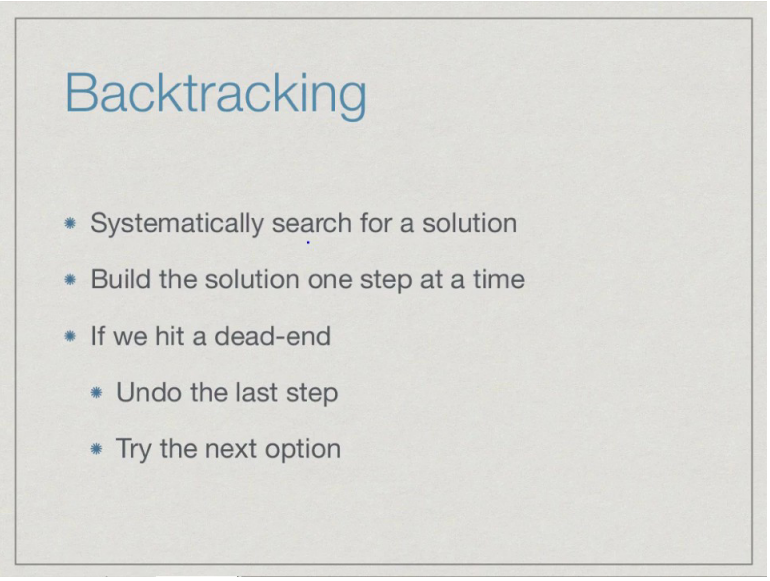
Programming, Data Structures and Algorithms in Python
Prof. Madhavan Mukund
Department of Computer Science and Engineering
Chennai Mathematical Institute,Madras

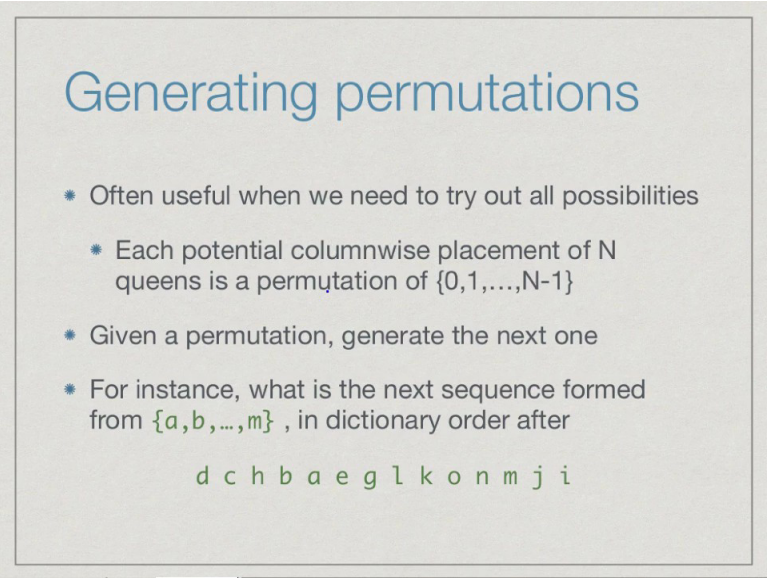
Week - 06
Lecture - 03
Generating Permutations

(Refer Slide Time: 00:03)



We will be looking at Backtracking. In backtracking we systematically search for a solution one step at a time and when we hit a dead end we undo the last step and try the next option.

(Refer Slide Time: 00:14)

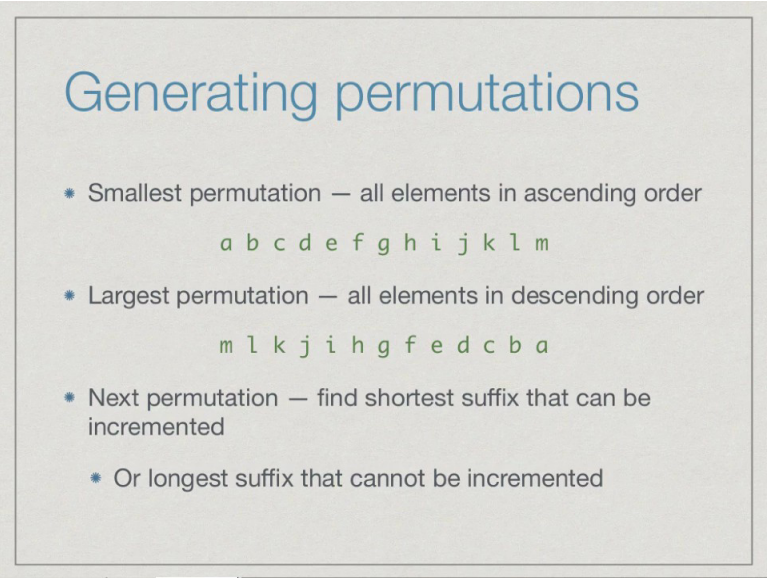


Now, in the process of doing the backtracking we need to generate all the possibilities. For instance, remember when we try to printout all the queens we ran through every possible position for every queen on every row, and if it was free then we tried it out and if the solution extended to a final case then we print it out. Now if we look at the solutions that we get for the 8 queens, each queen on row is in a different column from every other queen. The column numbers if we read then row by row, the column numbers form a permutation of 0 to N minus 1. So, each number 0 to N minus 1 occurs exactly once as a column number for the n queens.

So, one way of solving a problem like 8 queens or similar problems is actually it generate all permutations and keep trying them one at a time. This give rise to the following question; if we have a permutation of 0 to N minus 1 how do we generate the next permutation. This is like thinking of it as a next number, but this could be in an arbitrary number of symbols.

Suppose, we have the letters a to m. So, these are the first thirteen letters of the alphabet and we treat the dictionary order of words as the ordering of numbers, we think of them as digits if you want to think of it is base thirteen. Here for instance, is a number in a base thirty or now alternatively a rearrangement of a to m in some order. Now what we want to is, what is the next rearrangement after this you immediately next one in dictionary order.

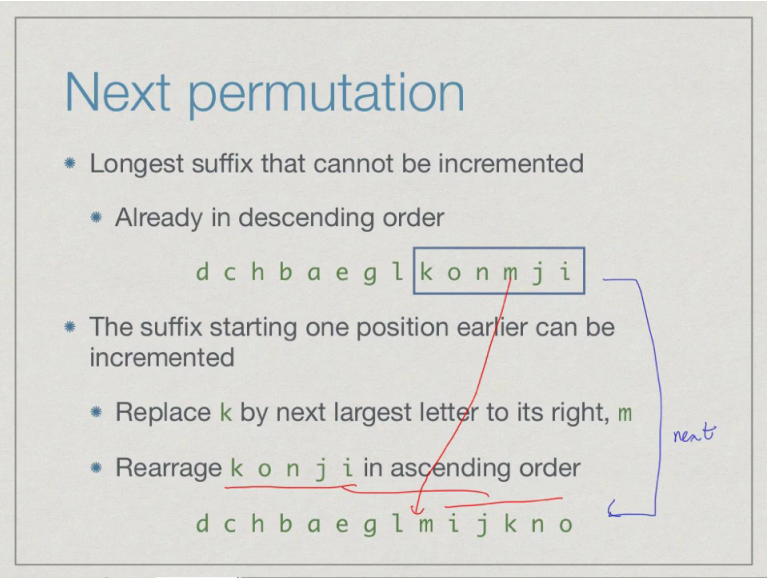
(Refer Slide Time: 01:54)



In order to solve this problem the first observation we can make is that, if we have a sequence of such letters or digits the smallest permutation is the order in which the elements are arranged in ascending order. So we start with a which is smallest one then b and c and so on and there is no smaller permutation than this one. Similarly, the largest permutation is one in which all the elements are in descending order, so we start with the largest element m and we work backwards down to a.

If we want to find the next permutation we need to find as short suffix as possible that can be incremented, it is probably easiest to do it in terms of a numbers but let us do it with letters. The shortest suffix that can be incremented consists of something followed by the longest suffix cannot be incremented. So this will become a little clear when we work through an example.

(Refer Slide Time: 02:56)



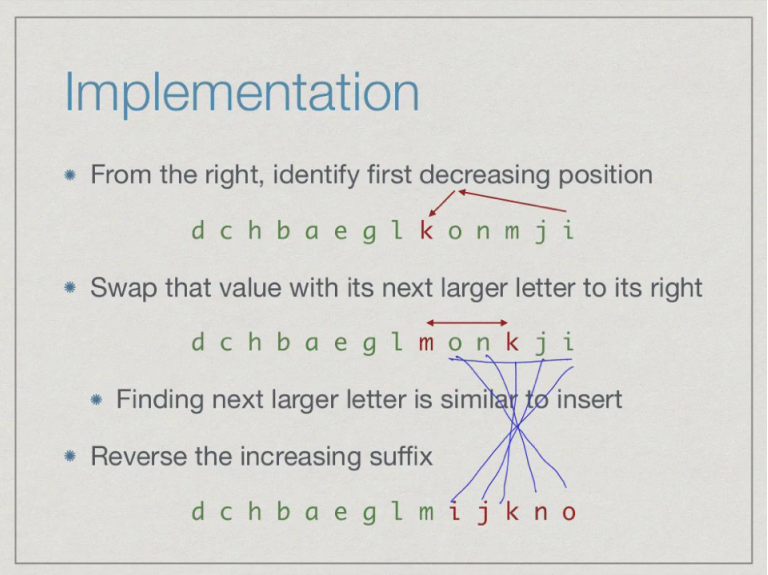
We want to find the longest suffix that cannot be incremented. So, a suffix that cannot be incremented is one which is as large as it could possibly be which means that it is already in descending order. If you look at example that we had before for which we want to define the next permutation, we find this suffix o n m j i these five letters are in descending orders so I cannot make any larger permutation using this.

So, if I fix the letter from d to k then this the largest permutation I can generate with d to k fixed. If I want to change it and need to increment something and I mean to increment it, I cannot increment it within this red box so I must extend this to find the shortest suffix namely; suffix started with k where something can be incremented. Now how do we increment this? Well, what we need to do is that now is like say that we have with k we cannot do any better so we have to replace k by something bigger and the something bigger has to be the smallest thing that we can replace it by, so we will replace k at the next largest letter to its right namely m.

Among these letters m n and o are bigger than k if I replace it by j or i, I will get a smaller permutation which I do not want, so I may replace it m n or o, but among these this m is the smallest I must now start a sequence where the suffix of length six begins with the letter m. And among suffix that begins with letter m I need the smallest one, that mean I rearrange the remaining letters k o n j i in ascending order to give me the smallest permutation to begin with m and has the letters k o n j i after it.

This gives me this permutation. So, I have now moved this m here and I have now taken these letters and rearrange them in an ascending order to get i j k n o. Therefore, this means that for this permutation the next permutation is this one.

(Refer Slide Time: 04:55)



So, algorithmically we can do it as follows, what we need to do is first identify the suffix that can be incremented. We begin by looking for suffix that cannot be incremented namely we go backwards so long as it is in descending order. So we keep looking for values as they increase. So, i is smaller than j, j is smaller than m, m is smaller than n, n is smaller than o, but o is bigger than k so that means than up to here we have a suffix that cannot be incremented and this is the first position where we can make an increment.

Having done this we now need to replace k by the letter to its right which is next bigger. Now this is a bit like insert we go one by one, we say that k smaller than n so we continue, and we say than k is bigger than j so we stop here. So this tells us that the letter m is the one we want. We can identify this in one scan, because this remember it is in descending order, it is in sorted order so we can go through and find the first position where we crossed on something bigger than k to something smaller than k and that is the position of the letter that we need to change. So, it is exactly like inserting something into a sorted list.

Now having done this, we have exchange this m and k now we need to put this in ascending order, but remember it was in descending order and what we did to the

descending order we replace m by k but what are the property of k? k was smaller than m but bigger than j so, o n k j i remains in descending order. If we want convert it to ascending order we do not need to sort it we just need to reverse it we just needed backwards, so this is just the reversal of this.

This is a concrete way in all to which find the next permutation, walk backwards from the end and see when the order stops increasing. So, wherever we first decrease this that is the suffix that you want to increment, of course if we go all the way and go back to the first letter and we are not found such a position then we have already reached the last permutation in the overall scheme of things.

Once we find such a position we find which letter to swap it with by doing equivalent of the search that we do for insertion sort. So, we do an insert kind of thing find the position in this case m to swap with k after swapping it we take the suffix after the new letter we put namely m and we reverse it to get the smallest permutation starting with that letter m.