Trees, AVL Trees & Heaps

Lecture 03
21 January 2010

Dr. Joan A. Smith/Emory University
Pop Quiz

• Define a tree (2 pts)
A Risky Wager?

• Let’s say I offer you $1000 if you can guess the number I have written on this piece of paper
  – The number is between 0 and 1000, inclusive
  – You are allowed up to 20 guesses
  – If you lose, you owe me $1000
  – If you win, I pay you $1000

• Should you take this wager?
• What’s the minimum number of guesses you would need?
• What’s the maximum number of (naïve) guesses?
• What’s the max number of systematic guesses?
Trees: Creation & Maintenance

• Creation expects organization – some kind of order
  – Divide & Conquer method for find, insert, delete

• Initialization options can vary with dataset

• Array is reasonable if:
  – Data set is a static size
  – Purpose is search vs insertion/deletion
  – Still requires some sorting process (quicksort, insertion sort, etc.)

• Dynamic creation
  – Data set size is variable and/or unknown
  – Frequent inserts/deletes
  – Initial creation involves multiple re-orderings

• Consider a few sorting algorithms…
### Table 4-8. Performance (in seconds) on ordered random 26-letter permutations of the alphabet (continued)

<table>
<thead>
<tr>
<th>n</th>
<th>Hash Sort 17,576 buckets</th>
<th>Quicksort median-of-three</th>
<th>Heap Sort</th>
<th>Median Sort</th>
<th>Quicksort BFPRRT minSize=4</th>
</tr>
</thead>
<tbody>
<tr>
<td>16,384</td>
<td>0.0037</td>
<td>0.0036</td>
<td>0.0062</td>
<td>0.0094</td>
<td>0.0161</td>
</tr>
<tr>
<td>32,768</td>
<td>0.0074</td>
<td>0.0082</td>
<td>0.0157</td>
<td>0.0216</td>
<td>0.0381</td>
</tr>
<tr>
<td>65,536</td>
<td>0.0161</td>
<td>0.0184</td>
<td>0.0369</td>
<td>0.049</td>
<td>0.0873</td>
</tr>
<tr>
<td>131,072</td>
<td>0.0348</td>
<td>0.0406</td>
<td>0.0809</td>
<td>0.1105</td>
<td>0.2001</td>
</tr>
</tbody>
</table>

### Table 4-9. Performance (in seconds) on killer median data

<table>
<thead>
<tr>
<th>n</th>
<th>Hash Sort 17,576 buckets</th>
<th>Heap Sort</th>
<th>Median Sort</th>
<th>Quicksort BFPRRT minSize=4</th>
<th>Quicksort median-of-three</th>
</tr>
</thead>
<tbody>
<tr>
<td>4,096</td>
<td>0.0011</td>
<td>0.0012</td>
<td>0.0021</td>
<td>0.0039</td>
<td>0.0473</td>
</tr>
<tr>
<td>8,192</td>
<td>0.0019</td>
<td>0.0028</td>
<td>0.0045</td>
<td>0.0087</td>
<td>0.1993</td>
</tr>
<tr>
<td>16,384</td>
<td>0.0038</td>
<td>0.0066</td>
<td>0.0101</td>
<td>0.0194</td>
<td>0.8542</td>
</tr>
<tr>
<td>32,768</td>
<td>0.0077</td>
<td>0.0179</td>
<td>0.024</td>
<td>0.0472</td>
<td>4.083</td>
</tr>
<tr>
<td>65,536</td>
<td>0.0171</td>
<td>0.0439</td>
<td>0.056</td>
<td>0.1127</td>
<td>17.1604</td>
</tr>
<tr>
<td>131,072</td>
<td>0.0338</td>
<td>0.1004</td>
<td>0.1292</td>
<td>0.2646</td>
<td>77.4519</td>
</tr>
</tbody>
</table>

* Because the performance of QUICKSORT median-of-three degrades so quickly, only 10 trials were executed; the table shows the average of eight runs once the best and worst performers were discarded.
Median Sort Method

- From *Algorithms In A Nutshell*

---

**Figure 4-8. Median Sort fact sheet**

- **Median Sort**
  - Best: $O(n \log n)$
  - Average: $O(n \log n)$
  - Worst: $O(n^2)$

---

**Figure 4-9. Median Sort in action on small array**
Quicksort Method

• From *Algorithms In A Nutshell*

The random pivot selection is what lets Quicksort outperform other sorting techniques *on average*.

![Quicksort Fact Sheet](image-url)
Heap Sort Method

Figure 4.14. Heap Sort fact sheet
Figure 4-15. (a) Sample heap of 16 unique elements; (b) labels of these elements; (c) heap stored in an array.
Balanced vs Unbalanced

• Why balance a tree?

• What happens when a tree is unbalanced?

• What about the work to maintain balance?
AVL Trees

- AVL Trees are a kind of BST
- BST:
  - The binary search tree $T$ is a decision tree, where the question asked at an internal node $v$ is whether the search key $k$ is less than, equal to, or greater than the key stored at $v$.
  - Pseudocode:
    
    ```
    Algorithm TreeSearch(k, v):
        Input: A search key $k$ and a node $v$ of a binary search tree $T$.
        Output: A node $w$ of the subtree $T(v)$ of $T$ rooted at $v$, such that either $w$ is an internal node storing key $k$ or $w$ is the external node encountered in the inorder traversal of $T(v)$ after all the internal nodes with keys smaller than $k$ and before all the internal nodes with keys greater than $k$.
        if $v$ is an external node then
            return $v$
        if $k = key(v)$ then
            return $v$
        else if $k < key(v)$ then
            return TreeSearch($k$, $T$.leftChild($v$))
        else
            $k > key(v)$
            return TreeSearch($k$, $T$.rightChild($v$))
    ```

21 January 2010
AVL Trees

• An AVL Tree is a binary search tree such that for every internal node $v$ of $T$, the heights of the children of $v$ can differ by at most 1.
AVL Trees

• Cf. pages 13-26 of AVL Trees