HW2

Rules & Instruction

- Compiler of programming problem.
  - C：`gcc -DONLINE_JUDGE -O2 -w -fmax-errors=3 -std=c11 main.c -lm -o main`
  - C++: `g++ -DONLINE_JUDGE -O2 -w -fmax-errors=3 -std=c++17 main.cpp -lm -o main`
  - Execution: `./main`
- Can I use theorem that haven't been mentioned in class ?
  - Programming: No restriction.(無限制)
  - Handwritten: Please include its proof.(請寫上證明。)
- Rules of cheating.
  - Programming: We use tools to monitor code similarity.(我們用機器抓抄襲)
  - Handwritten: Though we forbid plagiarism, we still encourage you to discuss with each other. Remember that after discussion you must answer problems **in your own words**. (禁止抄襲，鼓勵大家有問題可以互相討論，**但請用自己的話去作答**。)
  - **Copycats will be scored 0.** You also need to have a cup of coffee with professor Yeh. (**抄襲者與被抄襲者當次作業零分**，另外會跟教授喝杯咖啡☕。)
    - People who committed twice or more will be punished. (情節嚴重者，如累犯...，以校規處置)
- Rules of delay.
  NO LATE SUBMISSION IS ALLOWED.
  That is, zero toleration of late submission.

Non-Programming

> The following algorithms will make the input sequence ascending if it does not mention ascending or descending order.

> All the answers without explanation will be scored 0.

1. Quick Sort Attack

Quick Sort(A, p, r)

```
if p < r
    q = PARTITION(A, p, r)
    QUICKSORT(A, p, q - 1)
    QUICKSORT(A, q + 1, r)
```

PARTITION(A, p, r)

```
x = A[r]
i = p - 1
for j = p to r - 1
    if A[j] ≤ x
        i = i + 1
        exchange A[i] with A[j]
exchange A[i+1] with A[r]
return i + 1
```

The above is the quick sort algorithm pseudo code, please find the answer.
(1) $(5, 5, 15\ Points)$
Find out the worst, the best, and average time complexity(using Big-Theta notation) of the above quick sort algorithm, and prove or explain why your answer is correct.

(2) $(20\ Points)$
Please complete following C-code function to generate a sequence to make the above algorithm always sort it with the worst time complexity, and briefly explain why your code can do this.

```c
int *attack(int n){
    int *arr = malloc(n * sizeof(int));
    // your code here
    return arr;
}
```

2. Bubble Sort

Given a sequence $A = [a_1, a_2, a_3 \ldots, a_n]$, we could say (i, j) an inversion if $1 \leq i \leq j \leq n$ and $a_i > a_j$. Let $I(A)$ mean the number of inversions.

BubbleSort(A)

```
for i = 1 to A.length - 1
    for j = A.length downto i + 1
        if A[j] < A[j - 1]
            exchange A[j] with A[j-1]
```

Above is the pseudo code of bubble sort.

(1)

- (a) $(5\ Points)$
  Suppose a_i is $k - th$ small elements, which index is it at after the whole array is sorted by bubble sort? (index started from 1)
- (b) $(10\ Points)$
  Let x be the number of elements which are larger than a_i and are on his left-hand side in the initial array(before sorting), and let y be the number of elements which are smaller than a_i and are on his right-hand side.
  How many exchanges does a_i encounter during the bubble sort procedure?
- (c) $(10\ Points)$
  Following the previous question.
  How many inversions are formed by a_i ? That is, the number of inversions containing a_i.

(2)
Please try to explain or prove why the number of exchanges equal to $I(A)$, the pairs of inversion.

- (a) $(21\ Points)$
  Please prove why after one swapping, the $I(A)$ decrease exactly 1.
- (b) $(9\ Points)$
  Using the conclusion from (a), please prove or explain why $I(A)$ equals to the numbers of bubble sort swap.

Hint

```
int A[] = {1, 5, 2, 4, 3}
```

$I(A) = 4$
because (5, 2), (5, 3), (5, 4), (4,3) are all inversions.

Programming

pA: Ramen Ranking

> Time Limit: 1s
> Memory Limit: 268436KB

Description

After finishing $\mathrm{Handwritten\ 2.\ Bubble\ Sort}$, Da-Hei-Hei(a person) thinks this sorting algorithm can help him to rank ラーメン(ramen).

This is how he does to rank ramen with bubble sort:

1. First of all, Da-Hei-Hei buys N bowls of ramen from N different restaurants(Let's call them $R_1,\ R_2,\ \ldots,\ R_N$), the deliciousity of $R_1,\ R_2,\ \ldots,\ R_N$ is denoted by $D_1,\ D_2,\ \ldots,\ D_N$.
2. He lines up $R_1,\ R_2,\ \ldots$ to R_N on the table, and tastes from the first one to the last. For each pair of adjacent bowls, R_i and R_j for example, he will exchange them if he thinks R_i tastes better than R_j, or do nothing otherwise. Formally, to say R_i tastes better than R_j if and only if $D_i > D_j$.
3. This procedure(tastes every bowls) is called a $\mathrm{Round\ of\ Taste}$.
4. After some $\mathrm{Round\ of\ Tastes}$, he ranks ramen from the worse to the best successfully, that is, deliciousity is sorted in ascending order, and is on a full stomach also.

He wants to know how many $\mathrm{Round\ of\ Tastes}$ are needed to rank ramen in ascending order.

The following explained the first Sample Input/Output:
$[D_1, D_2, D_3, D_4, D_5]$ = $[2, 1, 4, 5, 3]$.
After the first $\mathrm{Round\ of\ Taste}$, it becomes $[1, 2, 4, 3, 5]$, after the second $\mathrm{Round\ of\ Taste}$, it becomes $[1, 2, 3, 4, 5]$.
Successfully rank them in ascending order by $2\ \mathrm{Round\ of\ Taste}$.

Input Format

First line contains an integer N, indicating Da-Hei-Hei buys N bowls of ramen.
Second line contains N space-separated integers $D_1,\ D_2,\ \ldots\ ,\ D_N$, indicating the deliciousity of each bowl of ramen, namely, deliciousity of $R_1,\ R_2,\ \ldots\ ,\ R_N$.

For all test data, it is guaranteed:

- $1 \leq N \leq 2 \times 10^5$
- $0 \leq D_i \leq 10^9,\ \text{for } 1 \leq i \leq N$

- for each pair of $(i,\ j),\ D_i \neq D_j$

Subtask1 (25%)

- $1 \leq N \leq 3 \times 10^3$

Subtask2 (75%)

- No other restrictions.

Output Format

Please print a single number, indicating how many $\mathrm{Round\ of\ Taste}$ that Da-Hei-Hei needs.

Sample Input

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

Sample Output

```
2
```

Sample Input

```
10
89 7 22 35 4 11 23 90 55 64
```

Sample Output

```
4
```

Hint

Because the input files are large, please add

```
std::ios_base::sync_with_stdio(false);
std::cin.tie(nullptr);
```

to the beginning of the main function if you are using std::cin.

Who is Da-Hei-Hei?
Don't ask... or...

pB: oo-pe̍h sort

> Time Limit: 1s
> Memory Limit: 268436KB

Description

So brilliant as A-Hua is, he not only passes Software Engineering, but also passes Algorithm, and he likes Algorithm, especially Sorting!

Moreover, A-Hua wants to create a new sorting algorithm. After a little while, he came up with a new sorting algorithm.
Since this sorting algorithm was created casually, he named this algorithm **oo-pe̍h Sort**(in Taiwanese).

Here's the pseudo code of oo-pe̍h sort.

```
// A is a 0-indexed array of integers
oo_pe̍h_sort(A[0..n-1])
    Sorted = false
    Counter = 0
    while Sorted == false
        Sorted = true
        for i=0; i<len(Arr)-2; i=i+1
            if A[i] > A[i+2]
                Sorted = false
                Reverse the subarray A[i..i+2]
                Couter = Couter + 1
```

The following is the demonstration with A = [3, 4, 5, 2, 1]

- First time in while loop:
  - check if `A[0] > A[2]`, `3 > 5`, nothing happened
  - check if `A[1] > A[3]`, `4 > 2`, A = [3 2 5 4 1], Counter = 1
  - check if `A[2] > A[4]`, `5 > 1`, A = [3 2 1 4 5], Counter = 2
- Second time in while loop:
  - check if `A[0] > A[2]`, `3 > 1`, A = [1 2 3 4 5], Counter = 3
  - check if `A[1] > A[3]`, `2 > 4`, nothing happened
  - check if `A[2] > A[4]`, `3 > 5`, nothing happened
- Third time in while loop: for all `A[i] > A[i+2]` is false, and program is over.

Unfortunately, A-Hua's roommates found that this method may not sort the sequence correctly, for example, A = [2 3 1].

But A-Hua will not give up at this time, he wants to do a research about what situation will oo-pėh sort fail.
Therefore, he needs your help, and he give your a sequence of N numbers, tell him whether or not this sequence can be sorted(ascending order) by oo-pėh sort.

He wants to analyze the performance as well, so please also tell him the value `Counter` after oo-pėh sort.

Input Format

First line contains an integer N, indicating the length of the sequence given by A-Hua.
Second line contains N space-separated integers $a_1,\ a_2,\ \ldots,\ a_N$, indicating the numbers in sequence.

For all test data, it is guaranteed:

- $3 \leq N \leq 2 \times 10^5$
- $0 \leq a_i \leq 10^9,\ \text{for } 1 \leq i \leq N$
- $\text{for each pair of } (i,\ j),\ a_i \neq a_j$

Subtask1 (25%)

- $3 \leq N \leq 3 \times 10^3$

Subtask2 (75%)

- No other restrictions.

Output Format

On first line, print "yes"(without quote), if the given sequence can be sorted in ascending order by oo-pėh sort, and "no" otherwise.
On second line, print an integer, indicating the value of `Counter`.

Sample Input

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

Sample Output

```
yes
3
```

Sample Input

```
6
3 7 2 6 1 5
```

Sample Output

```
no
6
```

Hint

Because the input files are large, please add

```
std::ios_base::sync_with_stdio(false);
std::cin.tie(nullptr);
```

to the beginning of the main function if you are using std::cin.

Bonus: Teleport Again

> Time Limit: 1s
> Memory Limit: 268436KB

Description

Bogay, one of the greatest wizard in NTNU, is sharpening the skill: $Teleport$. (See this problem first if you haven't!)

Thanks to your help, Bogay is now good at this skill.
But he met a new problem: every time he wants to move forward at least D distance, he has to check if he can move forward D, $D+1$..., until finding it or in vain.

So this time he wonders how many different ways can he launch $Teleport$ such that he can move forward **at least** K.

However, the Mastery Book that he bought was cursed! Some pages' magic number has turned into negative, and if Bogay launch $Teleport$ on this page, he'll move backward!

Input Format

First line contains two space-separated integers $N,\ D$, indicating the number of pages of Mastery Book, and the desired distance he wants to move forward at least.
Second line contains N space-separated integers $M_1,\ M_2,\ \ldots,\ M_N$, indicating the magic numbers on each page.

For all test data, it is guaranteed:

- $1 \leq N \leq 2 \times 10^5$
- $0 \leq D \leq 2 \times 10^9$
- $|M_i| \leq 10^5$

Subtask1 (100%)

- No other restrictions.

Output Format

Print a number, indicating how many different ways that Bogay can move forward at least D distance.

Sample Input

```
6 5
4 -6 3 1 -2 7
```

Sample Output

```
5
```

Sample Input

```
10 2
1 -2 3 -4 5 -6 -8 3 -1 1
```

Sample Output

```
9
```

Hint

In first sample,

$\{7\}, \{-2, 7\}, \{1, -2, 7\}, \{3, 1, -2, 7\}, \{4, -6, 3, 1, -2, 7\}$

are the 5 ways to move forward at least 5 distance by launching $Teleport$.

Because the input files are large, please add

```
std::ios_base::sync_with_stdio(false);
std::cin.tie(nullptr);
```

to the beginning of the main function if you are using std::cin.