CS 101
Problem Solving

Lecture 2
Problem Solving

As discussed in the last lecture, our running definition of Computer Science is:

The process of finding solutions to difficult or complex issues by defining a set of steps or instructions to be run by a computer for accomplishing a particular task

So let’s talk about problem solving.
Problem Solving

• Humans have an intuitive sense of what it means to “Solve a Problem”
  ○ If the kitchen is full of dirty dishes, we solve the problem by cleaning them one-by-one
  ○ If the grass in the lawn is too high, we solve the problem by getting out the lawn mower
  ○ If we are hungry, we solve the problem by systematically assembling a meal, and then eating it
Problem Solving

As with most problems, the solution to each of them can be broken down into a series of **steps** that need to be taken to arrive at the solution
Problem Solving - Dishes

Let’s define the “Dishes” problem in more detail:

The Problem: There is a stack of dirty dishes in the right side of the sink

The Ideal Outcome: All dishes are cleaned and in their proper place

The Solution: . . .
1) Turn on the water to the left sink
2) Grab the sponge
3) Put some soap on the sponge
4) As long as dishes remain in the right sink
   a) Grab a dish from the right sink
   b) Scrub it well
   c) Place it in the left sink
5) Grab the drying rag
6) As long as dishes remaining in the left sink
   a) Grab dish from the left sink
   b) Dry it well
   c) Put it in the proper cabinet
Problem Solving - Grass

Let’s define the “Grass” problem in more detail:

The Problem: The grass in the lawn is getting too high

The Ideal Outcome: All grass on the lawn should be no longer than 1 inch

The Solution: . . .
1) Get the lawn mower out of the shed
2) Fill with gasoline
3) Move lawn mower to SW corner of lawn
4) Start mower
5) While there is still more lawn to cover
   a) Push mower to north end
   b) Turn around
   c) Push back to south end
   d) Shift over 2 feet
6) Turn off mower
7) Put mower back in shed
Problem Solving

- These algorithms can also be represented with **decision trees**
- A **decision tree** is a diagram that can be used to represent the sequential steps of an algorithm
- Arrows are used to show which step is next
Problem Solving

How can this be converted into a decision tree?

1) Grab two slices of bread from pantry
2) Grab peanut butter from pantry
3) Grab jelly from the fridge
4) Spread peanut butter on one slice of bread
5) Spread jelly on the other
6) Combine slices
7) EAT!
8) If still hungry, go back to step 1
Problem Solving

Grab two slices of bread from pantry

Grab peanut butter from pantry

Grab jelly from the fridge

Spread peanut butter on one slice of bread

Spread jelly on the other

Combine slices

EAT!

Still hungry?

YES

NO

Done
Converting to Decision Tree

Mixing Concrete...
How can this be converted into a decision tree?

1) Pour/spray 0.75 gallons of water into mixer
2) Cut open 1 bag of concrete
3) Pour bag concrete into mixer
4) Turn on mixer
5) If too stiff
   a) Add more water
6) If too watery
   a) Add more concrete mix
7) Pour concrete out of mixer
Pour/spray 0.75 gallons of water into mixer

Cut open 1 bag of concrete

Pour bag concrete into mixer

Turn on mixer

Concrete too stiff?

- NO

Concrete too watery?

- YES
Add water

- NO
Add concrete

Pour Concrete

Pour it out
Converting to Decision Tree

1) Turn on the water to the left sink
2) Grab the sponge
3) Put some soap on the sponge
4) As long as dishes remain in the right sink
   a) Grab a dish from the right sink
   b) Scrub it well
   c) Place it in the left sink
5) Grab the drying rag
6) As long as dishes remaining in the left sink
   a) Grab dish from the left sink
   b) Dry it well
   c) Put it in the proper cabinet

How can this be converted into a decision tree?
1. Turn on the water
2. Grab the sponge
3. Put some soap on the sponge
4. Still dishes in right sink?
   - Yes: Grab dish from right
     - Scrub it
     - Place on drying rack
   - No: Grab drying towel

5. Still clean dishes in left sink?
   - Yes: Grab dish from left
     - Dry it
     - Put into cabinet
   - No: Done!
Conditions

- Notice that each of these algorithms/diagrams have *conditions*
- Some of the *conditions* have *repetitions* and some do not
- *Conditional statements* are an important concept in algorithm design, as well as programming
In-Class Exercise

- Create an algorithm and a decision-tree for the following
  - Baking N Cookies
Problem Solving

- We just described detailed steps for solving several problems
- Notice that each of them
  - Had a flow of steps. IE, start from the first step, and follow along sequentially
  - Steps were described in great detail
Problem Solving - Algorithm

- The previous example(s) are specific, detailed steps for solving a problem or completing a task
- In computer science, we call a detailed set of steps like this an Algorithm
- Algorithms are a fundamental concepts in computer science, which we will discuss and revisit throughout the semester
Problem Solving - Algorithm

- Computers are not like humans, in that they are not inherently intelligent.
- Computers are machines. Because of this, computers must be told *exactly* how to solve a problem by a human.
- Computers must be told exactly what to do. Thus, to get a computer so solve a problem, we have to write very detailed, specific instructions for it.
- We do this by coming up with **Algorithms** for computers to execute.
Problem Solving - Algorithm

• However, we can’t just give a computer instructions written in English like we came up with before. **Computers don’t speak English.**
Problem Solving - Programming Language

- This is where a **Programming Language** comes into play
- A **Programming language** is a language that computers “understand”
- When we want to tell a computer a set of instructions to use to solve a problem, we have to specify those instruction using a programming language, rather than a human language
Problem Solving - Programming Language

Thus, to *solve problems with computers*, we write sets of instructions (an *Algorithm*) with a *Programming language*, and the computer will do the task.
Thus, to *solve problems with computers*, we write sets of instructions (an *Algorithm*) with a *Programming language*, and the computer will do the task.
Problem Solving - Programming Language

- Throughout the semester, we will learn one (of many) programming languages named *Processing*
- Why Processing?
  - it’s “visual”
  - designed to make interactive graphics easy
  - focused on code that creates images, animations and interactions
Processing

- Processing
  - It’s “domain” is the computer screen
  - A computer screen is a grid of light elements called *pixels*
  - Initially, drawing in Processing is like drawing on graph paper (conceptually)
    - Q: How would we draw a line, a point, anything?
    - A: Specify the coordinates
Do you remember drawing on graph paper?
Let's look back at the cartesian coordinates
  o we'll experiment on the graph paper
  o *go to Elmo*
Do you remember drawing on graph paper?

Let's look back at the cartesian coordinates

- we'll experiment on the graph paper
- go to Elmo
Problem Solving - Programming Language

- Becoming proficient with a programming language takes time, but the reward is great.
- Throughout the semester, you will write several programs in Processing.
Problem Solving - Programming Language

- *Processing* will be the main focus of about ~half of the lectures in this course.
- The other ~half will focus on more *general topics* like algorithms, representing data, computer architecture, networks, the internet, and so on.
Problem Solving - Materials

• Required Materials
  ○ UTDW - Chapter 4
  ○ Computer Science is for everyone (TED video)  (10 mins)
  ○ Algorithms are taking over the world! (TED video)  (11 mins)
  ○ What is an Algorithm? (video)  (5 mins)