Graphs, Groups, and Isomorphisms

ian.mcloughlin@atu.ie

January 27, 2023

Are all programming languages equally capable? What are the limits of computation? How can we discuss computation without worrying about the details of specific machines? To answer these questions, we need a simple way of describing problems. In the following article we will define the basic building blocks of computing.

References

We will use two reference texts: Norman Biggs' *Discrete Mathematics*[1] and Michael Sipser's *Introduction to the Theory of Computation*.[2] Another good resource is the Open Logic Text.[3] On the practical side, I recommend also use The Python Tutorial,[4] The Python Software Foundation's official tutorial for Python.

[1] Biggs, Norman L. *Discrete Mathematics*. revised Edition. Oxford Science Publ., 1989.

[2] Sipser, Michael. *Introduction to the Theory of Computation*. Third. Boston, MA: Course Technology, 2013. ISBN: 113318779X.

[3] Open Logic Project. *Open Logic Project Builds*. Dec. 21, 2022. URL: `https://builds.openlogicproject.org/` (visited on 01/23/2023).

[4] *The Python Tutorial — Python 3.11.1 documentation*. Jan. 22, 2023. URL: `https://docs.python.org/3/tutorial/` (visited on 01/22/2023).

Graphs

A graph is

```
def f(x):
  ans = 1
  for i in range(x):
    ans = ans * i
  return ans
```

In the earlier definition of a function we made no reference to algorithms. To mimic those kinds of functions we can just use a dictionary.

```
def f(x):
  pairs = {1: 1, 2: 4, 3: 9}
  return pairs[x]
```

Further Topics

Now that we have defined our basic building blocks, we can discuss topics in computation and computability.