طراحی الگوریتم

۱۴ مهر ۹۸

ملکی مجد

Topic	Reference
Recursion and Backtracking	Ch.1 and Ch.2 JeffE
Dynamic Programming	Ch.3 JeffE and Ch.15 CLRS
Greedy Algorithms	Ch.4 JeffE and Ch.16 CLRS
Amortized Analysis	Ch.17 CLRS
Elementary Graph algorithms	Ch.6 JeffE and Ch.22 CLRS
Minimum Spanning Trees	Ch.7 JeffE and Ch.23 CLRS
Single-Source Shortest Paths	Ch.8 JeffE and Ch.24 CLRS
All-Pairs Shortest Paths	Ch.9 JeffE and Ch.25 CLRS
Maximum Flow	Ch.10 JeffE and Ch.26 CLRS
String Matching	Ch.32 CLRS
NP-Completeness	Ch.12 JeffE and Ch.34 CLRS

مباحث

- ادامه برنامه نویسی پویا
  - درخت بهینه جستجوی دودویی
    - برای پیاده سازی دیکشنری با کمترین زمان مورد انتظار برای پیدا کردن هر کلمه

- الگوریتم حریصانه
  - زمان بندی بیشترین تعداد فعالیت

Look-up time
Optimal Binary Search Trees

- 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
- total time spent searching to be as low as possible
  - ensure an $O(\lg n)$ search time per occurrence by using a red-black tree
- case that a frequently used word such as "the" appears far from the root while a rarely used word such as "mycophagist" appears near the root
  - slow down the translation

Optimal Binary Search Trees

given a sequence $K = \langle k_1, k_2, \ldots, k_n \rangle$ of n distinct keys in sorted order

$k_1 < k_2 < \cdots < k_n$

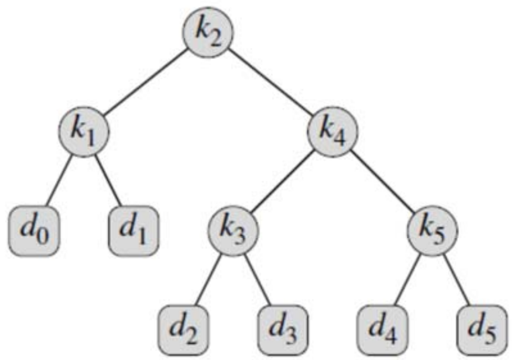
$d_0, d_1, d_2, \ldots, d_n$ representing values not in K

d_i represents all values between k_i and k_{i+1}.

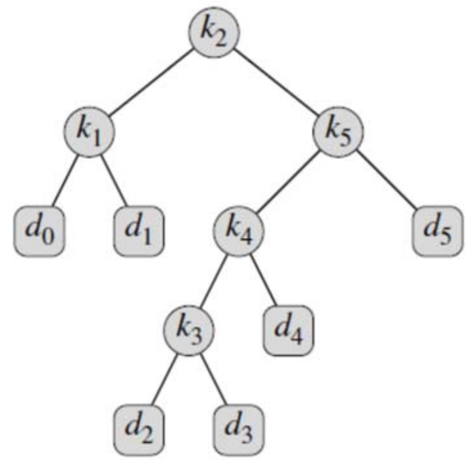
$$\sum_{i=1}^{n} p_i + \sum_{i=0}^{n} q_i = 1$$

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

Optimal Binary Search Trees example



expected search cost 2.80.



expected search cost 2.75.

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

Optimal Binary Search Trees

The structure of an optimal binary search tree (1)

range $k_i, \ldots, k_j$, for some $1 \leq i \leq j \leq n$

a subtree	keys $k_i, \ldots, k_j$	
	leaves	dummy keys $d_{i-1}, \ldots, d_j$

- if an optimal BS tree *T* has a subtree T' containing keys *i to j*
  - then this subtree T' must be optimal as well
  - cut-and-paste argument applies

Optimal Binary Search Trees

The structure of an optimal binary search tree (2)

$k_i, \ldots, k_j$, one of these keys, say k_r ($i \le r \le j$), will be the root of an optimal subtree

left subtree

$k_i, \ldots, k_{r-1}$ (and dummy keys $d_{i-1}, \ldots, d_{r-1}$)

right subtree

keys $k_{r+1}, \ldots, k_j$ (and dummy keys $d_r, \ldots, d_j$)

k_i's left subtree contains the keys $k_i, \ldots, k_{i-1}$

keys $k_i, \ldots, k_{i-1}$ has no actual keys but does contain the single dummy key d_{i-1}

Optimal Binary Search Trees
A recursive solution (1)

- e[1 , n]
- easy case occurs when $j = i - 1$.

Optimal Binary Search Trees
A recursive solution (2)

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

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

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

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

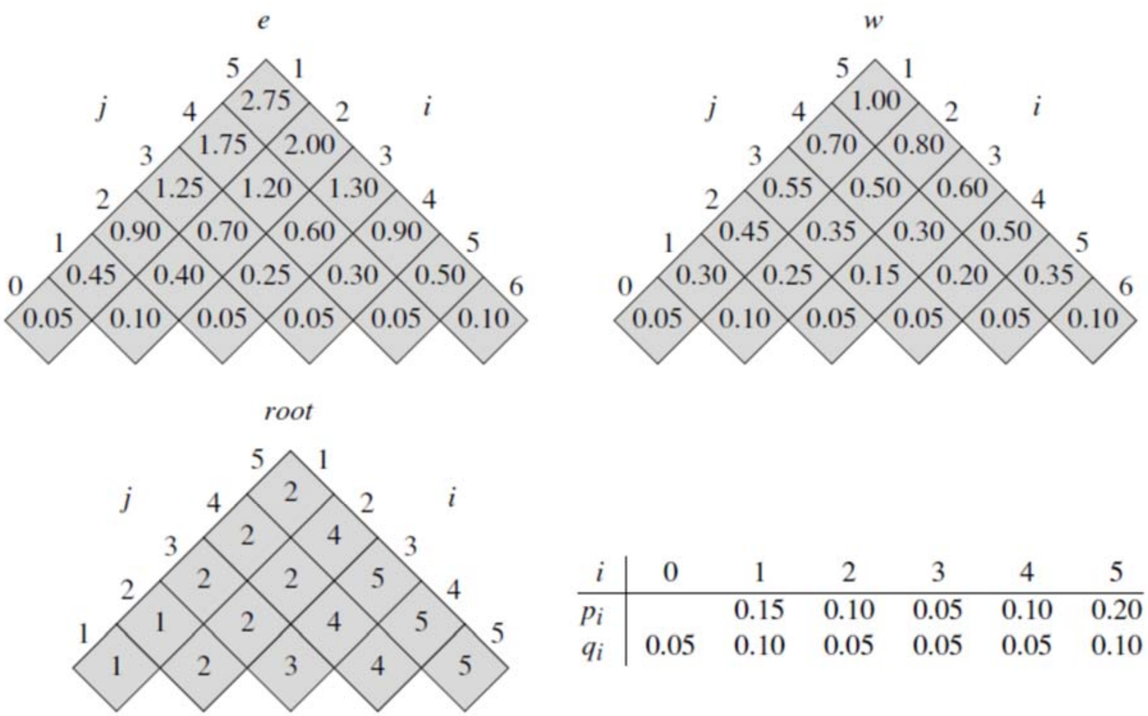
Optimal Binary Search Trees

Computing the expected search cost of an optimal binary search tree

```
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
```

Optimal Binary Search Trees

example



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

Greedy approach

An activity-selection problem

- select a maximum-size subset of mutually compatible activities
  - 2 4 9 11
  - 1 4 8 11

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	8	9	10	11	12	13	14