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

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

• Interfaces
  • Type hierarchy vs. class hierarchy
  • The Comparable Interface
  • Iterators & Iterables

• GUIs
  • Components, Containers, & Layout Managers
  • Events & Listeners

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

• 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

http://www.engineering.cornell.edu/CourseEval/
  • This link also appears on the CS2110 announcements page
Data Structure Concepts

- Induction
- Grammars & parsing
- Asymptotic analysis (big-O)
  - Solving recurrences
  - Lower bounds on sorting
- Basic building blocks
  - Arrays
  - Lists
    - Single- and doubly-linked
  - Trees
    - Binary Search Trees (BSTs)
- Searching
  - Linear vs. binary search
- Sorting
  - Linear vs. binary search
  - Insertion, Selection, Merge, Quick, and Heapsort

Useful ADTs (& implementations)

- Stacks & Queues
- Priority Queues
- Heaps
- 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)

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 \( O(n^2) \)

Some Unsolved Problems

- Complexity of \( X+Y \) Sorting?
- How long does it take to sort an \( n \)-by-\( n \) table of numbers?

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

<table>
<thead>
<tr>
<th>n-by-n</th>
<th>+</th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>13</td>
<td>15</td>
<td>17</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>13</td>
<td>15</td>
<td>17</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>15</td>
<td>17</td>
<td>19</td>
<td>22</td>
<td></td>
</tr>
</tbody>
</table>

- There is a technique that uses just \( O(n) \) comparisons [Friedman 77]
- But it uses \( O(n^2 \log n) \) time to decide which comparisons to use [Lambert 80]
- This problem is closely related to the problem of sorting the vertices of a line arrangement
Most researchers believe it's easy to show that $P \neq \text{NP}$, but at present, no proof has been found. 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.

The Big Question: Is P=NP?

P is the class of problems that can be solved in polynomial time.

- There are problems considered intractable.
- 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 P because we have an algorithm for the MST with runtime $O(m \cdot \log n)$.
- Given graph $G$, does $G$ have a Hamiltonian cycle (a simple cycle that visits all vertices)?
- This is NP because, given a possible solution, we can check in polynomial time that it's a cycle and it visits all vertices exactly once.

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!
😊