Design and Analysis of Algorithms

Introduction

Prof. Wilson Rivera

Spring 2018

Faster Algorithm vs. Faster CPU

- A faster algorithm running on a slower machine will always win for large enough instances
  - Suppose algorithm S1 sorts n keys in $2n^2$ instructions
  - Suppose computer C1 executes 1 billion instruc/sec
    - When n = 1 million, takes 2000 sec
  - Suppose algorithm S2 sorts n keys in $50n\log_2 n$ instructions
  - Suppose computer C2 executes 10 million instruc/sec
    - When n = 1 million, takes 100 sec

Modeling the Real World

- Cast your application in terms of well-studied abstract data structures

Concrete	*Abstract*
arrangement, tour, ordering, sequence	permutation
cluster, collection, committee, group, packaging, selection	subsets
hierarchy, ancestor/descendants, taxonomy	trees
network, circuit, web, relationship	graph
sites, positions, locations	points
shapes, regions, boundaries	polygons
text, characters, patterns	strings

Real-World Applications

- Hardware design: VLSI chips
- Compilers
- Computer graphics: movies, video games
- Routing messages in the Internet
- Searching the Web
- Distributed file sharing
- Computer aided design and manufacturing
- Security: e-commerce, voting machines
- Multimedia: CD player, DVD, MP3, JPG, HDTV
- DNA sequencing, protein folding
- and many more!

Some Important Problem Types

- Sorting
  - a set of items
- Searching
  - among a set of items
- String processing
  - text, bit strings, gene sequences
- Graphs
  - model objects and their relationships
- Combinatorial
  - find desired permutation, combination or subset
- Geometric
  - graphics, imaging, robotics
- Numerical
  - continuous math: solving equations, evaluating functions

Algorithm Design Techniques

- Brute Force & Exhaustive Search
  - follow definition / try all possibilities
- Divide & Conquer
  - break problem into distinct subproblems
- Transformation
  - convert problem to another one
- Dynamic Programming
  - break problem into overlapping subproblems
- Greedy
  - repeatedly do what is best now
- Iterative Improvement
  - repeatedly improve current solution
- Randomization
  - use random numbers

This Course

- Cover a variety of fundamental algorithm design and analysis techniques as applied to a number of basic problems
  - Organize by technique
- In the other direction, study some lower bounds, indicating inherent limitations for finding efficient algorithms
  - including NP-completeness
- Learn about undecidability: some problems are unsolvable