DS PROJECT REPORT

***ACKNOWLEDGEMENT***

I am over helmed in all humbleness and gratefulness to acknowledge my depth to all those who have helped us to put these ideas, well above the level of simplicity and into something concrete. I would like to express my special thanks of gratitude to my teacher as well as our teacher Mrs. Sherry Garg who gave me the golden opportunity to do this wonderful project , which also helped me in doing a lot of Research and i came to know about so many new things. I am really thankful to them. We would also be thankful to our college and other faculty of this course of JAYPEE INSTITUTE OF INFROMATION TECHNOLOGY for providing all the required facilities in completion of this project. I would like to thank my parents who helped me a lot in gathering different information, collecting data and guiding me from time to time in making this project, despite of their busy schedules, they gave me different ideas in making this project unique.

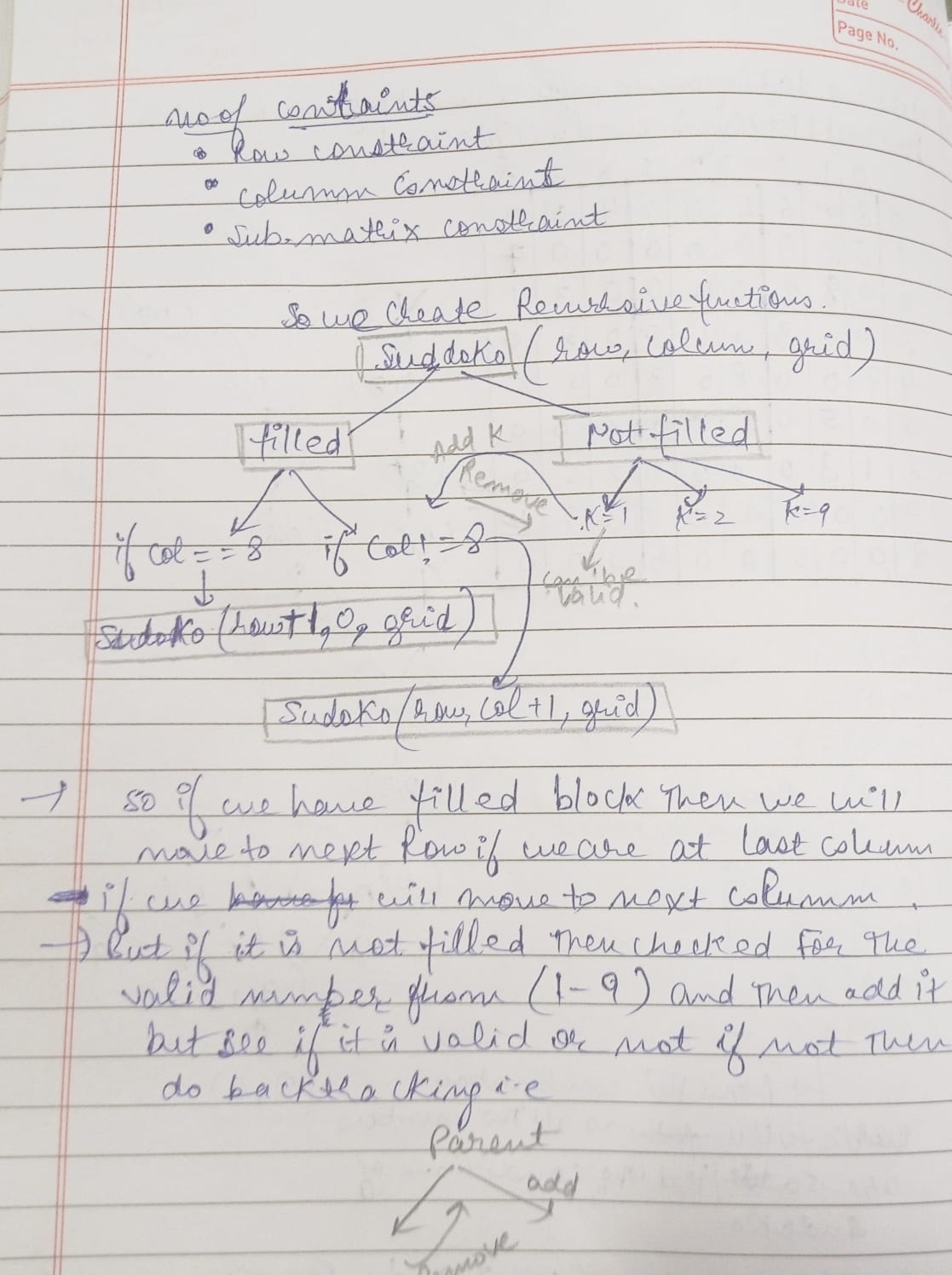
***Problem statement:***

We are making a joyful and mind cracking game named Sudoku which consists of **a 9 × 9 grid with numbers appearing in some of the squares**. The objective of the puzzle is to fill the remaining squares, using all the numbers 1–9 exactly once in each row, column, and the nine 3 × 3 sub grids objective:

***objective***

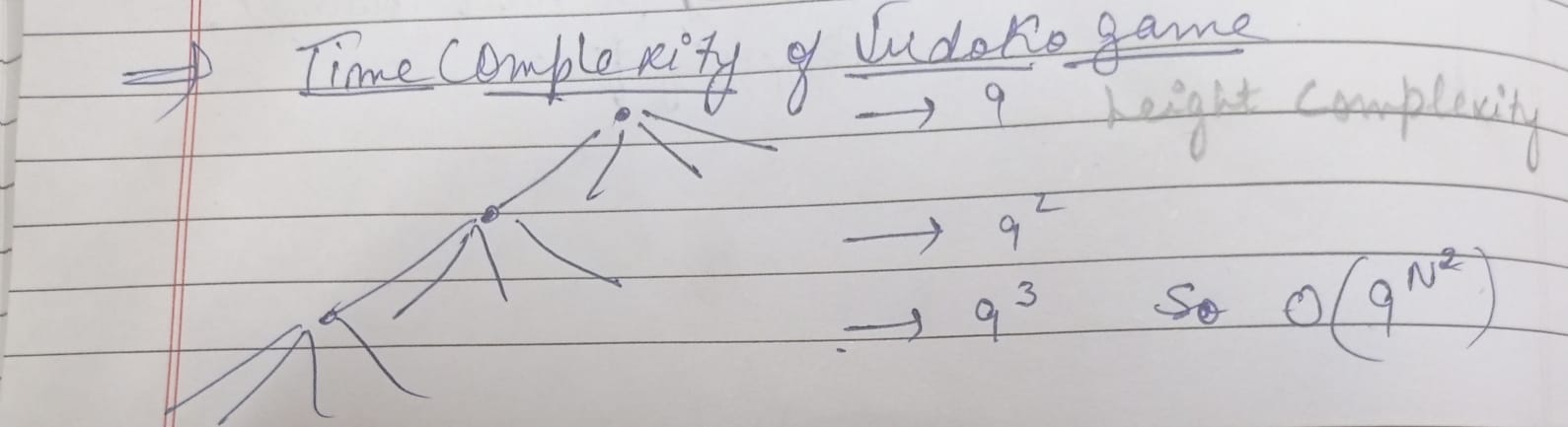
It is based on the objective of **backtracking algorithm**  which is a **recursive algorithm** that attempts to solve a given problem by testing all possible paths towards a solution until a solution is found. Each time a path is tested, if a solution is not found, the algorithm **backtracks** to test another possible path and so on till a solution is found or all paths have been tested.

The naive approach is to generate all possible configurations of numbers from 1 to 9 to fill the empty cells. Try every configuration one by one until the correct configuration is found, i.e. for every unassigned position fill the position with a number from 1 to 9. After filling all the unassigned position check if the matrix is safe or not. If safe print else recurs for other cases.



***Steps to be followed***

1. Create a function that will store the Sudoku game in it now then onwards transverse it till the end by checking the conditions.
2. Create a recursive function that takes a grid and the current row and column index.
3. Check some base cases. If the index is at the end of the matrix, i.e. i=N-1 and j=N then check if the grid is safe or not, if safe print the grid and return true else return false.
4. The other base case is when the value of column is N, i.e j = N, then we have to move to next row, i.e. i++ and j = 0 but if it is at same row and has valid input then put required number there and increase the count for column number.
5. If the current index is not assigned then fill the element from 1 to 9 and recursively call for all 9 cases with the index of next element, i.e. i, j+1. if the recursive call returns true then break the loop and return true.
6. if the current index is assigned then call the recursive function with index of next element, i.e. i, j+1
7. if the current index is not being able to assigned then transverse back to the previous indexes and then put the another possible combination in it ,which is basically known as backtracking .
8. The new change now will be particularly be best fitted to the new indexes as we choose the other possible combination as previous one got invalid.
9. When we will go back through backtracking then we have to again erase the previous numbers being given to the indexes ,as we have change the possible combinations so we assign them again 0
10. We have to also focus on the sub matrix of the Sudoku game also i.e in that also the 1 to 9 numbers must be there ,don’t replicate there also and don’t leave any number missing there which we choose could get its starting sub matrix index.



***Name of functions***

***Of our project***

Initially there will be given the Sudoku game in which some blank spaces would be there which identifies that there number has to be filled. so as according to the rules of the game the numbers will be consecutively being filled following the steps..

**i.e no rows ,no column and sub matrix of the Sudoku would contain the same numbers .**

**Numbers to be filled is between 0 to 9.**

For this various functions is made:

1)Solve function->

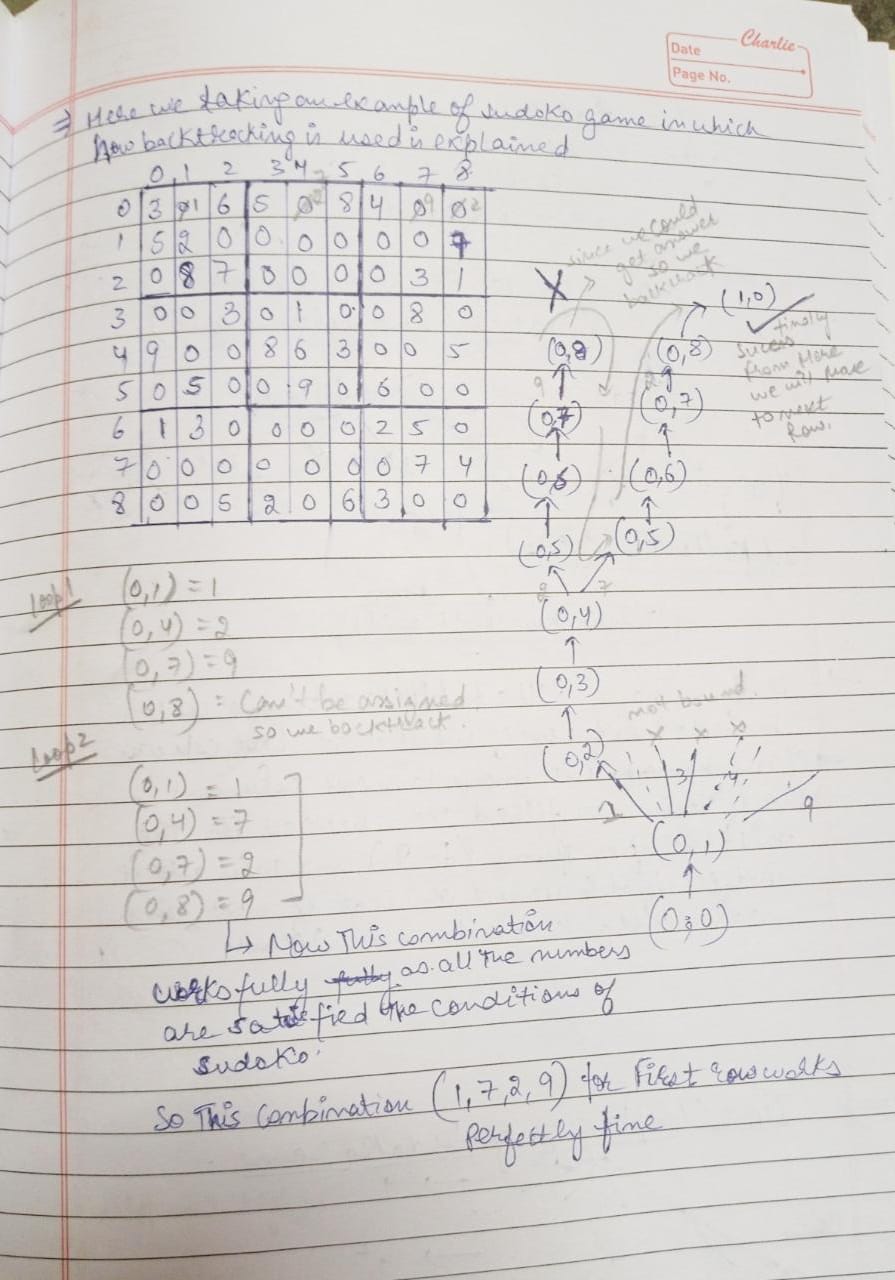
This function will fill the the values if index is not zero .Then will continue for the another numbers by calling itself again and again which are other then 0 .If the after checking all the conditions of the game if the number don’t satisfy then we will again put 0 in that column i.e will erase it.

Bounding function->

This function will check the boundary condition of the game i.e we know that row must be less than N and column must be less than the N-1 the next row is only when it reaches otherwise it will move to other column then only print the solution otherwise invalid values will appear and will show that solutions don’t exists.

3)Invalid function ->

if the number is coming in that present working row then that number is repeating again which is false so will not take it,same as with row. Also we have to check for the sub matrixes i.e in that 3\*3 matrix also the numbers from 0-9 should not be repeated so here we will use the formula which will check in that square sub matrix.



so our Sudoku is completed while taking all the games rule perfectly….