

Today's Content:

→ Sudoku → (Searching in 2D Matrix)

→ Rat in a maze ↳ Search in 2D Matrix

Sudoku: Given 9x9 matrix fill Sudoku

	0	1	2	3	4	5	6	7	8
0	5	3		2	7		4		
1				1	9	5			
2		9	8					6	
3	8		2	4	6		7		3
4	4			8		3			1
5	7		3		2				6
6	6	6		5			2	8	
7				4	1	9			5
8			1		8			7	9

Q) mat[9][9] valid sudoku, fill sudoku

→ In a row data cannot repeat

→ In a column data cannot repeat

→ In all 3x3 boxes data cannot repeat

→ We can only fill 1-9

cells: Start point of box →

x	y	$x - x \% 3$	$y - y \% 3$
1	4	→	0 3
4	8	→	3 6
7	1	→	6 0
8	5	→	6 3

	0	1	2	3	4	5	6	7	8
0	5	3	4	2	7	6	1	9	8
1	6	2		1	9	5	X	X	X
2		9	8					6	
3	8		2		6				3
4	4			8		3			1
5	7		3		2				6
6		6					2	8	
7				4	1	9			5
8			1		8			7	9

Diagram illustrating the sequence of states and transitions for the Huffman tree construction:

```

graph LR
    S1["(0, 8)  
8"] --> S2["(1, 0)  
7"]
    S2 --> S3["(1, 1)  
6"]
    S3 --> S4["(1, 2)  
6"]
    S4 --> S5["(1, 3)  
pre"]
    S5 --> S6["(1, 4)  
pre"]
    S6 --> S7["(1, 5)  
pre"]
    S7 --> S8["(1, 6)  
3"]
    S8 --> S9["(1, 7)  
4"]
    S9 --> S10["(1, 8)  
3"]
    S9 --> S11["(1, 7)  
4"]
    S11 --> S12["(1, 8)  
3"]
    S2 --> S13["(1, 1)  
6"]
    S13 --> S14["(1, 2)  
6"]
  
```

The diagram shows the progression of states (represented as pairs of numbers) and transitions (represented by arrows). The states are labeled with their respective values (e.g., 8, 7, 6, 3, 4, 6, 3, 4, 3). The transitions are labeled with values (e.g., 2, 2, 2, 3, 4, 3, 4, 3). The states are arranged in a sequence, with some states branching into multiple paths. The final state is (1, 8) with a value of 3.

A handwritten solution for a 9x9 grid puzzle. The grid contains numbers from 0 to 8. Some cells are highlighted in green. To the right of the grid, there is a note "→ Not possible" with a double underline. Below the grid, there is a circled number 8.

0	5	3	6	7	2	8	1	4
1	7	4	11	12	13	14	15	12
2	18	17	20	8	21	22	23	24
3	27	28	29	20	31	32	33	34
4	4		8		3			1
5	7		3		2			6
6		6				2	8	
7				4	1	9		5
8		1			8		7	20

$$9 \times 9 \times 9 - 9 \cdot 9 = 9$$

→ $32 \rightarrow \begin{cases} \text{row} : 32/9 \rightarrow 3 \\ \text{col} : 32 \% 9 \rightarrow 5 \end{cases}$

16 \rightarrow row: $16/9 \rightarrow 1$
col: $16/9 \rightarrow 7$

$$\begin{array}{l} 27 \rightarrow \text{row: } 27/9 \rightarrow 3 \\ \text{col: } 27\%9 = 0 \end{array}$$

2D Raden

↙ ↘


r c

$$\underline{x \rightarrow x/q \quad x \% q}$$

Back Tracking:

→ curr inden ps ID

parameters: $\text{mat}[\text{ }][\text{ }], \gamma,$

function calls: At mat[i][j] 

Vold

-How to know which cell data is filled, not filled

If cell is empty data is 0

void printSudoku (int mat[10][10], int n) { TC \rightarrow SC \rightarrow $O(81)$

if (n == 81) {

iterate & print mat[]

return

}

Base condition
Taco

int r = n/9, c = n%9

// we need to fill mat[r][c]

if (mat[r][c] != 0) { // got next cell

printSudoku (mat, n+1)

}

else {

for (int i = 1; i <= 9; i++) {

if (valid(mat, r, c, i)) {

mat[r][c] = i \rightarrow make change

printSudoku (mat, n+1) \rightarrow index

mat[r][c] = 0 \rightarrow revert changes

}

}

}

}

whether we can place
data 'i' in cell mat[r][c]?

```

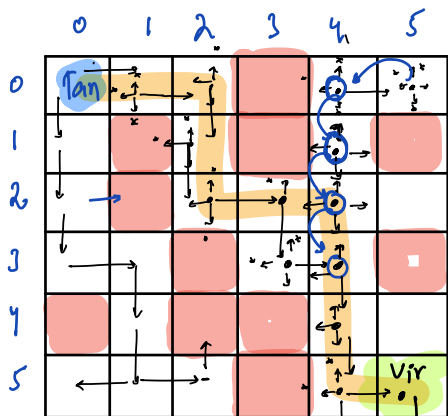
bool valid (int mat[7][7], int r, int c, int d) {
    for (int j = 0; j < 7; j++) {
        if (mat[r][j] == d) { return false; } // To check rth row
        if (mat[j][c] == d) { return false; } // To check cth col
    }
    int x = r - r%3, y = c - c%3;
    for (int i = x; i < x+3; i++) {
        for (int j = y; j < y+3; j++) {
            if (mat[i][j] == d) { return false; }
        }
    }
    return true;
}

```

10 → 15

10 → 28:

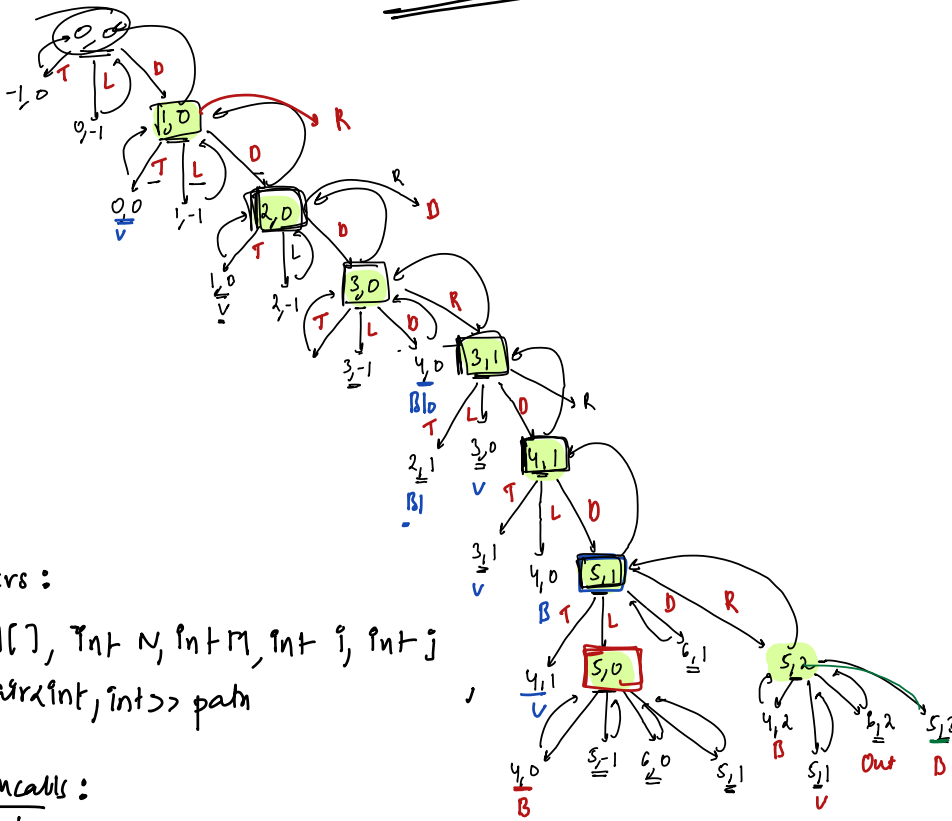
Rat in Maze :



1) cell $\leftarrow \begin{matrix} \uparrow \\ \leftarrow \text{cell} \rightarrow \\ \downarrow \end{matrix}$

2) Every cell we can use once

always at RP



Parameters :

mat[i][j], int N, int M, int i, int j
list<pair<int, int>> path

function calls :

↳ 4 calls :

$(i-1, j)$
 \uparrow
 $(i, j-1) \leftarrow (i, j) \rightarrow (i, j+1)$
 \downarrow
 $(i+1, j)$

$\text{mat}[i][j] = 2 \rightarrow \text{Blocked}$

$\text{mat}[i][j] = 1 \rightarrow \text{visited}$

$\text{mat}[i][j] = 0 \rightarrow \text{unvisited}$

Print the path :

```
void TanvsVir (int mat[N][M], int N, int M, int i, int j, list<pair<int, int>> path) {
```

```
    if ( i < 0 || i >= N || j < 0 || j >= M ) { return }
```

```
    if ( mat[i][j] == 1 || mat[i][j] == 2 ) { return }
```

```
    if ( i == N-1 && j == M-1 ) {
```

```
        // print path
```

```
        return
```

```
    }
```

```
    if ( mat[i][j] == 0 ) {
```

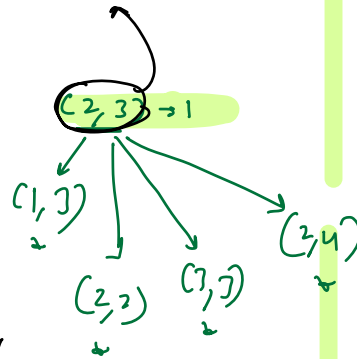
```
        mat[i][j] = 1;
```

```
        list.add( make_pair(i, j) )
```

```
        {
            TanvsVir(mat, N, M, i-1, j, path)
            TanvsVir(mat, N, M, i+1, j, path)
            TanvsVir(mat, N, M, i, j-1, path)
            TanvsVir(mat, N, M, i, j+1, path)
        }
```

```
        list.remove(1)
    }
```

Indirectly acting
on a check part



→ Recursion / Backtracking

↓
Spend 1hr/2hr → revise notes

→ Dynamic Programming : 7/8 Sessions

24th July → Dp1 7th Aug → Dp3 → 3 more problem solving m

31st July → Dp2 14th Aug → Gp1 → all previous topics
15th → 17th → 19th