Recursion will use more memory, while this problem can be solved by iteration.

The pattern was that:

say n = 4, you have {1, 2, 3, 4}

If you were to list out all the permutations, you have

1 + (permutations of 2, 3, 4)  
  
2 + (permutations of 1, 3, 4)  
  
3 + (permutations of 1, 2, 4)  
  
4 + (permutations of 1, 2, 3)

We know how to calculate the number of permutations of n numbers... n! So each of those with permutations of 3 numbers means there are 6 possible permutations. Meaning there would be a total of 24 permutations in this particular one. So if you were to look for the (k = 14) 14th permutation, it would be in the

3 + (permutations of 1, 2, 4) subset.

To programmatically get that, you take k = 13 (subtract 1 because of things always starting at 0) and divide that by the 6 we got from the factorial, which would give you the index of the number you want. In the array {1, 2, 3, 4}, k/(n-1)! = 13/(4-1)! = 13/3! = 13/6 = 2. The array {1, 2, 3, 4} has a value of 3 at index 2. So the first number is a 3.

Then the problem repeats with less numbers.

The permutations of {1, 2, 4} would be:

1 + (permutations of 2, 4)  
  
2 + (permutations of 1, 4)  
  
4 + (permutations of 1, 2)

But our k is no longer the 14th, because in the previous step, we've already eliminated the 12 4-number permutations starting with 1 and 2. So you subtract 12 from k.. which gives you 1. Programmatically that would be...

k = k - (index from previous) \* (n-1)! = k - 2\*(n-1)! = 13 - 2\*(3)! = 1

In this second step, permutations of 2 numbers has only 2 possibilities, meaning each of the three permutations listed above a has two possibilities, giving a total of 6. We're looking for the first one, so that would be in the 1 + (permutations of 2, 4) subset.

Meaning: index to get number from is k / (n - 2)! = 1 / (4-2)! = 1 / 2! = 0.. from {1, 2, 4}, index 0 is 1

so the numbers we have so far is 3, 1... and then repeating without explanations.

{2, 4}  
  
k = k - (index from pervious) \* (n-2)! = k - 0 \* (n - 2)! = 1 - 0 = 1;  
  
third number's index = k / (n - 3)! = 1 / (4-3)! = 1/ 1! = 1... from {2, 4}, index 1 has 4  
  
Third number is 4

{2}  
  
k = k - (index from pervious) \* (n - 3)! = k - 1 \* (4 - 3)! = 1 - 1 = 0;  
  
third number's index = k / (n - 4)! = 0 / (4-4)! = 0/ 1 = 0... from {2}, index 0 has 2  
  
Fourth number is 2

Giving us 3142. If you manually list out the permutations using DFS method, it would be 3142. Done! It really was all about pattern finding.