**Problem Name:** Pancake sorting

**Topics:** Array, Two-Pointers, Greedy, Sorting

**Companies:** Microsoft, Google

**Level:** Medium

**Language:** C++

**Problem Statement:** Given an array of integers arr, sort the array by performing a series of pancake flips.

In one pancake flip we do the following steps:

* Choose an integer k where 1 <= k <= arr.length.
* Reverse the sub-array arr[0...k-1] (**0-indexed**).

For example, if arr = [3,2,1,4] and we performed a pancake flip choosing k = 3, we reverse the sub-array [3,2,1], so arr = [1,2,3,4] after the pancake flip at k = 3.

Return *an array of the*k*-values corresponding to a sequence of pancake flips that sort*arr. Any valid answer that sorts the array within 10 \* arr.length flips will be judged as correct.

**Input Format:**

First line of the input contains integer n (length of array)

Second line contain n space separated integer.

Ex:

4

3 2 4 1

**Output Format:** Print an array of the k-values corresponding to a sequence of pancake flips that sort arr. Any valid answer that sorts the array within 10 \* arr.length flips will be judged as correct.

**Constraints:**

* 1 <= arr.length <= 100
* 1 <= arr[i] <= arr.length
* All integers in arr are unique (i.e. arr is a permutation of the integers from 1 to arr.length).

**Examples:**

**Input:** arr = [3,2,4,1]

**Output:** [4,2,4,3]

**Explanation:**

We perform 4 pancake flips, with k values 4, 2, 4, and 3.

Starting state: arr = [3, 2, 4, 1]

After 1st flip (k = 4): arr = [1, 4, 2, 3]

After 2nd flip (k = 2): arr = [4, 1, 2, 3]

After 3rd flip (k = 4): arr = [3, 2, 1, 4]

After 4th flip (k = 3): arr = [1, 2, 3, 4], which is sorted.

**Brute force Solution:**

**Explanation:**

**Code:**

#include <bits/stdc++.h>

using namespace std;

void flip(vector<int>& A, int ind) {

    int s = 0, e = ind;

    while(s<e) {

        swap(A[s++], A[e--]);

    }

}

int find(int tar, vector<int> A) {

    for(int i=0;i<A.size();i++)

        if(tar == A[i])

            return i;

    return -1;

}

vector<int> pancakeSort(vector<int>& A) {

    int n = A.size();

    vector<int> ans;

    for(int i=n;i>0;i--)  {

        int ind = find(i, A);

        if(ind != (i - 1)) {

            if (ind != 0) {

                flip(A, ind);

                ans.push\_back(ind + 1);

            }

            flip(A, i - 1);

            ans.push\_back(i);

        }

    }

    return ans;

}

int main() {

    int n;

    cin>>n;

    vector<int> arr(n);

    vector<int> result;

    for(int i=0; i<n; i++){

        cin>>arr[i];

    }

    result = pancakeSort(arr);

    for(int i=0; i<result.size(); i++){

        cout<<result[i]<<" ";

    }

    return 0;

}

**Time Complexity**: O(N2)

**Space Complexity:** O(1)

**Optimized Solution:**

**Explanation:** similar to selection sort starting with largest element.  
Start with the largest element and try to place it at it's correct position with at most 2 swaps.  
Find the cur position of the element and check if it is the right position, if not swap till that pos (so that element comes at the first pos) and then swap again till its right pos ( to place that element to its right position)

**Code:**

#include <bits/stdc++.h>

using namespace std;

int getPos(vector<int>& arr, int k)

{

    for(int i=0;i<arr.size();i++)

    {

        if(arr[i]==k)

            return i;

    }

    return -1;

}

void reverseArr(vector<int>& arr, int k)

{

    int i=0,j=k;

    while(i<j)

    {

        int temp = arr[i];

        arr[i]=arr[j];

        arr[j]=temp;

        i++;

        j--;

    }

}

vector<int> pancakeSort(vector<int>& arr) {

    int n = arr.size();

    vector<int> res;

    for(int i=n;i>0;i--)

    {

        int pos = getPos(arr,i);

        if(pos==i-1)

            continue;

        else if(pos!=0)

        {

            res.push\_back(pos+1);

            reverseArr(arr,pos);

        }

        res.push\_back(i);

        reverseArr(arr,i-1);

    }

    return res;

}

int main() {

    int n;

    cin>>n;

    vector<int> arr(n);

    vector<int> result;

    for(int i=0; i<n; i++){

        cin>>arr[i];

    }

    result = pancakeSort(arr);

    for(int i=0; i<result.size(); i++){

        cout<<result[i]<<" ";

    }

    return 0;

}

**Time Complexity**: O(N)

**Space Complexity:** O(1)