CS 457, Fall 2019

Drexel University, Department of Computer Science

Lecture 2

Today's Lecture

- Algorithms for a variety of problems
- How to measure the efficiency of an algorithm
- Asymptotic notation

Why study algorithms?

- What is an algorithm?
  - A well-defined **computational procedure** that takes some value(s) as **input** and produces some value(s) as **output**.
  - A **sequence of computational steps** that transform the **input** into the **output**.
- Goal of an algorithm: solve a **computational problem**
  - Sorting problem:
    - Input: (32, 20, 25, 10, 18, 1, 9)
    - Output: (1, 9, 10, 18, 20, 25, 32)
  - Shortest path, travelling salesman, knapsack, maximum flow ...
- Algorithm **design and analysis techniques**
  - greedy, divide & conquer, randomization, dynamic programming...

Why study algorithms?

- Given a computational problem, e.g., the sorting problem
  - A specific input, e.g., (32, 20, 25, 10, 18, 1, 9) is a **problem instance**
- When is an algorithm **correct**?
  - When it computes the desired output on every problem instance
- When is an algorithm **efficient**?
  - Time efficient (main focus of this class)
  - Space efficient
  - Amenable to Parallelism
  - Not consuming too much bandwidth
  - Simple to code up…

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

- What is the running time of this algorithm?
  - $O(n)$
  - $O(n \log n)$
  - $O(n^2)$
  - $O(\sqrt{n})$

Insertion Sort (Running Time)

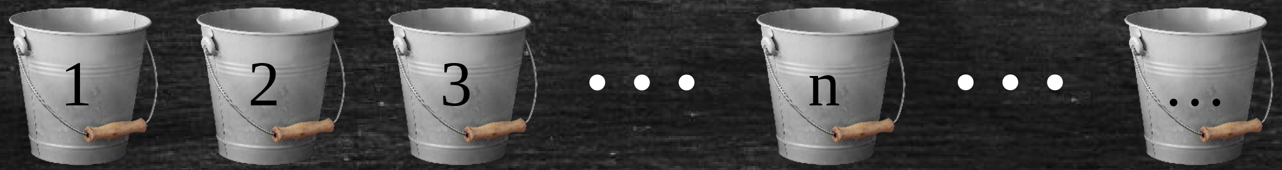
INSERTION_SORT (A)		COST	TIMES
for j =2 **to** A.length	• --------->	c_1	n
key = $A[j]$	• --------->	c_2	n-1
// Insert $A[j]$ into the sorted sequence $A[1 .. j-1]$.	• --------->	c_3=0	n-1
$i = j - 1$	• --------->	c_4	n-1
while $i > 0$ and $A[i] >$ key	• --------->	c_5	$\sum_{j=2}^{n} t_j$
$A[i+1] =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

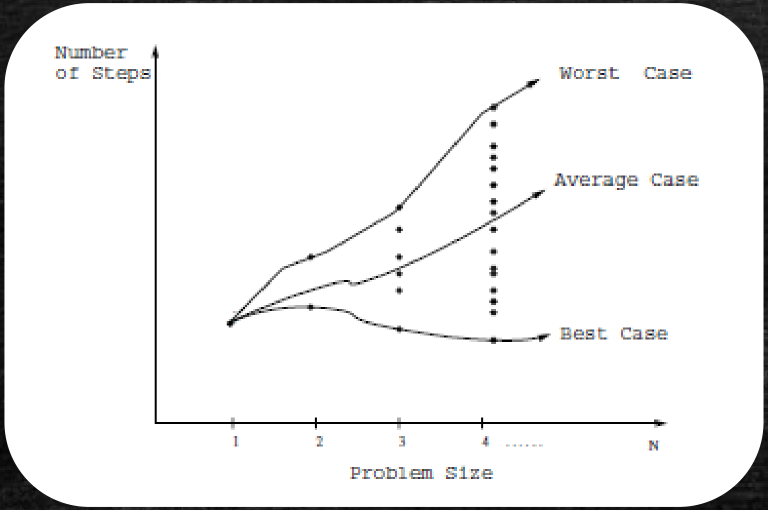
$$T(n) = c_1 n + (c_2 + c_4 + c_8)(n-1) + c_5 \sum_{j=2}^{n} t_j + (c_6 + c_7) \sum_{j=2}^{n}(t_j - 1)$$

- $t_j \geq 1$ 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 $T(n) \leq C_1 n^2 + C_2 n + C_3$

Running time as a function of input size

- For each possible input size come up with a “bucket”





This is a “lossy compression”! It is missing all the details that appear in the figure regarding the performance of the algorithm in general…

- Worst case running time of an algorithm is a function of n

Asymptotic Notation

- Worst-case running time of an algorithm is a **function $f(n)$**
- How does $f(n)$ grow as n grows larger?
  - $f(n) = n + \log n$
  - $f(n) = n + 100$
  - $f(n) = 2^n - 10n$
- 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 $f(n) = c_2 n \log n$. We will say that f(n) is $O(n \log n)$
  - Would you prefer a (worst case) running time of $1000\,n$ or just $n \log n$

Asymptotic Notation (Big-Oh)

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

- 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 $T(n) \le 10n^2 + 5n - 3$
  - 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 $n \log n = \Omega(\mathrm{n})$?

Asymptotic Notation (Theta)

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

- What do we know about insertion sort?

Asymptotic Notation (little-oh, little-omega)

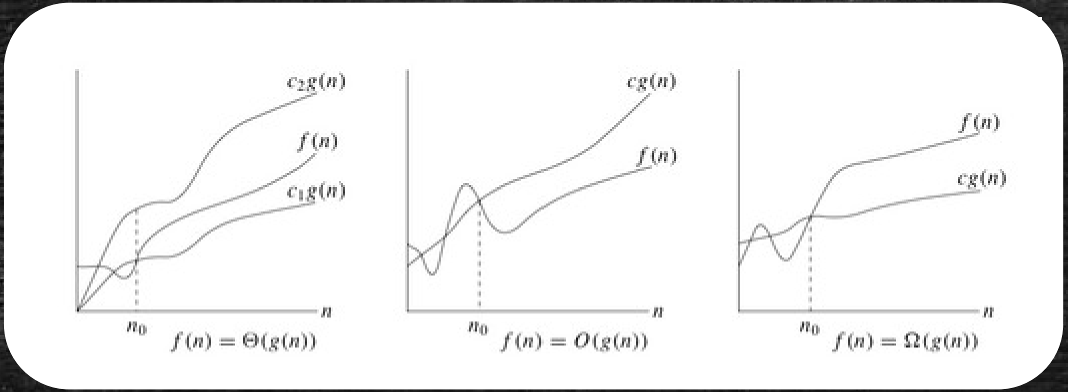
$$o(g(n)) = \left\{ \begin{array}{c} f(n) : \text{for any constant } c > 0, \text{ there exists a constant } n_0 > 0 \text{ s.t.} \\ 0 \le f(n) < cg(n) \text{ for all } n \ge n_0 \end{array} \right\}$$

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

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

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

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 $a > b$

1. Find a function $g(n)$ such that $5 + \cos(n) = \Theta(g(n))$
2. Show that $10n^2 - 100n - 20 \neq \Theta(n \log n)$
3. Show that $\max\{f(n), g(n)\} = \Theta(f(n) + g(n))$

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 $a < b$