CS/ENGRD 2110
Object-Oriented Programming and Data Structures

Spring 2011
Thorsten Joachims

Lecture 25:
Review and Open Problems

Course Overview

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

Operational Knowledge

2

Programming Concepts

- 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
- Recursion
  - Divide and conquer
  - Stack frames
  - Exceptions
- Interfaces and Types
  - Type hierarchy vs. class hierarchy
  - Generic types
  - The Comparable interface
  - Design patterns: Iterator, Observer (GUI), etc.
- GUIs
  - Components, Containers, Layout Managers
  - Events & listeners
- Concurrency and Threads
  - Locking
  - Race conditions
  - Deadlocks

3

Data Structure Concepts

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

4

Data Structure Concepts

- Graphs
  - Mathematical definition of a graph (directed, undirected)
  - Representations
    - Adjacency matrix
    - Adjacency list
  - Topological sort
  - Coloring
  - Searching (BFS & DFS)
  - Shortest paths
  - Minimum Spanning Trees (MSTs)
    - Prim's algorithm
    - Kruskal's algorithm

5

What else is there in CS?

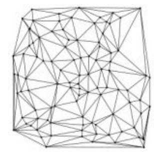
- CS2110 + Math is sufficient prerequisite for many 4000-level Computer Science classes!
- Areas of Computer Science:
  - Artificial Intelligence
  - Network Science
  - Software Engineering
  - Computer Graphics
  - Natural Language Processing
  - Programming Languages
  - Security and Trustworthy Systems
  - Databases
  - Operating Systems
  - Theory of Computing

6

Some Unsolved Problems

Complexity of Bounded-Degree Euclidean MST

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



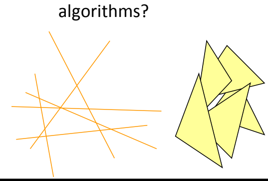
- Bounded-degree version:
  - Given n points in the plane, determine a MST where each vertex has degree ≤ d
    - Known to be NP-hard for d=3 [Papadimitriou & Vazirani 84]
    - O(n log n) algorithm for d=5 or greater
      - Can show Euclidean MST has degree ≤ 5
    - Unknown for d=4

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 O(n log n)?
  - 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 m = n(n-1)/2

3SUM in Subquadratic Time?

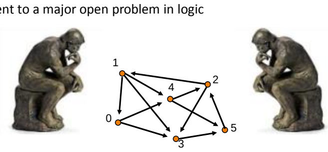
- Given a set of n integers, are there three that sum to zero?
  - $O(n^2)$ algorithms are easy (e.g., use a hashtable)
  - Are there better algorithms?



- This problem is closely related to many other “3SUM-Hard” 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?

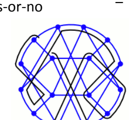
Winning Strategies for the Parity Game?

- Played on a directed graph with nodes 0, 1, 2, ..., n−1
  - Start with a pebble on node 0
  - Players Steven and Todd alternately choose edges along which to push the pebble
  - They play forever...
- Who wins?
  - Steven wins if the least-numbered vertex visited infinitely often is even
  - Todd wins if the least-numbered vertex visited infinitely often is odd
- It is known that for any graph, either Steven or Todd has a winning strategy – but can you determine which?
  - Equivalent to a major open problem in logic



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



- Example 1:
  - 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)
- Example 2:
  - Given graph G, does G have a Hamiltonian cycle (a simple 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 exactly once

Current Status: P vs. NP

- It's easy to show that P ⊆ NP
- Most researchers believe that P ≠ NP
  - 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
  - it 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

15

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 at least 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

16

Final Exam

- Time and Place
  - Thursday, May 12
  - 2:00pm - 4:30pm
  - Baker Laboratory 200 (BKL200)
- Review Session
  - Wednesday, May 11
  - 3:30pm – 5:00pm
  - Kimball B11
- Exam Conflicts
  - Email me TODAY!
- Office Hours
  - Continue until final exam
  - But there may be time changes…

17

Course Evaluations (2 Parts)

- CourseEval
  - Worth 0.5% of your course grade
  - Anonymous
    - We get a list of who completed the course evaluations and a list of responses, but no link between names & responses
  - http://www.engineering.cornell.edu/CourseEval
- CMS Survey
  - Worth another 0.5% of your course grade
  - Not anonymous
    - But no confidential questions

18

Becoming a Consultant

- Jealous of the glamorous life of a CS consultant?
  - We're recruiting next-semester consultants for CS1110 and CS2110
  - Interested students should fill out an application, available in 303 Upson

19

Good luck on the final!

Thanks for an enjoyable semester!

Have a great summer!

☺