A short Introduction To Algorithms

A bit of computational thinking

Caleb Kibet

February 22, 2019

Outline

Introduce yourself and answer the question...

What is you exposure to Linux and Computing?

- I have used Linux terminal
- I have used one or more programming language (R, Python, etc)
- I know about algorithms
- These are all new to me, but I am ready to learn

Why should you learn to about algorithms?

Alan Perlis

- One of the founders of computer science
- Argued in 1961 that Computer Science should be part of a liberal education: **Everyone should learn to program.**
  - Perhaps computing is more critical to a liberal education than Calculus
  - Calculus is about *rates*, and that's important to many.
  - Computer science is about *process*, and that's important to **everyone**.

Programming is about Communicating Process

An **algorithm** is a sequence of steps for solving a specific problem given its input data and the expected output data. An algorithm transforms the input to output.

A **program** is the most concise statement possible to communicate a process in a specific programming language.

..a set of instructions that instructs a computer to carry out certain operations

- translate DNA
- predict a motif
- perform statistical analysis
- align sequences

A Computer is Stupid, but this is about to change

"A computer is a stupid machine with the ability to do incredibly smart things, while computer programmers are smart people with the ability to do incredibly stupid things."
– Bill Bryson,

You have to tell it everything it needs to do, and how to do them...

An algorithm is a mechanical procedure that is guaranteed to eventually finish.

Algorithm to make coffee

For example, here is a procedure for making coffee, adapted from the directions that come with a major coffeemaker:

1. Lift and open the coffeemaker lid.
2. Place a basket-type filter into the filter basket.
3. Add the desired amount of coffee and shake to level the coffee.
4. Fill the decanter with cold, fresh water to the desired capacity.
5. Pour the water into the water reservoir.
6. Close the lid.
7. Place the empty decanter on the warming plate.
8. Press the ON button.

Describe a simple algorithm

Cook Ugali?

Programming language

Programming is the act of writing instructions that make the computer to solve a problem. A **programming language** is the tool that is used to implement an algorithm.

Figure: Some programming languages

What is a problem?

They are of two types:

1. A problem with algorithmic solution

- They have a clearly defined sequence of steps that would give the desired solution eg baking, adding two numbers...
- the sequence of steps or recipe for arriving at the solution is called the **algorithm**.

2. A problem with heuristic solution

- Solutions emerge largely from the process of trial and error based on knowledge and experience (learning) eg, winning tennis,
- **Machine Learning** is commonly used to solve such problems

Problem-solving steps

1. Defining the problem requirements (R)
   - Clearly define the problem in words, stating the input and output data as well as the processing logic. Requires familiarity with the problem environment and needs
2. Identifying Problem Components (C)
   - From the problem definition, identify the list of problem inputs, outputs, constraints and relationships between input and output data expressed in coherent formulas.
3. Possibly break problem solution into small modules (M)
   - Skip this step when dealing with small problems
4. 4. Design the Algorithm to solve the problem (A)
   - Chose the best among many alternative ways for solving the problem
   - Define algorithmic solution for all modules
5. Implementation and Coding (C)
   - Translate the algorithm into a program using a programming language
6. Test the program to ensure that it gives the expected output (R)

This course is concerned with problem definition to algorithm design with emphasis on specific types of algorithms for solving Bioinformatics Problems.

An **algorithm** is of interest to the programmer who will convert it to a computer program using a specific programming language. Therefore, an algorithm should be represented in a standard form easily clearly understandable by the programmer.

Pseudocode is a language computer scientists often use to describe algorithms: it ignores many of the details that are required in a programming language, yet it is more precise and less ambiguous than, say, a recipe in a cookbook.

Make PumpkinPie Recipe

Ingredients

- 1 1/2 cups canned or cooked pumpkin
- 1 cup brown sugar, firmly packed
- 1/2 teaspoon salt
- 2 teaspoons cinnamon
- 1 teaspoon ginger
- 2 tablespoons molasses
- 3 eggs, slightly beaten
- 12 ounce can of evaporated milk and 1 unbaked pie crust

Recipe

Combine pumpkin, sugar, salt, ginger, cinnamon, and molasses. Add eggs and milk and mix thoroughly. Pour into unbaked pie crust and bake in hot oven (425 degrees Fahrenheit) for 40 to 45 minutes, or until knife inserted comes out clean.

Pumpkin Pie Algorithm

```
MAKEPUMPKINPIE(pumpkin, sugar, salt, spices, eggs, milk, crust)
1  PREHEATOVEN(425)
2  filling ← MIXFILLING(pumpkin, sugar, salt, spices, eggs, milk)
3  pie ← ASSEMBLE(crust, filling)
4  while knife inserted does not come out clean
5        BAKE(pie)
6  output "Pumpkin pie is complete"
7  return pie

MIXFILLING(pumpkin, sugar, salt, spices, eggs, milk)
1  bowl ← Get a bowl from cupboard
2  PUT(pumpkin, bowl)
3  PUT(sugar, bowl)
4  PUT(salt, bowl)
5  PUT(spices, bowl)
6  STIR(bowl)
7  PUT(eggs, bowl)
8  PUT(milk, bowl)
9  STIR(bowl)
10 filling ← Contents of bowl
11 return filling
```

Figure: Make Pumpkin PseudoCode

Biological versus Computer Algorithms

Nature uses algorithm-like procedures to solve biological problems e.g DNA Replication

See Page 14 of "Introduction to Bioinformatics Algorithms

Change Problem

We'll use the Change problem to explain two important concepts in algorithms:

Does the algorithm give the correct output?

An algorithm is correct when it can translate every input instance into the correct output. An algorithm is incorrect when there is at least one input instance for which the algorithm does not produce the correct output. Unless you can justify that an algorithm is correct, alway assume it is incorrect. Some algorithms rely on approximation as no know correct algorithm exits.

How long does the algorithm take?

Rather than computing an algorithm's running time on every computer, we rely on the total number of operations that the algorithm performs to describe its running time

Important Algorithms Concepts

Recursive algorithms

An algorithm is recursive if it calls itself. A recursive algorithm is an algorithm which calls itself with 'smaller (or simpler)' input values, and which obtains the result for the current input by applying simple operations to the returned value for the smaller (or simpler) input.

Iterative algorithm

An iterative algorithm executes steps in iterations. It aims to find successive approximation in sequence to reach a solution. They are most commonly used in linear programs where large numbers of variables are involved.

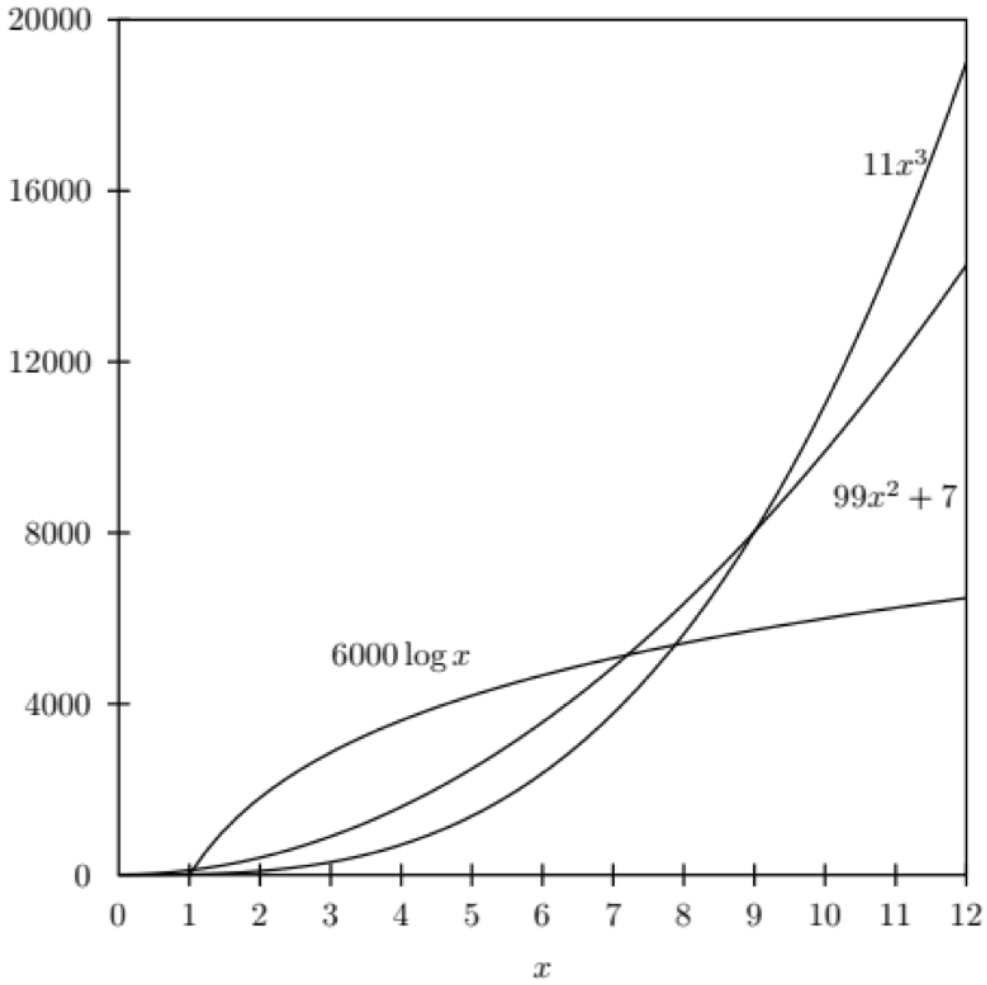

Figure 2.6 A comparison of a logarithmic ($h(x) = 6000 \log x$), a quadratic ($f(x) = 99x^2 + 7$), and a cubic ($g(x) = 11x^3$) function. After $x = 8$, both $f(x)$ and $g(x)$ are larger than $h(x)$. After $x = 9$, $g(x)$ is larger than $f(x)$, even though for values 0 through 9, $f(x)$ is larger than $g(x)$. The functions that we chose here are irrelevant and arbitrary: any three (positive-valued) functions with leading terms of $\log x$, x^2, and x^3 respectively would exhibit the same basic behavior, though the crossover points might be different.

You're a scientist first, not a programmer

Remember you are a **scientist** *and the* **quality of your research** *is what is important,* **not how pretty** *your source code looks.*

Perfectly written, extensively documented, elegant code that gets the answer wrong is not as useful as a basic script that gets it right.

Computational Thinking

Computational thinking is reformulating a seemingly difficult problem into one we know how to solve, perhaps by reduction, embedding, transformation, or simulation.

- **decomposition**: breaking down a complex problem or system into smaller, more manageable parts
- **pattern recognition**: looking for similarities among and within problems
- **abstraction**: focusing on the important information only, ignoring irrelevant detail
- **algorithms**: developing a step-by-step solution to the problem, or the rules to follow to solve the problem

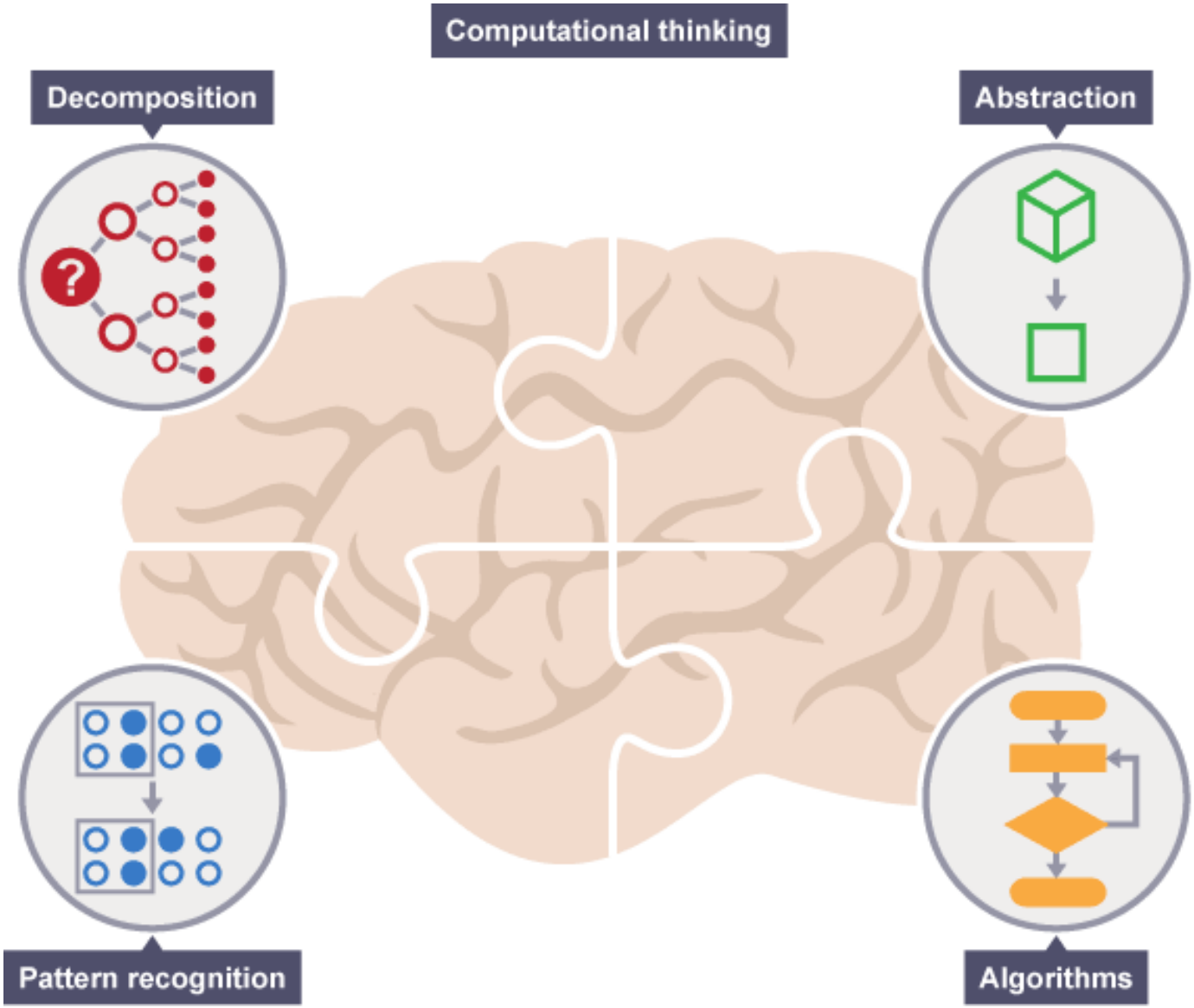

Figure: Computational Thinking. Source: https://www.bbc.com/bitesize/guides/zp92mp3/revision/1

What makes a good programmer?

- Logical thinking
- Creative thinking
- Problem solving skills
- The ability to "think outside of the box"

Be suspicious and trust nobody...

Computational analysis, especially with large data, **will always give you some results** (significant p-value).

Treat results **with great suspicion**, and carry out further tests to determine whether the results can be explained by experimental error or bias

Errors are opportunities to learn, embrace them

You will definitely have errors in your code. We all do. Learning to interpret the error message, and identify the problem, is an essential skill to acquire from the onset.

What's Next

- Learn Linux and the command line
- Learn Python

Training Format

We'll mostly use a combination of lectures and live coding.

References

To go deeper into this topic, read:

1. : Carey MA, Papin JA (2018) Ten simple rules for biologists learning to program. PLoS Comput Biol 14(1): e1005871. https://doi.org/10.1371/journal.pcbi.1005871
2. Loman, N., and Watson, M. (2013). So you want to be a computational biologist? Nat. Biotechnol. 31, 996.introduction