Algorithms Design and Analysis [ETCS-301]

Dr. A K Yadav
Amity School of Engineering and Technology
(affiliated to GGSIPU, Delhi)
akyadav1@amity.edu
akyadav@akyadav.in
www.akyadav.in
+91 9911375598

September 5, 2019

Step 1: The structure of an OBST I

- Let T is a optimal tree having subtree $T^{'}$ with keys $k_i \dots k_j$ and dummy keys $d_{i-1} \dots d_j$ for some $1 \leq i \leq j \leq n$
- Then $T^{'}$ will also be a optimal subtree
- We can find the optimal cost of the tree by optimal cost of the subtree
- Take k_r as the root from keys k_i to k_j and dummy keys d_{i-1} to d_j
- Now there are two subtrees one having keys from k_i to k_{r-1} with dummy keys from d_{i-1} to d_{r-1}
- The second tree having keys from k_{r+1} to k_j with dummy keys from d_r to d_j

Step 1: The structure of an OBST II

- Now the total cost of the tree will be sum of cost of left subtree having keys from k_i to k_{r-1} with dummy keys from d_{i-1} to d_{r-1} plus cost of right subtree having keys from k_{r+1} to k_j with dummy keys from d_r to d_j plus cost of key k_r
- d_i is the dummy key between the keys k_i and k_{i+1}
- If we choose k_i as the root then there is no key (keys from k_i to k_{i-1}) on left subtree but only one dummy key d_{i-1}
- If we choose k_j as the root then there is no key (keys from k_{j+1} to k_j) on right subtree but only one dummy key d_j
- By taking all key from k_i to k_j as root one by one, we can find the optimal cost of the tree

Step 2: A recursive solution I

- Let a subproblem with keys k_i to k_j and dummy keys d_{i-1} to d_j where $i \geq 1$, $j \leq n$ and $j \geq i-1$
- Let $e[i,j]$ is the expected cost of searching an optimal binary search tree containing the keys k_i to k_j and dummy keys d_{i-1} to d_j
- Finally we have to find out $e[1,n]$
- Let $w[i,j]$ is the sum of the probability of all the keys $k_i \dots k_j$ and dummy keys $d_{i-1} \dots d_j$ and $w[1,n] = 1$

$$w[i,j] = \sum_{k=i}^{j} p_k + \sum_{k=i-1}^{j} q_k$$

- When $j = i-1$ then no key but only dummy key so $e[i,i-1] = q_{i-1}$ and $w[i,i-1] = q_{i-1}$

Step 2: A recursive solution II

- When $j > i - 1$, then take a key k_r as root such that left subtree having keys from k_i to k_{r-1} and right subtree having keys from k_{r+1} to k_j is optimal
- When a tree becomes subtree of a root node then hight of the each node of the tree increases by one.
- The expected search cost of this subtree increases by the sum of all the probabilities in the subtree i.e.

$$e[i, r-1] = e[i, r-1] + w[i, r-1]$$

$$e[r+1, j] = e[r+1, j] + w[r+1, j]$$

Step 2: A recursive solution III

- Taking k_r as the root $e[i,j]$ becomes

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

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

$$\because w[i,j] = w[i,r-1] + p_r + w[r+1,j]$$

- The recursive solution will be:

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

Step 3: Computing the expected search cost of OBST I

- Optimal cost of OBST is stored in $e[1 \dots n+1, 0 \dots n]$ that is size of upper triangular matrix is $(n+1) \times (n+1)$.
- $e[1,0]$ is used to store the value of d_0 and $e[n+1,n]$ is used to store the value of d_n
- $e[i,j]$ is used to store the value of optimal search cost of the keys from k_i to k_j for $1 \le i \le n+1, 0 \le j \le n,$ and $j \ge i-1$
- $root[i,j]$ is used to store the root of the tree having keys from k_i to k_j for $1 \le i \le j \le n$
- $e[i,i-1] = w[i,i-1] = q_{i-1}$ for $1 \le i \le n+1$
- $w[i,j] = w[i,j-1] + p_j + q_j$ for $1 \le i \le n+1, j > i-1$
- $e[i,j] = \min\limits_{i \le r \le j} \{e[i,r-1] + e[r+1,j] + w[i,j]\}$ for $j > i-1$

Bottom-up Approach:

OPTIMAL-BST(p,q,n)

1. Let $e[1 \ldots n+1, 0 \ldots n]$, $w[1 \ldots n+1, 0 \ldots n]$ and $root[1 \ldots n, 1 \ldots n]$
2. for $i = 1$ to $n+1$ //when no keys, will be only one dummy key
3. $\quad w[i, i-1] = q_{i-1}$
4. $\quad e[i, i-1] = q_{i-1}$
5. for $l = 1$ to n // l number of keys in the subtree
6. $\quad$ for $i = 1$ to $n - l + 1$
7. $\quad\quad j = i + l - 1$
8. $\quad\quad e[i, j] = \infty$
9. $\quad\quad w[i, j] = w[i, j-1] + p_j + q_j$
10. $\quad\quad$ for $r = i$ to j

Step 3: Computing the expected search cost of OBST III

$q = e[i, r-1] + e[r+1, j] + w[i, j]$

if $q < e[i, j]$

  $e[i, j] = q$

  $root[i, j] = r$

return $e, root$

Top-down Approach:

MEMOIZED-OPTIMAL-BST(p,q,n)

Let $e[1 \ldots n+1, 0 \ldots n]$, $w[1 \ldots n+1, 0 \ldots n]$ and $root[1 \ldots n, 1 \ldots n]$
for $i = 1$ to $n+1$
    for $j = i-1$ to n
        $e[i,j] = \infty$
        if $j = i-1$
            $w[i,j] = q_{i-1}$
        else
            $w[i,j] = w[i,j-1] + p_j + q_j$
return LOOKUP-OBST(e,w,root,p,q,1,n)

Step 3: Computing the expected search cost of OBST V

```
LOOKUP-OBST(e,w,root,p,q,i,j)
if e[i,j] < ∞
    return e[i,j]
if j = i − 1
    e[i,j] = q_{i−1}
else for r = i to j
    q = LOOKUP-OBST(e,w,root,p,q,i,r-1)
        +LOOKUP-OBST(e,w,root,p,q,r+1,j)+w[i,j]
    if q < e[i,j]
        e[i,j] = q
        root[i,j] = r
return e[i,j]
```

Step 4: Constructing an OBST I

- Table $e[i,j]$ gives the cost of OBST for the keys k_i to k_j with dummy keys d_{i-1} to d_j
- Table $root[i,j]$ store the root node $k_r = k_{root[i,j]}$
- At $k_r = k_{root[i,j]}$ there will be two subtree: one left subtree with keys k_i to $k_{root[i,j]-1}$ and second right subtree with $k_{root[i,j]+1}$ to k_j
- First call will be:
- if $n \geq 1$ //atleast one key
- $r = root[1,n]$
- print k_r is the root of the OBST
- PRINT-OBST($root, i, r-1, r, "left"$)
- PRINT-OBST($root, r+1, j, r, "right"$)
- else
- print d_0 is the root of the OBST

Step 4: Constructing an OBST II

PRINT-OBST(root,i,j,r,child)

1. if $i \leq j$
2. $\quad c = root[i, j]$
3. $\quad$ print k_c is child of k_r
4. $\quad$ PRINT-OBST$(root, i, r-1, c, "left")$
5. $\quad$ PRINT-OBST$(root, r+1, j, c, "right")$

Complexity of the OBST is $O(n^3)$

Thank you

Please send your feedback or any queries to akyadav1@amity.edu, akyadav@akyadav.in or contact me on +91 9911375598