We are now doing the depth-first search. In this search tree, each node represents a grid and the depth represents the current row in the grid we’re editing. So a basic idea is that when the depth reaches the grid size and doesn’t contradict the hints, then that’s the solution. This function called “dfs” has five arguments as follows.

* (depth: int) - The current 0-indexed depth of node from the root
* (state: grid) - the current grid. The rows lower than our current depth are filled with BLANK.
* (grid\_size: int) – the grid size
* (vertical\_hints: int list list) – the hints shown on the left of the Nonogram
* (horizontal\_hints: int list list) – the hints shown on top of the Nonogram

First, we generate possible rows that fit the vertical hint that corresponds to the current row using “compute\_permutations.” If we couldn’t generate any row, which is the return value of the function is an empty list, we are going to raise an exception “NotFound. ” And after, we declares one local recursive function called “verify” which checks if each possible row doesn’t contradict the hints on the top. In this function, we first do a pattern-matching with the argument, which is a list of the generated argument. If it’s empty, which means we already searched until the last possible row in the given argument list, that means we couldn’t find a solution, so raise the exception. If not, we extract the head of the argument list. We declare a variable called “current\_state,” which is a grid that derives from the alredy existing grid we’re given as an argument of “dfs.” And also, it has been added one possible row generated from “compute\_permutations” to the current depth of row. And the rest of the grid down below that are filled with BLANK, which means uninitialized. The implementation here is a bit complicated because we can’t normally reassign a value to the existing list. So, we create a new list based on this idea using List utility functions. After that, we pass this new grid to “validate” function to check if this grid contradicts the hints on the top. If not, we are going to call “dfs” again one row deeper. We do this procedure to all the generated row candidates. And importantly, if the depth equals to the grid size minus one and the grid is valid, we print out the final grid.