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

The world of algorithms contains varied types of functions, and I have kept on exploring this world. From the basic sorting algorithms to its optimisations, from the simplest recursion function to dynamic programming, from the Binary search tree to the Red-Black trees, all those algorithms and data structures amazed me with their high consistency and efficiency in dealing with complicated puzzles. In many competitive programming competitions such as NOI(national olympics of informatics) and Code Jam, those algorithms become useful tools in solving the questions in a fastest approach.

Among them, sorting algorithms are the most frequently used. Quicksort is an example of them. As its name suggests, it is claimed to have the shortest average running time for sorting. However, since the worst case of Quicksort has a running time of O(n^2), it is unimaginable to link it with “quick”. I started to research online on why Quicksort has an average running time of O(nlgn) even it has a much longer worst case. Eventually, I figure it out through a recursion tree example. If a good partition which separates array into two (n-1)/2 takes place alternatively with the bad partition that divides the array into size 0 and n-1, the resultant time cost of this level's partition is still O(n), only with a variation on the constant term inside the O notation. Hence, to achieve the average case, a fair partition is vital. I start to ponder how to optimise the sorting.

At initial phase, a pivot is selected from a fixed position - the first or the last place of the array. However, when facing sorted array, this way of choosing pivot inevitably results in worst-case partition. Thus, to avoid this, a randomisation is required to select the pivot from a random position. This effectively reduced 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 could further reduce the occurrence of the worst case. However, I still find flaws in this method as the worst case still occurs in arrays with equal elements. After researching, I have finally encountered the best approach in ‘Data Structures and Algorithm Analysis in C’. Insertion sort is preferred over quick sort when sorting arrays of shorter length due to its higher efficiency. Additionally, by grouping all elements same as the selected pivot together, it is able to avoid worst-case partition in sub steps. With the combination of insertion and quick sort and the grouping of equal elements, the running time to sort arrays with repeated elements is greatly reduced. This approach of sorting is also known as Qsort. Eventually, I have finally understand the ‘quick’ of quick sort and the spirit to keep improving one approach attracts me deeply.

Beyond algorithms, logic is another aspect that draws my attention. Induction is a concept that is frequently used in Mathematic proof. With induction, to prove the validity of a statement requires only a valid base case and a correct induction case. This concept challenges the traditional way of proof, and it makes me wonder similar uses in coding. Like mathematic formulas, a line of code is also a statement which should be able to get validated. After reading relative books and searching online, I find out an important concept of logic in computer science - Hoare’s 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 post conditions. 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 new concept and it brings me further is other areas of Computer Science.

With the knowledge of logic and algorithms, I have applied them in real-life application beyond my curricular time. I have participated in Robotics competition and use the algorithms such as Greedy to find the shortest possible path inside the unexplored map. With that, the mission running time for a robot to navigate has greatly reduced. Through this experience, I have understood how useful and powerful algorithms and logics are even during daily life.

The knowledge I have builds up a solid foundation for my future study. I believe that the opportunity to study at such an outstanding university would guide me further.