**Largest Permutation**

Show Topic Tags   

Given a permutation of first **n** natural numbers as an array and an integer k. Print the lexicographically largest permutation after at most k swaps.

**Input:**  
The first line of input contains an integer T denoting the number of test cases. Each test case contains two integers n and k where n denotes the number of elements in the array a[]. Next line contains space separated n elements in the array a[].  
  
**Output:**  
Print space separated n integers which form the largest permutation after at most k swaps.  
  
**Constraints:**  
1<=T<=100  
1<=n<=1000  
1<=a[i]<=1000  
1<=k<=100 0  
  
**Example:  
Input:**  
2  
5 3  
4 5 2 1 3  
3 1  
2 1 3  
**Output:**  
5 4 3 2 1   
3 1 2

\*\*For More Examples Use Expected Output\*\*

<http://practice.geeksforgeeks.org/problems/largest-permutation/0>

-------ACEPTADO: PERO CON UN MAP PARA GUARDAR LOS INDICES EXCEDE EL TIEMPO LÍMITE-------

#include <iostream>

#include <stdio.h>

#include <map>

using namespace std;

void Swap(int arr[], int p1, int p2)

{

int temp = arr[p1];

arr[p1] = arr[p2];

arr[p2] = temp;

}

int main(){

int t;

scanf("%d", &t);

while(t--) {

int n,k;

scanf("%d %d", &n, &k);

int arr[n];

int indices[n];

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

scanf("%d", &arr[i]);

indices[arr[i]]=i;

}

/\*

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

{

indices[arr[i]] = i;

}\*/

int max = n;

int ubicacion = 0;

int cont = 0;

for (int i = max; i >= 1; i-- )

{

if (cont >= k)

{

break;

}

if (indices[i] != ubicacion)

{

indices[arr[ubicacion]] = indices[i];

Swap(arr, indices[i], ubicacion);

cont++;

}

ubicacion++;

}

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

{

printf("%d ", arr[i]);

}

printf("\n");

}

return 0;

}

----------------JAVA-------------

import java.util.\*;

import java.lang.\*;

import java.io.\*;

class GFG {

static void Swap(int[] arr, int p1, int p2)

{

int temp = arr[p1];

arr[p1] = arr[p2];

arr[p2] = temp;

}

public static void main(String[] args) throws IOException {

// TODO code application logic here

BufferedReader br = new BufferedReader(new InputStreamReader(System.in)); //

int t = Integer.parseInt(br.readLine());

while(t-- > 0) {

String[] nk = br.readLine().trim().split(" ");

int n = Integer.parseInt(nk[0]);

int k = Integer.parseInt(nk[1]);

String[] input = br.readLine().trim().split(" ");

int[] arr = new int[n];

int[] indices = new int[n+1];

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

arr[i] = Integer.parseInt(input[i]);

indices[arr[i]] = i;

}

int max = n;

int ubicacion = 0;

int cont = 0;

for (int i = max; i >= 1; i-- )

{

if (cont >= k)

{

break;

}

if (indices[i] != ubicacion)

{

indices[arr[ubicacion]] =indices[i];

Swap(arr, indices[i], ubicacion);

cont++;

}

ubicacion++;

}

for (int i = 0; i < arr.length; i++)

{

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

}

System.out.println();

}

}

}