Lecture 7: Analysis of Algorithms

CS 62
Fall 2017
Kim Bruce & Alexandra Papoutsaki

Assignment

- **WordStream**: Reads text word by word
  - Use `nextToken()` but make sure `hasMoreTokens()`
- **Pair**: of two elements
  - **StringPair**: Pair of Strings. Extends `Pair`
  - Assume two associations `<k,v>, <k',v'>`.
    - the equals method will return true iff the k and k’ are equal
- **List**
  - `indexOf(Object o)` finds index of o in a list
  - Return -1 if on not in list

FreqList

- list of associations holding words and their frequencies
- Instance variable `List<Association<String, Integer>> fList`
- Start with `toString()`
- Continue with `add()`
  - What to check when adding?

In general...

- Work on paper first!
- More demanding than assignment 1. Start early!
- Come to office hours
- Don’t forget Friday’s quiz
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 $n > k$.
- Used to measure time and space complexity of algorithms on data structures of size $n$.
- Examples:
  - $2n + 1$ is $O(n)$
  - $n^2 - n^4 + 83$ is $O(n^3)$
  - $2^n + n^2$ is $O(2^n)$
- Most common are:
  - $O(1)$ - for any constant
  - $O(\log n), O(n), O(n \log n), O(n^2), ..., O(2^n)$

Complexity

- 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>$3t$</td>
<td>$7t$</td>
<td>$10t$</td>
</tr>
<tr>
<td>$O(n)$</td>
<td>$10t$</td>
<td>$100t$</td>
<td>$1000t$</td>
</tr>
<tr>
<td>$O(n \log n)$</td>
<td>$&gt;10t$</td>
<td>$&gt;100t$</td>
<td>$&gt;1000t$</td>
</tr>
<tr>
<td>$O(n^2)$</td>
<td>$&gt;100t$</td>
<td>$&gt;10,000t$</td>
<td>$&gt;1,000,000t$</td>
</tr>
<tr>
<td>$O(2^n)$</td>
<td>$&gt;1^{10}$</td>
<td>$&gt;2^{10}$</td>
<td>$&gt;2^{1000}$</td>
</tr>
</tbody>
</table>
Rule of thumb

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 one at a time to end
    - Total cost of copying over arrays: $1 + 2 + 3 + \cdots + (n - 1) = n(n - 1)/2$
    - Total cost of $0(n^2)$
    - Average cost of each is $O(n)$

- What if double in size each time?
  - Suppose add $n = 2^m$ new elements to end
    - Total cost of copying over arrays: $1 + 2 + 4 + \cdots + n/2 = n - 1, O(n)$
    - Average cost of $O(1)$, but "lumpy"

ArrayList Operations

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