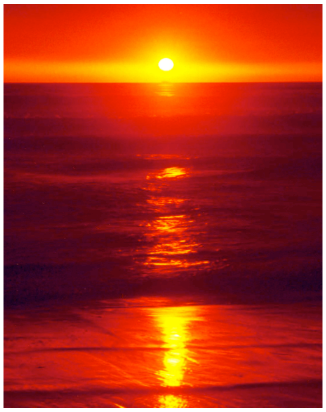


Course Review & A Few Unanswered Questions

Lecture 26
CS211 – Summer 2007

Announcements

- Final Exam
  - Tuesday August 7, 8:00-10:00 am, this room
- A5 due Sunday 11:59PM
  - A4 regrade requests due tonight 11:59PM
  - A5 will be graded by Tuesday morning
  - You must file A5 regrade requests by Tuesday 11:59PM
- Remember: Final is worth 25% of your grade, A5 is worth 10%
  - Balance studying for the final with working on A5!
  - That A5 is due Sunday doesn't imply you have to work on it until then!
- Check the website for office & consulting hours this weekend

Announcements

- Review problem sets on the website, along with solutions
- Remember to fill out course evaluations on-line
  - They're kept strictly anonymous. We see aggregated results only.
- Jealous of the glamorous life of a CS consultant?
  - We're recruiting consultants for CS100 and CS211
  - Fill out an application in 303 Upson Hall

Course Overview

- Programming concepts
  - We use Java, but the goal is to understand the ideas rather than to become a Java expert
  - Recursion
  - Object-Oriented Programming
  - Interfaces
  - Graphical User Interfaces (GUIs)
- Data structure concepts
  - The goal here is to develop skill with a set of tools that are widely useful
  - Induction
  - Asymptotic analysis (big-O)
  - Arrays, Trees, and Lists
  - Searching & Sorting
  - Stacks & Queues
  - Priority Queues
  - Sets & Dictionaries
  - Graphs

Programming Concepts

- Recursion
  - Stack frames
  - Exceptions
- Object-oriented programming
  - Classes and objects
  - Primitive vs. reference types
  - Dynamic vs. static types
  - Subtypes and Inheritance
    - Overriding
    - Shadowing
    - Overloading
    - Upcasting & downcasting
  - Inner & anonymous classes
- Interfaces
  - Type hierarchy vs. class hierarchy
  - The Comparable interface
  - Iterators & Iterable
- GUIs
  - Components, Containers, & Layout Managers
  - Events & listeners

Data Structure Concepts

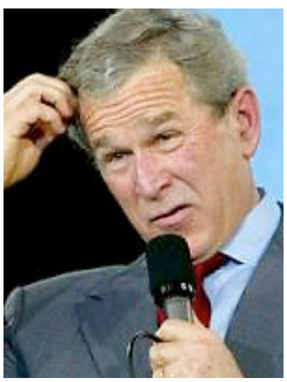
- Induction
- Grammars & parsing
- Asymptotic analysis (big-O)
  - Solving recurrences
  - Lower bounds on sorting
- Basic building blocks
  - Arrays
  - Lists
    - Singly- and doubly-linked
  - Trees
    - Binary Search Trees (BSTs)
- Searching
  - Linear- vs. binary-search
- Sorting
  - Insertion-, Selection-, Merge-, Quick-, and Heap-sort
- Useful ADTs (& implementations)
  - Stacks & Queues
    - Arrays & lists
  - Priority Queues
    - Heaps
    - Array of queues
  - Sets & Dictionaries
    - Bit vectors (for Sets)
    - Arrays & lists
    - Hashing & Hashtables
    - BSTs (& balanced BSTs)
  - Graphs...

Overview of Graphs

- Mathematical definition of a graph (directed, undirected)
- Representations
  - Adjacency matrix
  - Adjacency list
- Topological sort
- Coloring & planarity
- Searching (BFS & DFS)
- Dijkstra's shortest path algorithm
- Minimum Spanning Trees (MSTs)
  - Prim's algorithm (growing a single tree)
  - Kruskal's algorithm (build a forest by adding edges in order)

Where to go from here…

- CS 212: Java practicum
  - 1 credit, work on a major programming projects
- CS 113: C programming
  - 1 credit S/U, only 1 month
- CS 213: C++ programming
  - 2 credits S/U
- CS 280: Discrete math
- CS 312: Functional programming, more algorithms and data structures, language features, etc.
- CS 316: Systems programming

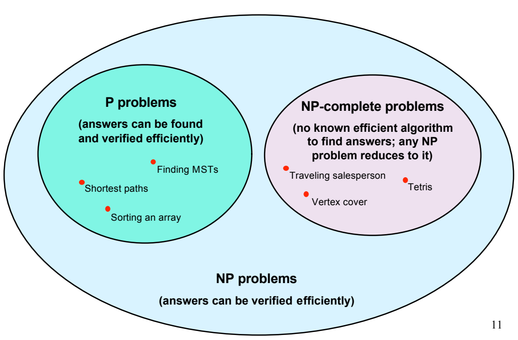


Some Unsolved Problems

The Big Question: Is P=NP?

- P is the class of problems that can be solved in polynomial time
  - These problems are considered *tractable*
  - Problems that are not in P are considered *intractable*
- NP represents problems that, for a *given solution*, the solution can be *checked* in polynomial time
  - But *finding* the solution may be hard
- For ease of comparison, problems are usually stated as yes-or-no questions
- Examples
  - Given a weighted graph G and a bound k, does G have a spanning tree of weight at most k?
    - This is in P because we have an algorithm for the MST with runtime O(m + n log n)
  - Given graph G, does G have a cycle that visits all vertices?
    - This is in NP because, given a possible solution, we can check in polynomial time that it's a cycle and that it visits all vertices

Complexity classes



Current Status: P vs. NP

- It's easy to show that P ⊆ NP
- Is P = NP?
  - Most researchers believe no
    - 61% no
    - 9% yes
    - 22% unsure
    - 8% impossible to prove/disprove
  - But at present, no proof
  - We do have a large collection of *NP-complete problems*
    - If *any* NP-complete problem has a polynomial time algorithm, then they *all* do
- A problem B is *NP-complete* if
  - is in NP
  - any other problem in NP reduces to it efficiently
- Thus by making use of an *imaginary* fast subroutine for B, any problem in NP could be solved in polynomial time
  - the Boolean satisfiability problem is NP-complete [Cook 1971]
  - many useful problems are NP-complete [Karp 1972]
  - By now thousands of problems are known to be NP-complete

Some NP-Complete Problems

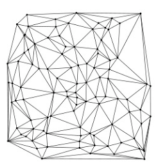
- Graph coloring: Given graph G and bound k, is G k-colorable?
- Planar 3-coloring: Given planar graph G, is G 3-colorable?
- Traveling salesperson: Given weighted graph G and bound k, is there a cycle of cost ≤ k that visits each vertex exactly once?
- Hamiltonian cycle: Give graph G, is there a cycle that visits each vertex exactly once?
- Knapsack: Given a set of items i with weights w_i and values v_i, and numbers W and V, does there exist a subset of at most W items whose total value is at least V?
- What if you really *need* an algorithm for an NP-complete problem?
  - Some special cases can be solved in polynomial time
    - If you're lucky, you have such a special case
  - Otherwise, once a problem is shown to be NP-complete, the best strategy is to start looking for an approximation
- For a while, a new proof showing a problem NP-complete was enough for a paper
  - Nowadays, no one is interested unless the result is somehow unexpected

A practical application: Cryptography

- RSA: a popular public-key cryptography algorithm
- Everyone has their own *public* and *private* keys
  - Private keys are kept secret
  - Public keys are distributed freely
- To send someone an encrypted message, you encode it with their public key. They decrypt it with their private key.
- Private key and public keys are related
  - Prime factors of a very large number (100's of digits long)
- Finding prime factors is in NP
  - Given prime factors and a number, it's easy to verify the product
- But no polynomial time algorithm is known to do the factoring
  - RSA security depends on this!

Complexity of Bounded-Degree Euclidean MST

- The Euclidean MST (Minimum Spanning Tree) problem:
  - Given n points in the plane, determine the MST
  - Can be solved in O(n log n) time by first building the Delaunay Triangulation



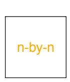
- Bounded-degree version:
  - Given n points in the plane, determine the MST where each vertex has degree ≤ d
    - Known to be NP-hard for d=3 [Papadimitriou & Vazirani 84]
    - O(n log n) algorithm for d=5 or greater
      - Can show Euclidean MST has degree ≤ 5
    - Unknown for k=4

Complexity of Euclidean MST in R^d

- Given n points in dimension d, determine the MST
  - Is there an algorithm with runtime close to the Ω(n log n) lower bound?
- Can solve in time O(n log n) for d=2
- For large d, it appears that runtime approaches $O(n^2)$
- Best algorithms for general graphs run in time linear in m = number of edges
  - But for Euclidean distances on points, the number of edges is n(n-1)/2

$O(n^2)$ Time for X+Y Sorting?

How long does it take to a sort an n-by-n table of numbers?



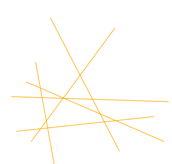
- $O(n^2 \log n)$ because there are n^2 numbers in the table
- What if it's an *addition table*?
  - Shouldn't it be easier to sort than an arbitrary set of n^2 numbers?

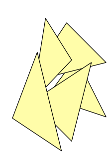
+	1	3	5	8
2	3	5	7	10
10	11	13	15	18
12	13	15	17	20
14	15	17	19	22

- There is a technique that uses just $O(n^2)$ *comparisons* [Fredman 76]
  - But it uses $O(n^2 \log n)$ time to decide *which* comparisons to use [Lambert 92]
- This problem is closely related to the problem of sorting the vertices of a line arrangement

3SUM in Subquadratic Time?

- Given a set of n integers, are there three that sum to zero?
  - $O(n^2)$ algorithms are easy (e.g., use a hashtable)
  - Are there better algorithms?

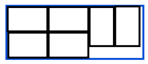




- This problem is closely related to many other problems [Gajentaan & Overmars 95]
  - Given n lines in the plane, are there 3 lines that intersect in a point?
  - Given n triangles in the plane, does their union have a hole?

Pallet Loading Problem

- Can n small rectangles of size a x b be packed into a large rectangle of size A x B?



  - No known polynomial time algorithm
  - Unknown even whether this is in NP
  - There have been several published “proofs” that it is in NP, but they were all later found to be invalid