JOHNS HOPKINS UNIVERSITY | Engineering for Professionals

# Introduction to Algorithms

**Foundations of Algorithms**
Guven

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

# Reading Assignment

- Cormen, Chapter 1, 2, 3
- Check out Beginning to Python: https://wiki.python.org/moin/BeginnersGuide/Programmers
- From Blackboard, study the Project rubric and Jupyter notebook example

# Outline

- Design and analysis of algorithms
- Real world algorithms
- Insertion sort
- Asymptotic analysis
- Integer multiplication
- Merge sort
- Notes on Proofs

# This Course

- The theoretical study of design and analysis of computer algorithms
- Basic goals for an algorithm
  - always correct
  - always terminates
  - performance
    - speed
    - space
    - complexity

# Design and Analysis of Algorithms

- Analysis: Predict the **cost** of an algorithm in terms of resources and performance

- Design: Design **correct** algorithms which minimize the **cost**

# Our Machine Model

- Generic Random Access Machine (RAM)
- Executes operations sequentially
- Set of primitive operations
  - Arithmetic
  - Logical
  - Comparisons
  - Function calls
- Simplifying assumption: all operations cost 1 unit
- No dependence on the speed of the computer

# Most Basic Algorithmic Problems

- Pattern matching, pattern searching
- Sorting
- Searching
- Dynamic programming
- Hashing
- Optimization
- Modeling, higher level representations

# 10 Real-World Algorithms

- Merge Sort, Quick Sort and Heap Sort
- Fourier Transform and Fast Fourier Transform (FFT)
- Dijkstra’s algorithm
- RSA algorithm
- Secure hash algorithm
- Integer factorization
- Link analysis
- Proportional Integral Derivative (PID) Algorithm
- Data compression algorithms
- Random number generation

# Insertion Sort (find the bug!)

```
def insertion_sort(A,n):   # A[0..n-1]
    for j in range(1,n):
        key = A[j]
        i = j-1
        while i>0 and A[i]>key:
            A[i+1] = A[i]
            i = i-1
        A[i+1] = key
```

- A: [0....i→→→→→j...............n-1]

**sorted** **key**

# Time Complexity

- The running time depends on the input
  - An already sorted sequence is faster to sort
- Major simplifying convention
  - Parameterize the running time by the size of the input, since short sequences are quicker to sort than long ones
- $T_A(n)$ = Time of A on inputs with length n
- Generally, we seek upper bounds on the running time, to have guaranteed performance

# Analyses Types

- Worst-case: (occasional)
  - T(n) = maximum running time of algorithm
- Average-case: (most of the times)
  - T(n) = expected time of algorithm
  - Assumption of statistical distribution of inputs
- Best-case: (seldom)
  - A slow algorithm might work fast on some input
  - Lucky case

# Machine-independent Time

- What is insertion sort's worst-case time?
  - Ignore machine dependent constants
  - Otherwise impossible to verify and to compare algorithms
- Look at growth of $\mathrm{T}(\mathrm{n})$ as $\mathrm{n} \to \infty$
- Asymptotic analysis

# Θ-notation

- Definition

$$\Theta(g(n)) = \{\ f(n) : \exists \text{ positive constants } c_1, c_2, n_0 \text{ such that } 0 \le c_1 g(n) \le f(n) \le c_2 g(n)\ \forall\ n_0 \le n\}$$

- Example
  - $3n^3+90n^2-5n+3000 = \Theta(n^3)$
  - Drop low-order terms, ignore leading constants

# Asymptotic Performance

- When n gets large enough, a $\Theta(n^2)$ algorithm always better than a $\Theta(n^3)$ algorithm
  - *i.e.* runs faster
- Asymptotic analysis is a useful tool to help analysis and design of algorithms
  - Don't ignore asymptotically slower algorithms
  - Real-world design situations need balancing

# Insertion Sort Analysis

- Best case, already sorted

  $\Theta(n)$

- Worst case, sorted array (!)

  $\Theta(n^2)$

- Average case, all permutations equally likely

  $T(n) = \sum_{j=1}^{n-1} \Theta\left(\frac{j}{2}\right) = \Theta(n^2)$

# Integer Multiplication

- Let $X=[A:B]$ and $Y=[C:D]$ where $A,B,C,$ and $D$ are $n/2$ bit integers
- Straightforward method
  $XY = (2^{n/2}A+B)(2^{n/2}C+D)$
  $\quad =2^{n}AC+2^{n/2}AD+2^{n/2}BC+BD$
- Recurrence
  $T(n) < 4T(n/2) + \Theta(n)$
- $T(n) = \Theta(n^2)$  (apply master theorem case 3)

# Integer Multiplication - Better

- Let $X=[A:B]$ and $Y=[C:D]$ where $A,B,C$, and $D$ are $n/2$ bit integers
- Karatsuba:
  $XY = (2^{n/2}+2^{n})AC + 2^{n/2}(A\text{-}B)(C\text{-}D) + (2^{n/2}+1)BD$
- Recurrence
  $T(n) < 3T(n/2) + \Theta(n)$
- $\Theta(n) = \Theta(n^{\log 3})$

# Merge Sort

```
def merge_sort():   # A[0..n-1]
    if n==1:
        done
    recursively sort A[0..⌈n/2-1⌉]
    recursively sort A[⌈n/2⌉..n-1]
    merge 2 sorted lists
```

- Time to merge a total of n elements $\Theta(n)$ - linear time

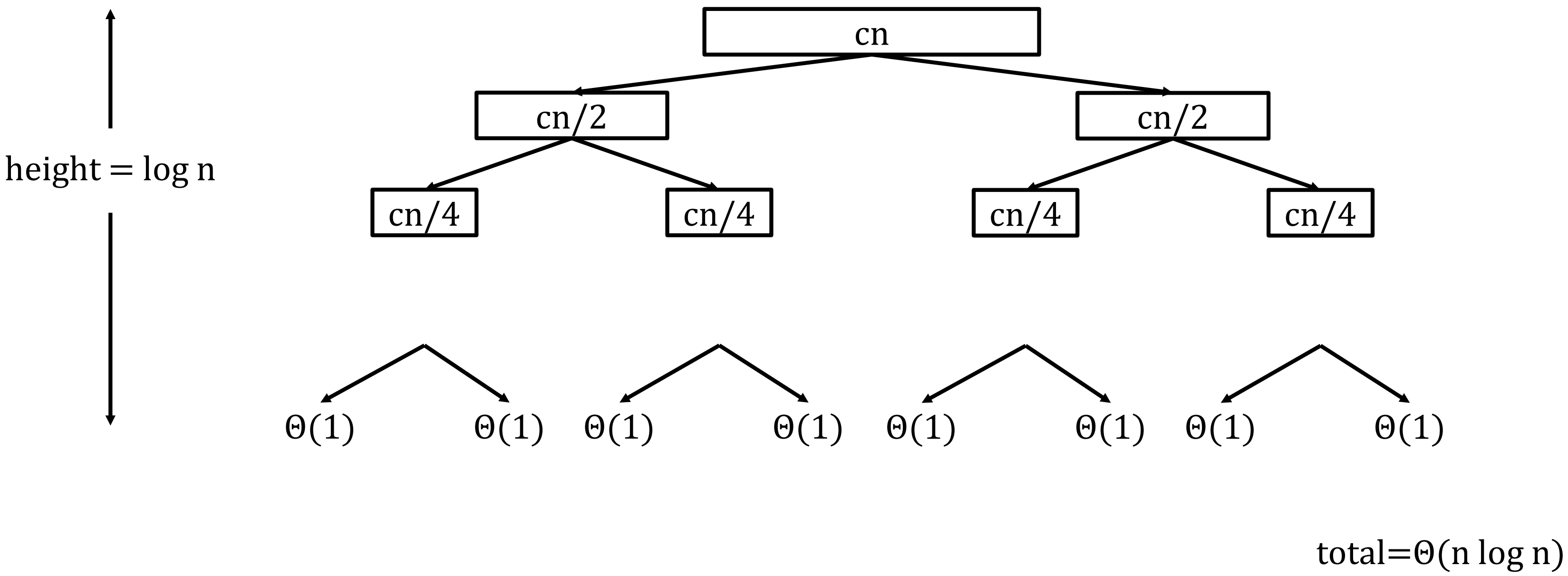
# Analyzing Merge Sort

```
T(n)        def merge_sort():   # A[0..n-1]
Θ(1)            if n==1:
                    done
T(n/2)          recursively sort A[0..⌈n/2-1⌉]
T(n/2)          recursively sort A[⌈n/2⌉..n-1]
Θ(n)            merge 2 sorted lists
```

- $T(n) = \Theta(1)+T(n/2)+T(n/2)+\Theta(n)$
  $T(n) = 2T(n/2) + \Theta(n)$

# Recursion Tree

- Solve T(n)=2T(n/2)+cn, where c>0 is constant



# Notes on Proofs

- Algorithm correctness
- Loop Invariance – invariant variable
  - Initialization: True prior to the first iteration of the loop
  - Maintenance: True in initialization, remains true before next iteration
  - Termination: Loop terminates, the invariant property helps show algorithm correctness
- Proof by induction
- Proof by contradiction

# Example Proof by Induction

- Show that $\forall n \in \mathbb{Z}^+ \; 1+2+..+n = n(n+1)/2$
- **Initial step**
  Verify that P(1) is true, where P(n)=n(n+1)/2
  P(1)=1(1+1)/2 $\Longrightarrow$ clearly true
- **Inductive step**
  Assume P(k) is true and show P(k+1) is true
  Show P(k+1) = 1+2+..+k+(k+1) = (k+1)(k+2)/2
  $\Longrightarrow$ P(k)+(k+1) = k(k+1)/2 + (k+1)
  $\Longrightarrow$ P(k+1) = (k+1)(k/2 + 1) = (k+1)(k+2)/2
  QED "Quod Erat Demonstrandum"

# Example Proof by Contradiction

- Show that $\forall n \in \mathbb{Z}$ if $n^2$ is odd then $n$ is odd
- **Assume opposite of the statement is true**
  i.e. $\exists n$ such that $n^2$ is odd while $n$ is even
  n=2k where $k \in \mathbb{Z}$
  $n^2 = (2k).(2k)$
  Clearly, $n^2$ is even $\Longrightarrow$ Contradiction
  *Proof complete*