Lecture 6: Complexity

CS 62
Spring 2012
Kim Bruce & Kevin Coogan

Lab This Week

• Timing ArrayList operations
  • Encourage working in pairs
  • Stopwatch class: start(), stop(), getTime(), reset()
• Java has Just-In-Time compiler
  • Must “warm-up” before you get accurate timing
  • What can mess up timing?
• Uses Vector from Bailey rather than ArrayList from Java libraries because can change way it increases in size.

Programming Assignment This Week

• Weak AI/Natural Language Processing:
  • Generate text by building frequency lists based on pairs of words. ArrayList of Associations of String (words) and Integer (count of that word).

Order of Magnitude

• Definition: We say that $g(n)$ is $O(f(n))$ if there exist two constants $C$ and $k$ such that $|g(n)| \leq C |f(n)|$ for all $n > k$.
• Examples: $2n+1$, $n^3-n^2+83$, $2^n+n^2$
• Used to measure time and space complexity of algorithms on data structures of size $n$.
• Most common are
  • $O(1)$ - for any constant
  • $O(\log n)$, $O(n)$, $O(n \log n)$, $O(n^2)$, ..., $O(2^n)$

Use simplest version in...
Comparing Orders of Magnitude

- Suppose have ops w/complexities given & problem of size \( n \) taking time \( t \).
- How long if increase size of problem?

<table>
<thead>
<tr>
<th>Problem Size:</th>
<th>10 ( n )</th>
<th>100 ( n )</th>
<th>1000 ( n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( O(\log n) )</td>
<td>( 3 + t )</td>
<td>( 7 + t )</td>
<td>( 10 + t )</td>
</tr>
<tr>
<td>( O(n) )</td>
<td>( 10 t )</td>
<td>( 100 t )</td>
<td>( 1000 t )</td>
</tr>
<tr>
<td>( O(n \log n) )</td>
<td>( &gt; 10 t )</td>
<td>( &gt; 100 t )</td>
<td>( &gt; 1000 t )</td>
</tr>
<tr>
<td>( O(n^2) )</td>
<td>( 100 t )</td>
<td>( 10,000 t )</td>
<td>( 1,000,000 t )</td>
</tr>
<tr>
<td>( O(2^n) )</td>
<td>( \approx t^{10} )</td>
<td>( \approx t^{100} )</td>
<td>( \approx t^{1000} )</td>
</tr>
</tbody>
</table>

Adding to ArrayList

- Suppose \( n \) elements in ArrayList and add 1.
- If space:
  - Add to end is \( O(1) \)
  - Add to beginning is \( O(n) \)
- If not space,
  - What is cost of ensureCapacity?
  - \( O(n) \) because \( n \) elements in array

EnsureCapacity

- What if only increase in size by 1 each time?
  - Adding \( n \) elements to end
    - Total cost of copying over arrays: \( 1 + 2 + 3 + \cdots + (n-1) = n(n-1)/2 \)
    - Total cost of \( O(n) \)
    - Average cost of each is \( O(n) \)
- What if double in size each time?
  - Suppose add \( n = 2^m \) new elts to end
    - Total cost of copying over arrays: \( 1 + 2 + 4 + \cdots + n/2 = n-1, O(n) \)
    - Average cost of \( O(n) \), but “lumpy”
ArrayList Ops

• Worst case
  • $O(1)$: size, isEmpty, get, set
  • $O(n)$: remove, add
• Add to end, on average $O(1)$