Course Review & A Few Unanswered Questions

Lecture 26
CS2110 – Fall 2011
Announcements

• Final Exam
  ▪ Monday, Dec 12
  ▪ 9–11:30am
  ▪ Barton Hall Main Floor West

• Review Session
  ▪ Sunday, Dec 11
  ▪ 7:30 - 9pm and 9 - 10:30pm
  ▪ Upson B17
  ▪ Both sessions the same

• For exam conflicts:
  ▪ Notify Michelle Eighmey today
  ▪ You must provide
    ◆ Your entire exam schedule
    ◆ Include the course numbers

• Definition of exam conflict:
  ▪ Two exams at the same time, or
  ▪ Three+ exams within 24 hours

• A5 due Sunday, Dec 4, 11:59pm
  ▪ Sorry, no more extensions
Announcements

• Check the course website for additional announcements as the final exam approaches
  ▪ Consulting ends this week
  ▪ Office hours continue until Final Exam
    ✷ There may be changes (TAs have exams, too)
    ✷ Any changes will be announced on the course website

• 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

Check the course website for additional announcements as the final exam approaches.
Course Evaluations

• **Worth one assignment point**
  - Will count as 1% of your course grade
  - This is a regular point, *not* a bonus point
  - Anonymity
    - 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/
  - This link also appears on the CS2110 announcements page
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 Heapsort

- 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)
Some Unsolved Problems
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 a MST where each vertex has degree $\leq d$
    ▪ 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 $\leq 5$
    ▪ Unknown for $d = 4$
Complexity of Euclidean MST in $\mathbb{R}^d$

- Given $n$ points in dimension $d$, determine the MST
  - Is there an algorithm with runtime close to the $\Omega(n \log n)$ 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 $m = \text{number of edges}$
  - But for Euclidean distances on points, the number of edges is $n(n-1)/2$
O(n²) Time for X+Y Sorting?

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

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>13</td>
<td>15</td>
<td>18</td>
</tr>
<tr>
<td>12</td>
<td>13</td>
<td>15</td>
<td>17</td>
<td>20</td>
</tr>
<tr>
<td>14</td>
<td>15</td>
<td>17</td>
<td>19</td>
<td>22</td>
</tr>
</tbody>
</table>

- O(n² log n) because there are n² numbers in the table

- What if it’s an addition table?
  - Shouldn’t it be easier to sort than an arbitrary set of n² numbers?

- There is a technique that uses just O(n²) comparisons [Fredman 76]
  - But it uses O(n² 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?
3-Colorability of Great-Circle Graphs?

- Build a graph by drawing great-circles on a sphere
  - Create a vertex for each intersection
  - Assume no three great circles intersect in a point
- Is the resulting graph 3-colorable?
- All arrangements for up to 11 great circles have been verified as 3-colorable

- For general circles on the sphere (or for circles on the plane) the graph can require 4 colors
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 ...
- 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

• 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 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 $P \subseteq NP$
• Most researchers believe that $P \neq NP$
  ▪ 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
  1. it is in $NP$
  2. any other problem in $NP$ reduces to it efficiently

• With a 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 $\leq k$ that visits each vertex at least once?
- Hamiltonian cycle: Given 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
Good luck on the final!

Thanks for an enjoyable semester!

Have a great winter break!

😊