Computational Problem Solving

Debswapna Bhattacharya, PhD
Assistant Professor
Computer Science and Software Engineering
Auburn University

Solving problems computationally

- What do you do first?
- Write code? Bad idea!
- Specify the problem
- Come up with multiple strategies
- Design corresponding algorithms (or understand, modify and reuse existing ones)
- Then choose the best

CPS in more detail….

1. Specify the problem well
2. Come up with at least one solution strategy
3. Develop algorithm(s) corresponding to the strategy(ies)
    1. Find and understand existing algorithms that use these strategies to solve the problem
    2. If no such algorithms can be found, find and understand existing algorithms that solve a similar problem, then modify them appropriately to implement your strategies
    3. Design new algorithms
4. Ensure/prove correctness of all algorithms
5. Analyze and compare their performance/efficiency
    1. Theoretically: Using a variety of mathematical tools
    2. Empirically: Code, run and collect performance data
6. Choose the most efficient algorithm to implement

Specifying the problem:
Make it "well-defined"

A **well-defined** problem is a problem specified with its **inputs**, expected **outputs** and **correctness criteria** explicitly stated.

Well-defined problem specification

A well-defined problem specification contains enough information for you to estimate the **inherent computational complexity** of the problem, and to start thinking about its **computational solution** (i.e. **strategies** to solve it, which will lead to algorithms).

The Sorting Problem

(in the ascending order)

- Input: A sequence A of *n* numbers arranged in some order $<a_1, a_2, a_3,\ldots,a_n>$

- Output: A sequence B of n numbers $<b_1, b_2, b_3,\ldots,b_n>$

Correctness criterion: B is a permutation of A such that

$b_1 \leq b_2 \leq b_3 \leq,\ldots, \leq b_n$

What is the inherent complexity of a problem?

A **lower-bound** estimate of the number of computational steps **any** computational solution to the problem (i.e. algorithm) will have to execute in order to solve that problem.

What is the inherent complexity of sorting?

At least n basic operations – i.e., $\Omega(n)$

Why? – you have to *at least* look at all the numbers!

Inherent problem complexity

vs.

Algorithm complexity

Inherent problem complexity can provide you with a sense of how fast/efficient an algorithm to solve the problem can possibly be...

For example, no sorting algorithm can sort n numbers in less than n steps or basic operations.

The "search" problem: find an item in a bag of n items

- What does a well-defined problem specification for "searching for a number x in a set of n numbers" look like?
- What is its inherent complexity?
- How about the Binary Search algorithm?
- Binary Search is solving a problem with a *slightly different* specification.
- What is a well-defined problem specification for Binary Search?

Computational Strategies

Precursors of algorithms

Stated in plain English with no notation

Details can be omitted

Strategy vs. Algorithm

What is the difference?

Strategy: In English, not detailed

Algorithm: In Pseudocode, detailed

Can one strategy give rise to many algorithms?

Yes, e.g., different algorithms implementing the same strategy may use different data structures

What are some of the strategies to solve the "search" problem?

What is the inherent complexity of finding the max among n numbers?

Finding the min?

Finding both?

The “moving zeroes” problem:
given an array of numbers move any and all zeroes in it to the end of the array

1. What is the inherent complexity of this problem?

2. Come up with multiple strategies to solve this problem. Several computational strategies are possible

3. Will sorting the array work…how?

4. Can you come up with a space-efficient strategy, i.e. one that leads to an **in-place** algorithm?

5. The order of non-zero numbers in the output is not relevant to correctness

6. But if it were, which of the strategies we have discussed in class is/are incorrect?

The “string search” problem:
find a string (such as “cat”) in a piece of text

Thinking Assignment

1. Write a problem specification so that the string search problem becomes well-defined.
2. Come up with one computational strategy to solve this problem.

A note about Thinking Assignments

Most lecture slides will have thinking assignments which are problems I make up or from the textbook. You should try to work out these problems as soon as possible, while your memory of the material is fresh. Try to do as many or as few as you have time for. If you have difficulty, ask your friends (you may want to schedule study sessions with friends to work out these problems), search the web, ask me or the TA (meet us during our office hours or set up an appointment). **These need not be turned in and will not be graded; nor will their solutions be discussed in class**. But working out the thinking assignments will definitely improve your grasp of the technical material and computational problem solving skills.

Make sure you grasp three key notions...

- problem specification: **a well-defined** problem
- **inherent computational complexity** of a problem
- **computational strategies** to solve a problem, as distinct from algorithms.

Thinking Assignment

1. Take an app you use a lot
2. What is/are the key computational problems it has to solve?
3. Can you come up with well-defined specifications of at least one of these problems?
4. Estimate its inherent complexity?
5. Come up with one or more solution strategies?

Thinking Assignments

- Come up with a strategy other than sorting to solve the problem of reordering an array of numbers to have any negative numbers appear first in any order, then any zeroes and finally any positive numbers.

- Which of the sorting algorithms you know are **in-place**?

- Which of the sorting algorithms you know are **anytime**?

- Can you come up with an anytime sorting algorithm that reads numbers one at a time, and processes them in such a way that if it is interrupted any time, it can output the numbers it has seen so far in the ascending order?

Reading & Thinking Assignment

Read all of Chapter 1 and try solving some of the exercises in that chapter.