Data Structures and Algorithms II: Introduction

Sattiraju Prabhakar
CS560: Class I
Wichita State University

Topics

- Introduction to Course
- Introduction to Algorithms
- Formalizing the Ideas

Introduction to Course

Overview

- Each Class + Lab:
  - Instruction
    - Algorithms
    - Implementing Algorithms
  - Tutorials
    - Problem Solving in Class
- Assignments:
  - Take Home Problem Solving: 4
  - Programming: 2
- Exams:
  - Mid Term
  - Final

Algorithm Instruction

- Stages in the Instruction of Algorithms
  - ***Problem***: What is the algorithm supposed to do?
  - ***Examples*:** Can I see how the algorithm works?
  - ***Algorithm:*** Formal Specification: Can I see how the algorithm looks like?
  - ***Analysis:*** Why is the algorithm good or bad?
  - ***Implementation:*** How do I convert the algorithm into a form I can use to write my programs?

Tutorials

- In each class and lab
  - We solve a number of problems together
  - Assessment: Not evaluated
  - Purposes:
    - To learn to apply algorithm on a number of examples
    - To see, clearly how the algorithm works at different stages
    - To clarify any doubts
    - Preparation for for problem solving assignments, quizzes and exams

Implementational Instruction

- To discuss implemental issues for algorithms
  - "C like" pseudo Language is Used
  - Purpose: Issues involved in implementing various parts of an algorithm
  - Prepare for programming assignments
- NOTE: There is no instruction in any programming language
  - There is no instruction in how to write algorithms in a specific programming language

Analysis

- Analyzed
  - ***Informally:***
    - By using one or more examples
  - ***Time complexity:***
    - How much time does it take to run the program for various inputs?
  - ***Space complexity:***
    - How much memory does it take to run the program for various inputs.
  - ***Variation in Inputs:***
    - What is the range of inputs over which my program will work?

Problem Solving Assignments

- Total Number: 4
- These are take home
- When are they given out?
  - After a topic is fully covered in the class
- What are you required to do?
  - You are given a set of problems similar to
    - The ones we solve during tutorials
    - Those present in the text book
  - You need to solve the problems and submit solutions
- Typically, you are given one week for submissions
- They will be marked and returned to you along with
  - a marking scheme
  - instructor's solutions
- These are done by individuals, no groups

Programming Assignments

- Total Number: 2
- You are required to implement the algorithms
- You can choose any one of the programming languages: C, C++, Java.
- You can form a group of two people.
- Each group submits one solution.
- Typically, two weeks are given for submissions.
- Masters groups demo their programs in the class

Exams

- Mid Term:
  - Problem Solving Type
  - The problems are similar to the ones in assignments and tutorials
- Final Exam:
  - Problem Solving Type
  - The problems are similar to the ones in assignments, tutorials and mid term exam
  - Comprehensive

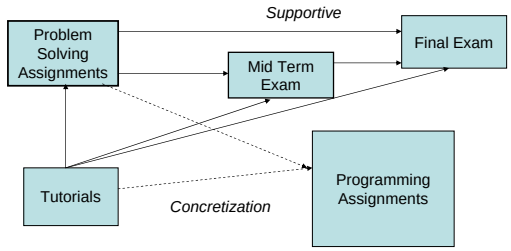
Attendance

- Attendance is not assessed
- If you fail to attend a quiz or exam, you will automatically get a "0" mark.
- There are no alternate exams
- Mid Term Exam:
  - October 12th, 2005
- The final exam:
  - Dec 7th, 2005

Mark Distribution

- Problem Solving Assignments: 30%
- Programming Assignments: 20% (approx)
- Mid Term Exam: 20%
- Final Exam: 30%

Seeing the Links



Syllabus (Tentative)

1. Introduction to Algorithms
2. Analysis and Design of Algorithms
3. Notations
4. Recurrences
5. Heapsort
6. Greedy Algorithms
7. Quicksort
8. Sorting in Linear Time
9. Binary Search Trees
10. Red-black Trees
11. Hash Tables
12. Elementary Graph Algorithms
13. Minimum Spanning Trees

Why there are no time bounds?

- Instruction and assessment are understanding bound
- Typically, each assessments (problem solving) is linked to a topic from syllabus.

Constraints

- The due dates for assignments are strictly followed
- Plagiarism is strongly discouraged
- The alternate exams will be organized according to University policies

How to contact your instructor?

- Address:
  - Room 241, Jabara Hall
  - Telephone: 978-3928
  - Email: prabhakar@cs.twsu.edu
- Office Hours:
  - MW: 4:30PM – 5:30PM
  - Other Times: By prior appointment only (send email)
- Before you ask a question, first find whether that information is already present in course outlines or not.

Your GTA

- Ayodele Fakalujah
- Room no : ?? JH
- Ph. No : ?
- Email: ?

Where is the stuff?

- Blackboard: http://blackboard.wichita.edu
- Your course outlines will be present there
- Class notes: PDF files of Power Point Presentations are added one hour before each class.
- Assignments in PDF format
- Announcements: Regularly posted

Where is the stuff for class_n?

- The class notes is given out as files.
- Each file is for a topic.
- The topic may be covered over several classes.

Course Evaluation and Feedback

- Feedback from you is important to make this course more accessible.
- Feedback is collected mainly using:
  - Class discussions
  - Assignments, Exams

Tasks for Today

- Get acquainted with Blackboard
- Link the email address, you normally use, to your email address on the blackboard.
  - NOTE: If I need to contact you immediately, I will use the email on the blackboard.
- Buy the text book: “Introduction to Algorithms” by Cormen et al.
- Fill up the slip given to you with your name and a number
  - Add the last four digits of your shocker card with last four digits of your phone number, and give the last four digits
    - Example: 9877 + 9418 → 9295
  - This is the number against which all your marks will appear
  - Remember this number and don’t give it anyone.

Introduction to Algorithms

Consider a problem…

- Sorting Problem:
  - Write a program informally that converts the Input1 into Output1
    - Input1: [7, 1, 3, 5, 2]
    - Output1: [1, 2, 3, 5, 7]

Consider another sorting problem

- New Sorting Problem:
  - Input2: [200, 150, 350, 550, 300]
  - Output2: [150, 200, 300, 350, 550]
- Questions:
  - Is it possible to write a single program for both the programs?
  - How can we write such a program?

Consider another sorting problem

- New Sorting Problem:
  - Input3: [2.12, 0.56, 0.35, 0.002, 3.5, 5.6, 21.2, 0.55]
  - Output3: [0.002, 0.35, 0.55, 0.56, 2.12, 3.5, 5.6, 21.2]
- The input length is different from the previous ones. Can the program work?

Consider another sorting problem

- Input4: [5, 1, 5, 8, 32 ,12, 9, 76, 45, 12, 90, 56, 14, 65, 78, 09, 4, 42, 78, 12, 89, 34, 67, 78, …] (10, 000 entries)
- Output4: […]
- Question: Can our program terminate at a reasonable time? Or will it continue forever?

Let us figure out how to solve the sorting problem

– Input1: [7, 1, 3, 5, 2]
– Output1: [1, 2, 3, 5, 7]

Now you try it!

– Input2: [200, 150, 350, 550, 300]
– Output2: [150, 200, 300, 350, 550]

Let us write some conclusions

- There are some stages identifiable for programming
- The stages are about
  - Changing some data based on position
  - Test the data to satisfy a condition
- These stages produce output that passes the same test.
- They all the same kind of problem solving: *sorting*
- The time for computation seems to depend upon the length of the input.

Here is the *algorithm* informally: As a generalized procedure

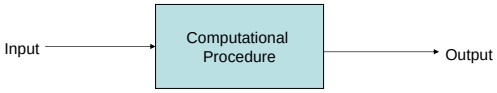
- Step1: Select an element to move from input sequence
- Step2: Move the element to a new position in the input sequence
- Step3: Test if there is an ordering in the input sequence
- Step4: If not, then go to step1

Differences Between A Program and an Algorithm as a generalized procedure

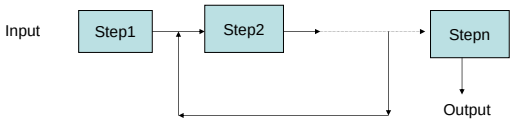
- Programs (if you write them for an input and output) work only for that input and output
- Algorithms work on several sets of inputs and outputs
- The steps can have mathematical properties
- The steps in algorithm are non-specific to any input or output values
- The algorithm can be seen as satisfying a mathematical transformation

Formalizing the Ideas

Algorithm



Algorithm has a sequence of steps



What is an Algorithm?

- It is any *well-defined* computational procedure that takes some value, or set of values, as **input** and produces some value, or set of values, as **output**.
- It is a tool for solving a well-specified **computational problem**.

Algorithm Specification for Sorting Problem

- Input: A sequence of numbers <a1, a2, …, an>
- Output: An ordered sequence of input numbers <a1', a2', …an'> where a1' ≤ a2' ≤ …an'

Instances vs Generic Problems

- A problem (Input, Output) is ***generic***. It does not have specific values.
  - Example: (<a1, a2, …, an>, <a1' ≤ a2' ≤ …an'>)
- An instance of a problem: It has specific values.
  - Example: (< 7, 1, 3, 5, 2>, <1, 2, 3, 5, 7>)

Algorithm and Problem

- An algorithm is a ***solution*** to the (generic) problem.
  - That is, given any input, it can come up with a unique output.
- ***Correct Algorithm***:
  - For every instance of the input (of the problem) it finds the correct output.
- Correct algorithm solves the problem.

Real World Examples

- Internet: Each query requires thousands of servers to contacted. Each server may have thousands of files. Each file may have thousands of words to search. The search is: 1000 * 1000 * 1000 = 10^9
- Electronic Commerce: Searching data bases for matching items