Discrete Mathematics: Introduction

Slides by: Christopher M. Bourke
Instructor: Berthe Y. Choueiry

Fall 2007

Computer Science & Engineering 235
Introduction to Discrete Mathematics
cse235@cse.unl.edu

Notes

Computer Science & Engineering 235

Discrete Mathematics

- Roll
- Syllabus
- Lectures: M/W/F 1:30 – 2:20 (Avery 109)
- Recitations: Tuesdays 5:30 – 6:20 (Avery 108)
- Office hours:
  - Instructor: M/W 2:30 – 3:30 (Avery 123B)
  - TA: Chris Bourke: 1:00 – 2:00 (Wed in Avery 123C and Thu in Avery 13A)
- Must have cse account
- Must use webhandin
- Bonus points: report bugs

Notes

Why Discrete Mathematics? I

You have to.

Computer Science is *not* programming.

It is not even Software Engineering.

> *"Computer Science is no more about computers than astronomy is about telescopes."* –*Edsger Dijkstra*

Computer Science is *problem solving*.

Notes

Why Discrete Mathematics? II

Mathematics is at the heart of problem solving.

Often, even *defining* a problem requires a level of mathematical rigor.

Competent use and analysis of models/data structures/algorithms requires a solid foundation in mathematics.

Justification for why a particular way of solving a problem is *correct* or *efficient* (i.e., better than another way) requires analysis within a well defined mathematical model.

Notes

Why Discrete Mathematics? III

Abstract thinking is necessary to applying knowledge.

Rarely will you encounter a problem in an abstract setting (your boss is not going to ask you to solve MST). Rather, it is up to you to determine the proper model of such a problem.

Notes

Scenario I

A limo company has hired you (or your company) to write a computer program to automate the following tasks for a large event.

Task 1 – In the first scenario, businesses request limos and drivers for a fixed period of time (specifying a start-date/time and end-date/time) and charged a flat rate. The program should be able to generate a schedule so that the maximum number of customers can be accommodated.

Notes

Scenario II

Task 2 – In the second scenario, the limo service is considering allowing customers to *bid* on a driver (so that the highest bidder gets a limo/driver when there aren't enough available). The program should thus make a schedule a feasible (i.e., no limo can handle two customers at the same time) while at the same time, maximizing the profit by selecting the highest *overall* bids.

Notes

Scenario III

Task 3 – In a third scenario, a customer is allowed to specify a *set* of various times and bid an amount for the entire event. A driver must choose to accept the entire set of times or reject it all. The scheduler must still maximize the profit.

Notes

Scenario

What's your solution?

How can you *model* such scenarios?

How can you develop algorithms for these scenarios?

How can you justify that that they work? That they actually guarantee an optimal (i.e., maximized profit) solution?

Notes