[**Next Permutation**](https://leetcode.com/problems/next-permutation/)

**public** **class** NextPermutation {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**int**[] nums = {6, 5 , 1 , 4 , 3 , 2 , 0};

*nextPermutation*(nums);

**for**(**int** i : nums) {

System.***out***.print(i + " ");

}

}

**public** **static** **void** nextPermutation(**int**[] nums) {

**if**(nums == **null** || nums.length == 0) {

**return**;

}

**int** i = nums.length - 2;

**while**(i >= 0 && nums[i] >= nums[i + 1]) {

i--; //loop till find a divergence from strictly increasing number

}

**if**(i >= 0) { //if there is any divergent point, swap that no with a number greater than it to its right

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

**int** temp = nums[i];

nums[i] = nums[j];

nums[j] = temp;

}

**public** **static** **void** reverse(**int**[] nums , **int** i , **int** j) {

**if**(nums == **null** || nums.length == 0) {

**return**;

}

**while**(i <= j) {

**int** temp = nums[i];

nums[i] = nums[j];

nums[j] = temp;

i++;

j--;

}

}

}

Time Complexity : O(n) , n is length of given num array

Space Complexity : O(1), constant space