COP 4516: Competitive Programming and Problem Solving

Giri Narasimhan & Kip Irvine
Phone: x3748 & x1528
{giri, irvinek}@cs.fiu.edu
Problems to think about!

- What is the least number of comparisons you need to sort a list of 3 elements? 4 elements? 5 elements?
- How to arrange a tennis tournament in order to find the tournament champion with the least number of matches? How many tennis matches are needed? How to arrange a tennis tournament in order to find the runner up to the champion with the least number of matches?
- How to randomize the order of a list?
Sorting Algorithms

- SelectionSort
- InsertionSort
- BubbleSort
- ShakerSort
- MergeSort
- HeapSort
- QuickSort
- Bucket & Radix Sort
- Counting Sort
Data Structure Evolution

• **Standard operations on data structures**
  - Search
  - Insert
  - Delete

• Linear Lists
  - Implementation: **Arrays** *(Unsorted and Sorted)*

• **Dynamic** Linear Lists
  - Implementation: **Linked Lists**

• **Dynamic** Trees
  - Implementation: **Binary Search Trees**
# Data Structures Comparison

<table>
<thead>
<tr>
<th>Data Structure \ Operation</th>
<th>Search</th>
<th>Insert</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unsorted Array</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sorted Array</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unsorted Linked List</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sorted Linked List</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Binary Search Trees</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Balanced Binary Search Trees</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

9/8/11 COP 4516 5
BST: Search

\textbf{TreeSearch}(node \, x, \, key \, k)

\begin{itemize}
  \item Search for key \, k \, in subtree rooted at node \, x
  \item 1 \quad \textbf{if} \ ((x = NIL) \, \text{or} \, (k = key[x]))
  \item 2 \quad \text{then} \, \text{return} \, x
  \item 3 \quad \textbf{if} \ (k < key[x])
  \item 4 \quad \text{then} \, \text{return} \, \textbf{TreeSearch}(left[x], \, k)
  \item 5 \quad \text{else} \, \text{return} \, \textbf{TreeSearch}(right[x], \, k)
\end{itemize}

Time Complexity: O(h)

\textbf{Not O}(\log n) \quad \text{— Why?}

h = \text{height of binary search tree}
BST: Insert

TREEINSERT(tree $T$, node $z$)

$\triangleright$ Insert node $z$ in tree $T$

1. $y \leftarrow \text{NIL}$
2. $x \leftarrow \text{root}[T]$
3. while ($x \neq \text{NIL}$)
   4. do $y \leftarrow x$
   5. if ($\text{key}[z] < \text{key}[x]$)
      6. then $x \leftarrow \text{left}[x]$
      7. else $x \leftarrow \text{right}[x]$
   8. $p[z] \leftarrow y$
9. if ($y = \text{NIL}$)
   10. then $\text{root}[T] \leftarrow z$
   11. else if ($\text{key}[z] < \text{key}[y]$)
      12. then $\text{left}[y] \leftarrow z$
      13. else $\text{right}[y] \leftarrow z$

Time Complexity: $O(h)$

$h = \text{height of binary search tree}$
BST: Delete

Time Complexity: $O(h)$

$h = \text{height of binary search tree}$

$\text{TreeDelete}(tree \ T, \ node \ z)$

$\triangleright \text{Delete node } z \text{ from tree } T$

1. if ($(left[z] = \text{NIL}) \text{ or } (right[z] = \text{NIL}))$
2. then $y \leftarrow z$
3. else $y \leftarrow \text{Tree-Successor}(z)$
4. if $(left[y] \neq \text{NIL})$
5. then $x \leftarrow left[y]$
6. else $x \leftarrow right[y]$
7. if $(x \neq \text{NIL})$
8. then $p[x] \leftarrow p[y]$
9. if $(p[y] = \text{NIL})$
10. then $\text{root}[T] \leftarrow x$
11. else if $(y = \text{left}[p[y]])$
12. then $\text{left}[p[y]] \leftarrow x$
13. else $\text{right}[p[y]] \leftarrow x$
14. if $(y \neq z)$
15. then $\text{key}[z] \leftarrow \text{key}[y]$
16. copy $y$'s satellite data into $z$
17. return $y$
## Data Structures Comparison

<table>
<thead>
<tr>
<th>Data Structure \ Operation</th>
<th>Search</th>
<th>Insert</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unsorted Array</td>
<td>$O(n)$</td>
<td>$O(1)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Sorted Array</td>
<td>$O(\log n)$</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Unsorted Linked List</td>
<td>$O(n)$</td>
<td>$O(1)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Sorted Linked List</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Binary Search Trees</td>
<td>$O(h)$</td>
<td>$O(h)$</td>
<td>$O(h)$</td>
</tr>
<tr>
<td>Balanced Binary Search Trees</td>
<td>$O(\log n)$</td>
<td>$O(\log n)$</td>
<td>$O(\log n)$</td>
</tr>
</tbody>
</table>
Animations

- BST:  
  http://babbage.clarku.edu/~achou/cs160/examples/bst_animation/BST-Example.html
- Rotations:  
  http://babbage.clarku.edu/~achou/cs160/examples/bst_animation/index2.html
- RB-Trees:  
Example

- \([0,6], [1,4], [2,13], [3,5], [3,8], [5,7], [5,9], [6,10], [8,11], [8,12], [12,14]\)

- **Simple Greedy Selection**
  - Sort by start time and pick in “greedy” fashion
  - Does not work. WHY?
    - \([0,6], [6,10]\) is the solution you will end up with.

- **Other greedy strategies**
  - Sort by length of interval
  - Does not work. WHY?
Example

- [0,6], [1,4], [2,13], [3,5], [3,8], [5,7], [5,9], [6,10], [8,11], [8,12], [12,14]
- [1,4], [3,5], [0,6], [5,7], [3,8], [5,9], [6,10], [8,11], [8,12], [2,13], [12,14] -- Sorted by finish times
- [1,4], [3,5], [0,6], [5,7], [3,8], [5,9], [6,10], [8,11], [8,12], [2,13], [12,14]
- [1,4], [3,5], [0,6], [5,7], [3,8], [5,9], [6,10], [8,11], [8,12], [2,13], [12,14]
Greedy Algorithms

• Given a set of activities \((s_i, f_i)\), we want to schedule the maximum number of non-overlapping activities.

• **GREEDY-ACTIVITY-SELECTOR** \((s, f)\)
  1. \(n = \text{length}[s]\)
  2. \(S = \{a_i\}\)
  3. \(i = 1\)
  4. for \(m = 2\) to \(n\) do
  5. \(\text{if } s_m \text{ is not before } f_i \text{ then}\)
  6. \(S = S \cup \{a_m\}\)
  7. \(i = m\)
  8. return \(S\)
Why does it work?

• **THEOREM**
  Let $A$ be a set of activities and let $a_1$ be the activity with the earliest finish time. Then activity $a_1$ is in some maximum-sized subset of non-overlapping activities.

• **PROOF**
  Let $S'$ be a solution that does not contain $a_1$. Let $a'_1$ be the activity with the earliest finish time in $S'$. Then replacing $a'_1$ by $a_1$ gives a solution $S$ of the same size.

  Why are we allowed to replace? Why is it of the same size?

Then apply induction! How?
Greedy Algorithms – Huffman Coding

- Huffman Coding Problem
  
  **Example:** Release 29.1 of 15-Feb-2005 of TrEMBL Protein Database contains \(1,614,107\) sequence entries, comprising \(505,947,503\) amino acids. There are 20 possible amino acids. What is the minimum number of bits to store the compressed database?
  
  \(\sim 2.5 \text{ G bits or 300MB}\.\)

- How to improve this?
- **Information:** Frequencies are not the same.

<table>
<thead>
<tr>
<th>Amino Acid</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ala (A)</td>
<td>7.72</td>
</tr>
<tr>
<td>Arg (R)</td>
<td>5.24</td>
</tr>
<tr>
<td>Asn (N)</td>
<td>4.28</td>
</tr>
<tr>
<td>Asp (D)</td>
<td>5.28</td>
</tr>
<tr>
<td>Cys (C)</td>
<td>1.60</td>
</tr>
<tr>
<td>Gln (Q)</td>
<td>3.91</td>
</tr>
<tr>
<td>Glu (E)</td>
<td>6.54</td>
</tr>
<tr>
<td>Gly (G)</td>
<td>6.90</td>
</tr>
<tr>
<td>His (H)</td>
<td>2.26</td>
</tr>
<tr>
<td>Ile (I)</td>
<td>5.88</td>
</tr>
<tr>
<td>Leu (L)</td>
<td>9.56</td>
</tr>
<tr>
<td>Lys (K)</td>
<td>5.96</td>
</tr>
<tr>
<td>Met (M)</td>
<td>2.36</td>
</tr>
<tr>
<td>Phe (F)</td>
<td>4.06</td>
</tr>
<tr>
<td>Pro (P)</td>
<td>4.87</td>
</tr>
<tr>
<td>Ser (S)</td>
<td>6.98</td>
</tr>
<tr>
<td>Thr (T)</td>
<td>5.52</td>
</tr>
<tr>
<td>Trp (W)</td>
<td>1.18</td>
</tr>
<tr>
<td>Tyr (Y)</td>
<td>3.13</td>
</tr>
<tr>
<td>Val (V)</td>
<td>6.66</td>
</tr>
</tbody>
</table>

- **Idea:** Use shorter codes for more frequent amino acids and longer codes for less frequent ones.
Huffman Coding

2 million characters in file.


**IDEA 1:** Use ASCII Code
Each need at least 8 bits,
Total = 16 M bits = 2 MB

**IDEA 2:** Use 4-bit Codes
Each need at least 4 bits,
Total = 8 M bits = 1 MB

**IDEA 3:** Use Variable Length Codes

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Frequency</th>
<th>Code Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>22</td>
<td>11</td>
</tr>
<tr>
<td>T</td>
<td>22</td>
<td>10</td>
</tr>
<tr>
<td>C</td>
<td>18</td>
<td>011</td>
</tr>
<tr>
<td>G</td>
<td>18</td>
<td>010</td>
</tr>
<tr>
<td>N</td>
<td>10</td>
<td>001</td>
</tr>
<tr>
<td>Y</td>
<td>5</td>
<td>00011</td>
</tr>
<tr>
<td>R</td>
<td>4</td>
<td>00010</td>
</tr>
<tr>
<td>S</td>
<td>4</td>
<td>00001</td>
</tr>
<tr>
<td>M</td>
<td>3</td>
<td>00000</td>
</tr>
</tbody>
</table>

How to Decode?
Need Unique decoding!
Easy for Ideas 1 & 2.
What about Idea 3?

Percentage Frequencies

2 million characters in file.
Length = ?
Expected length = ?
Sum up products of frequency times the code length, i.e.,

\[(.22 \times 2 + .22 \times 2 + .18 \times 3 + .18 \times 3 + .10 \times 3 + .05 \times 5 + .04 \times 5 + .04 \times 5 + .03 \times 5) \times 2\ M\ \text{bits} = 3.24\ M\ \text{bits} = .4\ MB\]