Dynamic Programming

Optimal BSTs: Optimal Substructure

Algorithms: Design and Analysis, Part II

Problem Definition

Input: frequencies $p_1, p_2, \ldots, p_n$ for items $1, 2, \ldots, n$.
[assume items in sorted order, $1 < 2 < 3 < \cdots < n$]

Goal: compute a valid search tree that minimizes the weighted (average) search time:

$$C(T) = \sum_{\text{items } i} p_i \cdot [\text{search time for } i \text{ in } T]$$

[search time for i in T] = depth of i in T + 1

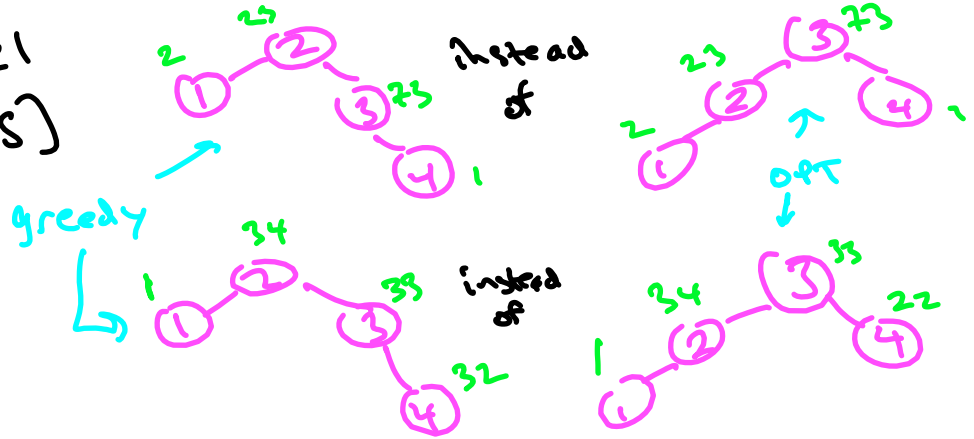
Greedy Doesn't Work

Intuition: want the most (respectively, least) frequently accessed items closest (respectively, furthest) from the root.

Ideas for greedy algorithms:

- bottom-up [populate lowest level with least frequently accessed keys]
- top-down [put most frequently accessed item at root, recurse]

Counter examples:



Choosing the Root

Issue: with the top-down approach, the choice of root has hard-to-predict repercussions further down the tree.

[strymies both greedy and naive divide+conquer approaches]

Idea: what if we knew the root?

(i.e., may be can try all possibilities within a dynamic programming algorithm!)

Optimal Substructure

Question: Suppose an optimal BST for keys $\{1, 2, \ldots, n\}$ has root r, left subtree T_1, right subtree T_2.

Pick the strongest statement that you suspect is true.

(A) Neither T_1 nor T_2 need be optimal for the items it contains.

(B) At least one of T_1, T_2 is optimal for the items it contains.

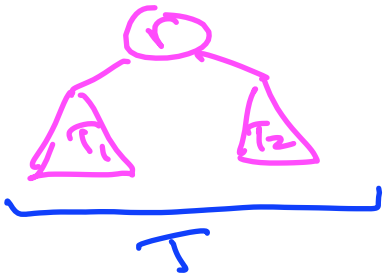
(C) Each of T_1, T_2 is optimal for the items it contains.

(D) T_1 is optimal for the keys $\{1, 2, \ldots, r-1\}$ and T_2 for the keys $\{r+1, r+2, \ldots, n\}$

Tim Roughgarden

Proof of Optimal Substructure

Let T be an optimal BST for keys $\{1, 2, \ldots, n\}$ with frequencies $p_1, \ldots, p_n$. Suppose T has root r.



Suppose for contradiction that T_1 is not optimal for $\{1, 2, \ldots, r-1\}$ [other case is similar]

with $C(T_1^*) < C(T_1)$.

Obtain T^* from T by "cutting + pasting" T_1^* in for T_1.

Note: to complete contradiction + proof, only need to show that $C(T^*) < C(T)$

Proof of Optimal Substructure (con'd)

A Calculation:

$$C(T) = \sum_{i=1}^{n} p_i \cdot [\text{search time for } i \text{ in } T]$$

$$= p_r \cdot 1 + \sum_{i=1}^{r-1} p_i \cdot [\text{search time for } i \text{ in } T] + \sum_{i=r+1}^{n} p_i \cdot [\text{search time for } i \text{ in } T]$$

$= 1 +$ search time for i in T_1

$= 1 +$ search time for i in T_2

$$= \sum_{i=1}^{n} p_i + \sum_{i=1}^{r-1} p_i \cdot [\text{search time for } i \text{ in } T_1] + \sum_{i=r+1}^{n} p_i \cdot [\text{search time for } i \text{ in } T_2]$$

- $\sum_{i=1}^{n} p_i$: a constant (independent of T)
- $\sum_{i=1}^{r-1} p_i \cdot [\text{search time for } i \text{ in } T_1] = C(T_1)$
- $\sum_{i=r+1}^{n} p_i \cdot [\text{search time for } i \text{ in } T_2] = C(T_2)$

Similarly: $C(T^*) = \sum_{i=1}^{n} p_i + C(T_1^*) + C(T_2)$

Upshot: $C(T_1^*) < C(T_1)$ implies $C(T^*) < C(T)$, contradicting optimality of T. QED!