“Find one bottle that contains poison among 1000 using 10 mice given that mouse would not die immediately.” This question on Quora captured my attention when I was in Year 9. After reading the answer, I found it necessary to solve this question with binary notation. Indeed, using 10 mice could find the poisoned bottle of water among 1024 bottles at most. With a further search for knowledge on binary, I was astonished by the incredible concept that using simply 0 and 1, all those complicated arithmetic and fancy video game scenes could be achieved, which aroused my initial interest in computers.

After getting exposure to programming in high school, my interest was completely ignited. With an understanding of how high-level language was transformed into assembly language, I realised that each piece of code in programming was like a stream of 0 and 1. It seemed that with the wand of binary, I could solve any problem using spells called algorithms, which brought me a sense of accomplishment that I had never had before. Accordingly, I tended to unconsciously apply general algorithms to address problems encountered.

To stimulate the intellectual curiosity of the class, the math teacher once gave a tricky puzzle: how to get exactly 4L of water with a 5L beaker and a 3L beaker? The idea of using Computer Science concept to build a general solution sprung into my mind. After researches and inquiries, it came away clear to me that this was about DFS and BFS algorithms. I started to use the same concept following figuring out the principle. I found that I could derive the next state with different actions of the current state, and then get another state with the iterative method. All possible states could then be obtained by comparing whether the new state has been previously visited. I further understood that searching algorithms can be optimised by applying a heuristic to choose an appropriate next state.

As I progressed in programming, I increasingly realised the importance of understanding the mathematical model behind the question. That's why I took part in the UKMT competition to strengthen my math skills. In Year 11 I won the UKMT Silver Award, and following one year of further study, I got the UKMT Gold Award in Year 12. To test my ability to transfer Maths knowledge to programming concept, I attended 2019 SPC Suzhou programming competition. Regardless of the third place I won, I did not manage to code an algorithm called KNN classifier and I put it into my mind deeply. After the competition, I did some research on this algorithm. During the process, I learned that artificial neural networks might perform better in the field of classification and recognition, which also provided an idea for my later thoughts.

For some of my peers, including myself, regardless of an alarm, oversleeping always results in tardiness. I came up with an idea to make a clock that could automatically recognise whether the user accidentally falls asleep after the alarm has rung. My initial thought was to install a miniature camera on the clock and judge whether the user is in bed or not by comparing the RGB total values of the pictures before and after the ring. But the results of the test were unsatisfactory. After consulting the teacher, I realised that the method of Euclidian distance could significantly improve the success rate. However, following the review of these images over and over again, I found their accordance in a great contrast of light and shade resulted in the inaccuracy of colour recognition. At this moment, the artificial neural network gave me a new idea that could directly identify whether someone is in the bed. However, due to my limited knowledge of mathematics and programming, it is impossible for me to get further into this idea for the moment.

Successfully, I have figured out a lot of problems with computer science knowledge, but like what Isaac Newton said, the great ocean of truth lay all undiscovered before me, I only own a few shells on the beach, so I am determined to keep exploring to perfect and realise my ideas.