CS146: Data Structures and Algorithms
Lecture 1

INTRODUCTION

INSTRUCTOR: KATERINA POTIKA
CS SJSU

Basic Data Structures

Data Structure	Search	Insertion	Deletion	Traversal
Array				
Ordered array				
Linked list				
Ordered linked list				
Binary tree (average)				
Binary tree (worst case)				
Balanced tree (average and worst case)				
Hash table				

Efficiency

- Same problem can have different algorithms that solve them
- Example:

  Sorting problem (sort n numbers)

Algorithm for sorting	Worst time
Insertion Sort	$c_1 n^2$
Merge Sort	$c_2 n \, logn$

Difference in running time

- Sort 10 million numbers use:

1. Insertion Sort: with 10 billion instructions/sec machine and running time (instructions) $2n^2$
   $$\frac{2(10^7)^2 \; instructions}{10^{10} \; instructions/sec} = 20.000 \text{ sec } (5.5 \text{ h})$$
2. Merge Sort: with 10 million instructions/sec and running time 50nlogn
   $$\frac{50 * 10^7 log 10^7 \quad instructions}{10^7 \; instructions/sec} = 1163 \text{ sec } (20 \text{ min})$$

Asymptotic Performance

5

- In this course, we care most about *asymptotic performance*
  - How does the algorithm behave as the problem size gets very large?
    - Running time
    - Memory/storage requirements
    - Bandwidth/power requirements/logic gates/etc.

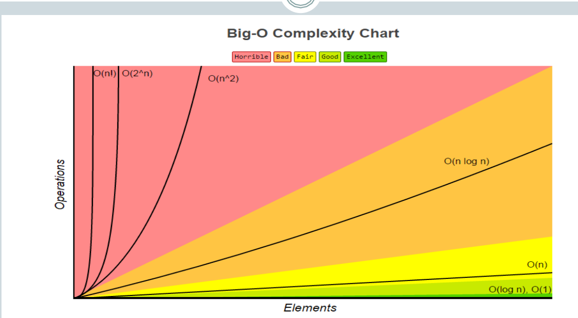
Asymptotic Notation

6

- Intuitive feel for asymptotic (big-O) notation:
  - *What does $O(n)$ running time mean? $O(n^2)$? $O(n \lg n)$?*
  - *How does asymptotic running time relate to asymptotic memory usage?*
- We will define this notation more formally and completely

Practical Complexity

7



bigocheatsheet.com

Analysis of Algorithms

8

- Analysis is performed with respect to a computational model
- We will usually use a generic uniprocessor random-access machine (RAM) – all operations cost const
  - All memory equally expensive to access
  - No concurrent operations
  - All reasonable instructions take unit time
    - Except, of course, function/method calls
  - Constant word size
    - Unless we are explicitly manipulating bits

Input Size

9

- Time and space complexity
  - This is generally a function of the **input size**
    - E.g., sorting, multiplication
  - How we characterize input size depends:
    - Sorting: number of input items
    - Multiplication: total number of bits
    - Graph algorithms: number of nodes & edges
    - Etc

Running Time

10

- Number of primitive steps that are executed
  - Except for time of executing a function/method call most statements roughly require the same amount of time
    - y = m * x + b
    - c = 5 / 9 * (t - 32)
    - z = f(x) + g(y)

Analysis

11

- Worst case
  - Provides an upper bound on running time
  - An absolute guarantee
- Average case
  - Provides the expected running time
  - Very useful, but treat with care: what is "average"?
    - Random (equally likely) inputs
    - Real-life inputs

Worst vs Average case Analysis

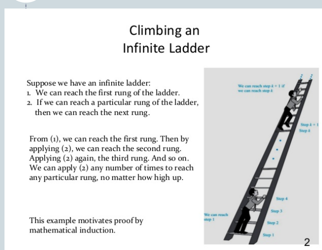
12

Worst case running time analysis

1. Gives an upper bound on the running time for any input (guarantee)
2. For some algorithms, worst case occurs fairly often
3. Average case as bad as worst case

Review: Induction

- Suppose
  - $S(k)$ is true for fixed constant k
    - Often k = 0
  - $S(n) \Rightarrow S(n+1)$ for all n >= k
- Then $S(n)$ is true for all n >= k



Proof By Induction

- Claim: $S(n)$ is true for all n >= k
- Basis:
  - Show formula is true when n = k
- Inductive hypothesis:
  - Assume formula is true for an arbitrary n
- Step:
  - Show that formula is then true for n+1

Induction Example: Gaussian Closed Form

- Prove $1 + 2 + 3 + \dots + n = n(n+1) / 2$
  - Basis:
    - If n = 0, then $0 = 0(0+1) / 2$
  - Inductive hypothesis:
    - Assume $1 + 2 + 3 + \dots + n = n(n+1) / 2$
  - Step (show true for n+1):

$1 + 2 + \dots + n + n+1 = (1 + 2 + \dots + n) + (n+1)$

$= n(n+1)/2 + n+1 = [n(n+1) + 2(n+1)]/2$

$= (n+1)(n+2)/2 = (n+1)(n+1 + 1) / 2$

Induction Example: Geometric Closed Form

- Prove $a^0 + a^1 + \dots + a^n = (a^{n+1} - 1)/(a - 1)$ for all $a \neq 1$
  - Basis: show that $a^0 = (a^{0+1} - 1)/(a - 1)$

$a^0 = 1 = (a^1 - 1)/(a - 1)$

  - Inductive hypothesis:
    - Assume $a^0 + a^1 + \dots + a^n = (a^{n+1} - 1)/(a - 1)$
  - Step (show true for n+1):

$a^0 + a^1 + \dots + a^{n+1} = a^0 + a^1 + \dots + a^n + a^{n+1}$

$= (a^{n+1} - 1)/(a - 1) + a^{n+1} = (a^{n+1+1} - 1)/(a - 1)$

Induction

- We've been using *weak induction*
- *Strong induction* also holds
  - Basis: show S(0)
  - Hypothesis: assume S(k) holds for arbitrary k <= n
  - Step: Show S(n+1) follows
- Another variation:
  - Basis: show S(0), S(1)
  - Hypothesis: assume S(n) and S(n+1) are true
  - Step: show S(n+2) follows