Oh Notation

Growth of Functions

S.V. N. (vishy) Vishwanathan

University of California, Santa Cruz
vishy@ucsc.edu

January 12, 2016
Outline

1. Why Compare Algorithms?

2. Oh Notation

3. Omega and Theta Notation

4. Little o and ω Notation
There is a Zoo of Algorithms for Sorting

- Insertion Sort
- Bubble Sort
- Quick Sort
- Merge Sort
- Heap Sort
- Radix or Bucket Sort
- and many many more . . .
Why Compare Algorithms?

Which one is the Best?

- Idea 1: Take the particular array you want to sort and try all the algorithms and see which one takes the least amount of time
- Problems:
  - I don’t want to spend all my time coding all these different algorithms
  - My array might be so large that each algorithm takes many hours to finish
Is There a Better Way?

- Idea: Can we count how many *elementary* operations are required by the algorithm to solve a given input.
- Advantage: Definitely better than simply looking at wall clock time.
- Problems:
  - What is *elementary* is not well defined. Different machines have different capabilities.
  - Same code might take different amounts of time on different computers.
  - If my input changes, the number of operations needed by the algorithm will change.
Let’s be Pessimistic

- From among all inputs, let us pick the worst one, that is, the one for which the algorithm takes the maximum number of operations.
- Taking this one step further: For an input of size $n$, can we analytically write out how many operations the algorithm needs?
- We will get a function $f(n) > 0$.
Still does not Solve my Problem

- If the number of operations of one algorithm grows as $f(n)$ and another one as $g(n)$, which one is better?
Outline

1. Why Compare Algorithms?
2. Oh Notation
3. Omega and Theta Notation
4. Little $o$ and $\omega$ Notation
Basic Idea

Focus on the asymptotic growth of a function

- Asymptotic: Fancy way of saying that we will focus on how the function grows as $n \to \infty$. 
Given functions $g(n)$ and $f(n)$ we say that $f(n) \in O(g(n))$ if
- there exists constants $c$ and $n_0$
- such that

$$0 \leq f(n) \leq c \cdot g(n) \text{ for all } n \geq n_0.$$ 

Notational convention: $f(n) = O(g(n))$

Typically $g(n)$ is chosen to be a *simple* function
Oh Notation

In Pictures

\[ c \cdot g(n) \]

\[ f(n) \]

\[ n_0 \]

\[ n \]
Example - 1

- $2n + 30 = O(n)$
- Need to show that there exists $c$ and $n_0$ such that
  \[0 \leq 2n + 30 \leq c \cdot n \text{ for all } n \geq n_0\]
- Let $c = 3$ and $n_0 = 30$. 

Example - 2

- $2n + 30 = O(n^2)$
- Need to show that there exists $c$ and $n_0$ such that
  $$0 \leq 2n + 30 \leq c \cdot n^2 \text{ for all } n \geq n_0$$
- Let $c = 2$ and $n_0 = 6$. 
Example - 3

- $5n^3 + 10n \notin O(n^2)$
- Need to show that there exists $c$ and $n_0$ such that
  \[ 0 \leq 5n^3 + 10n \leq c \cdot n^2 \text{ for all } n \geq n_0 \]
- Rewrite as
  \[ 0 \leq 5n + \frac{10}{n} \leq c \text{ for all } n \geq n_0 \]

  But as $n \to \infty$ we have $5n + \frac{10}{n} \to \infty$. 
How Does This Help me Compare Algorithms? - Insertion Sort

\textbf{Insertion-Sort}(A)

1. \textbf{for} $j = 2$ \textbf{to} $A.length$ // Loop runs $n - 1$ times
2. \hspace{1em} key = $A[j]$
4. \hspace{1em} $i = j - 1$
5. \hspace{1em} \textbf{while} $i > 0$ and $A[i] >$ key // Loop can run up to $j - 1$ times
6. \hspace{2em} $A[i + 1] = A[i]$
7. \hspace{2em} $i = i - 1$
8. \hspace{1em} $A[i + 1] =$ key
How Does This Help me Compare Algorithms? - Insertion Sort

- Total operation count in the worst case:
  - When \( j = 2 \), inner loop can run \( j - 1 = 1 \) time
  - When \( j = 3 \), inner loop can run \( j - 1 = 2 \) times
  - \( \ldots \)
  - When \( j = n \), inner loop can run \( j - 1 = n - 1 \) times

- Summing everything up
  - \( 1 + 2 + 3 + \ldots + n - 1 = ? \)

- Therefore the insertion sort algorithm is \( O(?) \).
Question

What about the operations inside the loop? Why don’t they matter?

**Insertion-Sort(A)**

```plaintext
1 for j = 2 to A.length
2   key = A[j]
3   // Insert A[j] into the sorted sequence A[1..j - 1].
4   i = j - 1
5   while i > 0 and A[i] > key
6       A[i + 1] = A[i]
7       i = i - 1
8   A[i + 1] = key
```
How Does This Help me Compare Algorithms? - Bubble Sort

**BubbleSort**($A$)

1. for $i = 1$ to $A$.length$ - 1$
2. for $j = A$.length downto $i + 1$
3. if $A[j] \leq A[j - 1]$
Keep in Mind

- Remember that saying that an algorithm is $O(n^2)$ does \textbf{not} imply that the algorithm takes $cn^2$ time for every input.
Outline

1. Why Compare Algorithms?
2. Oh Notation
3. Omega and Theta Notation
4. Little $o$ and $\omega$ Notation
Given functions $g(n)$ and $f(n)$ we say that $f(n) \in \Theta(g(n))$ if

- there exists constants $c_1$, $c_2$ and $n_0$
- such that

$$0 \leq c_1 g(n) \leq f(n) \leq c_2 \cdot g(n) \text{ for all } n \geq n_0.$$

Notational convention: $f(n) = \Theta(g(n))$

Typically $g(n)$ is chosen to be a *simple* function
An Alternate View Point

\[ \lim_{n \to \infty} \frac{f(n)}{g(n)} = \text{const.} > 0 \]
Example - 4

- $5n^2 + 10n = \Theta(n^2)$
- Need to show that there exists $c_1$, $c_2$ and $n_0$ such that
  \[
  0 \leq c_1 n^2 \leq 5n^2 + 10n \leq c_2 \cdot n^2 \text{ for all } n \geq n_0
  \]
- Rewrite as
  \[
  0 \leq c_1 \leq 5 + \frac{10}{n} \leq c_2 \text{ for all } n \geq n_0
  \]

But as $n \to \infty$ we have $5 + \frac{10}{n} \to 5$. So select $c_1 = 4$ and $c_2 = 6$, with $n_0 = 10$. 
Omega Notation

- Given functions $g(n)$ and $f(n)$ we say that $f(n) \in \Omega(g(n))$ if
  - there exists constants $c$ and $n_0$
  - such that
    \[
    0 \leq cg(n) \leq f(n) \text{ for all } n \geq n_0.
    \]

- Notational convention: $f(n) = \Omega(g(n))$

- Typically $g(n)$ is chosen to be a simple function
Omega and Theta Notation

In Pictures

(a) $f(n) = \Theta(g(n))$

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

(c) $f(n) = \Omega(g(n))$
Fun Fact. Try this at Home.

For any two functions \( f(n) \) and \( g(n) \), we have

\[
f(n) = \Theta(g(n)) \iff f(n) = O(g(n)) \text{ and } f(n) = \Omega(g(n)).
\]
Outline

1. Why Compare Algorithms?
2. Oh Notation
3. Omega and Theta Notation
4. Little $o$ and $\omega$ Notation
### Little $o$ and $\omega$ Notation

**$o$ Notation**

- $O$ notation is not asymptotically tight. For instance $n = O(n)$ and also $n = O(n^2)$.

- Given functions $g(n)$ and $f(n)$ we say that $f(n) \in o(g(n))$ if
  - there exists a constant $n_0$
  - such that
    
    \[
    0 \leq f(n) \leq cg(n) \text{ for all } c > 0 \text{ and } n \geq n_0.
    \]

- Carefully note the order of the qualifiers.
- Intuitively, $f(n)$ becomes insignificant compared to $g(n)$

\[
\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0.
\]
Example - 5

- $5n^2 + 10n \notin o(n^2)$
- $\lim_{n \to \infty} 5 + \frac{10}{n} = 5 \neq 0$
- $5n^2 + 10n = o(n^3)$
- $\lim_{n \to \infty} \frac{5}{n} + \frac{10}{n^2} = 0$
Given functions $g(n)$ and $f(n)$ we say that $f(n) \in \omega(g(n))$ if

- there exists a constant $n_0$
- such that

$$0 \leq cg(n) \leq f(n) \text{ for all } c > 0 \text{ and } n \geq n_0.$$

Carefully note the order of the qualifiers.

Intuitively, $f(n)$ grows much faster as compared to $g(n)$

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty.$$
Questions?