Johnson Trotter 2

locked

You are given an array 'nums' containing distinct elements, an array 'dir' of the same length representing the initial direction of each element, and an integer k. Your task is to find the k-th next permutation in Johnson-Trotter order starting from the given permutation. If no such permutation exists print 'Permutation does not exist'.

The Johnson-Trotter algorithm is used to generate permutations by identifying the largest mobile element—A directed integer is said to be mobile if it is greater than its immediate neighbor in the direction it is looking at.—and swapping it with its adjacent element. Further the directions of all the elements greater than the mobile element is reversed. The process is repeated to generate successive permutations.

Your task is to implement this algorithm and determine the k-th next permutation after performing k iterations from the given state.

Boilerplate

```c
#include <stdio.h>
#include <stdbool.h>

//Complete the function
void johnsonTrotter(int n, int nums[], int dir[], int k) {

}

int main() {
    int n, k;

    scanf("%d", &n);

    int nums[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &nums[i]);
    }

    int dir[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &dir[i]);
    }

    scanf("%d", &k);

    johnsonTrotter(n, nums, dir, k);

    return 0;
}
```

Input Format

- The first line contains an integer n (size of the array).
- The second line contains n space-separated distinct integers representing the array nums.
- The third line contains n space-separated integers (0 or 1) representing the array dir, where:

- 0 indicates the element is moving left.
- 1 indicates the element is moving right.

- The fourth line contains an integer k, representing the number of steps to generate the k-th next permutation.

Constraints

1 ≤ n ≤ 10 1 ≤ each element in nums ≤ 100 nums contains distinct elements. 1 ≤ k ≤ min(100,n!)

Output Format

- Print n space-separated integers representing the k-th permutation after performing k iterations using the Johnson-Trotter algorithm.
- 'Permutation does not exist' if it is not possible to find the k-th permutation from the current permutation

Sample Input 0

```
1
5
0
1
```

Sample Output 0

```
Permutation does not exist
```

Explanation 0

There is no mobile component, 5 is the last possible permutation, no permutation exists after it

Sample Input 1

```
3
1 2 3
0 0 1
2
```

Sample Output 1

```
2 3 1
```

Explanation 1

initially: 1< 2< 3> Iteration 1: mobile element: 2 permutation: 2< 1< 3<

Iteration 2: mobile element: 3 permutation: 2< 3< 1<

Submissions: 59
Max Score: 10
Difficulty: Medium

Rate This Challenge:

More

C

```c
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>

int main() {

    /* Enter your code here. Read input from STDIN. Print output to STDOUT */
    return 0;
}
```

Line: 1 Col: 1

Upload Code as File Test against custom input Run Code Submit Code