My interest in coding has developed since I was eight. That was when the virus ‘Whboy’ first infected my father’s computer. Shocked by how complicated the virus could be, its composition became a childhood myth and eventually motivated me to pursue further studies in computer science and figure more about algorithms and logics.

I have kept on exploring the world of algorithms. From the fundamental sorting algorithms to its optimisations, from the purest recursion function to dynamic programming, from the Binary search tree to the Red-Black tree, all those algorithms and data structures amazed me with their high consistency and efficiency in solving complicated puzzles. In many competitive programming competitions such as the National Olympics of Informatics and Code Jam, those algorithms become my useful instruments in solving the questions in the fastest approach.

Among them, sorting algorithms are the most common. QuickSort, for example, is claimed to have the least average running time for common sorting algorithms. However, since QuickSort has a worst-case running time of O(N^2), it is unimaginable for me to link it with “quick”. Hence, I start to explore how we may optimise QuickSort. The key to do so is to ensure a fair partition, namely choosing a suitable pivot. In a naive implementation, a pivot is selected from a fixed position - the first or the last place of the array. However, when facing sorted arrays, this way of choosing pivots inevitably results in a worst-case partition. Thus, to avoid this, a shuffle of the array may effectively reduce the possibility of getting the worst case when facing ordered arrays. A more advanced way of randomisation is to use Median-of-3 partition: by selecting the first, last and another random element and using their median as pivot may increase the probability of having a best-case partition. The concept of probability here may seem irrelevant to algorithms, yet it plays a significant role in implementations. However, I still find flaws in this method as the worst-case still occurs in arrays with repeated elements. After researching, I have finally encountered the best approach in ‘Data Structures and Algorithm Analysis in C’: grouping all elements the same as the selected pivot, which could avoid worst-case partition in sub-steps. The process of optimisation may appear vigorous, but in real life, many algorithms are running simultaneously within the computer system. Thus, any small ‘steps’ in the algorithm could yield a tremendous ‘leap’ of efficiency, and optimisation is indispensable.

Logic, on the other hand, is another aspect that drew my attention. Induction is a fundamental proof method in Mathematics. With induction, to prove the validity of a statement requires only a valid base case and a correct inductive step. This concept makes me wonder similar uses in coding. Like mathematic formulas, we should be able to know when a loop ends and whether the consistency keeps while looping. After doing some research, I find out an important concept of logic in computer science - Hoare Logic. Just like a base case in induction, the precondition of the Hoare’s triple indicates the initial state before running the code. The inductive step, in this case, becomes the postcondition after running the code and each line of code is the process to change from pre to postconditions. Like invariant quantity in maths, there is also invariant in the logic triple to prove the validity of code. This parallelism between Math and coding helps me better understand the new concept and it brings me further is other areas of Computer Science.

Beyond curriculum, I have joined Robotics Club. When I am composing the functions for robots, I have deeply understood how useful the theoretical knowledges of algorithms and logics are in real-life application. The knowledge I have learned builds up a solid foundation for my future study. I believe that studying at an outstanding university would guide me further.