Today’s Outline

• Announcements
  – Assignment #1 due Thurs, April 9 at 11:45pm

• Asymptotic Analysis

Exercise

```c
bool ArrayFind(int array[], int n, int key) {
    // Insert your algorithm here
}
```

What algorithm would you choose to implement this code snippet?

Analyzing Code

<table>
<thead>
<tr>
<th>Basic Java operations</th>
<th>Constant time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consecutive statements</td>
<td>Sum of times</td>
</tr>
<tr>
<td>Conditionals</td>
<td>Larger branch plus test</td>
</tr>
<tr>
<td>Loops</td>
<td>Sum of iterations</td>
</tr>
<tr>
<td>Function calls</td>
<td>Cost of function body</td>
</tr>
<tr>
<td>Recursive functions</td>
<td>Solve recurrence relation</td>
</tr>
</tbody>
</table>

Analyze your code!

Linear Search Analysis

```c
bool LinearArrayFind(int array[], int n, int key) {
    for (int i = 0; i < n; i++) {
        if (array[i] == key) {
            // Found it!
            return true;
        }
    }
    return false;
}
```

Best Case:
Worst Case:

Binary Search Analysis

```c
bool BinArrayFind(int array[], int low, int high, int key) {
    // The subarray is empty
    if (low > high) return false;
    // Search this subarray recursively
    int mid = (high + low) / 2;
    if (key == array[mid]) {
        return true;
    } else if (key < array[mid]) {
        return BinArrayFind(array, low, mid-1, key);
    } else {
        return BinArrayFind(array, mid+1, high, key);
    }
}
```

Best case:
Worst case:
Solving Recurrence Relations

1. Determine the recurrence relation. What is the base case(s)?

2. “Expand” the original relation to find an equivalent general expression in terms of the number of expansions.

3. Find a closed-form expression by setting the number of expansions to a value which reduces the problem to a base case.

Linear Search vs Binary Search

<table>
<thead>
<tr>
<th>Method</th>
<th>Worst Case</th>
<th>Best Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Search</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Binary Search</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

So … which algorithm is better? What tradeoffs can you make?

Fast Computer vs. Slow Computer

- Linear search on Pentium IV: $T(n) = 3n + 2 \in \Theta(n)$
- Binary search on Pentium IV: $T(n) = 4 \log_2 n + 4 \in \Theta(\log n)$

Remember: the fastest algorithm has the slowest growing function for its runtime.

Asymptotic Analysis

- Asymptotic analysis looks at the order of the running time of the algorithm.
  - A valuable tool when the input gets “large”
  - Ignores the effects of different machines or different implementations of the same algorithm

- Intuitively, to find the asymptotic runtime, throw away the constants and low-order terms.
  - Linear search is $T(n) = 3n + 2 \in \Theta(n)$
  - Binary search is $T(n) = 4 \log_2 n + 4 \in \Theta(\log n)$
Asymptotic Analysis

- Eliminate low order terms
  - $4n + 5 \Rightarrow 0.5n \log n + 2n + 7 \Rightarrow n^3 + 2n + 3n \Rightarrow$
  - $0.5n \log n \Rightarrow n \log n$

- Eliminate coefficients
  - $4n \Rightarrow 0.5n \log n$

Order Notation: Intuition

Although not yet apparent, as $n$ gets “sufficiently large”, $f(n)$ will be “greater than or equal to” $g(n)$.

Definition of Order Notation

- Upper bound: $T(n) = O(f(n))$  
  Exist constants $c$ and $n'$ such that $T(n) \leq c f(n)$ for all $n \geq n'$
- Lower bound: $T(n) = \Omega(g(n))$  
  Exist constants $c$ and $n'$ such that $T(n) \geq c g(n)$ for all $n \geq n'$
- Tight bound: $T(n) = \Theta(f(n))$  
  When both hold: $T(n) = O(f(n))$  
  $T(n) = \Omega(f(n))$

Order Notation: Definition

$O(f(n))$: a set or class of functions

$g(n) \in O(f(n))$ if there exist consts $c$ and $n_0$ such that:

$g(n) \leq c f(n)$ for all $n \geq n_0$

Example: $g(n) = 1000n$ vs. $f(n) = n^2$

Is $g(n) \in O(f(n))$?

Pick: $n_0 = 1000$, $c = 1$

Order Notation: Example

$100n^2 + 1000 \leq n^3 + 2n^2$ for all $n \geq 19$

So $f(n)$ is $O(g(n))$

Notation Notes

Note: Sometimes, you’ll see the notation:

$g(n) = O(f(n))$.

This is equivalent to:

$g(n) \in O(f(n))$.

However: The notation $O(f(n)) = g(n)$ is meaningless!

(in other words big-O “equality” is not symmetric)
Big-O: Common Names

- constant: $O(1)$
- logarithmic: $O(\log n)$ ($\log n$, $\log n^2$ is $O(\log n)$)
- linear: $O(n)$
- log-linear: $O(n \log n)$
- quadratic: $O(n^2)$
- cubic: $O(n^3)$
- polynomial: $O(n^k)$ (k is a constant)
- exponential: $O(c^n)$ (c is a constant > 1)

Meet the Family

- $O( f(n) )$ is the set of all functions asymptotically less than or equal to $f(n)$
- $o( f(n) )$ is the set of all functions asymptotically strictly less than $f(n)$
- $\Omega( f(n) )$ is the set of all functions asymptotically greater than or equal to $f(n)$
- $\omega( f(n) )$ is the set of all functions asymptotically strictly greater than $f(n)$
- $\Theta( f(n) )$ is the set of all functions asymptotically equal to $f(n)$

Meet the Family, Formally

- $g(n) \in O( f(n) )$ iff
  There exist $c$ and $n_0$ such that $g(n) \leq c \cdot f(n)$ for all $n \geq n_0$
- $g(n) \in o( f(n) )$ iff
  There exists a $n_0$ such that $g(n) < c \cdot f(n)$ for all $c$ and $n \geq n_0$
- $g(n) \in \Omega( f(n) )$ iff
  Equivalent to: $\lim_{n \to \infty} \frac{g(n)}{f(n)} = 0$
  There exist $c > 0$ and $n_0$ such that $g(n) \geq c \cdot f(n)$ for all $n \geq n_0$
- $g(n) \in \omega( f(n) )$ iff
  There exists a $n_0$ such that $g(n) > c \cdot f(n)$ for all $c$ and $n \geq n_0$
- $g(n) \in \Theta( f(n) )$ if
  Equivalent to: $\lim_{n \to \infty} \frac{g(n)}{f(n)} = 1$
  $g(n) \in O( f(n) )$ and $g(n) \in \Omega( f(n) )$

Big-Omega et al. Intuitively

<table>
<thead>
<tr>
<th>Asymptotic Notation</th>
<th>Mathematics Relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O$</td>
<td>$\leq$</td>
</tr>
<tr>
<td>$\Omega$</td>
<td>$\geq$</td>
</tr>
<tr>
<td>$\Theta$</td>
<td>$=$</td>
</tr>
<tr>
<td>$o$</td>
<td>$&lt;$</td>
</tr>
<tr>
<td>$\omega$</td>
<td>$&gt;$</td>
</tr>
</tbody>
</table>

Pros and Cons of Asymptotic Analysis

Types of Analysis

Two orthogonal axes:

- bound flavor
  - upper bound ($O$, $o$)
  - lower bound ($\Omega$, $\omega$)
  - asymptotically tight ($\Theta$)

- analysis case
  - worst case (adversary)
  - average case
  - best case
  - “amortized”
Which Function Grows Faster?

\[ n^3 + 2n^2 \text{ vs. } 100n^2 + 1000 \]

Which Function Grows Faster?

\[ n^0.1 \text{ vs. } \log n \]

Which Function Grows Faster?

\[ 5n^5 \text{ vs. } n! \]
Nested Loops

for $i = 1$ to $n$
  for $j = 1$ to $n$
    sum = sum + 1

for $i = 1$ to $n$
  for $j = 1$ to $n$
    sum = sum + 1

16$n^3$ log$_8$(10$n^2$) + 100$n^2$ = $O(n^3 \log(n))$

- Eliminate low order terms
- Eliminate constant coefficients

4/06/09 31

16$n^3$ log$_8$(10$n^2$) + 100$n^2$ = $O(n^3 \log(n))$

- Eliminate low order terms
- Eliminate constant coefficients

4/06/09 32

16$n^3$ log$_8$(10$n^2$) + 100$n^2$ = $O(n^3 \log(n))$

- Eliminate low order terms
- Eliminate constant coefficients

4/06/09 33

16$n^3$ log$_8$(10$n^2$) + 100$n^2$ = $O(n^3 \log(n))$

- Eliminate low order terms
- Eliminate constant coefficients

4/06/09 34