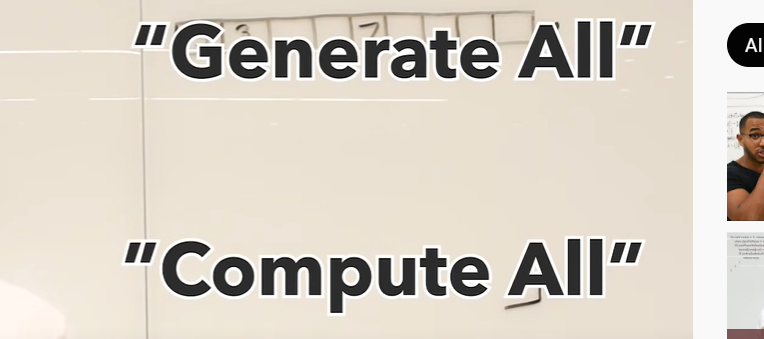
<https://www.youtube.com/watch?v=Zq4upTEaQyM&t=390s&ab_channel=BackToBackSWE>

#===================🡺

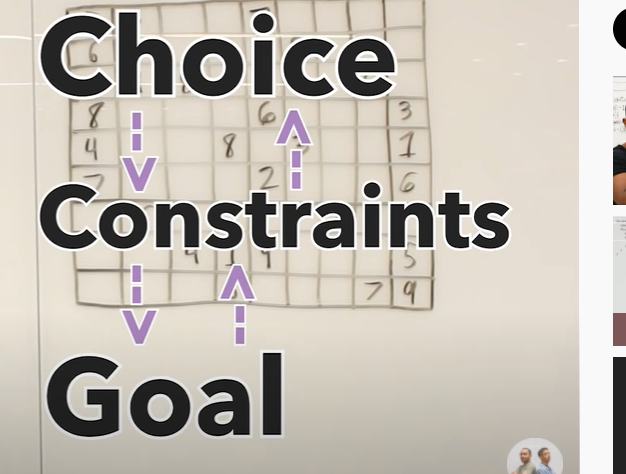
How do we know if it is a backtracking problem:

If it says:

1. Generate All
2. Compute All
3. Words employing exhaustion of decision space.



The Backtracking Blueprint: The Legendary 3 Keys To Backtracking Algorithms



3 keys:

1. Choices: Decision space to choose from. What will be the core decision space in any problem. ?
2. Constraints : Decision space is restricted somehow
3. Goals: is to do something.

# what each of these means?

# how we shape our code.

For sudoku solver

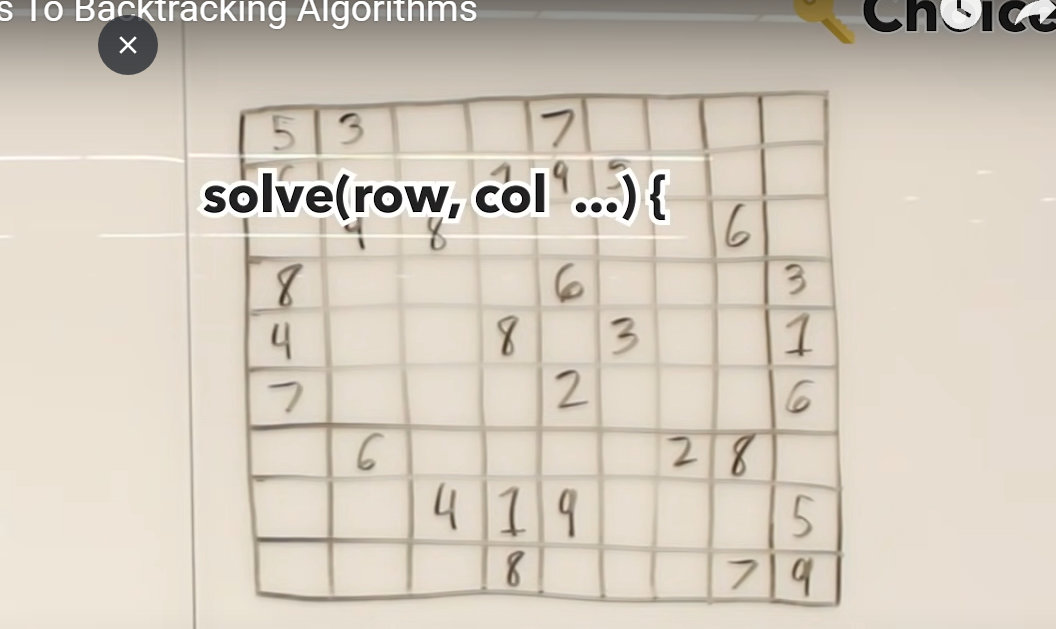
# The Choice: to choose from the number to fill.

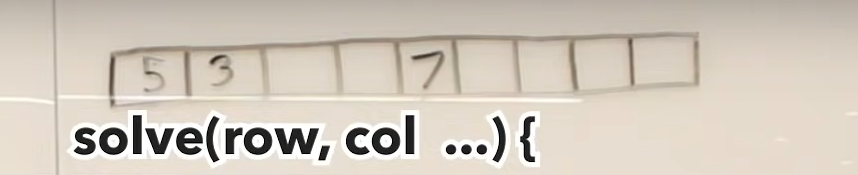
We have to fundamentally think about the core choice that we are making at each step.

We can craft a recursive function to craft a problem like this.

In sudoku board the choice is to fill a cell. So choice will be made on a cell.

Cell sits in a row and a column.





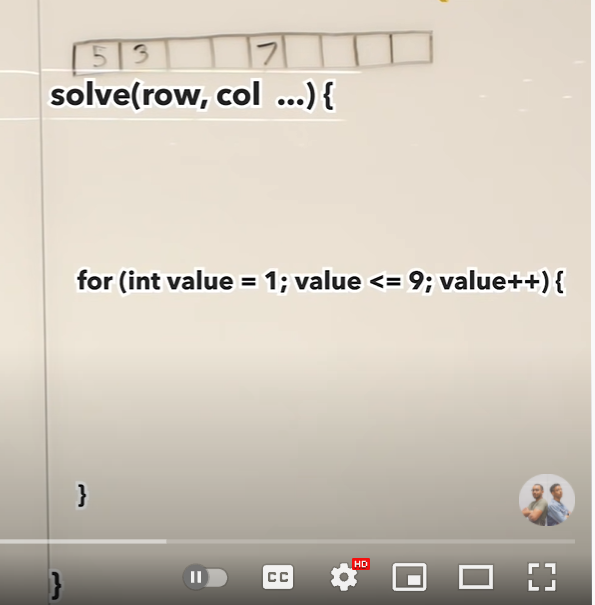
Lets first think about the first row. How can we compartmentalize this into subproblems?

We first solve all the cells in a row. And then we will solve cells in the next row and the next row.

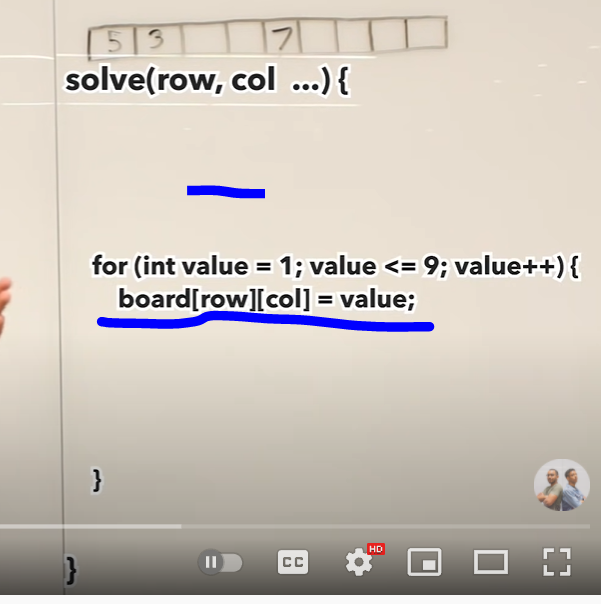
Choice: is to place a number at every single cell. Number from 1 to 9.

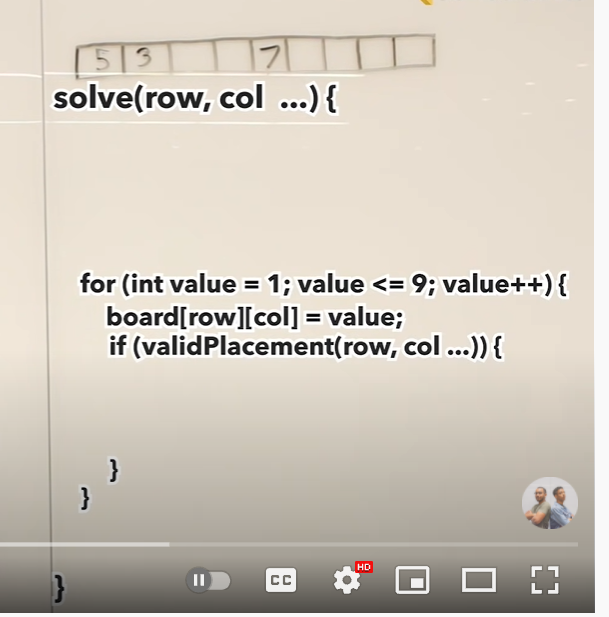
So using a for-loop lets put numbers in an empty cell.

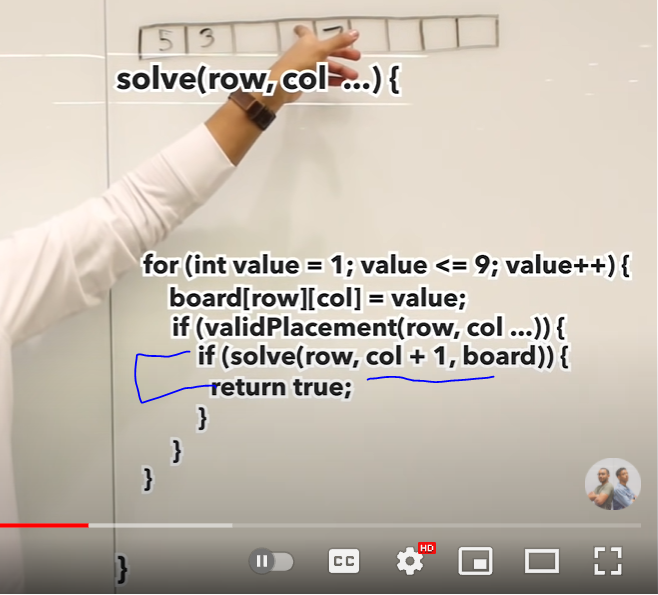
Express the decision space.. decision is to fill numbers from 1 to 9.



Now I have to place an item into the board. How is that done?







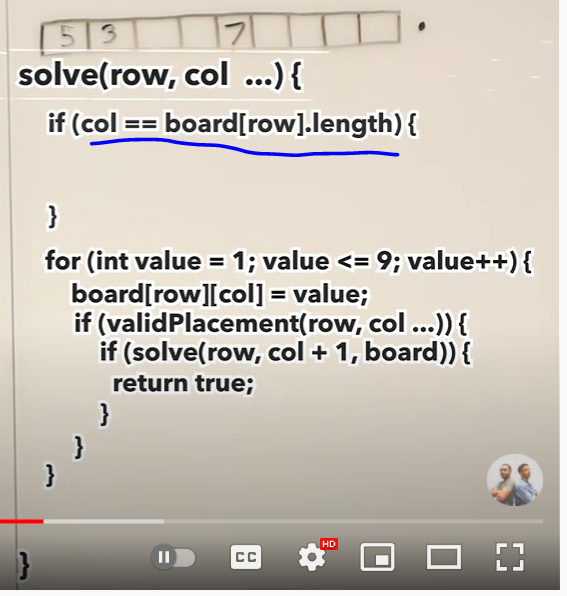
Okay.. I made a choice of putting the value. What are my constraints?

When we place a number, we can validate whole sudoku board. But we do not wanna do that. We want to validate the row and the col and the subgrid.

**So checking, row, col and a subgrid.. if the placement is valid, I recurse on it. We recurse on our decision. We make a choice on the current cell and then we recurse on the decision. If decision doesn’t work, we come back and we undo it. We make another choice.**

**# so we explore and we undo.**

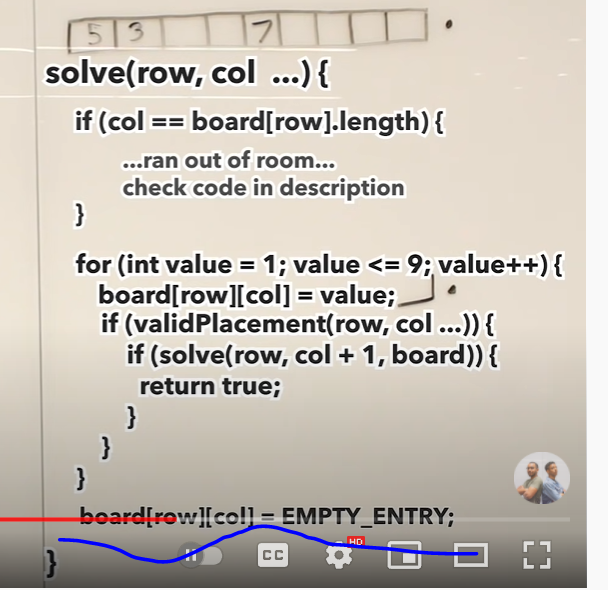
(col+1).. because if current col. Is solved next logical step would be to solve the next column.



While recursing and moving through columns. When we reach to the end of the column in first row, we have to go to the 2nd row. And that will be leading to our base case. Which is the Gioal.

Goal is to reach our base cases. So 1 base case is we have finished all the columns of the row and we move to the next row.

Now another base case would be going out of bounds on the final row. Which will mean all the rows are finished.



What if decision doesn’t work out? What do we do then? We need to eject our decision. That is done in a way marked by blue.

