COL100: Introduction to Computer Science

1.1 What is computer science?

What is COL100 about?

- “How to use a computer” —No
- “How to do programming” —Partially. But not the main focus!

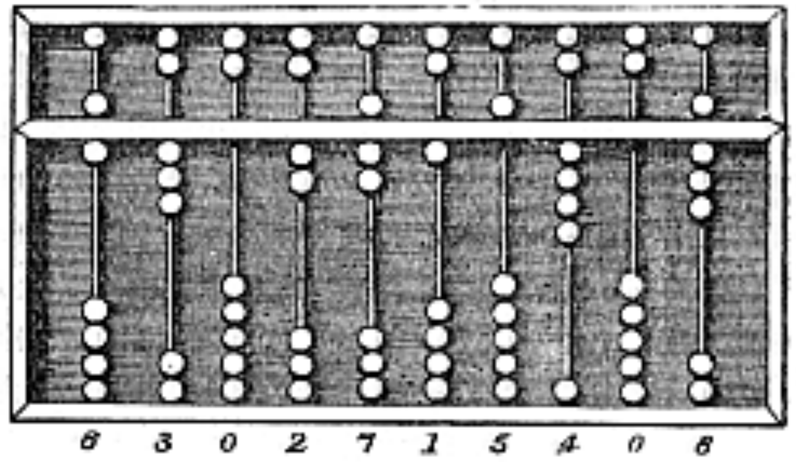
COL100 is about *computation*.

Computation can be done even without a computer

Examples of computational problems

Adding, subtracting, multiplying, dividing integers

$$
\begin{array}{r}
\;3\;8\;4 \\
\times\qquad 5\;6 \\
\hline
\;2\;3\;0\;4 \\
1\;9\;2\;0 \\
\hline
2\;1\;5\;0\;4
\end{array}
$$



Examples of computational problems

Longest common subsequence

“this is an older version of the sentence I wrote”

“this is the new sentence after some changes”

“this is ~~an older version of~~ the new sentence ~~I wrote~~ after some changes”

Examples of computational problems

Sorting

→

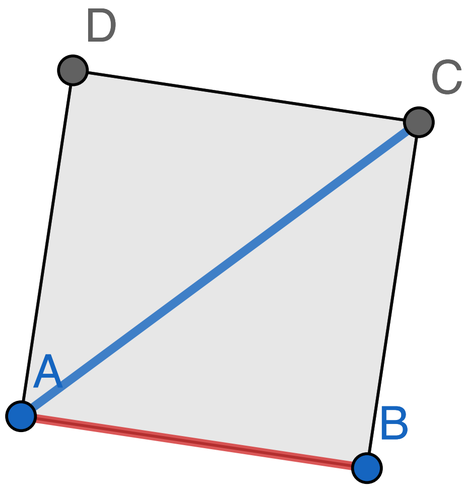
An unconventional example

Given a positive real number x, find $\sqrt{2} \cdot x$.

Given a line segment AB, construct a line segment whose length is $\sqrt{2}$ times the length of AB using a straightedge and compass.

1. Construct a square which has AB as one edge.

How?



2. Draw a diagonal of the square.
   This is the desired line segment.

Geometry as computation

1. Construct a square which has *AB* as one edge. ?
2. Draw a diagonal of the square. This is the desired line segment. ✓

Geometry as computation

1. Construct a square which has *AB* as one edge. ?
    1. Construct a line through *A* perpendicular to *AB*. ?
    2. Draw an arc centered at *A* with length *AB*. ✓
    3. …
2. Draw a diagonal of the square. This is the desired line segment. ✓

Geometry as computation

1. Construct a square which has *AB* as one edge. ?
    1. Construct a line through *A* perpendicular to *AB*. ?
        1. Draw a circle centered at *A* with small radius. ✓
        2. …
    2. Draw an arc centered at *A* with radius *AB*. ✓
    3. …
2. Draw a diagonal of the square. This is the desired line segment. ✓

Models of computation

A *model of computation* defines primitive objects/values & operations on them.

- **Straightedge and compass:** points, lines, arcs; drawing and intersection
- **Decimal arithmetic:** sequences of digits; add/subtract/multiply two digits

An *algorithm* is a precise, unambiguous specification of the solution to a given problem in terms of such operations.

Why algorithms?

Algorithm is precise and unambiguous

→ Carrying out algorithm requires no ingenuity / insight / intelligence

→ Can be done by anyone/anything (that can perform primitive operations)

→ Can be done by computer!

Intelligence is required in *designing the algorithm* in the first place.

Algorithms and programs

An algorithm can be expressed in many ways…

English:

Given two numbers *a* and *b*, if *a* is greater than *b* then the maximum is *a*, otherwise it is *b*.

Mathematical notation:

$$\max(a, b) = \begin{cases} a & \text{if } a > b, \\ b & \text{otherwise} \end{cases}$$

Standard ML:

```
fun max(a, b) =
  if a > b then a
  else b
```

A *programming language* is an artificial language for expressing algorithms

- …in a specific model of computation (which can be executed on a computer)

Algorithm design and programming

“Write a program to solve problem XYZ”

1. Design the algorithm — abstractly, in words and/or maths
2. Prove that the algorithm is correct
3. Implement the algorithm in a programming language

Programming languages

Hundreds of different programming languages out there! Which to learn???

… C C++ C# Java JavaScript ….

… Perl PHP Python Ruby ….

… Scala Go Rust Lisp Scheme ….

… ML OCaml Haskell Prolog ….

Algorithms don’t depend on language, only on model of computation.
…And many languages have same / similar computational models!

This course

First half of semester: **functional model of computation**

- Programming language: Standard ML
- Closer to mathematics
- Easier to analyze and prove correctness
- Encourages thinking about programs at higher level

Second half: **imperative model of computation**

This course

First half of semester: **functional model of computation**

Second half: **imperative model of computation**

- Programming language: Python
- Can still write programs in functional style
- Imperative model common in other “industrial” languages (C, C++, Java, etc.)
- Proving correctness is more complicated

Reading material

- Read the course webpage in full!
  http://www.cse.iitd.ac.in/~narain/courses/col100/
- Lecture notes by Profs. Subhashis and Arun-Kumar:
  http://www.cse.iitd.ac.in/~suban/COL100/lecture.pdf
  - This lecture: Ch. 1