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/

Go through above links to get a very fine idea of recursion .

Let us discuss a nice example . There is a 2D array filled with some values and we are at (1,1) cell (assume 1-based indexing) . Let us define some directions , w.r.t (i,j) cell

(i+1,j) be direction 1

(i+1,j+1) be direction 2

Now from (I,j) cell we can go in either of these two directions and as we go down we take the sum of values we visit . Now we want to maximize this sum . Suppose the array is :

	1	2	3	4
1	5			
2	10	15		
3	30	20	15	
4	100	200	300	50

Maximum sum which we can create is 5 + 15 + 20 + 300 = 340

Now a possible solution is to create all possible sequences and output the maximum one . Now how do we get an idea that this problem can be solved with recursion ?? Now suppose the max. sum which we can create by starting from (2,1) be x and (2,2) be y . Then the our answer is max(5+x,5+y) . Now our problem is to know the values of x and y . And similarly to get value x we need max. sum sequence starting from (3,1) and (3,2) and for y we need starting ones from (3,2) and (3,3) . So this problem can be broken down into similar subproblems with similar property . So this can be solved with recursion

Here is the pseudocode for that:

I,j represent the current cell and n,m represent the size of matrix.

```
Int max_sum( int I , int j ){
        If(i>n || j>m)return 0;
        Return max( a[i][j] + max_sum(i+1,j) , a[i][j] + max_sum(i+1,j+1) );
}
```

This solution has exponential complexity but this can be solved in a much better way in O(N^2) with dynamic programming approach but let’s leave that for some other tutorial .