Module 1 Critical Thinking: Algorithms in Practice

Donghwan Kim

Colorado State University Global

CSC 506: Design and Analysis of Algorithms

Dr. Lori Farr

April 28, 2022

Module 1 Critical Thinking: Algorithms in Practice

This Module 1 Critical Thinking assignment discusses what an algorithm is including my personal perception of it and then as a practice design an algorithm that, given an arrangement of digits from zero to nine, finds the next larger value of it by just rearranging the digits. Algorithm is a sequence of specified instructions to solve a problem. The next section deals with some aspects of the algorithm and, the section of Algorithms in Practice takes a close look at the problem of finding the smallest number larger than a given number and then proposes an algorithm based on the patterns found.

Algorithm: An overview

As the building block of computing, algorithm is a set of specified instructions to solve a computational problem. Here a definition of an algorithm: "An algorithm can be described in English, pseudocode, a programming language, hardware, etc. A computational problem specifies an input, a question about the input that can be answered using a computer, and the desired output"(Lysecky and Vahid 2019: Section 1.2). The frequently used keywords to define algorithm in English dictionaries and literature including Algorithm (2022) and Cormen et al. (2009) are "step-by-step procedure", "problem-solving", "computation", and so on. The next section exemplifies a computational problem and its algorithm design after briefly reviewing the type of algorithms.

Type of algorithms

There are lots of algorithms developed to perform a certain computational task and problem, and there are several ways to view the algorithms and categorize it. And the computational problems include searching, sorting, string processing, numerical problems, and so on. Searching problem is to find a certain value in a given elements. The practice below involves the searching algorithm. It is straightforward to search the elements sequentially (which is termed `sequential search` or `linear search`). If the elements are sorted in a certain order, an alternative `Binary search algorithm` can reduce the runtime. Iteration and recursion is a way to categorize algorithms. Algorithms have their own constrains and

strengths. One consideration in evaluating algorithm is to measure how fast the algorithm performs the task as the input size (which is usually denoted by n), which is called `algorithm efficiency` or `computational complexity` (Lysecky and Vahid 2019; Cormen et al. 2009).

More on algorithms

Designing and coding an algorithm require logical thinking and attention to detail. Although agile approach of modern software development emphasizes the important of interaction with users - Indeed, it is critical - and it is warned not to be too logical when communicating with stakeholders and users, logic and attention to detail are important personalities for algorithm design. A basic algorithm is the bisection method to find the roots of a continuous function. Numerical analysis includes data structure and numerical analysis of matrix, and optimization algorithms in statistical learning, which deals with gradient descent, newton's method, and even Bayesian method. The workflow of Machine learning (in the case of supervised learning) consists of the choice of hypothesis, loss function, and optimization. For most of hypothesis and loss function, optimization is available numerically. The numerical analysis is becoming important with the AI and ML.

Algorithms in Practice

This section designs an algorithm that, when given an arrangement of the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, rearranges the digits to find the next larger value. For example, 5647382901 would produce 5647382910. More precisely, this problem is to produce the smallest number which is larger than the given number with the same set of digits. As will be shown, the solution to the problem is independent of the number of digits used. That is, the number of digits used does not affect the design of the algorithm. This section takes a close at the problem, seeks patterns, and then designs an algorithm.

A close look at the problem

Let's first think of the two extreme cases: the smallest number and the highest number possible. The highest number is 9876543210. It is the one in which all the digits are sorted in descending order. The comparison of numbers starts with comparing the leftmost digit,

and then, if it is same, moves to the next digit to the right and so on. The number with a higher leftmost digit is greater. As will be proved, the second highest number is 9876543201. Next, the smallest number is 0123456789, which is sorted in ascending order, And the next smallest number larger than 0123456789 is 0123456798 and the next one is 0123456879.

It tells that, given an arrangement of the digits from zero to nine, sorting it in ascending and descending orders is the way to get the lowest and highest numbers, respectively. It is applied to any subsets of the digits. For example, 3210 is the highest number possible with the digits 0, 1, 2, an 3. And the smallest number is 0123. This property plays an important role in finding the solution. With this in mind, let's discuss an algorithm of the problem.

The key to increase a number is in the position whose number is lower than the previous one from the rightmost side. Therefore, the first step to find the solution is to look at the digits from the rightmost of the given number and find the first position whose digit is smaller than previous one. If there is no such position like in 9876543210, it means that the number is the highest number among them. Once the position is found, the right-hand side of it including the position is the only part considered - the left-hand side of it is irrelevant anymore.

Two more steps are necessary to obtain the smallest larger than the given number. To get a higher number, the digit of the position found should be replaced with a higher number which comes from the right-hand side. It should be replaced with the smallest number among the larger numbers in the right-hand side. After the replacement, sorting the right-hand side in ascending order gives the solution, the smallest one.

An algorithm to find the next larger number

Below is a description of an algorithm. The algorithm to solve this computational problem can be used for any subsets of the 10 digits. For example, if the input number is 3842, it will produce 4238, which is next larger number by reordering the four digits 2, 3, 4, and 8. That is, the output is the smallest number larger than the input. The algorithm below involves searching (in Step 2 and 3) one of which is performed for an ordered (sub)array and one

sorting (in Step 5).

Algorithm to find the next larger number

// Input: Array of digits representing a number

// Question: What is the smaller number larger than the input by reordering the digits

// Output: the next larger number than the input

Step 1 Store a number in an iterable array (e.g. Python list)

Step 2 Iterate from the reverse to find the first position whose number is less than the previous one.

Step 3 Find in the right-hand side the index of the smallest digit which is greater than the digit found in Step 2

Step 4 Replace the two digits found in Step 2 and 3

Step 5 Sort the right-hand side in ascending order

Conclusion

Algorithm is a sequence of rules to solve a computational problem, whose activity requires logical thinking and attention to detail. This Module 1 Critical Thinking assignment reviews what an algorithm is, and then designs an algorithm to find the next larger number by rearranging given digits. To do that, it takes a close look at the problem, seek patterns, and finally designs an algorithm based on the patterns. The key to solution is to find the first position whose digit is lower than the previous one from the rightmost. The digit is replaced with the smallest larger than the digit in the right-hand side of it and then the final step is to sort the right-hand side in ascending order.

References

Algorithm (2022). In *Wikipedia*. `https://en.wikipedia.org/wiki/Algorithm`.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms*. The MIT Press.

Lysecky, R. and Vahid, F. (2019). Design and analysis of algorithms. In Lysecky, R. and Vahid, F., editors, *Data structures essential: Pseudocode with python examples*. Zybooks.