Using a "Binary Search Tree" to solve the Dictionary Problem (Search, Insert, Delete)

- each internal node contains a key from an ordered set

  key(node) $>$ keys in left subtree
  $<$ keys in right subtree

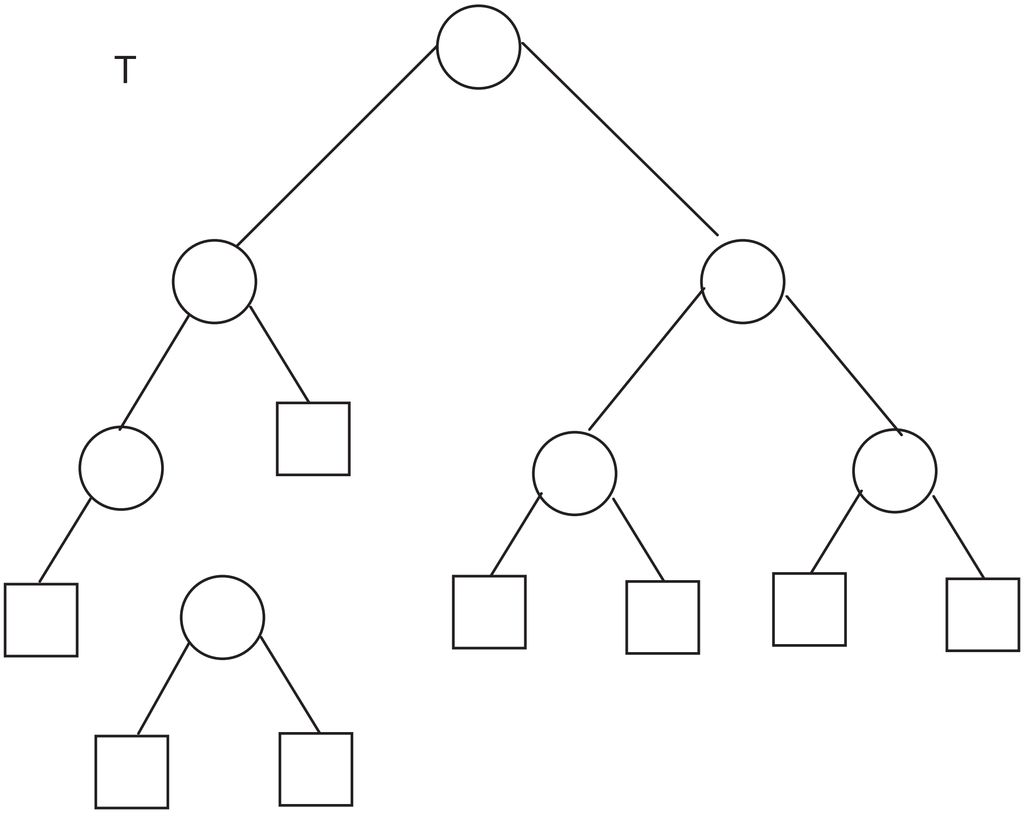
- external nodes correspond to unsuceesful searches

- We use $I(T)$ to denote *internal path length* of tree T (sum of all path lengths from an internal node to root).

- We use $E(T)$ to denote *extermal path length* of tree T. (sum of all path lengths from an external node to root) External nodes correspond to unsuccessful searches.

Properties of Binary Trees:

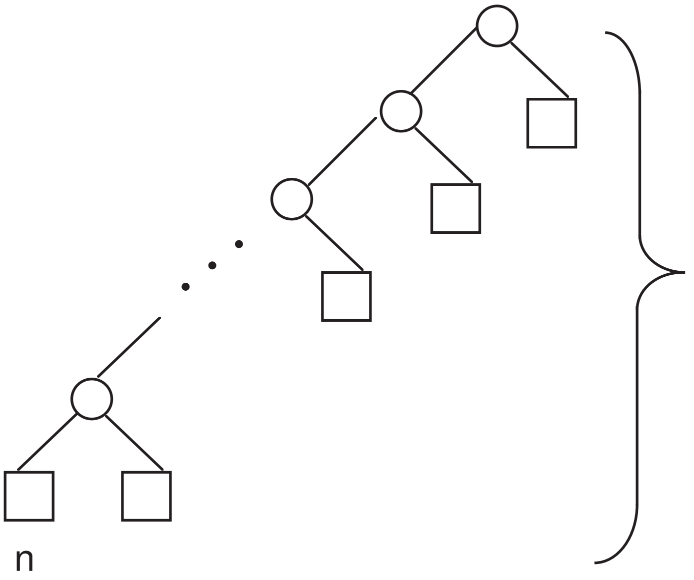
- root $\sim$ level 0
- others $\sim$ 1+level (parent)
- height(T) = maximum level



Internal path length $I(T) = 0 + 1 + 1 + 2 + 2 + 2 + 3 = 11$

External path length $E(T) = 3 + 4 + 4 + 2 + 3 + 3 + 3 + 3 = 25$

Maximum $I(T)\&E(T)$ for n internal node tree



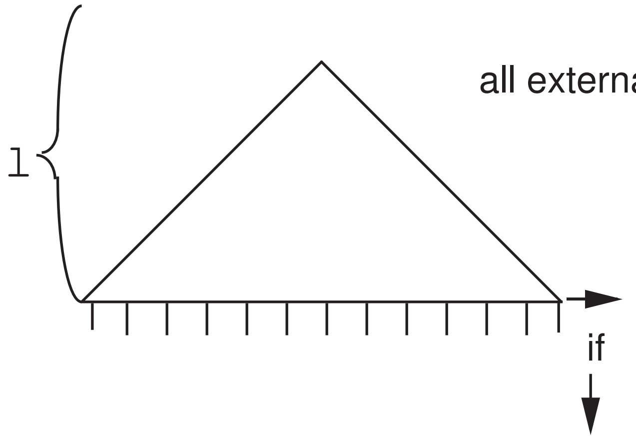
$$I(T) = \sum_{0}^{n-1} i = \frac{1}{2} n(n-1)$$

$$E(T) = I(T) + n + n$$

each node increase by 1

last node

Tree that minimizes $I(T)$ and $E(T)$:



all external nodes on level l or l+1

$n=2^k \sim k = \log n$

if

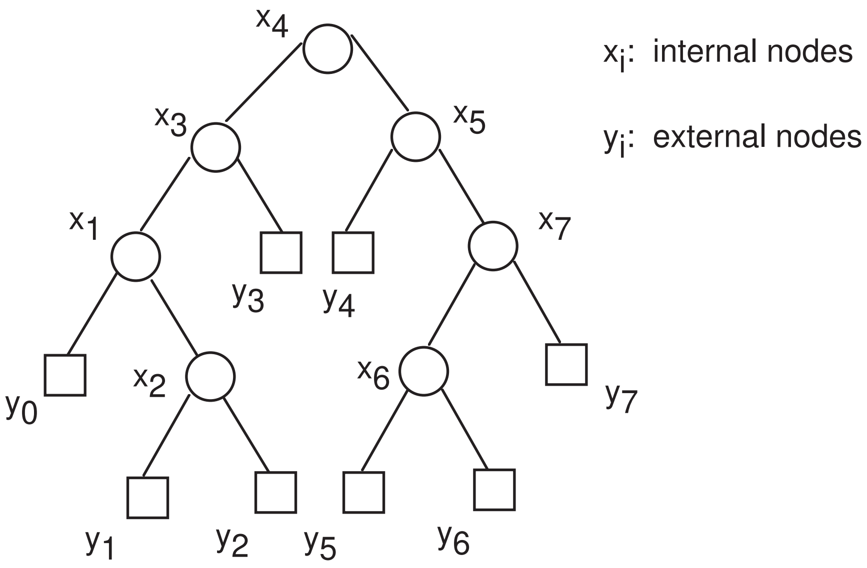
(Proof by contradiction)

$$E(T) = (n+1)k$$

$$I(T) = \sum_{0}^{k-1} 2^i i \approx n(k-2)$$

Binary Search Tree continues ... Let

$\beta_i = Prob$ (access key x_i)
$\alpha_i = Prob$ (access x, key $x_i < x <$ key x_{i+1})
$\alpha_0 + \sum_{i=1}^{n}(\beta_i + \alpha_i) = 1$



Average Search Cost is

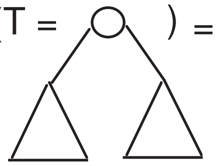
$$W(T) = \sum_{i=1}^{n} \beta_i(1 + \text{level}(x_i)) + \sum_{i=0}^{n} \alpha_i \text{level}(y_i)$$

• Big Question: How do we find the optimal tree minimizing $W(T)$, given α_i, β_i.

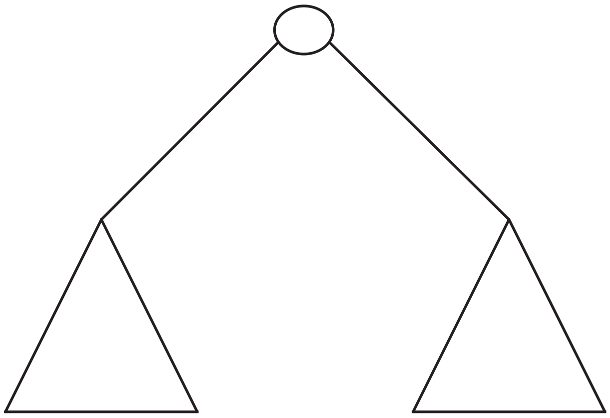
- We know there are a <u>lot</u> (exponentially many) of trees on n nodes. So we can't search one by one.

- Note:

$W(\square) = 0$

$W(T =$  $) = W(T_l) + W(T_r) + \Sigma\alpha_i + \Sigma\beta_i$

- Note: If T has minimum $W(T)$, then so are T_ℓ, T_r



- Solution: Dynamic Programming. To find optimal tree, find best root and optimal subtree.

Dynamic Programming: Optimal solution to a problem consists of optimal solutions to its subproblems.

$C_{ij} \equiv$ cost of optimal tree over$\alpha_i \beta_{i+1} \cdots \beta_j \alpha_j$

C_{ij} is the cost of $x_{i+1} \cdots x_j$ including y_k gaps

We want to compute $C_{0,n}$, and corresponding tree. Key Observation:

$$C_{ij} = \min_{i < k \leq j} (C_{i,k-1} + C_{k,j}) + \sum_{t=i}^{j} \alpha_t + \underbrace{\sum_{t=i+1}^{j} \beta_t}_{\text{probability of tree nodes}}$$

over all possible roots

cost of subtrees

For portion $y_i x_{i+1} \cdots x_j y_j$, let R_{ij} be the root of the optimal tree.

w_{ij} be $\alpha_i + \beta_{i+1} + \cdots + \beta_j + \alpha_j$, weight of this part $(x_{i+1} \ldots x_j)$.

We will construct substrees in increasing order by size $(j - i)$.

for $i :=0$ to n do
    $R_{ii} = 0; W_{ii} := \alpha_i; C_{ii} := 0;$
for $\ell := 1$ to n do
    begin {constr. tree of ℓ consecutive internal nodes }
    for $i := 0$ to $n - \ell$ do
        begin {constr ℓ node opt. tree after node x_i }
        $j := i + \ell;$
        $R_{ij} := i + 1;$
        for $k := i + 2$ to j do
            if $C_{ik-1} + C_{k,j} < C_{i,(R_{ij})-1} + C_{R_{ij},j}$
            then $R_{ij} := k;$
        $W_{ij} := W_{ij-1} + \beta_j + \alpha_j;$
        $C_{ij} := C_{i,(R_{ij})-1} + C_{R_{ij},j} + W_{ij}$
    end;

Analysis:

Main loop:

for $\ell := 1$ to n
  for $i := 0$ to $n - \ell$
    for $k := i + 2$ to j

costs

$$
\begin{aligned}
\sum_{\ell=1}^{n} \sum_{i=0}^{n-\ell} \sum_{k=i+2}^{i+\ell} 1 &= \sum_{\ell=1}^{n} \sum_{0}^{n-\ell} (\ell - 1) \\
&= \sum_{\ell=1}^{n} (n - \ell + 1)(\ell - 1) \\
&\approx O(n^3)
\end{aligned}
$$

With care, time complexity can be reduced to $O(n^2)$. (Space is also $O(n^2)$.)