⇒ 2 months

↳ 1-2 hr. → 2-3 hr | 3

⇒ Tomorrow

↳ 1-2 hr → Coding.

↳ Exam

↳ SDE →

↳ Data Engineer

↳ Support Engineer

↳

Knights Tour (Day 28)

Problem | Submissions | Leaderboard | Discussions

1. You are given a number n, the size of a chess board.

2. You are given a row and a column, as a starting point for a knight piece.

3. You are required to generate the all moves of a knight starting in (row, col) such that knight visits all cells of the board exactly once.

4. Calculate and print all configurations of the chess board representing the route of knight through the chess board. Use sample input and output to get more idea.

Note -> When moving from (r, c) to the possible 8 options give first precedence to (r - 2, c + 1) and move in clockwise manner to explore other options. Note -> The online judge can't force you to write the function recursively but that is what the spirit of question is. Write recursive and not iterative logic. The purpose of the question is to aid learning recursion and not test you.

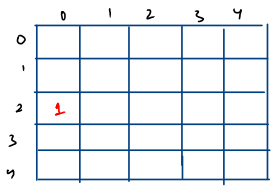
Sample Input 0

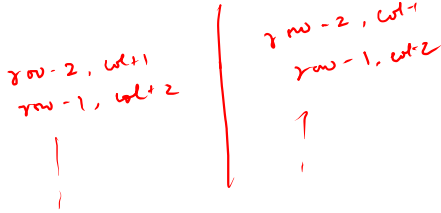
```
5
2
0
```

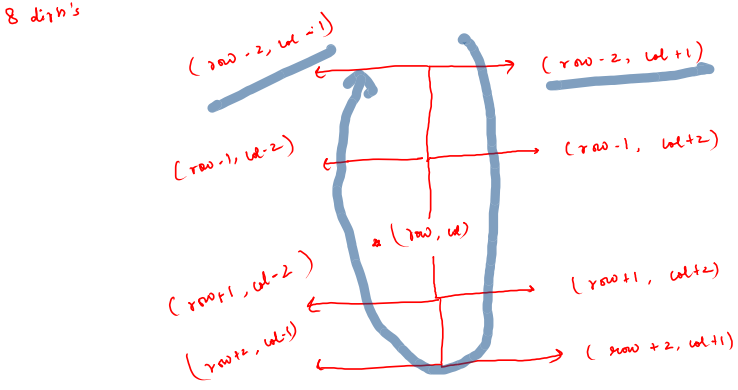
n → 5; 2 → row; 0 → col

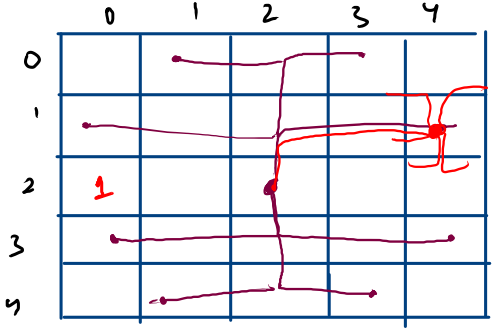
Sample Output 0

```
25 2 13 8 23
12 7 24 3 14
1 18 15 22 9
6 11 20 17 4
19 16 5 10 21
```









row-2, col+1
row-1, col+2
row+1, col+2 ✗
row+2, col+1 ✗
row+2, col-1 ✗
row+1, col-2 ✗
row-1, col-2
row-2, col-1

Negative

⇒ if(r<0, r>=n || c<=0 || c>=n || Dry Run + Code
chess[r][c]>0)
↳ return

5 min + 10 mins

Flood fill

11:08 pm

25

5×5 = 25

5×5=25

⇒ positive base case

(K == n×n)

Print configuration

	0	1	2	3	4
0	(0)	2 (3)	13 (5)	18 (1)	7 (6)
1	12 (2)	19 (4)	8 (3)	3 (4)	14 (5)
2	1 (1)	1	17 (1)	6 (1)	9 (5)
3	0	11 (8)	20 (6)	15 (6)	4 (4)
4	21	16 (1)	5 (1)	10 (7)	0

```java
public class Solution {
    public static void display(int chessboard[][]){
        for(int i=0;i<chessboard.length;i++){
            for(int j=0;j<chessboard.length;j++){
                System.out.print(chessboard[i][j] + " ");
            }
            System.out.println();
        }

        System.out.println();
    }
public static void knightTours(int chessboard[][],int r, int c, int steps){
    if(r<0 || r>=chessboard.length || c<0 || c>=chessboard.length || chessboard[r][c] > 0){
        return;
    }

    if(steps == chessboard.length * chessboard.length){
        chessboard[r][c] = steps;
        display(chessboard);
        chessboard[r][c] = 0;
        return;
    }

    chessboard[r][c] = steps;
    knightTours(chessboard, r-2,c+1,steps+1);
    knightTours(chessboard, r-1,c+2,steps+1);
    knightTours(chessboard, r+1,c+2,steps+1);
    knightTours(chessboard, r+2,c+1,steps+1);
    knightTours(chessboard, r+2,c-1,steps+1);
    knightTours(chessboard, r+1,c-2,steps+1);
    knightTours(chessboard, r-1,c-2,steps+1);
    knightTours(chessboard, r-2,c-1,steps+1);
    chessboard[r][c] = 0;
}
```

V.v.v. Imp.

```java
public static void main(String[] args) {
    /* Enter your code here. Read input from STDI
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int r = scn.nextInt();
    int c = scn.nextInt();

    int chessboard [][] = new int[n][n];

    knightTours(chessboard, r,c,1);
}
```

10 mins

(A)

```
public static int powerLinear(int x,int n){
    if(n == 0) return 1;
    int subAns = powerLinear(x,n-1);
    int ans = subAns * x;
    return ans;
}
```

$T(n) = T(n-1) + K$

$\Rightarrow$ T.C. $\Rightarrow O(n)$

$T(n) = T(n-1) + K$

$T(n-1) = T(n-2) + K$

$T(n-2) = T(n-3) + K$

$T(1) = T(0) + K$

$T(n) = K + K + K + \dots + K$

$T(n) = nK$

(B)

$T(n/2)$ $T(n/2)$

```
public static int powerLinear(int x,int n){
    if(n == 0) return 1;
    int ans = powerLinear(x,n/2) * powerLinear(x,n/2);
    if(n%2 != 0){
        ans *=x;
    }
    return ans;
}
```

$T(n) = T(n/2) + T(n/2) + K$

$T(n) = 2T(n/2) + K$ $\Rightarrow \log n$

$O(n)$

(C)

$x^n \rightarrow T(n)$

```
public static int powerLog(int x,int n){
    if(n == 0) return 1;

    int subAns = powerLog(x,n/2);
    int ans = subAns*subAns;
    if(n%2 != 0) ans*=x;

    return ans;
}
```

by map

$T(n) = T(n/2) + K$

$\Rightarrow$ T.C. $= O(\log n)$

1st $\rightarrow$ $T(n) = T(n/2) + K$

$T(n/2) = T(n/4) + K$

$T(n/4) = T(n/8) + K$

$\vdots$

$T(1) = T(0) + K$

$T(n) = xK$

$T(n) = K \log_2 n$

$T(n) = T\left(\frac{n}{2}\right)$ ✓

$T(n/2) = T\left(\frac{n}{2^2}\right)$ ✓

$T(n/4) = T\left(\frac{n}{2^3}\right)$

$T(1) = T\left(\frac{n}{2^{x-1}}\right)$

$\frac{n}{2^{x-1}} = 1$ $\quad 2^{x-1} \approx 2^x$

$n = 2^{x-1}$

$n \approx 2^x$

$\log_2 n = \log_2 2^x$

$x = \log_2 n$

$T(n) = 2T(n/2) + K$

$T(n/2) = 2T(n/4) + K$] $\times 2$

$T(n/4) = 2T(n/8) + K$] $\times 4$

$T(n/8) = 2T(n/16) + K$] $\times 8$

$\vdots$

$T(1) = T(0) + K$

x terms

$\Rightarrow 2^1 T(n/2) = 2T(n/2) + K$

$\Rightarrow 2^1 T(n/2) = 2^2 T(n/4) + 2K$

$\Rightarrow 2^2 T(n/4) = 2^3 T(n/8) + 4K$

$\Rightarrow 2^3 T(n/8) = 2^4 T(n/16) + 8K$

$\vdots$

$\Rightarrow 2^{x-1} T(1) = 2^x T(0) + 2^{x-1} \cdot K$

$T(n) = K + 2K + 4K + 8K + \cdots + 2^{x-1}$ (x terms)

$T(n) = K(1 + 2 + 4 + 8 + \cdots + 2^{x-1})$

G.P.

$\Rightarrow K\left[\dfrac{1 \cdot (2^x - 1)}{2 - 1}\right]$

$\Rightarrow 2^x \cdot K$ — (1)

G.P. $\dfrac{a(r^n - 1)}{r - 1}$ $\quad \dfrac{y}{2} = 2$

$T(1) = 2^{x-1} T\left(\dfrac{n}{2^{x-1}}\right)$

$\dfrac{n}{2^{x-1}} = 1$

$n = 2^{x-1}$

$n \approx 2^x$ — (2)

$\Rightarrow$ From eq^n (1) & (2),

$T(n) = n \cdot K$

$T(n) = O(n)$

$T(n) = n$

4 - 5 mins

```java
ArrayList<String> subAns1 = getStairPaths(n-1); // 1 step
ArrayList<String> subAns2 = getStairPaths(n-2); // 2 steps
ArrayList<String> subAns3 = getStairPaths(n-3); // 3 steps

ArrayList<String> ans = new ArrayList<>();
```

Approx :- (1) $T(n-1) \geq T(n-2)$

(2) $T(n-1) \geq T(n-3)$

$T(n) = T(n-1) + T(n-2) + T(n-3) + K$

$\Rightarrow T(n) \approx T(n-1) + T(n-1) + T(n-1) + K$

$\Rightarrow T(n) = 3T(n-1) + K$

$T(n-1) = 3T(n-2) + K \quad] \times 3$

$T(n-2) = 3T(n-3) + K \quad] \times 9$

$T(n-3) = 3T(n-4) + K \quad] \times 27$

$T(1) = 3T(0) + K$

$\Rightarrow$ 1st $\rightarrow 3^0\, T(n) = 3^1\, T(n-1) + K$

2nd $\rightarrow 3^1\, T(n-1) = 3^2\, T(n-2) + 3K$

3rd $\rightarrow 3^2\, T(n-2) = 3^3\, T(n-3) + 3^2 K$

$3^3\, T(n-3) = 3^4\, T(n-4) + 3^3 K$

$\vdots$

$3^{n-1}\, T(1) = 3^n\, T(0) + 3^n K$

$\Rightarrow T(n) = K + 3K + 3^2K + 3^3K + \cdots 3^nK \Rightarrow$ n terms.

G.P. $\Rightarrow T(n) = K(1 + 3 + 3^2 + 3^3 + \cdots 3^n)$

$T(n) = K\left[\frac{3^n - 1}{3-1}\right]$

$= K\left[\frac{3^n - 1}{2}\right]$

$\Rightarrow$ 4-5 mins

$T(n) = T(n-1) + T(n-2) + T(n-3) + K(n)$

T.C. $\Rightarrow O(4^n)$

T.C. $\Rightarrow O(3^n)$ ←

$T(1) = T(3$

2 months

4 classes

- ⇒ Tuesday
- ⇒ Thursday
- ⇒ Saturday

→ 9 ; 11 pm

- ⇒ Sunday → (10 - 1) morning

n → rows, m → cols

```
getMazePaths(sr, sc+1, dr, dc);
getMazePaths(sr+1, sc, dr, dc);
```

n = 5
m = 5

h
v

n + m = a

2 digit 3-4 mins

$T(n+m) = T(n+m-1) + T(n+m-1) + k$

$T(n+m) = 2T(n+m-1) + k$

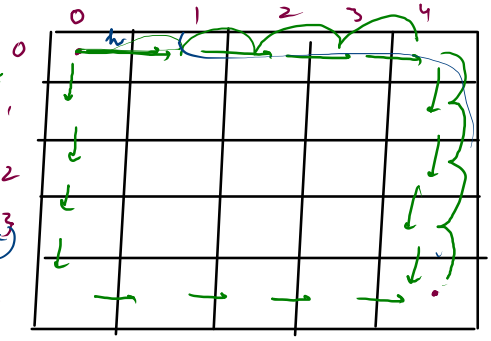
$T(a) = 2T(a-1) + k$ $\quad T(n) = 2T(n-1) + k$

$T(a-1) = 2T(a-2) + k$

$T(1) = 2T(0) + k$

$T(n) = k \cdot 2^a$

$T(n) = k \cdot 2^{n+m}$



(no. of calls)^n

h h h h v v v v

4 (n-1) 4 (m-1)

T.C. ⇒ $O(2^{n+m})$