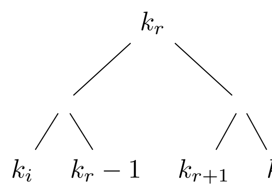
Optimal Binary Search Trees



To find an optimal BST:

1. Examine all candidate roots k_r, for $i \leq r \leq j$
2. Determine all optimal BSTs containing $k_i, \ldots, k_{r-1}$ and containing $k_{r+1}, \ldots, k_j$

If k_r is the root of an optimal BST for $k_i, \ldots, k_j$:

$$\begin{aligned} e[i,j] &= p_r + (e[i,r-1] + w(i,r-1)) + (e[r+1,j] + w(r+1,j)) \\ &= e[i,r-1] \text{ (Optimal e[search cost] considering } k_i, \ldots, k_{r-1})+ \\ &\quad e[r+1,j] \text{ (Optimal e[search cost] considering } k_{r+1}, \ldots, k_j)+ \\ &\quad w(i,j) \text{ (derived from } w(i,r-1) + p_r + w(r+1,j)) \end{aligned}$$

Considering different k_r and chose the best one.

$$e[i,j] = \begin{cases} 0 & \text{if } j = i-1 \\ \min\limits_{i \leq r \leq j} \{e[i,r-1] + e[r+1,j] + w(i,j)\} & \text{if } i \leq j \end{cases}$$

For each subproblem (i,j), store: e[search cost] in a table $e[1 \ldots n+1, 0 \ldots n]$ Will only use entries $e[i,j]$, where $j \geq i-1$.

$$root[i,j] = \text{ root of subtree with keys} k_i, \ldots, k_j, \text{ for } 1 \leq i \leq j \leq n$$

```
Optimal-BST(p, q, n)
for i <- 1 to n + 1
    do e[i, i-1] <- 0
    w[i, i-1] <- 0

for l <- 1 to n
    do for i <- 1 to n - l + 1
        do j <- i + l - 1
            e[i, j] <- infinity
            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
```

Example:

k_1	k_2	k_3	k_4
0.1	0.2	0.4	0.3

$$0.4 = min(e[1,0] + e[2,2] + 0.3, e[1,1] + e[3,2] + 0.3)$$

$$\begin{aligned} 1.1 &= min(e[1,0] + e[2,3] + .7, e[1,1] + e[3,3] + .7, e[1,2] + e[4,3] + .7) \\ &= min(0.8 + 0.7, 0.1 + 0.4 + 0.7, 0.4 + 0.7) \end{aligned}$$