First paragraph:

Why you choose this subject?

3D printing winter camp/ colour map

Second paragraph

Extra knowledge

The extended project/

The process you work out the soduko algorithm

* Asymptotic notation
* Graph theory
* DFS and BFS
* Heaps and Queues
* Some contents in the introduction to the algorithm

Third paragraph

Awards and achievements

Fourth paragraph

Something to end (Why you go to the UK)

When I first touched programming in computer science, I found that it is not only amusing when you are making your own program but also really useful that it can create many amazing functions which only computers can do. Thus, it is such a wonderful way for people to solve the problems which they are no able to do before by using the computers. Just like the color map problem which was the first major theorem to be proved by using a computer. By using the computer tests, it can help mathematicians to proof some theorems. Besides, computer science can also help some vulnerable groups in the society. I have also attended a winter camp which taught us how to use 3D printing to make the artificial limbs for the hands disables children. I realized how computer science can change our lives and make our lives more convenient and comfortable. This is the reason why I want to learn computer science.

1. Senior Mathematical Challenge 2017 and 2018 Gold Certificate
2. ASDAN 3D printing 2019 Feburary: Use 3D printing technology to make artificial limbs for the hands disable children
3. First Place Certificate for the Problem Solving Round in the High School Team Maths Competition between 16th and 17th March 2018
4. School House Programming Competition 2019 April: Write the program for the problem and try to make it within the smallest memory space and least time
5. Extended Project 2019 May-August: Do the research about how machine learning can help to improve the security
6. Sudoku Project 2019 7.1-7.16: Try to use algorithms to solve the Sudoku problem and do some comparisons in a group

I realize that it is important for us to corporate and divide our tasks when we do a group project so that we can work efficiently. We should frequently have some conversations about the project and combine our thoughts together. We can learn from each other and create more great ideas throughout the discussions. Everyone should be responsible for their parts and manage their time well, so we can finish the project in time.

I once attended the summer camp for using the programming to solve the Sudoku problems. I firstly thought about letting the computer solve the Sudoku in the human ways that does the eliminations by checking each row, column and box (contains 9 cells/grids). As a result, I wrote the if statements for checking if we could put this number in and a for loop to try number 1 to 9 for each cell. It was quite complicated, but it worked. For some simple Sudoku, it worked really fast about only 0.03s. However, I found out that if at some point, the number in the cell which needed to be guessed, then the elimination method could not solve it. Then, I tried to use the algorithms that computer could try numbers on each cell and returned the correct one. Thus, the ideas came up in my mind is using the Depth First Search Algorithms (DFS) and Breadth First Search Algorithm (BFS). For BFS, it would try the number in each cell and save the correct Sudoku in a stack which we called it candidate, then it would move to the next blank. We also had two checking function at the end. One was checking if the Sudoku was completed, another one was checking if the Sudoku was correct that met all the constraints (1-9 unique integers in each row, column and box). If this Sudoku was not valid, it would pop out the last one in the stack. The process would continue until the Sudoku was completed and valid. I had also tried the BFS which the only difference was using the queue and popped out the first candidate if it was not valid. I had tried different Sudoku and the graph was made to show the comparisons that I found that the time taken for the BFS is usually longer than the DFS since BFS discovers all the vertices at a certain distance before moving to further vertices whereas the DFS tries to go as far as possible. Thus, BFS will try almost all the possibilities and takes longer time. However, for some more difficult Sudoku, it still took a lot of time, so I tried to think about the ideas that could reduce the possibilities needed to be guessed for each candidate. Therefore, I tried to put the elimination method after the candidate was popped out in BFS and DFS. It could skip lots of wrong guessed, so it was able to decrease the time taken significantly.

However, when I tried the hardest Sudoku which contains 17 number initially, it still took about 60s to finish it. In order to reduce the time taken more further, we have tried to create a function to reduce the number of possible integers could be put into the cell with 0 instead of trying all the integers from 1 to 9. At the same time, trying to find a new way to reduce the time spent on checking if the Sudoku is valid. In this code, we have a function called zeroposition to go through all the cells in the Sudoku and record all the indices of the cell with zero. Besides, there is a function called possibility to calculate the possible integers can be put in for each cell with zero. At first, it will create a list with 1-9 integers in it. Then it will check the row and column which the cell belongs to, if some integers have already appeared in the other cells in the row and column, it will remove the integers in the list which equal to those integers. Next, it will calculate the box number for this cell and check the integers in the box and remove the integers in the list which equal to the integers appeared already in the box. After these operations, the integers left in the list are the only possible values for this cell. Then, we need to do the DFS function, this time, it will be different. We will have 0 for the value of m at the beginning. If the m equals the length of the zerolist, or the number of zeroes in the original Sudoku. It will return True. For each index, it will create a numlist which contains all the possible integers can be put into the cell. The validness of the Sudoku is done by checking the if numlist is empty. If it is empty, then it means that the Sudoku is not correct since it is not completed yet. If the Sudoku is correct, then it will go to the next position of cell with zero, the m will increase by 1. On the other hand, if the Sudoku is not correct, it will try the other possible integers in the numlist. If all the values are wrong, it will make this cell zero and recurs backward to the previous loop and try the number to the previous cell again. At the end, after all of cell in the Sudoku is finished, it will return True and all the previous function call will also return True. Then, the function is finished and our Sudoku has the correct solution.

Although the time taken for this method is random between 2 and 10 seconds, when the number of blank cells increases, the time taken will still remain in this range. In different situations, these methods have different performance. As a result, we need to choose the most efficient one when we meet different Sudoku.