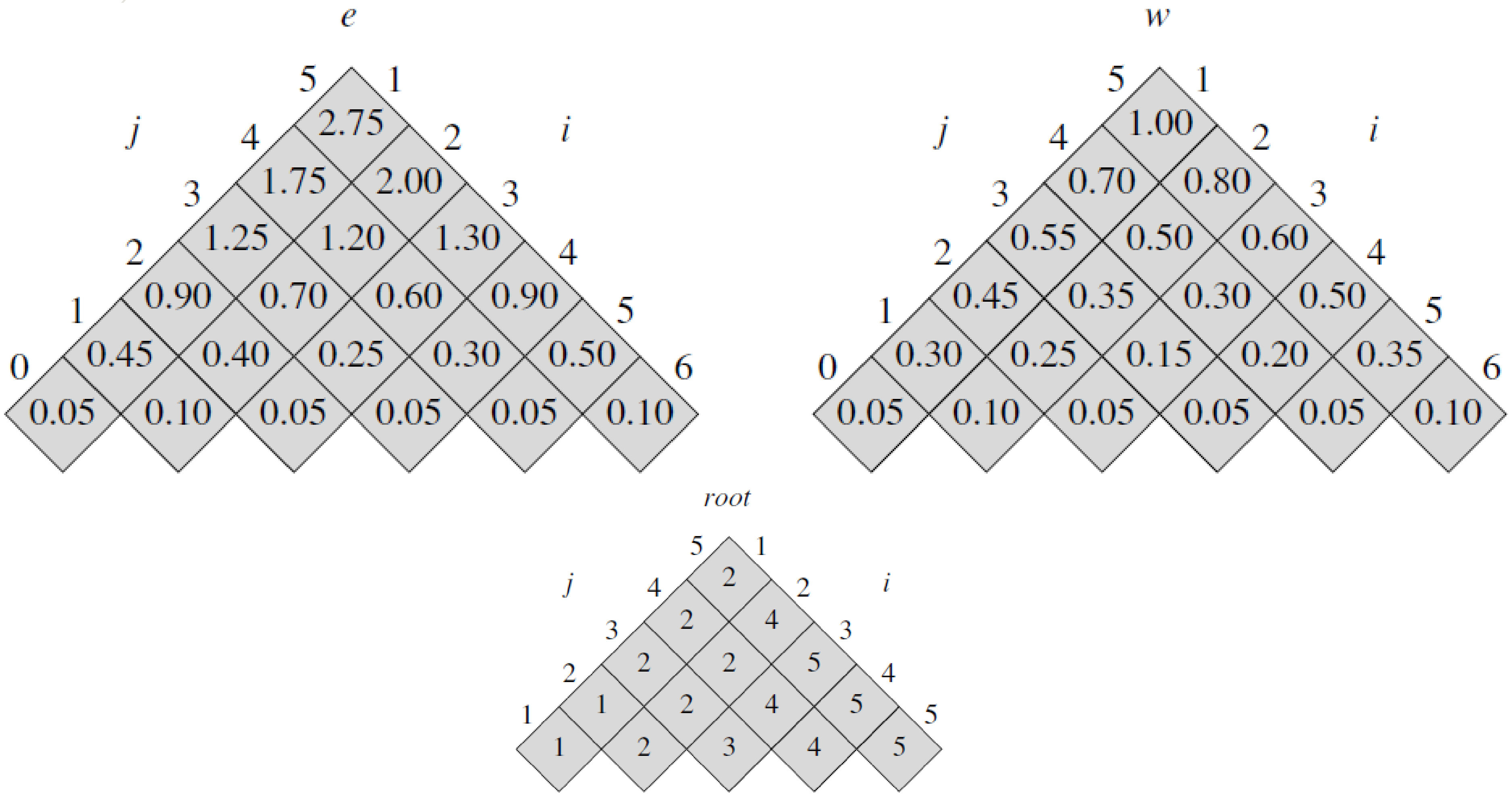
$w[i, i-1]$

$$w[i, j] = w[i, j-1] + p_j + q_j\,.$$

4. Optimal binary search trees

```
OPTIMAL-BST(p, q, n)
for i ← 1 to n + 1
    do e[i, i − 1] ← q_{i−1}
       w[i, i − 1] ← q_{i−1}
for l ← 1 to n
    do for i ← 1 to n − l + 1
           do j ← i + l − 1
              e[i, j] ← ∞
              w[i, j] ← w[i, j − 1] + p_j + q_j
              for r ← i to j
                  do t ← e[i, r − 1] + e[r + 1, j] + w[i, j]
                     if t < e[i, j]
                        then e[i, j] ← t
                             root[i, j] ← r
return e and root
```

4. Optimal binary search trees



4. Optimal binary search trees

15.5-4 ★

Knuth [184] has shown that there are always roots of optimal subtrees such that $root[i, j-1] \leq root[i, j] \leq root[i+1, j]$ for all $1 \leq i < j \leq n$. Use this fact to modify the OPTIMAL-BST procedure to run in $\Theta(n^2)$ time.

15-7 Scheduling to maximize profit

Suppose you have one machine and a set of n jobs $a_1, a_2, \ldots, a_n$ to process on that machine. Each job a_j has a processing time t_j, a profit p_j, and a deadline d_j. The machine can process only one job at a time, and job a_j must run uninterruptedly for t_j consecutive time units. If job a_j is completed by its deadline d_j, you receive a profit p_j, but if it is completed after its deadline, you receive a profit of 0. Give an algorithm to find the schedule that obtains the maximum amount of profit, assuming that all processing times are integers between 1 and n. What is the running time of your algorithm?