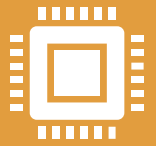
Algorithms

19CSE100 – Problem Solving and Algorithmic Thinking

Algorithms

An algorithm is a detailed *step-by-step instruction set* or formula for solving a problem or completing a task.

In computing, programmers write algorithms that instruct the computer how to perform a task.



When you think of an algorithm in the most general way (not just in regard to computing), algorithms are everywhere.

Computational Concepts

The algorithm for an even simpler task, like Brushing the TEETH or preparing CORNFLAKES.

Without knowingly, we are exploring computational concepts like

Repetition (brush bottom left teeth five times)

Sequencing (put cereal in bowl and then put in milk),

Conditional Logic (if the bowl is empty, stop eating).

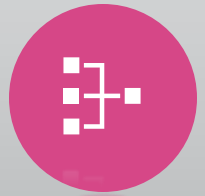
Computers don't understand our intentions, so if we don't specify that we need to get out the bowl first, we will end up pouring milk on the floor!

Benefits of Algorithmic Thinking

Algorithmic thinking allows to break down problems and conceptualize solutions in terms of discrete steps in a procedure.

Algorithmic thinking is thinking about how to solve a problem in a systematic way. It's about:

Defining the problem clearly



Breaking the problem down into small, simple parts

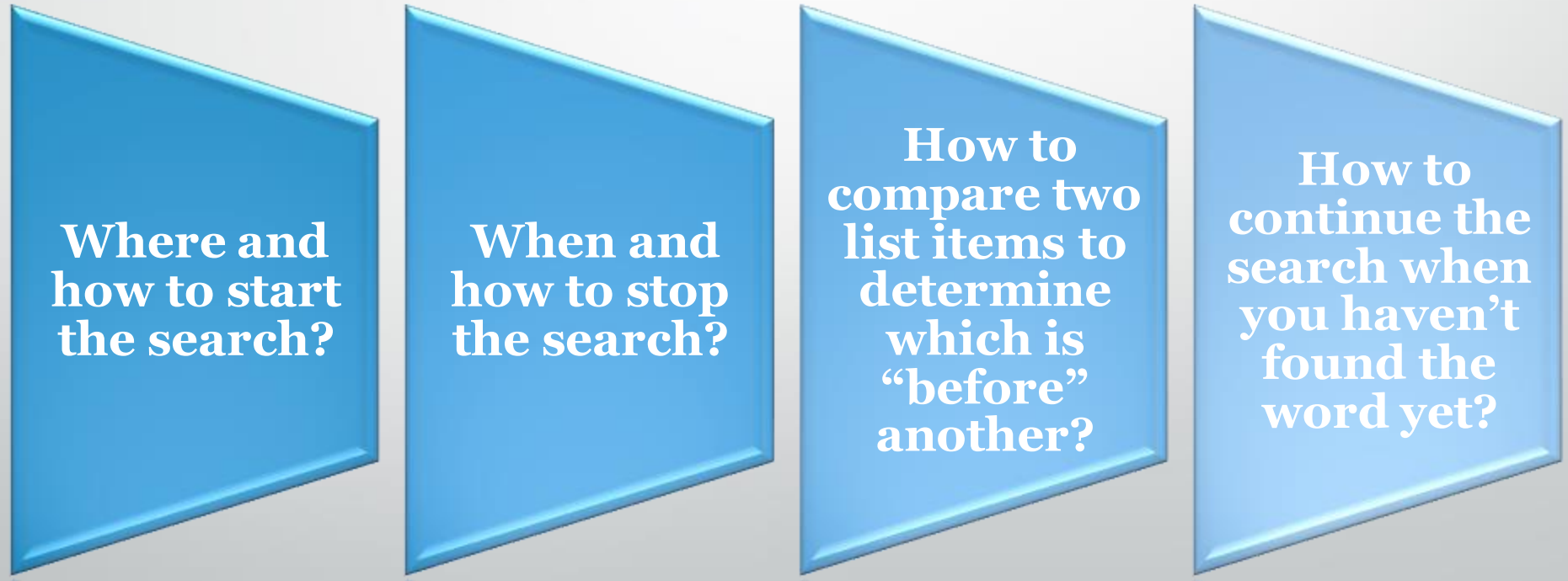
Define the solution for each part of the problem

Implementing the solution

Making it efficient (eventually)

Algorithm Example

Take the problem of looking up the word “bumfuzzle” up in a dictionary (bumfuzzle means “to confuse”).



Characteristics of Algorithm

Unambiguous – Algorithm should be clear and unambiguous. Each of its steps (or phases), and their inputs/outputs should be clear and must lead to only one meaning.

Input – An algorithm should have 0 or more well-defined inputs. **Output** – An algorithm should have 1 or more well-defined outputs, and should match the desired output.

Finiteness – Algorithms must terminate after a finite number of steps. Stopping may mean that you get the expected output OR you get a response that no solution is possible.

Effectiveness - It means that all those steps that are required to get to output must be **feasible** with the available resources. It should not contain any unnecessary and redundant steps which could make an algorithm ineffective. *For Eg: suppose you are cooking a recipe and you chop vegetables which are not be used in the recipe then it is a waste of time.*

Independent – An algorithm should have step-by-step directions, which should be independent of any programming code.

Completeness

1. Go to shop
2. Buy Chocolate Muffins
3. Return home

Is this complete?

Completeness

1. Go to shop
2. If shop has Chocolate Muffins
   - Then buy Chocolate Muffins
   - Else if the shop has Croissants
   - Then buy Croissants
   - Else buy nothing
3. Return home

Determinism

1. Go to shop
2. If shop has Chocolate Muffins
   - Then buy Chocolate Muffins
   - Else if the shop has Croissants
   - Then buy Croissants
   - Else buy nothing
3. Return home

Determinism

1. Go to shop
2. If shop has Chocolate Muffins
   Then buy Chocolate Muffins
   Else if the shop has Croissants
   Then buy Croissants
   Else buy anything
3. Return home

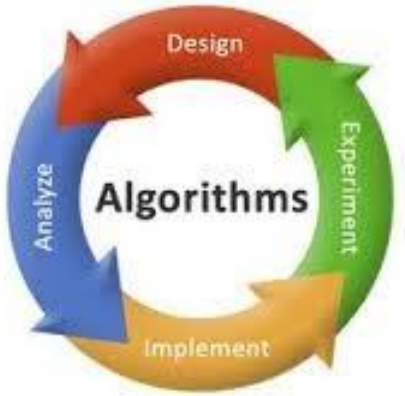
Is this deterministic?

Computing without computers, Paul Curzon

Termination

An algorithm should always finish and produce a result

How to write Algorithms



Type I: Sequential

Flow of the algorithm is **sequential**

Ex: Find the area of the circle

Algorithm:

1.Start
2. Declare variables area, radius.
3: Read the value, radius.
4: Compute area= pi *radius^2.
5. Display "The area is", area
6. Stop

Type II: Selection/ Branching

Flow of the algorithm is **Branching**

Ex: Find the largest number among two different numbers

Algorithm:

1.Start
2.Declare variables a,b.
3. Read the values, a,b.
4.If a > b is true
    4.1 Display a is the largest number.
    4.2 goto step 6
5. otherwise
    5.1 Display b is the largest number.
6.Stop

Type III: Repetition/ Looping

Flow of the algorithm is Repetition

Ex: Display the numbers from 1 to 10

Algorithm:

1. Start
2. Initialize number=1
3. Print the value of number.
4. Let number =number + 1.
5. Repeat the step 3,4 until number > 11.
6. Stop