Gopika shree M

22AD034

AI & DS

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

A **permutation** of an array of integers is an arrangement of its members into a sequence or linear order.

* For example, for arr = [1,2,3], the following are all the permutations of arr: [1,2,3], [1,3,2], [2, 1, 3], [2, 3, 1], [3,1,2], [3,2,1].

The **next permutation** of an array of integers is the next lexicographically greater permutation of its integer. More formally, if all the permutations of the array are sorted in one container according to their lexicographical order, then the **next permutation** of that array is the permutation that follows it in the sorted container. If such arrangement is not possible, the array must be rearranged as the lowest possible order (i.e., sorted in ascending order).

* For example, the next permutation of arr = [1,2,3] is [1,3,2].
* Similarly, the next permutation of arr = [2,3,1] is [3,1,2].
* While the next permutation of arr = [3,2,1] is [1,2,3] because [3,2,1] does not have a lexicographical larger rearrangement.

Given an array of integers nums, *find the next permutation of* nums.

The replacement must be [**in place**](http://en.wikipedia.org/wiki/In-place_algorithm) and use only constant extra memory.

**Example 1:**

**Input:** nums = [1,2,3]

**Output:** [1,3,2]

**Example 2:**

**Input:** nums = [3,2,1]

**Output:** [1,2,3]

**Example 3:**

**Input:** nums = [1,1,5]

**Output:** [1,5,1]

**Constraints:**

* 1 <= nums.length <= 100
* 0 <= nums[i] <= 100

import java.util.\*;

public class Main {

public int[] nextPermutation(int[] arr) {

int n = arr.length;

int b = -1;

for (int i = n - 2; i >= 0; i--) {

if (arr[i] < arr[i + 1]) {

b = i;

break;

}

}

if (b == -1) {

int a = 0, l = n - 1;

while (a < l) {

int temp = arr[a];

arr[a] = arr[l];

arr[l] = temp;

a++;

l--;

}

return arr;

}

for (int i = n - 1; i > b; i--) {

if (arr[i] > arr[b]) {

int temp = arr[b];

arr[b] = arr[i];

arr[i] = temp;

break;

}

}

int k = b + 1;

int m = n - 1;

while (k < m) {

int temp = arr[k];

arr[k] = arr[m];

arr[m] = temp;

k++;

m--;

}

return arr;

}

public static void main(String args[]) {

Main obj = new Main();

int[] res = obj.nextPermutation(new int[]{2,1,3});

System.***out***.println(Arrays.*toString*(res));

}

}

[**2) Spiral Matrix**](https://leetcode.com/problems/spiral-matrix/)

Given an m x n matrix, return *all elements of the* matrix *in spiral order*.

**Example 1:**



**Input:** matrix = [[1,2,3],[4,5,6],[7,8,9]]

**Output:** [1,2,3,6,9,8,7,4,5]

**Example 2:**



**Input:** matrix = [[1,2,3,4],[5,6,7,8],[9,10,11,12]]

**Output:** [1,2,3,4,8,12,11,10,9,5,6,7]

**Constraints:**

* m == matrix.length
* n == matrix[i].length
* 1 <= m, n <= 10
* -100 <= matrix[i][j] <= 100

import java.util.\*;

public class Main {

public int[] spiralMatrix(int[][] arr) {

int l = 0, r = arr[0].length;

int t = 0, b = arr.length;

int k = 0;

int[] ans = new int[(r)\*(b)];

while (l < r && t < b) {

for (int i = l; i < r; i++) {

ans[k] = arr[t][i];

k+= 1;

}

t++;

for (int i = t; i < b; i++) {

ans[k] = arr[i][r-1];

k += 1;

}

r--;

if (l >= r && t >= b) {

break;

}

for (int i = r-1; i >= l; i--) {

ans[k] = arr[b-1][i];

k += 1;

}

b--;

for (int i = b-1; i >= t; i--) {

ans[k] = arr[i][l];

k += 1;

}

l++;

}

return ans;

}

public static void main(String args[]) {

Main obj = new Main();

int[] res = obj.spiralMatrix(new int[][]{{1,2,3},{4,5,6},{7,8,9}});

System.***out***.println(Arrays.*toString*(res));

}

}

[**3) Longest Substring Without Repeating Characters**](https://leetcode.com/problems/longest-substring-without-repeating-characters/)

Given a string s, find the length of the **longest** **substring** without repeating characters.

**Example 1:**

**Input:** s = "abcabcbb"

**Output:** 3

**Explanation:** The answer is "abc", with the length of 3.

**Example 2:**

**Input:** s = "bbbbb"

**Output:** 1

**Explanation:** The answer is "b", with the length of 1.

**Example 3:**

**Input:** s = "pwwkew"

**Output:** 3

**Explanation:** The answer is "wke", with the length of 3.

Notice that the answer must be a substring, "pwke" is a subsequence and not a substring.

**Constraints:**

* 0 <= s.length <= 5 \* 104
* s consists of English letters, digits, symbols and spaces.

import java.util.\*;

public class Main {

public int lengthOfLongestSubstring(String s) {

int left = 0, right = 0;

List<Character> q = new ArrayList<>();

int max = 0;

while (right < s.length()) {

char current = s.charAt(right);

if (q.contains(current)) {

while (q.get(0) != current) {

q.remove(0);

}

q.remove(0);

}

q.add(current);

right++;

max = Math.*max*(max, q.size());

}

return max;

}

public static void main(String[] args) {

Main obj = new Main();

System.***out***.println(obj.lengthOfLongestSubstring("abcabcbb"));

System.***out***.println(obj.lengthOfLongestSubstring("bbbbb"));

System.***out***.println(obj.lengthOfLongestSubstring("pwwkew"));

}

}