Optimal Binary Search Tree (BST)

Solving the optimal binary search tree can be viewed as finding a binary search tree for an array of sorted elements with individual access frequencies, specified as input, such that the average search time of the BST is minimized.

Recurrence

$$\mathrm{B}(\varphi,\, \tau,\, \delta) = \min_{i=\varphi}^{\tau} \big(\Phi_i \times \delta + \mathrm{B}(\varphi,\, i-1,\, \delta+1) + \mathrm{B}(i+1,\, \tau,\, \delta+1)\big)$$

where

φ	:	from index
τ	:	to index
δ	:	depth
Φ_i	:	access frequency of index i
$\mathrm{B}(i,\, j,\, *)$	:	0 if $j < i$

For example, say we have 50 elements in the array. The minimum average search time of the corresponding optimal BST can be computed by $\mathrm{B}(1, 50, 1)$. In words, We compute the minimum average search time using the recurrence by specifying the range from 1 to 50 (spanning the entire BST) starting with depth 1.

Note

Observe how the recurrence structure in dynamic programming (DP) relates to the strong form of mathematical induction. In the case of strong induction, assuming $P_1, \cdots, P_k$ are all true, we need to show that P_{k+1} follows. In the case of DP, we compute P_{k+1} based on the results of all previous i.e. smaller cases.

The key to formulating such recurrence seems to be to come up with a notation that captures the collective result of what we try to compute, and then express a larger result in terms of the smaller ones.