Properties of Algorithms

- Necessary properties
- What about desirable properties?
Attributes of Algorithms

Correctness: First, make it correct!
- Not as straightforward as it seems…
- Giving correct results to the wrong problem
- Must give correct results for all possible input values
- Issue of accuracy

Other practical considerations:
- Ease of understanding
- Elegance (style)
- Efficiency
Resources

- Time
- Space

“Efficiency” is the term used to describe an algorithm’s careful use of resources.

Question: how to measure the efficiency of an algorithm?
Benchmarking

- Use the same input data and timing an algorithm on different machines gives a comparison of machine speeds on identical tasks.
- Winbench: http://www.winbench.com
- Different from measuring the efficiency of an algorithm.
Formal Definition

- Count the number of instruction executions
- It is the number of steps each algorithm requires, not the time the algorithm takes on a particular machine, that is important for comparing two algorithms that do the same task.
Example

Data cleanup problem, removing 0 entries from a list before the average is computed.

\[0\ 24\ 16\ 0\ 36\ 42\ 23\ 21\ 0\ 27\]

Legit: legitimate element
A Choice of Algorithms

- The shuffle-left algorithm (Figure 3.1)
- The copy-over algorithm (Figure 3.2)
- The converging-pointers algorithm (Figure 3.3)
- Comparisons
Measuring Efficiency

- Sequential search
  - Best case complexity
  - Worst case complexity
  - Average case complexity
Order of Magnitude

- $O(n)$
- $O(n^2)$
## Analysis of Algorithms

### Data cleanup

<table>
<thead>
<tr>
<th></th>
<th>Shuffle-left</th>
<th>Copy-over</th>
<th>Converging pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time</strong></td>
<td>O(n)</td>
<td>O(n)</td>
<td>O(n)</td>
</tr>
<tr>
<td><strong>Space</strong></td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td><strong>Best Case</strong></td>
<td>O(n)</td>
<td>O(n)</td>
<td>O(n)</td>
</tr>
<tr>
<td><strong>Worst Case</strong></td>
<td>O(n^2)</td>
<td>O(n)</td>
<td>O(n)</td>
</tr>
<tr>
<td><strong>Average case</strong></td>
<td>O(n^2)</td>
<td>O(n)</td>
<td>O(n)</td>
</tr>
</tbody>
</table>

- Shuffle-left: Copy-over
- Converging pointer: [n, 2n]
Selection Sort

- Step 1: Get values for n and the n list items
- Step 2: Set the marker for the unsorted section at the end of the list
- Step 3: Repeat steps 4 through 6 until the unsorted section of the list is empty
  - Step 4: Select the largest number in the unsorted section of the list
  - Step 5: Exchange the number with the last number in the unsorted section of the list
  - Step 6: Move the marker for the unsorted section forward on position
- Step 7: Stop
Binary Search Algorithm

- Search an ordered list
- Figure 3.19
- Order-of-magnitude time efficiency?
Pattern Matching

- Forward match algorithm
- Best case: $O(n)$
- Worst case: $O(mxn)$
When Things Get Out Of Hand

- Polynomially bounded
- Hamiltonian circuit
- Exponential algorithm
- Approximation algorithm