Generalization
What if n is large?

2 Trees

- **Definition**: As 2-tree is a tree in which every vertex except the leaves has exactly two children.

- **Lemma 6.1**: The number of vertices on each level of a 2-tree is at most twice the number on the level immediately above.

- **Lemma 6.2**: In a 2-tree, the number of vertices on level \( t \) is at most \( 2^t \) for \( t \geq 0 \).
Analysis of Binary 1 Search

- Both successful and unsuccessful search terminates at leaves.
- Number of leaves $2n$.
- They are in the last two levels.
- The height is also the smallest integer $t$ for which:
  - $2^t \geq 2n$
  - $2^{t-1} \geq n$
  - $t-1 = \log n$
  - $t = \log n + 1$

Analysis of Binary 2 Search

Unsuccessful Search:
- The tree is full at top
- All F are leaf nodes.
- Leafs are in last two levels.
- Number of leafs = $n+1$
  - $h = \log (n+1)$
- 2 comparison per node
- number of comparisons
  - $2 \times \log (n+1)$
Theorem 6.3

- **Theorem 6.3** Denote the external path length of a 2-tree by \( E \), the internal path length by \( I \), and let \( q \) be the number of vertices that are not leaves. Then

\[
E = I + 2q
\]

Proof of Theorem 6.3

- For a tree with only root: \( I=0, E=0, q=0 \)
- Let \( v \) be an immediate parent to any two leaf nodes:
  - Let's now delete these two children:
    - New \( E = E - 2(k+1) + k \)
    - New \( I = I - k \)
    - New \( q = q - 1 \)
  - Does the relationship still holds?
    - \( E - k - 2 = I + 2q - k - 2 \)
  - With reorganization:
    - \( E - 2(k+1) + k = (I-k) + 2(q-1) \)
    - New \( E = new I + 2 * new q \)
  - That’s proof by induction!
Analysis of Binary 2 Search

Successful Search:

- Number of leaves (n+1)
  - Therefore \( E = (n+1) \log (n+1) \)
- Number of internal nodes n.
- From Theorem 6.3 \( I = E - 2q \):
  - \( I = (n+1) \log (n+1) - 2n \)
- Average internal path length: \( I/n \)
- Average number of nodes in internal paths:
  - \( I/n + 1 \)
- Each node except the last one has two comparisons:
  - number of comparisons:
    - \( 2 \times (I/n + 1) - 1 = 2 \times \lceil (n+1)/n \log (n+1) - 2 + 1 \rceil - 1 \)
    - \( 2 \times (n+1)/n \log (n+1) - 3 \approx 2 \log n - 3 \)

Comparison of Methods

<table>
<thead>
<tr>
<th></th>
<th>Successful search</th>
<th>Unsuccessful search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary1Search</td>
<td>( \lg n + 1 )</td>
<td>( \lg n + 1 )</td>
</tr>
<tr>
<td>Binary2Search</td>
<td>( 2 \lg n - 3 )</td>
<td>( 2 \lg n )</td>
</tr>
</tbody>
</table>
Graphs

Logarithmic Plot
Can there ever be any better algorithm?

Lower Bound on Path Length

**Lemma 6.5** Let $T$ be a 2-tree with $k$ leaves. Then the height $h$ of $T$ satisfies $h \geq \lceil \lg k \rceil$ and the external path length $E(T)$ satisfies $E(T) \geq k \lg k$. The minimum values for $h$ and $E(T)$ occur when all the leaves of $T$ are on the same level or on two adjacent levels.
Proof

Lemma 6.5 Let $T$ be a 2-tree with $k$ leaves. Then the height $h$ of $T$ satisfies $h \geq \lceil \log k \rceil$ and the external path length $E(T)$ satisfies $E(T) \geq k \log k$. The minimum values for $h$ and $E(T)$ occur when all the leaves of $T$ are on the same level or on two adjacent levels.

\begin{itemize}
  \item $E(T') = E(T) - 2r + (r - 1) - s + 2(s - 1) = E(T) - r + s + 1 < E(T)$
  \item Even after all the compaction when all leaves are just in two adjacent levels.
  \item Lemma 2 will still hold,
    \begin{itemize}
      \item If all the k leaves are in last level h then $k \leq 2^h$.
      \item If some of them in upper level for every hole there is one less leaf.
      \item So always $k \leq 2^h$, or $h \geq \lceil \log k \rceil$.
    \end{itemize}
  \item From further analysis it can be shown that $E(T) \geq k (\log k + 1 + e - 2e)$ for all $0 \leq e < 1$.
  \item $(1 + e - 2e)$ is between 0 and .0861.
  \item Thus the minimum external path length is at least $\lceil k \log k \rceil$.
\end{itemize}

Lowest Bound on Search

Theorem 6.6 Suppose that an algorithm uses comparisons of keys to search for a target in a list. If there are $k$ possible outcomes, then the algorithm must make at least $\lceil \log k \rceil$ comparisons of keys in its worst case and at least $\log k$ in its average case.

\begin{itemize}
  \item Irrespective of the search method, any comparison based search will have $2n+1$ outcomes.
  \item Each comparison will result in a two way fork. Thus, the comparison tree will always be a 2-tree.
  \item From Lemma 6.5: The external path length
  \item $E(T) \geq (2n+1) \log (2n+1)$
  \item The best possible average worst-case:
    \begin{itemize}
      \item $\geq \lceil \log(2n+1) \rceil \geq \lceil \log 2n \rceil = \lceil \log n \rceil + 1$
    \end{itemize}
\end{itemize}
Other ways of Searching

**Corollary 6.7** BinarySearch is optimal in the class of all algorithms that search an ordered list by making comparisons of keys. In both the average and worst cases, BinarySearch achieves the optimal bound.

- **Other ways:**
  - Keys are all integers 1-n.
- **Interpolation Search:**
  - If keys are uniformly distributed.
  - \( \log \log n \)
  - For \( n = 1,000,000 \) Binary1 will require 21 comparisons.
  - Interpolation search will require about 4.32 comparison.

Next Class:

- Classification of algorithms’ running time.
- Asymptotics.