The Essence of Algorithms

Feng Zonghao[1]

To learn what is an algorithm, an intuitive thought might be firstly figure out the defintion of algorithms. But the fact is that giving a formal definition of algorithms still remains a challenging problem [2].

Let's take a look at how Donald E. Knuth gave his formal definition of algorithms in "the bible of computer science", *The Art of Computer Programming* [1]. He defined an explicit instruction set so that the algorithms he presented could be defined precisely, and their performance analyzed at better quality. This presentation is a mathematical model, even though it looks like "implementation". Other formal definitions have similar characteristic, as they all use abstact mathematical models, such as Turing-complete models.

As formal definitions are usually complex and obscure, most times we only talk about the informal definitions of algorithms. We can define an algorithm in a more descriptive way as follows:

Definition 1.

An algorithm is a **finite** set of **unambiguous** steps that defines a **efficient** process for solving a specfic type of problem. It receives given **inputs**, and generates expected **outputs** for the problem.

Pay attention to the bold words, as they constitute the core and essential ideas of algorithms.

1. **Finite** is for finiteness. An algorithm must always terminate after a finite number of steps for any given inputs, even if the number of steps could be extremely large.
2. **Unambiguous** is for definiteness. Each step of an algorithm must be precisely defined. The instructions must be unambiguously specified for each case. The languagues we speak may not meet this restriction, so programming languages are designed for delivering the ideas to computers. Computers can "understand" and execute binary codes, which are compiled from programming languages.
3. **Efficient** is for effectiveness. The steps of an algorithm must all be sufficiently basic and executable. Notice that the "effectiveness" we mentioned here does not require the complexity of the algorithm to be as low as possible, as there might be numerous algorithms with different complexity for solving a same problem.
4. **Inputs** can be regarded as an instance of the given problem. An algorithm produces **output** for the problem instance, which can be regarded as a solution. We say an algorithm can solve a specfic type of problem, it means that for all the problem instances, the algorithm could give correct solutions.

References

1. Donald Ervin Knuth. *The Art of Computer Programming: Fundamental Algorithms*. Addison-Wesley, 1973.

2. Yiannis N Moschovakis. What is an algorithm. *Mathematics unlimited–2001 and beyond*, pages 919–936, 2001.

[1]Computer Science Class 2013, Taishan College, Shandong University. Student number: 201300130013