**1)Next Permutation**

import java.util.Arrays;

public class NextPermutation {

public static void main(String[] args) {

NextPermutation obj=new NextPermutation();

int[] arr={2,4,1,7,5,0};

int[] res=obj.nextpermutation(arr);

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

}

int[] nextpermutation(int[] arr){

int i=arr.length-2;

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

i--;

}

if(i>=0){

int j=arr.length-1;

while(j>=0 && arr[j]<=arr[i]){

j--;

}

swap(arr,i,j);

}

reverse(arr,i+1);

return arr;

}

void swap(int[] arr,int i,int j){

int temp=arr[i];

arr[i]=arr[j];

arr[j]=temp;

}

void reverse(int[] arr,int i){

int end=arr.length-1;

swap(arr,i,end);

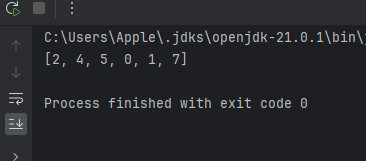
i++;

end--;

}

}

Output:



TimeComplexity:O(n)

SpaceComplexity:O(1)

**2)Spiral Matrix**

Program:

import java.util.Scanner;

public class SpiralMatrix {

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

System.out.println("Enter the number of rows (n) and columns (m):");

int n = sc.nextInt();

int m = sc.nextInt();

int[][] matrix = new int[n][m];

System.out.println("Enter the elements of the matrix:");

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++) {

matrix[i][j] = sc.nextInt();

}

}

SpiralMatrix obj=new SpiralMatrix();

obj.matrix(matrix);

}

public void matrix(int[][] matrix){

int left=0;

int right=matrix[0].length-1;

int top=0;

int bottom=matrix.length-1;

while(top<=bottom && left<=right){

for(int i=left;i<=right;i++){

System.out.print(matrix[top][i]+" ");

}

top++;

for(int i=top;i<=bottom;i++){

System.out.print(matrix[i][right]+" ");

}

right--;

if(top<=bottom){

for(int i=right;i>=left;i--){

System.out.print(matrix[bottom][i]+" ");

}

bottom--;

}

if(left<=right){

for (int i=bottom;i>=top;i--){

System.out.print(matrix[i][left]+" ");

}

left++;

}

}

System.out.println();

}

}

TimeComplexity:O(m\*n)

SpaceComplexity:O(1)

**2)Spiral matrix**

Program:

class Solution {

public List<Integer> spiralOrder(int[][] matrix) {

List<Integer> list=new ArrayList<>();

int left=0;

int right=matrix[0].length-1;

int top=0;

int bottom=matrix.length-1;

while(left<=right && top<=bottom){

for(int i=left;i<=right;i++){

list.add(matrix[top][i]);

}

top++;

for(int i=top;i<=bottom;i++){

list.add(matrix[i][right]);

}

right--;

if(top<=bottom){

for(int i=right;i>=left;i--){

list.add(matrix[bottom][i]);

}

bottom--;

}

if(left<=right){

for(int i=bottom;i>=top;i--){

list.add(matrix[i][left]);

}

left++;

}

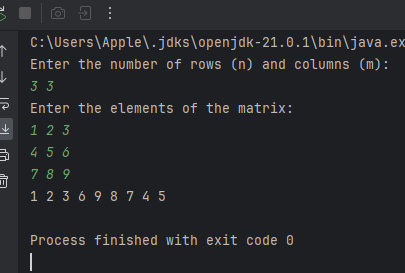
}

return list;

}

}

Output:



TimeComplexity:O(n\*m)

SpaceComplexity:O(n\*m)