Lecture 7: Analysis of Algorithms

CS 62
Fall 2018
Alexandra Papoutsaki & William Devanny
Order of Magnitude

• **Definition:** We say that \( f(n) \) is \( O(g(n)) \) iff there exist two constants \( C \) and \( k \) such that
  \[ |f(n)| \leq C |g(n)|, \text{ for all } n > k. \]

• Used to measure time and space complexity of algorithms on data structures of size \( n \).

• **Examples:**
  • \( 2n + 1 \) is \( O(n) \)
  • \( n^3 - n^2 + 83 \) is \( O(n^3) \)
  • \( 2^n + n^2 \) is \( O(2^n) \)
Order of Magnitude

• Most common are:
  • $O(1)$ - constant
  • $O(\log n)$ - logarithmic
  • $O(n)$ - linear
  • $O(n^2)$ - quadratic
  • $O(n^c)$ - polynomial
  • $O(c^n)$ - exponential
  • $O(n!)$ - factorial

• Growth:
  • $O(1), O(\log n), O(n), O(n \log n), O(n^2), O(2^n), O(n!), O(n^n)$
Complexity

Figure 5.2  Near-origin details of common curves. Compare with Figure 5.3.
Asymptotic Analysis

Figure 5.3  Long-range trends of common curves. Compare with Figure 5.2.
Comparing Orders of Magnitude

• Suppose we have the operations with complexities given and that a problem of size $n$ takes time $t$.
• How long would it take if we increase size of problem?

<table>
<thead>
<tr>
<th>Problem Size:</th>
<th>$10n$</th>
<th>$100n$</th>
<th>$1000n$</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>$10t$</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>$100t$</td>
<td>$10,000t$</td>
<td>$1,000,000t$</td>
</tr>
<tr>
<td>$O(2^n)$</td>
<td>$\sim t^{10}$</td>
<td>$\sim t^{100}$</td>
<td>$\sim t^{1000}$</td>
</tr>
</tbody>
</table>
Rule of thumb

![Diagram showing different time complexities](bigocheatsheet.com)
Adding to ArrayList

• Suppose there are \( n \) elements in ArrayList and you want to add one more. What is the cost of this operation?

• If enough space (\( \text{size}<\text{capacity} \)):
  • Add to end is \( O(1) \)
  • Add to beginning is \( O(n) \)

• If not space:
  • What is the cost of \texttt{ensureCapacity}?
  • \( O(n) \) because \( n \) elements in array
Amortized Time Analysis

As the arraylist increases in size, the doubling happens half as often but costs twice as much.
Amortized Time Analysis

We will use the aggregate method - the simplest method for amortized time analysis.

Others: accounting (banker’s) and potential (physicist’s).

Think of it as an average. For a total of \( n \) operations: \( O(\text{total cost}) \) / (number of operations) (in this case additions)

\[
0(\text{total cost of operations}) = \sum \text{cost of insertions} + \sum \text{cost of copying}
\]

Total cost of insertions: \( n \) of them, each \( O(1) \) cost, therefore \( nO(1) = O(n) \)

Total cost of copying: \( 1 + 2 + 2^2 + \cdots + 2^{\lfloor \log_2 n \rfloor} \leq 2n \) which is \( O(n) \)

\[
0(\text{total cost}) = O(n) + O(n) = O(n)
\]

Amortized time = \( O(n)/n = O(1) \) but “lumpy”
EnsureCapacity

• What if we only increase the capacity by 1 element each time?
  • Adding \( n \) elements one at a time to end
    • Total cost of \( n \) insertions: \( 1 + 2 + 3 + \cdots + (n - 1) = n(n - 1)/2 \)
    • Total cost of \( O(n^2) \)
  • Average cost of each is \( O(n) \)
ArrayList Operations

• Best case:
  • $O(1)$: size(), isEmpty(), get(int i), set(int i, E ele), remove(), add()

• Worst case:
  • $O(1)$: size(), isEmpty(), get(int i), set(int i, E ele)
  • $O(n)$: remove, add()

• add() runs in amortized constant time: adding $n$ elements requires $O(n)$ time.
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'>`.
  - Useful methods: `get(i)` and `getValue()`
  - 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 o 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