Python Algorithms: Introduction

Jeffrey Fisher

Logistics

- Do you have a computer at home that you can use?
- Materials:
  - Personal notebook

Notebook

We have to first solve the problem, then *implement* it (write the code).

Code can help with some parts of investigating a problem, but for many of the problems it is inefficient to start with coding.

So everyone will need to have their own notebook.

Class structure

To learn problem solving, you must solve a lot of problems.

The core process will be:

Try to solve a problem → get stuck → come back to it → get further → get stuck → look at the answer/explanation → repeat.

Competition

- Codeforces
- In-person competitions
  - mBIT, Montgomery Blair High School

Metaskills

Skills that help you:

- test-taking strategies apply here. You have a limited amount of time and want to get as many points in the competition as possible.
- Must focus on the problem, but then let your brain work on it in the background (diffuse mode). Either switch to a different problem, or

Metaskills

- team strategies. Competitive programming is a team sport.
  - Learn together
  - Become familiar with each others strengths and take on specific roles in the competition.
    - Everyone will be solving problems, but maybe you're especially good at:
      - A specific kind of problem
      - Simplifying problem statements

Metaskills

- Recognizing when complicated problems have a simple mathematical solution
- Tricky implementations of solved problems

Algorithms and data structures

An algorithm is a “finite sequence of mathematically rigorous instructions” used to solve a problem (Wikipedia).

A data structure is a specific way to store data on a

“Mathematically rigorous instructions”

By “mathematically rigorous”, we mean “no confusion possible”. If you mailed the instructions to somebody trained in algorithms, they would be able to follow them without asking you any questions.

Rigorous instructions:

- Computer programming languages, like Python
- pseudocode

Data structures

A data structure is a specific way of storing data on a computer. We'll get into it more later. An example is storing objects in 3D space, for example in a robot or video game. We have multiple options for storing objects:

- a list of all the objects, with each object tracking its (x, y, z) position
- a matrix of the points in 3D space, storing every point
- a quadtree
- ...

Pseudocode

Pseudocode is a way of thinking about and communicating algorithms. It looks similar to code from a language like Python, but focused purely on being understandable by humans. When writing pseudocode we ignore all details that are not important for understanding the core of the algorithm. So it may take some work to translate pseudocode to code you can actually run on a computer.

There are no exact rules. Different people write pseudocode differently.

I'll show an example of pseudocode later.

Importance of pseudocode

Many of the algorithms we learn or the problems we solve might be hard to understand for the first time.

Pseudocode lets us focus only on understanding the algorithm or solving the problem, and worry about writing code to implement the algorithm later.

We don't want to get stopped in our train of thought by writing valid Python code.

Let's get started

1. Solve some basic problems.
2. Introduce pseudocode.
3. ... setup on USACO Guide.
4. ... Python lists, strings