Lecture 17
Trees, Logs and Time Analysis

Instructor: George Wolberg
Department of Computer Science
City College of New York
Topics

- Big-O Notation
- Worse Case Times for Tree Operations
- Time Analysis for BSTs
- Time Analysis for Heaps
- Logarithms and Logarithmic Algorithms
## Big-O Notation

- The order of an algorithm generally is more important than the speed of the processor.

<table>
<thead>
<tr>
<th>Input size: ( n )</th>
<th>( O(\log n) )</th>
<th>( O(n) )</th>
<th>( O(n^2) )</th>
</tr>
</thead>
<tbody>
<tr>
<td># of stairs: ( n )</td>
<td>( \lfloor \log_{10} n \rfloor + 1 )</td>
<td>( 3n )</td>
<td>( n^2 + 2n )</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>30</td>
<td>120</td>
</tr>
<tr>
<td>100</td>
<td>3</td>
<td>300</td>
<td>10,200</td>
</tr>
<tr>
<td>1000</td>
<td>4</td>
<td>3000</td>
<td>1,002,000</td>
</tr>
</tbody>
</table>
Worst-Case Times for Tree Operations

- The worst-case time complexity for the following are all $O(d)$, where $d =$ the depth of the tree:
  - Adding an entry in a BST, a heap or a B-tree;
  - Deleting an entry from a BST, a heap or a B-tree;
  - Searching for a specified entry in a BST or a B-tree.

- This seems to be the end of our Big-O story...but
What’s d, then?

- Time Analyses for these operations are more useful if they are given in term of the number of entries \( n \) instead of the tree’s depth \( d \).

- Question:
  - What is the maximum depth for a tree with \( n \) entries?
Time Analysis for BSTs

- Maximum depth of a BST with n entries: n-1

- An Example:
  Insert 1, 2, 3, 4, 5 in that order into a bag using a BST
Worst-Case Times for BSTs

- Adding, deleting or searching for an entry in a BST with n entries is O(d), where d is the depth of the BST.
- Since d is no more than n-1, the operations in the worst case is (n-1).
- Conclusion: the worst case time for the add, delete or search operation of a BST is O(n).
Time Analysis for Heaps

- A heap is a complete tree
- The minimum number of nodes needed for a heap to reach depth $d$ is $2^d$:
  \[ = (1 + 2 + 4 + \ldots + 2^{d-1}) + 1 \]
  - The extra one at the end is required since there must be at least one entry in level $d$
- Question: how to add up the formula?
Time Analysis for Heaps

- A heap is a complete tree
- The minimum number of nodes needed for a heap to reach depth $d$ is $2^d$:
- The number of nodes $n \geq 2^d$
- Use base 2 logarithms on both side
  - $\log_2 n \geq \log_2 2^d = d$
  - Conclusion: $d \leq \log_2 n$
Worst-Case Times for Heap Operations

- Adding or deleting an entry in a heap with \( n \) entries is \( O(d) \), where \( d \) is the depth of the tree
- Because \( d \) is no more than \( \log_2 n \), we conclude that the operations are \( O(\log n) \)

- Why we can omit the subscript \( 2 \)?
Logarithms (log)

- **Base 10:** the number of digits in \( n \) is \( \lfloor \log_{10} n \rfloor + 1 \)
  - \( 10^0 = 1 \), so that \( \log_{10} 1 = 0 \)
  - \( 10^1 = 10 \), so that \( \log_{10} 10 = 1 \)
  - \( 10^{1.5} = 32^+ \), so that \( \log_{10} 32 = 1.5 \)
  - \( 10^3 = 1000 \), so that \( \log_{10} 1000 = 3 \)

- **Base 2:**
  - \( 2^0 = 1 \), so that \( \log_2 1 = 0 \)
  - \( 2^1 = 2 \), so that \( \log_2 2 = 1 \)
  - \( 2^3 = 8 \), so that \( \log_2 8 = 3 \)
  - \( 2^5 = 32 \), so that \( \log_2 32 = 5 \)
  - \( 2^{10} = 1024 \), so that \( \log_2 1024 = 10 \)
Logarithms (log)

- Base 10: the number of digits in $n$ is $\lceil \log_{10} n \rceil + 1$
  - $10^{1.5} = 32^+$, so that $\log_{10} 32 = 1.5$
  - $10^3 = 1000$, so that $\log_{10} 1000 = 3$

- Base 2:
  - $2^3 = 8$, so that $\log_2 8 = 3$
  - $2^5 = 32$, so that $\log_2 32 = 5$

- Relation: For any two bases, $a$ and $b$, and a positive number $n$, we have
  - $\log_b n = (\log_b a) \log_a n = \log_b a^{(\log_a n)}$
  - $\log_2 n = (\log_2 10) \log_{10} n = (5/1.5) \log_{10} n = 3.3 \log_{10} n$
Logarithmic Algorithms

- Logarithmic algorithms are those with worst-case time $O(\log n)$, such as adding to and deleting from a heap.
- For a logarithm algorithm, doubling the input size ($n$) will make the time increase by a fixed number of new operations.
- Comparison of linear and logarithmic algorithms:
  - $n = m = 1$ hour $\rightarrow \log_2 m \approx 6$ minutes
  - $n = 2m = 2$ hour $\rightarrow \log_2 m + 1 \approx 7$ minutes
  - $n = 8m = 1$ work day $\rightarrow \log_2 m + 3 \approx 9$ minutes
  - $n = 24m = 1$ day & night $\rightarrow \log_2 m + 4.5 \approx 10.5$ minutes
Summary

- **Big-O Notation**: Order of an algorithm versus input size (n)
- **Worse Case Times for Tree Operations**: $O(d)$, $d = \text{depth of the tree}$
- **Time Analysis for BSTs**: worst case: $O(n)$
- **Time Analysis for Heaps**: worst case $O(\log n)$
- **Logarithms and Logarithmic Algorithms**: doubling the input only makes time increase a fixed number