

Lecture10

Sunday, 2 February 2020 2:25 PM

Q = printkeypad paths

1 → "abc" ✓

2 → def

3 → ghi

4 → jkl

5 → mno

6 → pqr

7 → stu

8 → vwx

9 → yz.

"12"

$$\begin{Bmatrix} ad & ae & af \\ bd & be & bf \\ cd & ce & cf \end{Bmatrix}$$
"123"

$$\{ adg \quad adh \quad adi \quad aeg \quad aeh \\ \quad \quad \quad aci \}$$

Q ⇒

	0	1	2	3
0	○	○	9	○
1	○	○	9	○
2	9	○	○	○
3	○	9	○	○

4x4

(0,0) → (3,3).

Ans 1 →

1	1	9	0
0	1	9	0
0	1	1	1
0	9	0	1

Ans 2 →

1	1	9	0
0	1	9	0

0 1 1 0
0 9 1 1

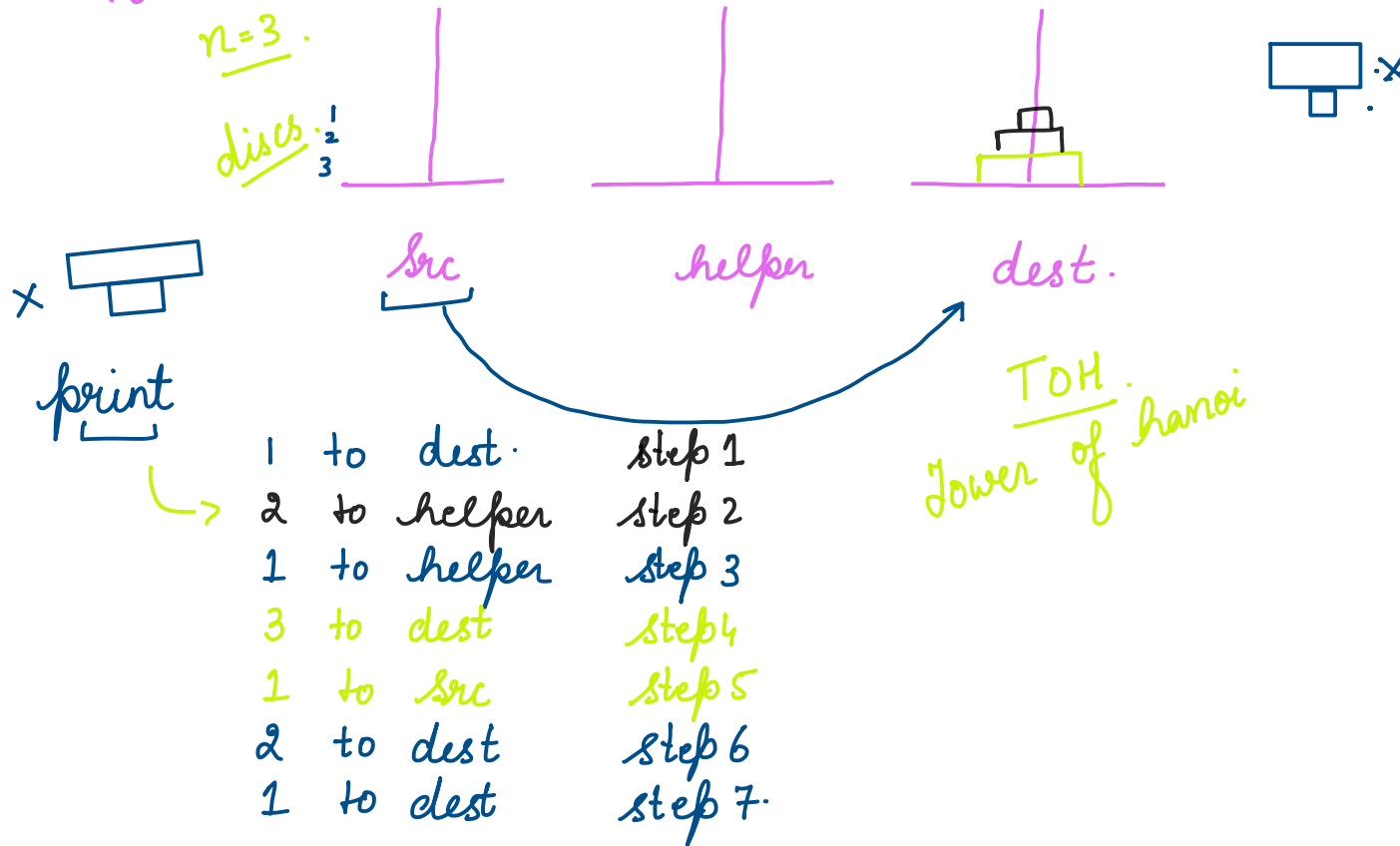
Ans 3 → ✓

Ans 4 → ✓

Q3 →

n=3.

discs. $\frac{1}{2}$
3



public static void printkeypad("12" " " String str, String res) {

```

if (str.length() == 0) {
    syso(res);
    return;
}

```

```

char ch = str.charAt(0); // '1'
String res = str.substring(1); // "2";

```

```

String code = getCode(ch); // "abc"

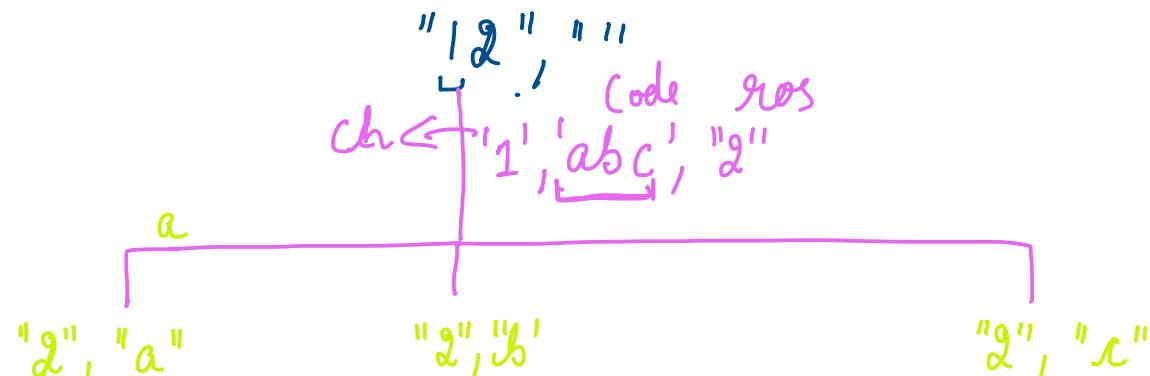
```

```

for (int i = 0; i < code.length(); i++) {
    printkeypad(res, res + code.charAt(i));
}

```

1





```

public static void ppWH(int[][] arr
    if (cr == er && cc == ec) {
        display(arr);
        return;
    }
    if (cr > er || cc > ec) {
        return;
    }
    if (arr[cr][cc] == 9) {
        return;
    }
    arr[cr][cc] = 1;
    ppWH(arr, cr+1, cc, er, ec);
    ppWH(arr, cr, cc+1, er, ec);
}

```

```

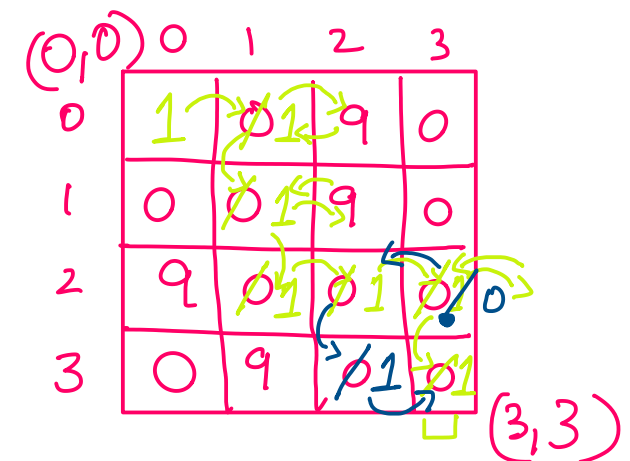
if (arr[cr][cc] == 9) {
    return;
}

```

```

arr[cr][cc] = 1;



```



→ ~~pp~~ WH (arr, er, ec, cr, cc+1),
 ✓ ~~pp~~ WH (arr, er, ec, cr+1, cc);
 arr[cr][cc] = 0;

decision ✓.

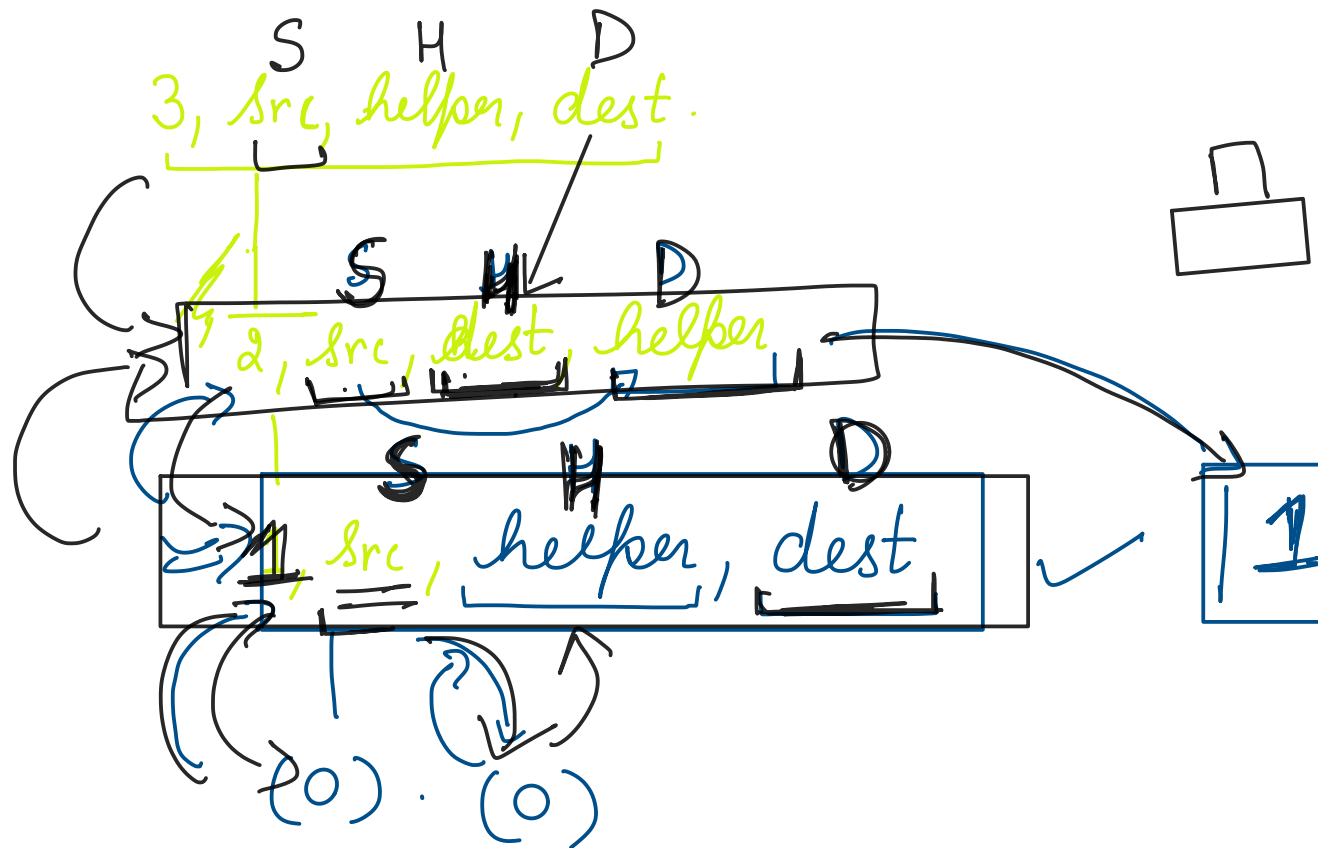

}

public static void TOH (int n, String src — "src"
 String helper, — "hel"
 String dest) { — "d"
 if (n == 0) {
 return;
 }
 ✓ TOH (n-1, src, dest, helper) 
 ⇒ Syso("Move nth disc from  src helper dest.
 ———— disc 1 to dest ————

$$\rightarrow \text{TOH}(n-1, \text{helper}, \text{src}, \text{dest});$$

}

src



Move 1st disc from src to dest

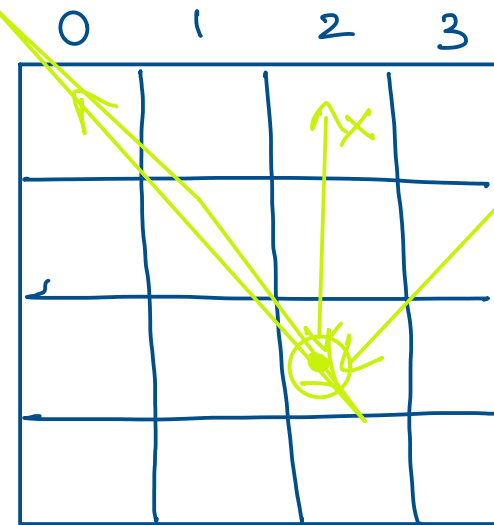
Move 2 disc from src to help
 Move 1th — from dest to help

Q \Rightarrow N Queens \rightarrow

0 \rightarrow 3

0
1

2
3



N x N

N

4

4

4

s.t

ar

Main.

int n = 4;

boolean[][] board = new boolean

Count N Queens (board, row) & ;

```

public static int CountNQueens(int boardLength) {
    if (row == board.length)
        return 1;

    Count = 0;
    for (int col = 0; col < board.length; ) {
        if (isItSafe(board, row, col)) {
            board[row][col] = true;
            Count = Count + CountNQueens(board, row + 1);
            board[row][col] = false;
        }
    }
    return Count;
}

```


public static boolean isItS

Q = Sudoku Solver.

if choose a number
check
Can be place
- d or not

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

9x

0	0	0	0	8	0	0	7	9
---	---	---	---	---	---	---	---	---

```
public static void main(String[] args)
```

```
    int n = 9;
    boolean[][] fixedCells = new
    setFixedCell(board, fixedCells
    boolean solveSudoku = solve(
    if (solveSudoku) {
        display(arr board);
    } else {
    }
}
```

```
public void setFixedCells (board
```

+	+				
+	8	+			

9x9

≡
≡
≡
≡

wherever you
other than zero
index true is

5

```
public void display (int[][] a
```

```
    }
    }
    }
```

```
}
```

```
public boolean solve (boolean
    if (row == n) {
        return true;
    }
    if (col == n) {
        return solve (board, fixedCells
    }
    if (fixedCells (row) (col)) {
        return solve (board,
    }
}
```

```
for (int setnum = 1; setnum
    boolean check =
    if (check) {
        board [row] [
    }
```

```

    boolean solve
    if (solve
        return
    ) else {
        board
    }
}

```

3

```

public static void main (String[] args) {
    int n=9; n;
    int[][] arr = {{ { } }}; → board, arr
    boolean[][] fixedCells = new boolean[9][9];
    setFixedCells(arr, fixedCells, n);
    boolean solve = SudukoSolver(arr, fixedCells,
    if (solve) {
        display(arr);
    } else {
        syso("can't be solved");
    }
}

```

```
public static void setFixedCells (int[][] arr,
                                   boolean[][] fixedCells;
```

3

```
public static boolean SudukoSolver(int[][] arr,
                                     boolean[][] fixed,
                                     int row,
                                     int col,
                                     int n){
```

}

```
public boolean CanPlace(Board, int row, int col,
                        int n, Set num) {
```

3

—
—
—

—