Basically we write these functions:

1. main() method
2. bool solveProblem()
3. int/bool solveProblemUntil()
4. bool isSafe()
5. Void printSolution()

Flow:--

Main()-🡪 solveProblem()🡪solveProblemUntil()🡪 IsSafe()

|

|

printSolution()

main()

{

1.//initialization of boar,starting value,starting index

2.//call solveProblem()

3.//return

}

--------------------

bool solveProblem(starting condition)

{

1.//Intialization of solution matrix or array

2.//If( solveProglemUntil(// with starting index)== False)

{

//print solution does not exist

Return false;

}

3.//printSolution()

return true;

}

-------------------------------

Bool/int solveProblemUntil(//initialize value with increase of 1 call by solveProblemUntil, for first time call by solveProblem)

{

1.//base case . – if reached upto last condition or completed the problem return true

If(v==V) or if(col>=n) such kind of codition

Return true;

---

2.//use looping condition or call for all possible case

In each case{

2.1//if(isSafe(with changed value of new case)==true)

{

2.2//modify the value

2.3//call next the same function with modified value and with one increased index like ------- if(sovleProblemUntil(index+1 or row+1 or any new condition)==true)

Return true;//// if u will get final solution true then here u will return true.and this will give a solution for u.

2.4//since this is failure so change again 2.2 value with previous one so that it can not affect actual values for new case

}

3.// After increasing value u didn’t get the solution so now return here false so that u can move to another possible case.

Return false; // backtrack

}

}

--------------------------

Bool isSafe(// with modified value for checking that this modified move is valid or not for given condtions)

{

//we can do this in 2 way either use if condition for true validating all condition in if statement or you can use if statement for all false condition .Generally use if condition for those (either for true or false) which one is easy to write and have less number of condtions.

Check for one by one for all conditions if true for all then return true

Else

Return false

}

------

Void printSolution(solution matrix or aray)

{

Print solution matrix or array which is define in function no2 i.e SolveProblem()

}

Links:-

<https://www.hackerearth.com/practice/basic-programming/recursion/recursion-and-backtracking/tutorial/>

<https://en.wikibooks.org/wiki/Algorithms/Backtracking>

<https://www.topcoder.com/community/data-science/data-science-tutorials/computational-complexity-section-1/>

<https://www.topcoder.com/community/data-science/data-science-tutorials/computational-complexity-section-2/>

<https://www.topcoder.com/community/data-science/data-science-tutorials/an-introduction-to-recursion-part-1/>

<https://www.topcoder.com/community/data-science/data-science-tutorials/an-introduction-to-recursion-part-2/>