Optimal Binary Search Tree

Yu-Tai Ching
Department of Computer Science
National Chiao Tung University

- A dictionary
- Static Case: We have all of the records, we build the binary search tree, and no insertion or deletion are allowed.
- Dynamic Case: Insertion and deletions are allowed.
- What if the probability of accessing a node is not uniform (static case).
- Figure 10.2 (a) at most 4 comparisons, (b) atmost 3 comparisons, the worst case.
- Average case, (a): $1 + 2 \times 2 + 1 \times 3 + 4 \times 1 = 12$, $12/5 = 2.4$.
- (b): $1 \times 1 + 2 \times 2 + 3 \times 2 = 11$, $11/5 = 2.2$.

- Figure 10.3 is obtained from Figure 10.2 by adding the external nodes.
- the square nodes: represent the failure node, an unsuccessful search.
- original nodes: the internal nodes, a successful search.
- the "external path length": sum over all external nodes of the lengths of the paths from the root to those nodes.
- the "internal path length": sum over all internal nodes of the lengths of the paths from the root to those nodes.
- 10.3 (a): $I = 0 + 1 + 1 + 2 + 3 = 7$, $E = 2 + 2 + 2 + 3 + 4 + 4 = 17$,
- 10.3 (b): $I = 0 + 1 + 1 + 2 + 2 = 6$, $E = 2 + 2 + 3 + 3 + 3 + 3 = 16$,

- $E = I + 2n$ (why?),
- Binary tree with maximum E also has maximum I.
- Tree is skew: $I = \sum i = n(n-1)/2$.
- Tree is balance
  $0 + 2 \cdot 1 + 4 \cdot 2 + 8 \cdot 3 + ... = \sum_{1 \le i \le n} \lfloor \lg i \rfloor = O(n \log n)$.

- If there are associated probabilities with nodes,
- $a_1, a_2, \ldots, a_n$, $a_1 < a_2 < \ldots < a_n$ element keys,
- a_i has probability p_i to be accessed.
- Total cost for tree, $\sum_{1 \leq i \leq n} p_i \cdot \mathit{level}(a_i)$, when only successful searches are made.
- if there are unsuccessful search, the search stops at an external node,
- there are $n + 1$ external nodes, each one has an associated probability q_i, $i = 0, \ldots, n$.
- They contribute $\sum_{0 \leq i \leq n} q_i \cdot (\mathit{level}(\mathit{failure\ node\ i}) - 1)$.
- $\sum p_i + \sum q_j = 1$.

- Total cost

$$\sum_{1 \le i \le n} p_i \cdot level(a_i) + \sum_{0 \le i \le n} q_i \cdot (level(failure\ node\ i) - 1.)$$

- An optimal binary search tree for $a_1, a_2, \ldots, a_n$ is a tree that minimize the total cost.
- Can we enumerate all tree?
  - Suppose that we have $a_1(5)$, $a_2(10)$, and $a_3(15)$. $p_i = q_j = 1/7$, Figure 10.4, costs are 15/7, 13/7, 15/7, 15/7, 15/7.
  - if $p_1 = 0.5$, $p_2 = 0.1$, $p_3 = 0.05$, $q_0 = 0.15$, $q_1 = 0.1$, $q_2 = 0.05$, $q_3 = 0.05$, cost are 2.65, 1.9, 1.5, 2.05, 1.6, (c) is optimal.
- Given n nodes, there are $O(\frac{4^n}{n^{\frac{3}{2}}})$ trees.

- Solve by using dynamic programming
- Given $a_1 < a_2 < a_3 \ldots, < a_n$, n keys,
- Let $T_{i,j}$ be the optimal binary tree for $a_{i+1}, \ldots, a_j$, $i < j$ and $c_{i,j}$ is the cost for $T_{i,j}$. $T_{i,i}$ is empty and $c_{i,i} = 0$.
- $r_{i,j}$ the root if $T_{i,j}$, $r_{i,i} = 0$.
- $w_{i,j}$, the weight of $T_{i,j}$, $w_{i,j} = q_i + \sum_{k=i+1}^{j}(q_k + p_k)$, $w_{i,i} = q_i$.
- $T_{0,n}$ is te tree we are looking for and $c_{0,n}$ is the cost.

- If $T_{i,j}$ is an optimal binary search tree for $a_{i+1}, \ldots, a_j$, $r_{i,j} = k$, $i < k \leq j$, the $T_{i,j}$ has two subtree L and R.
- L covers $a_{i+1}, \ldots, a_{k-1}$, and R covers $a_{k+1}, \ldots, a_j$,
- $c_{i,j} = p_k + cost(L) + cost(R) + weight(L) + weight(R)$,
- note that $cost(L)$ and $cost(R)$ must be optimal (why?), thus $cost(L) = c_{i,k-1}$ and $cost(R) = c_{k+1,j}$.
- $c_{i,j}$ can be rewritten as $c_{i,j} = w_{i,j} + c_{i,k-1} + c_{k,j}$.
- Since we don't know what is the root, so we have to try every one, and the optimal tree should be
- $c_{i,j} = \min_{i<l<j}\{w_{i,j} + c_{i,l-1} + c_{l,j}\} = w_{i,j} + \min_{i<l<j}\{c_{i,l-1} + c_{l,j}\}$.