Question: https://leetcode.com/problems/next-permutation/

Constraints: The replacement must be **[in place](http://en.wikipedia.org/wiki/In-place_algorithm" \t "https://leetcode.com/problems/next-permutation/_blank)** and use only constant extra memory.

See, We convert a permutation into greater lexicographical order by converting the last part into largest lexicographical order  
For eg 1 2 4 3 5  
We will not change 1 2 till we convert 4 3 5 into the largest order ie 5 4 3 , then only we can change 1 2

So the first step is to find the string from the back which is already in the largest lexicographical order  
We do this by finding i from the back as all elements after i will be in largest lexicographical order.  
eg 1 2 5 4 3 , we will find i to be 1 (nums[i]=2),

In second step we will find element greater than nums[index] to replace it with.  
In example we now need to change 2, that is why we start from end because we know the string 5 4 3 will be in descending order  
We will find j to be 4(nums[j]=3)

After that we swap nums[i] with nums[j]  
So order becomes 1 3 5 4 2 ,  
Then we reverse the 5 4 2 string so as to convert it into smallest order

So order becomes 1 3 2 4 5

Code:  
class Solution {

public void nextPermutation(int[] nums) {

int i=nums.length - 2;

while(i>=0 && nums[i]>=nums[i+1]) i--;

if(i>=0){

int j=nums.length - 1;

while(j>=i && nums[i]>=nums[j]) j--;

swap(nums, i, j);

}

reverse(nums, i+1, nums.length-1);

}

public void swap(int[] nums, int i, int j){

int temp=nums[i];

nums[i]=nums[j];

nums[j]=temp;

return;

}

public void reverse(int[] A, int i, int j){

while(i<j){

swap(A, i, j);

i++;

j--;

}

return;

}

}

Github Link :<https://lnkd.in/ecwtJeaz>