122COM: Introduction to algorithms

David Croft

Coventry University

david.croft@coventry.ac.uk

2016

Overview

Introduction

Introduction to algorithms module.

- What is an algorithm?

Introduction

Introduction to algorithms module.

- What is an algorithm?
- Not the same as code.
- Not the same as a program.

Introduction

Introduction to algorithms module.

- What is an algorithm?
- Not the same as code.
- Not the same as a program.

Task/algorithm/code

A task is a problem that needs to be solved.

- I.e. bake me a cake.

Task/algorithm/code

A task is a problem that needs to be solved.

- I.e. bake me a cake.

An algorithm is a generalised set of instructions to perform a specific task.

- A strategy to solve a given problem.
  - Many different strategies to solve same task.
- Like a recipe.

Task/algorithm/code

A task is a problem that needs to be solved.

- I.e. bake me a cake.

An algorithm is a generalised set of instructions to perform a specific task.

- A strategy to solve a given problem.
  - Many different strategies to solve same task.
- Like a recipe.

Code is a specific set of instructions to perform a specific task.

- An implementation of a strategy in a specific language/system.
- Have to adapt the recipe to your kitchen/oven etc.

Fibonacci sequence algorithm

C

Task - calculate the fibonacci sequence.

Fibonacci sequence algorithm

C

Task - calculate the fibonacci sequence.

Algorithm

1. Starting with 0 and 1.
2. Sum the two numbers to make a third.
3. Discard the lowest number.
4. Repeat from step 2.

Fibonacci sequence algorithm

C

Task - calculate the fibonacci sequence.

Algorithm

1. Starting with 0 and 1.
2. Sum the two numbers to make a third.
3. Discard the lowest number.
4. Repeat from step 2.

Code

```
def fibonacci( a, b ):
    c = a + b
    a, b = b, c

    print( a )
    fibonacci( a, b )

fibonacci( 0, 1 )
```

```
for( int a=0, b=1, c;
     a>=0;
     c=a+b, a=b, b=c )
{
    cout << a << endl;
}
```

Module content

Looking at search algorithms today.

Looking at sorting algorithms in a later week.

Will be tested on some algorithmic concepts.

- Implement simple algorithms.
- Describe advantages/disadvantages of certain algorithms.
- Big *O* notation.
  - How algorithms scale.
- Calculate an algorithms *O*() notation.

Most important

Thinking algorithmically.

- Learning how to break down a problem into small steps.
- Think through algorithms.
- Evaluate algorithms.
  - Does this algorithm actually work?

The End