CS 457, Fall 2016

Drexel University, Department of Computer Science

Lecture 2

Today's Lecture

- Insertion Sort
  - Correctness
  - Running time
- Asymptotic Notation
  - Big Oh, Big Omega, Theta, Little Oh, Little Omega
- First Homework

Insertion Sort

```
INSERTION_SORT (A)

for j = 2 to A.length
        key = A[j]
        // Insert A[j] into the sorted sequence A[1 .. j − 1].
        i = j − 1
        while i > 0 and A[i] > key
                A[i + 1] = A[i]
                i = i − 1
        A[i + 1] = key
```

Execution:

7 3 5 8 1 2
3 7
3 **5** 7
3 5 7 **8**
1 3 5 7 8
1 **2** 3 5 7 8

Insertion Sort (Correctness)

```
INSERTION_SORT (A)

for j =2 to A.length
        key = A[j]
        // Insert A[j] into the sorted sequence A[1 .. j − 1].
        i = j − 1
        while i > 0 and A[i] > key
                A[i +1] = A[i]
                i = i − 1
        A[i + 1] = key
```

Loop Invariant:

At the start of each iteration of the **for** loop, the subarray A[1 ,..., j-1] consists of elements originally in A[1 ,..., j-1], but in sorted order

Things to show about invariant:

1. Initialization
2. Maintenance
3. Termination

Insertion Sort (Running Time)

INSERTION_SORT (*A*)	COST	TIMES
for j = 2 **to** A.length	c_1	n
key = *A*[*j*]	c_2	n-1
// Insert *A*[*j*] into the sorted sequence *A*[1 .. *j* – 1].	$c_3 = 0$	n-1
i = *j* – 1	c_4	n-1
while *i* > 0 and *A*[*i*] > key	c_5	$\sum_{j=2}^{n} t_j$
A[*i* +1] = *A*[*i*]	c_6	$\sum_{j=2}^{n}(t_j - 1)$
i = *i* – 1	c_7	$\sum_{j=2}^{n}(t_j - 1)$
A[*i* + 1] = key	c_8	n-1

$$T(n) = c_1 n + (c_2 + c_4 + c_8)(n-1) + c_5 \sum_{j=2}^{n} t_j + (c_6 + c_7) \sum_{j=2}^{n}(t_j - 1)$$

- $t_j \geq 1$ so $\sum_{j=2}^{n} t_j \geq n - 1$ and $T(n) \geq (c_1 + c_2 + c_4 + c_5 + c_8)n - (c_2 + c_4 + c_5 + c_8)$
- $t_j \leq j$ so $\sum_{j=2}^{n} t_j \leq \frac{n(n+1)}{2} - 1$ and $T(n) \leq C_1 n^2 + C_2 n + C_3$

Asymptotic Notation

- Worst-case running time as a function of input size n is a **function $f(n)$**

- How does $f(n)$ grow **as a function of n**? (Fooplot, Google)
  - $f(n) = n + logn$
  - $f(n) = n + 100$
  - $f(n) = 2^n - 10n$

- Comparing algorithms for sorting:
  - Insertion sort is roughly $f(n) = c_1 n^2$. We will say that $f(n)$ is $O(n^2)$
  - Merge sort is roughly $f(n) = c_2 n \, logn$. We will say that $f(n)$ is $O(n \, logn)$

Asymptotic Notation (Big-Oh)

$$O(g(n)) = \left\{ \begin{array}{c} f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that} \\ 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0 \end{array} \right\}$$

- This is a **set** of functions! We should say $f(n) \in O\big(g(n)\big)$, but for notational simplicity, we will use $f(n) = O\big(g(n)\big)$
- Say the running time of Insertion Sort is $T(n) \leq 10n^2 + 5n - 3$
  - Worst-case running time is $O(n^2)$
- Is $n \log n = O(n)$?

Asymptotic Notation (Big-Omega)

$$\Omega(g(n)) = \left\{ \begin{array}{c} f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that} \\ 0 \leq cg(n) \leq f(n) \text{ for all } n \geq n_0 \end{array} \right\}$$

- What do we know about insertion sort?
  - Best-case running time is $\Omega(n)$
- Are these the best bounds that we can get?
  - Worst-case running time is $\Omega(n^2)$ and best-case is $O(n)$
- Is $n\, logn = \Omega(\mathrm{n})$?

Asymptotic Notation (Theta)

$$\Theta(g(n)) = \left\{ \begin{array}{l} f(n) : \text{there exist positive constants } c_1, c_2, \text{and } n_0 \text{ such that} \\ \quad 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ for all } n \geq n_0 \end{array} \right\}$$

- What do we know about insertion sort?

Asymptotic Notation (little-oh, little-omega)

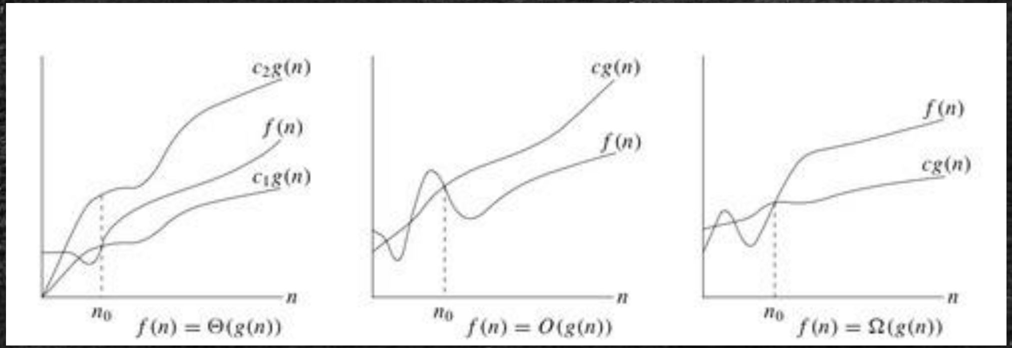
$$\mathrm{o}(g(n)) = \left\{ \begin{array}{c} f(n) : \text{for any constant } c > 0 \text{, there exists a constant } n_0 > 0 \text{ s.t.} \\ 0 \leq f(n) < cg(n) \text{ for all } n \geq n_0 \end{array} \right\}$$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

$$\omega(g(n)) = \left\{ \begin{array}{c} f(n) : \text{for any constant } c > 0 \text{, there exists a constant } n_0 > 0 \text{ s.t.} \\ 0 \leq cg(n) < f(n) \text{ for all } n \geq n_0 \end{array} \right\}$$

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

Asymptotic Notation



$f(n) = O(g(n))$ is like $a \leq b$
$f(n) = \Omega(g(n))$ is like $a \geq b$
$f(n) = \Theta(g(n))$ is like $a = b$
$f(n) = o(g(n))$ is like $a < b$
$f(n) = \omega(g(n))$ is like $a > b$

- More examples?

Asymptotic Notation Properties

- Transitivity:
  - $f(n)=O(g(n))$ and $g(n)=O(h(n))$, then $f(n)=O(h(n))$
  - $f(n)=\Omega(g(n))$ and $g(n)=\Omega(h(n))$, then $f(n)=\Omega(h(n))$
- Reflexivity: $f(n)=\Theta(f(n))$
- Transpose Symmetry: $f(n)=O(g(n))$ if and only if $g(n)=\Omega(f(n))$
- Symmetry: $f(n)=\Theta(g(n))$ if and only if $g(n)=\Theta(f(n))$
- Trichotomy: For any two real numbers a and b:
  - $a > b$, or $a = b$, or $a < b$

First Homework

- Available: Thursday 9/22
- Due: Thursday 9/29

- Email me with any questions

- Gradescope accounts