

Almost Sorted

Given an array with n elements, can you sort this array in *ascending order* using only one of the following operations?

- 1. Swap two elements.
- 2. Reverse one sub-segment.

Input Format

The first line contains a single integer, n , which indicates the size of the array.
The next line contains n integers separated by spaces.

```
n
d1 d2 ... dn
```

Constraints

$2 \leq n \leq 100000$
 $0 \leq d_i \leq 1000000$
All d_i are distinct.

Output Format

- 1. If the array is already sorted, output *yes* on the first line. You do not need to output anything else.
 - 1. If you can sort this array using one single operation (from the two permitted operations) then output *yes* on the first line and then:
 - a. If you can sort the array by swapping d_l and d_r , output *swap / r* in the second line. l and r are the indices of the elements to be swapped, assuming that the array is indexed from 1 to n .
 - b. Else if it is possible to sort the array by reversing the segment $d[l..r]$, output *reverse / r* in the second line. l and r are the indices of the first and last elements of the subsequence to be reversed, assuming that the array is indexed from 1 to n .
- $d[l..r]$ represents the sub-sequence of the array, beginning at index l and ending at index r , both inclusive.
- If an array can be sorted by either swapping or reversing, stick to the swap-based method.
- 2. If you cannot sort the array in either of the above ways, output *no* in the first line.

Sample Input #1

```
2
4 2
```

Sample Output #1

```
yes
swap 1 2
```

Sample Input #2

```
3
3 1 2
```

Sample Output #2

```
no
```

Sample Input #3

```
6
1 5 4 3 2 6
```

Sample Output #3

```
yes
reverse 2 5
```

Explanation

For #1, you can both $swap(1, 2)$ and $reverse(1, 2)$, but if you can sort the array using swap, output swap only.

For #2, it is impossible to sort by one single operation (among those permitted).

For #3, you can reverse the sub-array $d[2...5] = "5\ 4\ 3\ 2"$, then the array becomes sorted.