Report PWS

A competitive sudoku game

A square puzzle with numbers and squares

Description automatically generatedA logo of a company

Description automatically generatedA screenshot of a computer

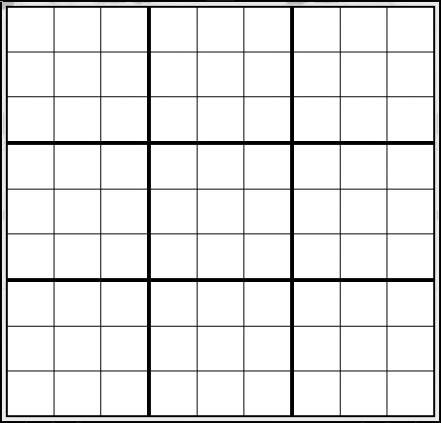
Description automatically generated

Index

* [Introduction](#Introduction)
* [How did we choose our PWS topic](#How_did_we_choose_our_PWS_topic)
  + Choosing subject and topic
  + Inspirations for PWS
* [The choices we made regarding PWS](#Choices_PWS)
* [Code Analysis – Interface](#Analysis_interface)
* [Code Analysis – Algorithms](#Analysis_Algorithms)
* [Discussion](#Discussion)
* [Reflection](#Reflection)
  + What we should have done/changed
  + Unused techniques
* [Conclusion](#Conclusion)
* [Sources](#Sources)

Introduction

Have you ever played the puzzles on a newspaper, like word finders or mazes or maybe even Sudokus? Sudoku is a single-person game about finding the solution in a 9x9 grid with 9 squares each square is a 3x3 sector, the objective of the game is to fill the 9x9 grid with numbers 1 through 9. However, you have to adhere to three rules to be allowed to place a number:



1

1

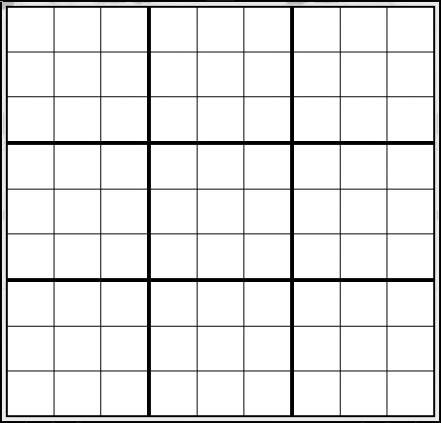
1

First, you cannot play a number in a row if there is a number in the row with the same value as the number you chose, for example (Board 1), you play a 1 in the second row (red is the newest move), but there already is a 1 in the same row, then you played an illegal move and it must be retracted.

Secondly, just like the first rule you cannot play a number in a column if there is one with an equal value. There is a visual aid in the same example as for the first rule (blue is the move for this example).

Board 1

Thirdly, inside a 3x3 square of which there are 9 you cannot have the same number twice, so in the example (Board 2), it would be illegal to play said move (red is the newest move). However, it would be legal to move the red 1 a single square to the right as that would be a different 3x3 square that does not yet contain a 1.



1

1

These simple rules can lead to very tricky puzzles that can keep you busy for hours. For our PWS we decided to make sudoku algorithms for a variation on the Sudoku puzzles. This variation belonged to a coding competition known as the Code Cup (CoDECUP 2025 - Past Contests, n.d.), the Code Cup is a competition that hosts a competition each year about a new game, which you have to write an algorithm for and send it in. The competition that we based our PWS on in the 2024 competition, is about Sudoku with a twist you play against another player, instead of by yourself with a predetermined solution, and there is one added rule: every move has to reduce the maximum amount of solutions.

Board 2

How did we choose our PWS topic?

The PWS started on Tuesday, August 27th. There, we formed a group. Our first objective as a group was to decide on a subject. The choice quickly fell between physics, science, and computer science. We chose computer science due to our interest in games.

The decision on what we would do was a greater hurdle, as computer science is a grand forest of which we did not have a map. With the guidance of the teachers there, we had three fields of computer science that we decided on: AI (the art of teaching computers), building simulations or algorithms. The AI idea was a top contender but was removed due to the difficulty of running one, this made it unusable for those in our group who did not have a pc. The simulations were going to be based on a YouTuber called Primer, but we wanted to be creative so we ultimately decided against the simulations and went with algorithms.

To make an algorithm we first had to find a game or topic that we could create an algorithm for, sadly this took us a longer time than we would like to admit, part of the problem came from our stubbornness in making an algorithm for something that did not have one yet or at the least, one that was not yet solved. We first came across the prisoner's dilemma and a video about strategy about a variant of the dilemma made by Veritasium (Veritasium, 2023), but decided against it, because they basically gave the solution. We then stumbled upon a YouTube video about a game called Chomp (William Spaniel, 2022), a simple game about avoiding eating a deadly cholate and forcing your opponent to eat it instead, it was a good idea for the PWS or so we thought as Cas, due to brute force found a pattern in a few days. Then we got advice from our guidance counsellor Mr. Gnodde to look at Code Cup. Furthermore, if we wanted we could have joined the competition that would be hosted in 2025, but after reading the instructions of the chosen game this year we gave up the idea of participating because we had trouble even understanding how to make a legal move. Finally, we discovered the 2024 competition, which was about sudoku a game that we all knew, and the twist on the game being turned into a 1v1 was intriguing enough to make it our PWS topic.

The Choices we made regarding the PWS

For the algorithm, we chose to program in C#, because we were familiar with the coding language and several of us had used it regarding last year’s final project.

We started with the idea of making an interface of the board, to which we could add the algorithms and one big algorithm, which we would pit against one of the algorithms of the competition. We were going to acquire the competition algorithms by contacting one of the individuals who participated via email or social media, to our chagrin that did not end up happening, as we only found a few of them, and those turned out to be in coding languages that we had no experience in and as we researched why none were in C# we found that the Code Cup simply did not allow C# code to be submitted, so we wasted a lot of time on a fleeting idea. We compromised by making two simpler and smaller algorithms and letting them duke it out.

//add choices regarding algorithms

A screenshot of a computer

Description automatically generatedCode Analysis – Interface

Our interface consists of 5 main sections of code:

1. **The A, B, C & human algorithms:** in “AlgorithmA” and “AlgorithmB” we have our algorithms, which will be discussed in the other sections of this report and in “AlgorithmHuman” is the code that would allow a person to play themselves instead of needing to build a AI to play.
2. **Cell:** this section keeps track of certain variables and adds the made moves to a list, while also making sure no one can play moves on squares with a number on it.

Picture 1

1. **Player:** is used to keep track of the players
2. **Rules:** this is the coded version of the rules, which where stated in the introduction and the competition rule of reducing the amount of possibilities every move
3. **Turn:** this is what allows the two players to act after one another and not twice and it keeps track of what move a player made and if they claimed a unique solution and it also has a turn counter.

A computer code on a black background

Description automatically generated*Now we will explain the code based on the sections (section 1 will be in the other part of the code analysis):*

Picture 2

Cell:

*The code on the right is the code we used for this section*

For this section the value and the validity of a move in a peculiar cell is recorded with the [ObservalbleProperty] and it keeps track if this value is changed.

The \_possibleValues keeps a list of the possible values of a cell, so that we cannot have a cell with the number 20 in it or a word and the private bool \_uniqueSolutionClaimed looks if an algorithm claimed a unique solution. The public strings keep track of each cells x and y coordinate.

A black rectangular object with a white border

Description automatically generatedThe last part of this section sends the acquired possible values and if you claimed an unique solution to the rest of the code, while putting the possible values in a list, that is visualised on the right hand side of picture 1 and will keep track of who and when that move was played using the turn code. (picture 2)

Picture 3

Player:

This section is the shortest of them all. The IsHuman bool looks to see if one of the algorithms is a human and then gets the name and score of that person. (picture 3)

Rules:

*A computer screen with text on it

Description automatically generatedThis is the largest part of the code and is a code version of the sudoku ruleset:*

The first part of the code is checking if the player made a move that is legal on an empty board. This is done by the if functions to check if the cell the player wants to fill in is even on the board and if the value that was selected is between 1 and 9

Picture 4

the second part of the code is intended to check when a player claims an unique solution meaning that they claim that the sudoku is now solved. It does this by first checking if a player claimed an unique solution using an if statement, then it creates a duplicate of the current board state and checks left to right and down to up if there are any cells with only one possible value left and if that is true it fills that cell with that value, then it updates the board with the added value, which might make it so that other cells now also have a single solution left, so when it fills in a value of a single value cell it starts over to check for any new cells with a single possible value. It repeats this process until it cannot continue . (picture 5)

Picture 5

A computer screen shot of a program code

Description automatically generatedAfter it has finished with this process of filling the board as far as it can, while following the rules of the game, it continues to the next part, where it now checks the result of the process. This is done by checking if all the cells of the duplicate are now filled, if so the claim must be true and the player who made the claim wins. However, if the board is not filled, the claim must be false, so the player is now declared the loser of the game (image 6).

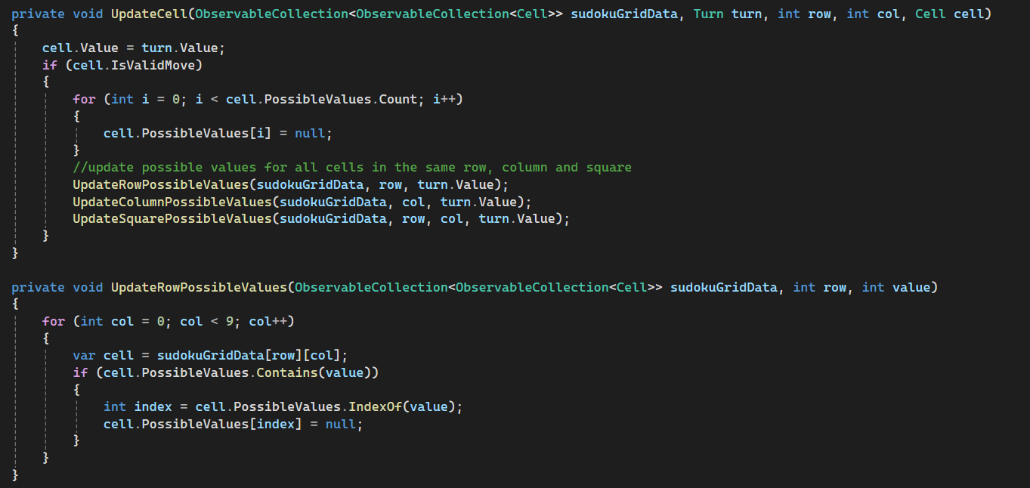
Picture 6

A screen shot of a computer program

Description automatically generatedThe last part of the code is updating the board with the new cell values, row and column possible values and the possible values within a square (image 7). Finally, this information is saved on a new gridA screen shot of a computer program

Description automatically generated and this data is then with the return command sent to the rest of the code to continue the program and start a new turn (image 8).

Picture 8



Picture 7

A computer screen shot of a code

Description automatically generatedThe last section of our code is the turn file, this code is what registers a turn, when it is finished, then using the rules code if it is legal, if an unique claim was made and if that claim was valid. It does this via the get command, as seen in picture 9. The IsValid bool is a continuation of the validity of the claim it checks if the move was valid and there was no unique solution claim or the move and the claim were valid.

Picture 9

The public string TurnAsString, which checks the value of the ClaimUniqueSolution variable. If it is 0 it means that the player did not put in a move, but directly claimed an unique solution, but if it has another value that is not null, it adds the move to the string. The total string is then put into the list that is on the side of picture 1.

A computer screen shot of a program code

Description automatically generatedA computer screen shot of white text

Description automatically generatedthe next and final part of the code, translates a human move, which consists of the same naming convention that is shown in the list to a person when the algorithms are making moves, for example: Ac4 is how a human could answer, but for the algorithm it is split into a x and y coordinate. It translates the human move by making the first part the y value (A in our example), the second part the x value (c in our example), and the value of the square is the third part is the number that becomes the value of the cell that has been selected with the x and y coordinates. To claim a unique solution, the move length has to be longer than three and the fourth move is an exclamation mark, so if you were tp claim an unique solution in our example it would be Ac4!.

Picture 10

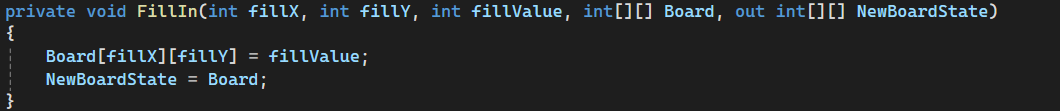
Code Analysis – Algorithms

*This part will individually break down the algorithms that we have made*

A screen shot of a computer

Description automatically generatedFirst algorithm A:

Image 1

It starts with filling in a list with the current board state for later use (image 1).

A screen shot of a computer

Description automatically generatedA computer screen with white and blue text

Description automatically generatedAfter this it fills in all possible values of every cell, so it creates a list for all 81 cells and attaches the number 1-9 to the cells (image 2).

Image 3

Image 2

A computer screen shot of a program

Description automatically generatedThe removePossibilites void creates a temporary board that will be used to keep track of all possibilities and the values that are on the board, it does this by going cell by cell and checking if there is a number and keeping track of them with the variables: value, x, y, found and instances, plus i and e, which are counters for the foreach loops (image 3). Found is used to remember where there where values, which is filled in using the FillValue function (image 4).

Image 4

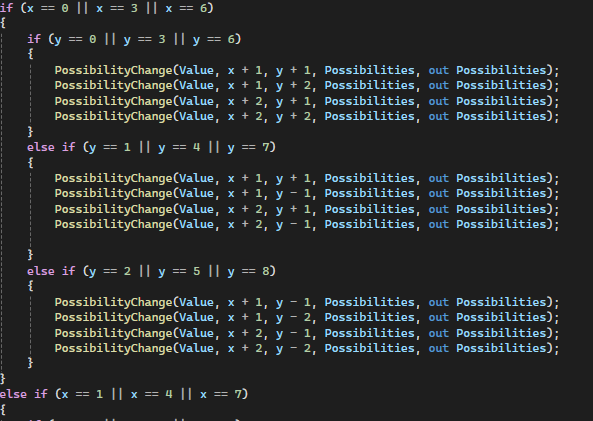
Furthermore, there are there are two other voids which remove the possibilities from the possibility list in from image 1: the possibility change function and the square check function

Image 5

The possibility change function is within the FillValue void and is used to not only remove the values from the list from an exact cell, but also removes the possibilities from all rows and column, it does so in groups of three, because whether it is the 1st row or 4th row or 7th row they all belong to the first row of a square, so they are identical same goes for the 2nd , 5th and 8th row and the 3rd, 6th and 8th row, as well as the columns (image 5).

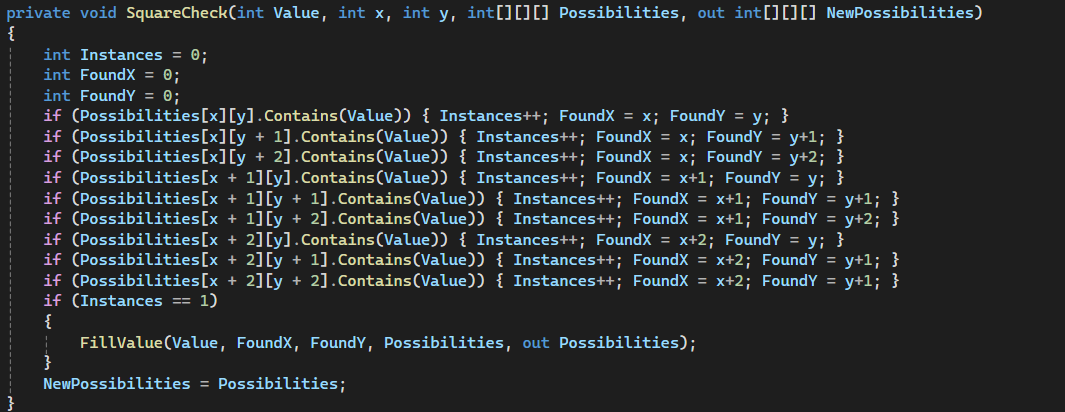
The square check function checks each cell within a square for numbers and removes them from the possibilities using the FillValue function, again using the variable instances to keep track of any numbers and FoundX and FoundY to keep track of their position (image 6).

Image 6

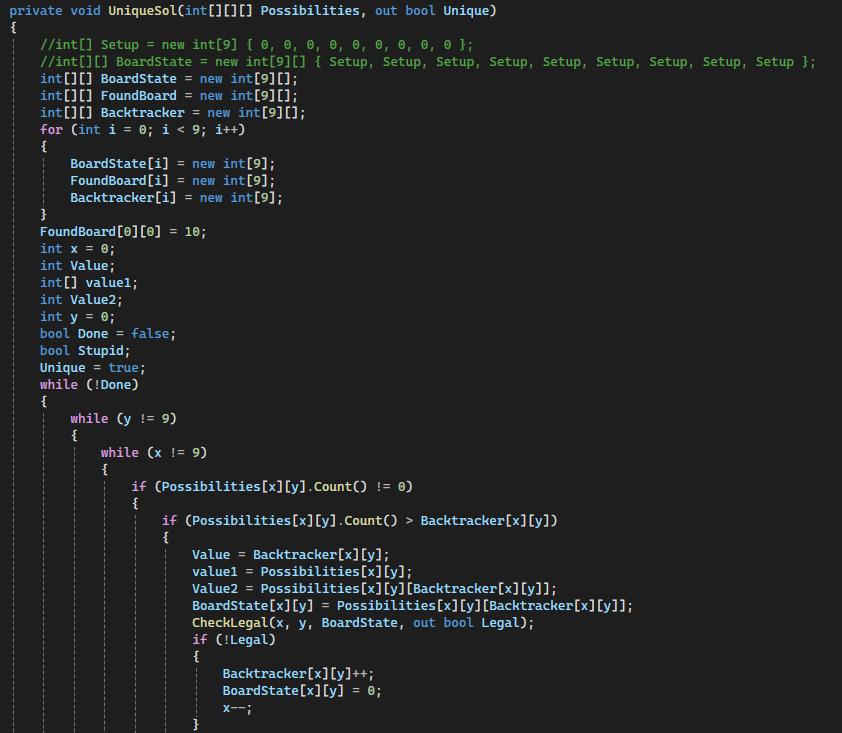
The last two functions are used to make a decision and check whether the board has a unique solution.

Image 7

UniqueSol uses backtracking to go through the now completed list of possibilities and while checking if the moves are legal, tries to solve the current board state by filling in cell that only have 1 possible value and then updating all the cells with the new possibilities and continuing the backtracking. It continues this process until it either cannot place anymore because the board is full or it does not have any cells left with only one possible value, in the first case it claims that there is an unique solution and in the other scenario it does not claim an unique solution and plays the chosen x and y (part of this code is shown in image 7).

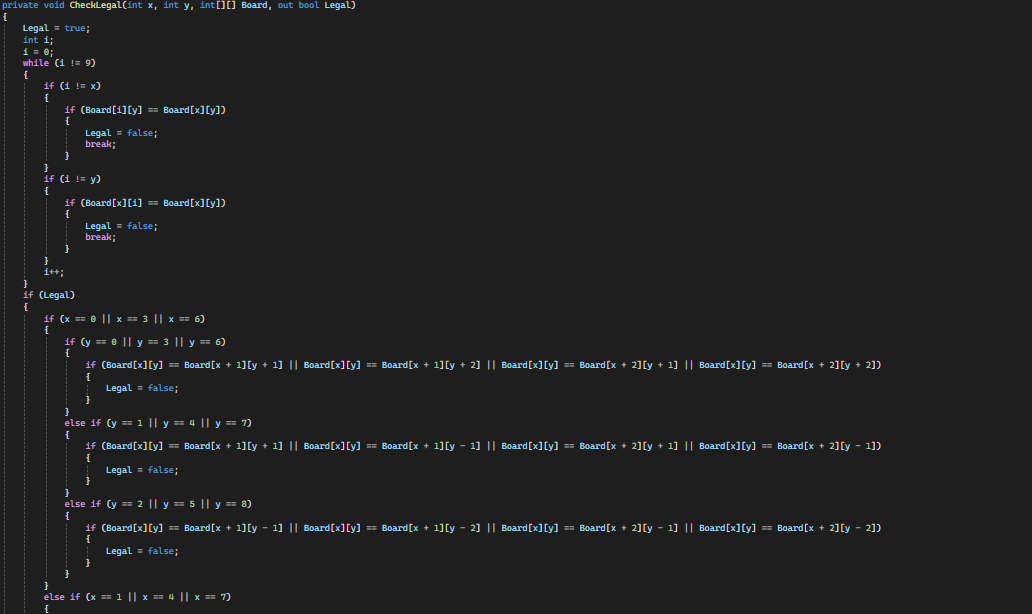
The validity of a move is checked in the last void CheckLegal and it uses the created board from removePossibilities and checks for legal moves using a similar method to that of the possibility change function (image 8).

Image 8

In the end its move is filling in the first possible cell starting at the top left cell with the first possible value from 1-9 and makes sure not to play any moves that are illegal.

A computer screen shot of a program code

Description automatically generated

Algorithm B has primarily focused on speed and simplicity.

Image 9

This algorithm uses a bit of randomness to find a valid move (image 9). It keeps track of two lists: TurnsMatchX and TurnsMatchY, it does this to check if the initial chosen position is free, aside from the position it keeps a list of the possible values for the randomly chosen cell and removes options from this list. It starts with checking what values the column already contain, then the row and finally the square (image 10).

Lastly, the code checks if the possible values do not go below 2 and if it does it again has to rechoose a new cell, but if the randomly chosen cell still has 3 or more possibilities and adhered to all the checks, it is finalized as the move that it makes this turn (image 11 & 12)

A computer screen shot of a code

Description automatically generated

A computer screen shot of a program code

Description automatically generatedA computer screen shot of a program

Description automatically generatedA computer screen shot of a program code

Description automatically generated

Image 12

Image 11

Image 10

Algorithm c is based on removing the most possibilities every single time. It starts by creating a list with every cell and the possibilities 1-9 for each (image 13).

Image 13

The first thing the code does when it comes to deciding what it will do in the turn is checking if the current board is unique, this is done by creating a temporary copy of the board and then filling in all the values on the board and then using backtracking similar to algorithm A to fill in the board and if it is full it then declares the board as solved and claims a unique solution (image 14). The backtracking uses the code in image 15 and image 16 to keep updating the possible values of each cell so that the backtracking can find more mono valued cells.

Image 15 is about removing the possibilities of cells that have numbers in the same row or column as them. The code checks for values by order of them being played in the turns and removes that number from all the other cells in the same row or column. Similarly, the code in image 16 does the same but then for the 3 by 3 squares in the sudoku.

A computer screen shot of a code

Description automatically generatedA computer screen shot of a code

Description automatically generatedA screenshot of a computer program

Description automatically generatedThe important part of the code is now to decide on one of the many possible moves and make sure that it is legal. The IsMoveValid bool is there to double check the possible values list and make sure that the move that is eventually chosen is legal (image 17).

Image 16

Image 15

Image 14

Image 17

As to how the move is chosen the code has a few rules that it should adhere to:

If the algorithm ever considers a move that makes it possible for the opponent to make a move and claim uniqueness at the same time (thereby winning) it should not make that move and consider the next possible move that best fits the described criteria

the algorithm should always check if the move it considers next causes to sudoku to become unique, because if it does it should make that move and put a ! behind it to make that move and claim uniqueness at the same time and win

if the only possible move/moves cause the sudoku to become solvable in 1 move it should still make that move since it has no other choice

A computer screen with text

Description automatically generated

It follows these rules using the private GetBestMove and the functions that this part uses the IsMoveValid code, CalculateImpact and the double check GetNextBestMove.

GetBestMove uses the functions CalculateImpact and IsMoveValid to decide on a move and ranks them based on the calculated impact of each move, then the move with the most impact and that is legal is returned as the chosen move (image 18).

The impact is calculated by the amount of possible values that a move would reduce and this is repeated for all the possible moves that could be played (image 19).

Lastly, after the best move has been decided the code checks if the move is valid again and tries to find a better than the chosen move to avoid playing a subpar move (image 20).

A computer screen shot of a program

Description automatically generatedEach of the algorithms has a similar, yet unique idea on how to optimally solve this problem. A focusses on creating temporary board states to simulate events, while B focusses on speed and perfecting a simple idea, and C tries to play moves with the biggest impact to the game, while not giving the opponent a chance to get an unique solution.

Image 18

A screen shot of a computer program

Description automatically generatedA computer screen with text on it

Description automatically generated

Image 20

Image 19

Discussion

Our PWS consists of a nice interface with algorithms that could best any human player in this game, yet it is still flawed in several ways. First, We would like to mention that there is no perfect solution for an algorithm that has to do it within a reasonable time, and even participants from the competition did not know how to create a program for the early game. However, compared to theirs, we still could have integrated more techniques.

Secondly, the interface we made is only compatible with C#, which is not an issue for us as that was the only real coding language that we could use with adequate proficiency to make a functioning project. Still, it could pose issues if we wanted to utilize a coding language more directed at algorithms or other people to join in.

Thirdly, the interface is workable but not that easy to understand. It does include explanations and proper naming of variables, but it still is a massive document with a lot of code that all has a different purpose, so for the group members that did not make the interface, it is tough to work with it.

Lastly, the algorithms are very different in size and complexity, but have not had rigorous testing to find the winner between the three. Furthermore, it is difficult to compare their strength to that of other codes, as the only codes that we know of that have been made under our rules are from the Code Cup and those are not in C#, so comparison is difficult and time consuming. This means that the result is difficult to judge on its efficiency and its intelligence.

Reflection

If we were to redo our PWS we would have made a few changes to our actions and behaviour mostly regarding communication and time management. Firstly, we did not have proper deadlines set for each part of the PWS, which resulted in us working progressively more as we got closer to the final deadline set by the school, we should have created deadlines that allowed us more than enough time to finish the project at a moderate pace instead of having to rush the final result.

Secondly, the communication between each other and our guidance counsellor was both lacking and when there was communication the responses often were delayed, which made fixing issues in the code a long and tiresome process. Furthermore, the lack of communication also was with our guidance counsellor made the PWS more difficult than necessary, we should have added him to our main way of communication WhatsApp or we should have used teams as our main way of communication.

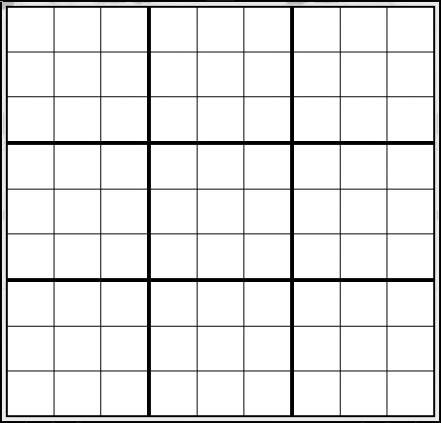
Thirdly, our initial idea of using the code of participants and using our own algorithm to beat them was a failure, but we would have liked to have gotten this to work, so if we had more time, we could have attempted to manually translate the code to C# or recreate the strategy perfectly in C#. It would take a considerable amount of time to translate the code, but we believe it would be a better representation of our algorithm if we beat one of the competition’s participants.

Lastly, the algorithms lack a few of the advanced techniques of sudoku, mostly due to the difficulty of implementing them and how niche some of them are, but if we had more time we would have liked to implement more of the techniques we spent time researching. To still show time was spent on research even if we could not implement it there will be a list of sudoku techniques with short simplified explanations:

1

1

1



1

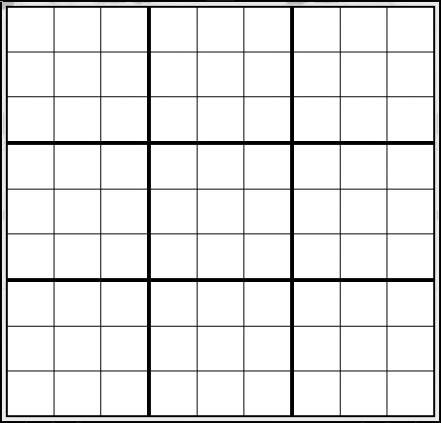
Board 3

**The X-wing**: if there is an imaginary square or rectangle of four possible positions of a single value, as depicted in board 3 (red is possible moves), you can group the top-left and bottom-right possibilities and the others to bring the four possibilities to 2 (green and blue), because if the top-left is correct the top-right and bottom-left have to be false because they cannot be in the same row or column as a number of equal value (*Х-wing - Sudoku Technique*, n.d.).

1

1

1



1

Board 4

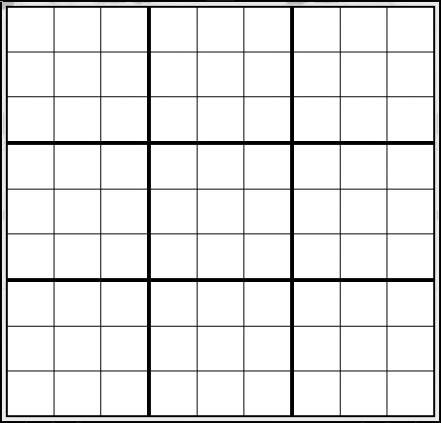
**The swordfish:** this is a more complicated version of the x-wing and is not only applicable to squares and rectangles but can be used as long as the numbers are in the same column or row. On board 4 it is demonstrated, by grouping the numbers that are not on the same row or column (blue or red) we can now test one of the grouped numbers if one is correct, then all others in the group go as well, because the other number would violate the first or second rule, so there can never be a red number correct and a blue one (*Sudoku Swordfish Strategy Explained - SudokuOnline.io*, n.d.).

1

1

**The XY-wing:** the XY-wing is one of the more complex sudoku techniques. It utilizes two terms called the pivot and the pincers, and it is used to eliminate possibilities from spaces with exactly two numbers that could go in a specific position. For this technique, you pick bi-valued cells that are buddy cells of the pivot, meaning they are in the same column, row, or square as your pivot point (see Board 5; 1,8 is the pivot point, and the blue & red are the pincers). The pivot contains numbers X and Y, while one of the pincers contains X and Z, while the other contains Y and Z (X, Y, and Z are numbers with different values). In the example, X = 1 and Y = 8, while our Z = 7, if our pivot contains either number, then blue or red would be 7, so any cell that has both blue and red as buddy cells cannot have 7 (in this case) as a possibility, as that would break one of the rules, so the 7 in the top left one is not a possibility, so there must be an 8 in there, so then the rest is solved (*Sudoku Y Wing | Sudoku Xy Wing Explained*, n.d.).

7,8



8,7

1,8

7,1

Board 5

**Nishio:** this technique got its name from a professional puzzler Tetsuya Nishio and this technique is regarded as a last resort, as it takes a long time because it uses guessing as a basis. How it works is that a player takes a bi-valued square, assumes it is correct, and continues on with the sudoku if the guess is incorrect, and this assumption leads to an unsolvable situation, then the player has to return to the point of the initial guess losing all progress and then takes the other option knowing that it is the correct answer (*Techniques: Nishio - Sudoku of the Day*, n.d.).

Conclusion

The PWS aimed to create our own version of the 2024 Code Cup competition by creating a functioning interface with code that detects illegal moves and an algorithm that could compete with those of the competition, sadly the last goal was not achievable, due to us not being able to use the code that we found, so in the end we switched to several smaller algorithms that can compete against each other. Furthermore, we added a feature were a human player can either participate against another person or one of the algorithms.

We are content with the final result, but there was so much more that we still could have added that it almost feels incomplete to hand it in. The time it took take to finish everything we handed in now was severely underestimated by us and is why we had to rush the final weeks. We would have liked to have translated one of the algorithms we found from participants to C# or to have recreated it, because now we cannot compare our algorithms to others that follow the same rules that ours do. Nevertheless, we got as far as we did and will be presenting this on the 6th of February.

Sources

*CoDECUP 2025 - Past Contests. (n.d.). https://www.codecup.nl/archives.php*

*Sudoku Swordfish strategy explained - SudokuOnline.io*. (n.d.). Sudoku Swordfish Strategy Explained - SudokuOnline.io. https://www.sudokuonline.io/tips/sudoku-swordfish-strategy

*Techniques: Nishio - Sudoku of the day. (n.d.). https://www.sudokuoftheday.com/techniques/nishio*

*Veritasium. (2023, December 23). What game theory reveals about life, the universe, and everything [Video]. YouTube. https://www.youtube.com/watch?v=mScpHTIi-kM*

*William Spaniel. (2022, February 3). The Long, Poisoned Chocolate Bar: A game theory puzzle [Video]. YouTube. https://www.youtube.com/watch?v=lmiPhe5zkk8*

*Х-wing - Sudoku technique*. (n.d.). https://sudoku.com/sudoku-rules/h-wing/