



# [DSA WEEKLY CONTEST T8 2025]. TEST 3. SẮP XẾP & TÌM KIẾM

[Info](#)[Statistics](#)[Rankings](#)[Participation](#)[Submissions](#)[Virtual join](#)[DSA](#)

## MEX [Submit](#)

**Time limit:** 1.0 / **Memory limit:** 256M

**Point:** 100

Cho mảng **A[]** gồm **N** phần tử, bạn hãy xác định số nguyên dương nhỏ nhất chưa xuất hiện trong mảng **A[]**.

### Đầu vào

Dòng 1 là **N** : số lượng phần tử trong mảng **A[]**

Dòng 2 là **N** số của mảng **A[]**

### Giới hạn

$$1 \leq N \leq 10^6$$

$$0 \leq A[i] \leq 10^6$$

### Đầu ra

In ra kết quả của bài toán

### Ví dụ :

#### Input 01

[Copy](#)



1 2 3 7 8 10

**Output 01**

4

Copy

# [Sắp Xếp - Tìm Kiếm]. Bài 52.

## Sắp xếp mảng

Submit

**Time limit:** 1.0 / **Memory limit:** 256M**Point:** 100

Cho mảng **A[]** gồm **N** phần tử, **28tech** muốn bạn kiểm tra xem liệu có thể lật ngược 1 dãy con liên tiếp bất kỳ trong mảng 1 lần duy nhất để tạo thành mảng tăng dần hay không?

Ví dụ **A** = [1, 2, 5, 4, 3, 7, 8, 9] bạn có thể lật ngược lại đoạn [5, 4, 3] để tạo thành mảng [1, 2, 3, 4, 5, 7, 8, 9]

**Gợi ý :** Tìm left là chỉ số bắt đầu của dãy con cần lật ( $a[\text{left}] > a[\text{left} + 1]$ ) và chỉ số right là chỉ số cuối cùng của dãy con cần lật ( $a[\text{right}] < a[\text{right} - 1]$ ). Nếu left ko tồn tại tức mảng đã tăng dần rồi, còn nếu left và right tồn tại thì lật ngược đoạn đó là và kiểm tra xem sau khi lật thì mảng có tăng dần không?

**Đầu vào**

Dòng 1 là **N** : các phần tử trong mảng

Dòng 2 là **N** số trong mảng

**Giới hạn**

$$1 \leq N \leq 10^6$$

$$0 \leq A[i] \leq 10^9$$



In ra **28tech** nếu có thể lật ngược mảng con để tạo thành mảng tăng dần, ngược lại in ra **29tech**

**Ví dụ :**

**Input 01**

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

Copy

**Output 01**

```
28tech
```

Copy

**Input 02**

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

Copy

**Output 02**

```
29tech
```

Copy

[Sắp Xếp - Tìm Kiếm]. Bài 57.  
Xếp hàng

Submit

**Time limit:** 1.0 / **Memory limit:** 256M

**Point:** 100

Lớp cấu trúc dữ liệu và giải thuật của **28Tech** có **N** bạn tham gia với chiều cao khác nhau đôi một, ban đầu **N** bạn này được xếp vào 1 hàng ngang với thứ tự ngẫu nhiên.



Mỗi em học sinh trong lớp, có một chiều cao duy nhất và phân biệt với mọi em khác trong lớp. Số bạn trong hàng.

Nhiệm vụ của bạn là hãy giúp **28Tech** đếm xem cần tối thiểu bao nhiêu hoán đổi vị trí các bạn học viên để hàng người tăng dần về chiều cao.

## Đầu vào

Dòng 1 là **N** : số lượng học viên lớp CTDL & GT

Dòng 2 gồm **N** số khác nhau tương ứng với chiều cao của các bạn trong lớp

## Giới hạn

$$1 \leq N \leq 2 \cdot 10^5$$

Chiều cao là số nguyên dương 32bit

## Đầu ra

In ra số hoán đổi tối thiểu để sắp tăng dần chiều cao của các bạn học viên

## Ví dụ :

### Input 01

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

[Copy](#)

### Output 01

```
2
```

[Copy](#)

[Submit](#)

# Loại bỏ 'đoạn thẳng'

**Time limit:** 1.0 / **Memory limit:** 256M**Point:** 100

Cho **N** đoạn thẳng trên trục tọa độ Ox, mỗi đoạn thẳng bắt đầu từ hoành độ **L** và kết thúc tại hoành độ **R**. 2 đoạn thẳng được coi là không giao nhau nếu điểm bắt đầu của đoạn thẳng này lớn hơn hoặc bằng điểm kết thúc của đoạn thẳng trước.

Ví dụ [2, 5] và [5, 10] là 2 đoạn thẳng không giao nhau, trong khi đó [1, 3] và [2, 5] là 2 đoạn thẳng giao nhau.

**28Tech** cảm thấy khó chịu khi phải nhìn những đoạn thẳng bị giao cắt nhau, bây giờ anh ấy muốn bạn xóa đi 1 số đoạn thẳng ít nhất để tất cả những đoạn thẳng còn lại sẽ không còn giao nhau.

## Đầu vào

Dòng 1 là **N** : số lượng đoạn thẳng

**N** dòng tiếp theo mỗi dòng là [**Li**, **Ri**] tương ứng với điểm bắt đầu và kết thúc của đoạn thẳng thứ **i**

## Giới hạn

$$1 \leq N \leq 10^5$$

$$0 \leq L[i] < R[i] \leq 10^9$$

## Đầu ra

In ra số lượng đoạn thẳng ít nhất cần loại bỏ để những đoạn thẳng còn lại không bị giao nhau

## Ví dụ :

## Input 01



```
4 5
2 3
1 4
6 7
5 9
```

### Output 01

```
2
```

[Copy](#)

### Giải thích test :

Loại bỏ đi đoạn thẳng  $[1, 4]$  và  $[5, 9]$  thì 3 đoạn thẳng còn lại sẽ không bị giao nhau

## [Sắp Xếp - Tìm Kiếm]. Bài 64. Painting matrix

[Submit](#)

**Time limit:** 1.0 / **Memory limit:** 256M

**Point:** 100

Cho một mảng 2 chiều gồm  $n$  hàng và  $m$  cột bao gồm  $nxm$  ô nhỏ, người ta sẽ sơn các ô trong mảng 2 chiều này theo từng dòng, mỗi lần sơn sẽ được biểu diễn bởi 3 số  $r, c1, c2$  tức là họ sẽ sơn ở hàng  $r$  từ cột  $c1$  tới cột  $c2$ . Có tất cả  $k$  lần sơn được thực hiện trên mảng 2 chiều này.

Tất nhiên là sẽ có những ô được sơn nhiều lần nhưng cũng có những ô không được sơn lần nào cả, nhiệm vụ của bạn là hãy đếm xem có tất cả bao nhiêu ô không hề bị sơn 1 lần nào cả.

### Đầu vào

- Dòng 1 là  $n, m$  và  $k$
- $k$  dòng tiếp theo mỗi dòng là bộ 3 số  $r, c1, c2$

### Giới hạn



- $0 \leq k \leq 1000$
- $1 \leq r \leq n$
- $1 \leq c1 \leq c2 \leq m$

### Đầu ra

- In ra số ô không bị sơn 1 lần nào cả.

### Ví dụ :

#### Input 01

```
1 7 3
1 1 2
1 2 4
1 3 5
```

[Copy](#)

#### Output 01

```
2
```

[Copy](#)

## [Sắp Xếp - Tìm Kiếm]. Bài 51. Đếm cặp

[Submit](#)

**Time limit:** 1.0 / **Memory limit:** 256M

**Point:** 100

Cho mảng **A[]** có **N** phần tử, mảng **B[]** có **M** phần tử. Gọi **a** là những phần tử thuộc mảng **A[]**, **b** là những phần tử thuộc mảng **B[]**, nhiệm vụ của bạn là hãy đếm số cặp **(a, b)** thỏa mãn  $a^b > b^a$ .

**Ví dụ** **A** = [3, 9, 7, 2] và **B** = [2, 5] thì có những cặp thỏa mãn là (3, 2), (3, 5), (2, 5)



... và ngược lại sắp xếp (x/1), (x/2), ... và ngược lại sắp xếp (x/2), (x/4), ... và ngược lại sắp xếp (x/4), (x/8), ... và ngược lại sắp xếp (x/8), (x/16), ... và ngược lại sắp xếp (x/16), (x/32), ... và ngược lại sắp xếp (x/32), (x/64), ... và ngược lại sắp xếp (x/64), (x/128), ... và ngược lại sắp xếp (x/128), (x/256), ... và ngược lại sắp xếp (x/256), (x/512), ... và ngược lại sắp xếp (x/512), (x/1024), ... và ngược lại sắp xếp (x/1024), (x/2048), ... và ngược lại sắp xếp (x/2048), (x/4096), ... và ngược lại sắp xếp (x/4096), (x/8192), ... và ngược lại sắp xếp (x/8192), (x/16384), ... và ngược lại sắp xếp (x/16384), (x/32768), ... và ngược lại sắp xếp (x/32768), (x/65536), ... và ngược lại sắp xếp (x/65536), (x/131072), ... và ngược lại sắp xếp (x/131072), (x/262144), ... và ngược lại sắp xếp (x/262144), (x/524288), ... và ngược lại sắp xếp (x/524288), (x/1048576), ... và ngược lại sắp xếp (x/1048576), (x/2097152), ... và ngược lại sắp xếp (x/2097152), (x/4194304), ... và ngược lại sắp xếp (x/4194304), (x/8388608), ... và ngược lại sắp xếp (x/8388608), (x/16777216), ... và ngược lại sắp xếp (x/16777216), (x/33554432), ... và ngược lại sắp xếp (x/33554432), (x/67108864), ... và ngược lại sắp xếp (x/67108864), (x/134217728), ... và ngược lại sắp xếp (x/134217728), (x/268435456), ... và ngược lại sắp xếp (x/268435456), (x/536870912), ... và ngược lại sắp xếp (x/536870912), (x/1073741824), ... và ngược lại sắp xếp (x/1073741824), (x/2147483648), ... và ngược lại sắp xếp (x/2147483648), (x/4294967296), ... và ngược lại sắp xếp (x/4294967296), (x/8589934592), ... và ngược lại sắp xếp (x/8589934592), (x/17179869184), ... và ngược lại sắp xếp (x/17179869184), (x/34359738368), ... và ngược lại sắp xếp (x/34359738368), (x/68719476736), ... và ngược lại sắp xếp (x/68719476736), (x/137438953472), ... và ngược lại sắp xếp (x/137438953472), (x/274877906944), ... và ngược lại sắp xếp (x/274877906944), (x/549755813888), ... và ngược lại sắp xếp (x/549755813888), (x/1099511627776), ... và ngược lại sắp xếp (x/1099511627776), (x/2199023255552), ... và ngược lại sắp xếp (x/2199023255552), (x/4398046511104), ... và ngược lại sắp xếp (x/4398046511104), (x/8796093022208), ... và ngược lại sắp xếp (x/8796093022208), (x/17592186044416), ... và ngược lại sắp xếp (x/17592186044416), (x/35184372088832), ... và ngược lại sắp xếp (x/35184372088832), (x/70368744177664), ... và ngược lại sắp xếp (x/70368744177664), (x/140737488355328), ... và ngược lại sắp xếp (x/140737488355328), (x/281474976710656), ... và ngược lại sắp xếp (x/281474976710656), (x/562949953421312), ... và ngược lại sắp xếp (x/562949953421312), (x/1125899906842624), ... và ngược lại sắp xếp (x/1125899906842624), (x/2251799813685248), ... và ngược lại sắp xếp (x/2251799813685248), (x/4503599627370496), ... và ngược lại sắp xếp (x/4503599627370496), (x/9007199254740992), ... và ngược lại sắp xếp (x/9007199254740992), (x/18014398509481984), ... và ngược lại sắp xếp (x/18014398509481984), (x/36028797018963968), ... và ngược lại sắp xếp (x/36028797018963968), (x/72057594037927936), ... và ngược lại sắp xếp (x/72057594037927936), (x/144115188075855872), ... và ngược lại sắp xếp (x/144115188075855872), (x/288230376151711744), ... và ngược lại sắp xếp (x/288230376151711744), (x/576460752303423488), ... và ngược lại sắp xếp (x/576460752303423488), (x/1152921504606846976), ... và ngược lại sắp xếp (x/1152921504606846976), (x/2305843009213693952), ... và ngược lại sắp xếp (x/2305843009213693952), (x/4611686018427387904), ... và ngược lại sắp xếp (x/4611686018427387904), (x/9223372036854775808), ... và ngược lại sắp xếp (x/9223372036854775808), (x/18446744073709551616), ... và ngược lại sắp xếp (x/18446744073709551616), (x/36893488147419103232), ... và ngược lại sắp xếp (x/36893488147419103232), (x/73786976294838206464), ... và ngược lại sắp xếp (x/73786976294838206464), (x/147573952589676412928), ... và ngược lại sắp xếp (x/147573952589676412928), (x/295147905179352825856), ... và ngược lại sắp xếp (x/295147905179352825856), (x/590295810358705651712), ... và ngược lại sắp xếp (x/590295810358705651712), (x/1180591620717411303424), ... và ngược lại sắp xếp (x/1180591620717411303424), (x/2361183241434822606848), ... và ngược lại sắp xếp (x/2361183241434822606848), (x/4722366482869645213696), ... và ngược lại sắp xếp (x/4722366482869645213696), (x/9444732965739290427392), ... và ngược lại sắp xếp (x/9444732965739290427392), (x/18889465931478580854784), ... và ngược lại sắp xếp (x/18889465931478580854784), (x/37778931862957161709568), ... và ngược lại sắp xếp (x/37778931862957161709568), (x/75557863725914323419136), ... và ngược lại sắp xếp (x/75557863725914323419136), (x/151115727451828646838272), ... và ngược lại sắp xếp (x/151115727451828646838272), (x/302231454903657293676544), ... và ngược lại sắp xếp (x/302231454903657293676544), (x/604462909807314587353088), ... và ngược lại sắp xếp (x/604462909807314587353088), (x/1208925819614629174706176), ... và ngược lại sắp xếp (x/1208925819614629174706176), (x/2417851639229258349412352), ... và ngược lại sắp xếp (x/2417851639229258349412352), (x/4835703278458516698824704), ... và ngược lại sắp xếp (x/4835703278458516698824704), (x/9671406556917033397649408), ... và ngược lại sắp xếp (x/9671406556917033397649408), (x/19342813113834066795298816), ... và ngược lại sắp xếp (x/19342813113834066795298816), (x/38685626227668133590597632), ... và ngược lại sắp xếp (x/38685626227668133590597632), (x/77371252455336267181195264), ... và ngược lại sắp xếp (x/77371252455336267181195264), (x/154742504910672534362390528), ... và ngược lại sắp xếp (x/154742504910672534362390528), (x/309485009821345068724781056), ... và ngược lại sắp xếp (x/309485009821345068724781056), (x/618970019642690137449562112), ... và ngược lại sắp xếp (x/618970019642690137449562112), (x/1237940039285380274899124224), ... và ngược lại sắp xếp (x/1237940039285380274899124224), (x/2475880078570760549798248448), ... và ngược lại sắp xếp (x/2475880078570760549798248448), (x/4951760157141521099596496896), ... và ngược lại sắp xếp (x/4951760157141521099596496896), (x/9903520314283042199192993792), ... và ngược lại sắp xếp (x/9903520314283042199192993792), (x/19807040628566084398385987584), ... và ngược lại sắp xếp (x/19807040628566084398385987584), (x/39614081257132168796771975168), ... và ngược lại sắp xếp (x/39614081257132168796771975168), (x/79228162514264337593543950336), ... và ngược lại sắp xếp (x/79228162514264337593543950336), (x/158456325028528675187087900672), ... và ngược lại sắp xếp (x/158456325028528675187087900672), (x/316912650057057350374175801344), ... và ngược lại sắp xếp (x/316912650057057350374175801344), (x/633825300114114700748351602688), ... và ngược lại sắp xếp (x/633825300114114700748351602688), (x/1267650600228229401496703205376), ... và ngược lại sắp xếp (x/1267650600228229401496703205376), (x/2535301200456458802993406410752), ... và ngược lại sắp xếp (x/2535301200456458802993406410752), (x/5070602400912917605986812821504), ... và ngược lại sắp xếp (x/5070602400912917605986812821504), (x/10141204801825835211973625643008), ... và ngược lại sắp xếp (x/10141204801825835211973625643008), (x/20282409603651670423947251286016), ... và ngược lại sắp xếp (x/20282409603651670423947251286016), (x/40564819207303340847894502572032), ... và ngược lại sắp xếp (x/40564819207303340847894502572032), (x/81129638414606681695789005144064), ... và ngược lại sắp xếp (x/81129638414606681695789005144064), (x/162259276829213363391578010288128), ... và ngược lại sắp xếp (x/162259276829213363391578010288128), (x/324518553658426726783156020576256), ... và ngược lại sắp xếp (x/324518553658426726783156020576256), (x/649037107316853453566312041152512), ... và ngược lại sắp xếp (x/649037107316853453566312041152512), (x/1298074214633706907132624082305024), ... và ngược lại sắp xếp (x/1298074214633706907132624082305024), (x/2596148429267413814265248164610048), ... và ngược lại sắp xếp (x/2596148429267413814265248164610048), (x/5192296858534827628530496329220096), ... và ngược lại sắp xếp (x/5192296858534827628530496329220096), (x/10384593717069655257060992658440192), ... và ngược lại sắp xếp (x/10384593717069655257060992658440192), (x/20769187434139310514121985316880384), ... và ngược lại sắp xếp (x/20769187434139310514121985316880384), (x/41538374868278621028243970633760768), ... và ngược lại sắp xếp (x/41538374868278621028243970633760768), (x/83076749736557242056487941267521536), ... và ngược lại sắp xếp (x/83076749736557242056487941267521536), (x/166153499473114484112975882535043072), ... và ngược lại sắp xếp (x/166153499473114484112975882535043072), (x/332306998946228968225951765070086144), ... và ngược lại sắp xếp (x/332306998946228968225951765070086144), (x/664613997892457936451903530140172288), ... và ngược lại sắp xếp (x/664613997892457936451903530140172288), (x/1329227995784915872903807060280344576), ... và ngược lại sắp xếp (x/1329227995784915872903807060280344576), (x/2658455991569831745807614120560689152), ... và ngược lại sắp xếp (x/2658455991569831745807614120560689152), (x/5316911983139663491615228241121378304), ... và ngược lại sắp xếp (x/5316911983139663491615228241121378304), (x/10633823966279326983230456482242756608), ... và ngược lại sắp xếp (x/10633823966279326983230456482242756608), (x/21267647932558653966460912964485513216), ... và ngược lại sắp xếp (x/21267647932558653966460912964485513216), (x/42535295865117307932921825928971026432), ... và ngược lại sắp xếp (x/42535295865117307932921825928971026432), (x/85070591730234615865843651857942052864), ... và ngược lại sắp xếp (x/85070591730234615865843651857942052864), (x/170141183460469231731687303715884105728), ... và ngược lại sắp xếp (x/170141183460469231731687303715884105728), (x/340282366920938463463374607431768211456), ... và ngược lại sắp xếp (x/340282366920938463463374607431768211456), (x/680564733841876926926749214863536422912), ... và ngược lại sắp xếp (x/680564733841876926926749214863536422912), (x/1361129467683753853853498429727072845824), ... và ngược lại sắp xếp (x/1361129467683753853853498429727072845824), (x/2722258935367507707706996859454145691648), ... và ngược lại sắp xếp (x/2722258935367507707706996859454145691648), (x/5444517870735015415413993718908291383296), ... và ngược lại sắp xếp (x/5444517870735015415413993718908291383296), (x/10889035741470030830827987437816582766592), ... và ngược lại sắp xếp (x/10889035741470030830827987437816582766592), (x/21778071482940061661655974875633165533184), ... và ngược lại sắp xếp (x/21778071482940061661655974875633165533184), (x/43556142965880123323311949751266331066368), ... và ngược lại sắp xếp (x/43556142965880123323311949751266331066368), (x/87112285931760246646623899502532662132736), ... và ngược lại sắp xếp (x/87112285931760246646623899502532662132736), (x/174224571863520493293247799005065324265472), ... và ngược lại sắp xếp (x/174224571863520493293247799005065324265472), (x/348449143727040986586495598010130648530944), ... và ngược lại sắp xếp (x/348449143727040986586495598010130648530944), (x/696898287454081973172991196020261297061888), ... và ngược lại sắp xếp (x/696898287454081973172991196020261297061888), (x/1393796574908163946345982392040522594123776), ... và ngược lại sắp xếp (x/1393796574908163946345982392040522594123776), (x/2787593149816327892691964784081045188247552), ... và ngược lại sắp xếp (x/2787593149816327892691964784081045188247552), (x/5575186299632655785383929568162090376495104), ... và ngược lại sắp xếp (x/5575186299632655785383929568162090376495104), (x/11150372599265311570767859136324180752990208), ... và ngược lại sắp xếp (x/11150372599265311570767859136324180752990208), (x/22300745198530623141535718272648361505980416), ... và ngược lại sắp xếp (x/22300745198530623141535718272648361505980416), (x/44601490397061246283071436545296723011960832), ... và ngược lại sắp xếp (x/44601490397061246283071436545296723011960832), (x/89202980794122492566142873090593446023921664), ... và ngược lại sắp xếp (x/89202980794122492566142873090593446023921664), (x/178405961588244985132285746181186892047843328), ... và ngược lại sắp xếp (x/178405961588244985132285746181186892047843328), (x/356811923176489970264571492362373784095686656), ... và ngược lại sắp xếp (x/356811923176489970264571492362373784095686656), (x/713623846352979940529142984724747568191373312), ... và ngược lại sắp xếp (x/713623846352979940529142984724747568191373312), (x/1427247692705959881058285969449495136382746624), ... và ngược lại sắp xếp (x/1427247692705959881058285969449495136382746624), (x/2854495385411919762116571938898990272765493248), ... và ngược lại sắp xếp (x/2854495385411919762116571938898990272765493248), (x/5708990770823839524233143877797980545530986496), ... và ngược lại sắp xếp (x/5708990770823839524233143877797980545530986496), (x/11417981541647679048466287755595961091061972992), ... và ngược lại sắp xếp (x/11417981541647679048466287755595961091061972992), (x/22835963083295358096932575511191922182123945984), ... và ngược lại sắp xếp (x/22835963083295358096932575511191922182123945984), (x/45671926166590716193865151022383844364247891968), ... và ngược lại sắp xếp (x/45671926166590716193865151022383844364247891968), (x/91343852333181432387730302044767688728495783936), ... và ngược lại sắp xếp (x/91343852333181432387730302044767688728495783936), (x/182687704666362864775460604089535377456991567872), ... và ngược lại sắp xếp (x/182687704666362864775460604089535377456991567872), (x/365375409332725729550921208179070754913983135744), ... và ngược lại sắp xếp (x/365375409332725729550921208179070754913983135744), (x/730750818665451459101842416358141509827966271488), ... và ngược lại sắp xếp (x/730750818665451459101842416358141509827966271488), (x/1461501637330902918203684832716283019655932542976), ... và ngược lại sắp xếp (x/1461501637330902918203684832716283019655932542976), (x/2923003274661805836407369665432566039311865085952), ... và ngược lại sắp xếp (x/2923003274661805836407369665432566039311865085952), (x/5846006549323611672814739330865132078623730171904), ... và ngược lại sắp xếp (x/5846006549323611672814739330865132078623730171904), (x/11692013098647223345629478661730264157247460343808), ... và ngược lại sắp xếp (x/11692013098647223345629478661730264157247460343808), (x/23384026197294446691258957323460528314494920687616), ... và ngược lại sắp xếp (x/23384026197294446691258957323460528314494920687616), (x/46768052394588893382517914646921056628989841375232), ... và ngược lại sắp xếp (x/46768052394588893382517914646921056628989841375232), (x/93536104789177786765035829293842113257979682750464), ... và ngược lại sắp xếp (x/93536104789177786765035829293842113257979682750464), (x/187072209578355573530071658587684226515959365500928), ... và ngược lại sắp xếp (x/187072209578355573530071658587684226515959365500928), (x/374144419156711147060143317175368453031918731001856), ... và ngược lại sắp xếp (x/374144419156711147060143317175368453031918731001856), (x/748288838313422294120286634350736906063837462003712), ... và ngược lại sắp xếp (x/748288838313422294120286634350736906063837462003712), (x/1496577676626844588240573268701473812127674924007424), ... và ngược lại sắp xếp (x/1496577676626844588240573268701473812127674924007424), (x/2993155353253689176481146537402947624255349848014848), ... và ngược lại sắp xếp (x/2993155353253689176481146537402947624255349848014848), (x/5986310706507378352962293074805895248510699696029696), ... và ngược lại sắp xếp (x/5986310706507378352962293074805895248510699696029696), (x/11972621413014756705924586149611790497021399392059392), ... và ngược lại sắp xếp (x/11972621413014756705924586149611790497021399392059392), (x/23945242826029513411849172299223580994042798784118784), ... và ngược lại sắp xếp (x/23945242826029513411849172299223580994042798784118784), (x/47890485652059026823698344598447161988085597568237568), ... và ngược lại sắp xếp (x/47890485652059026823698344598447161988085597568237568), (x/95780971304118053647396689196894323976171195136475136), ... và ngược lại sắp xếp (x/95780971304118053647396689196894323976171195136475136), (x/191561942608236107294





# [Sắp Xếp - Tìm Kiếm]. Bài 54. Thấp hơn gần nhất

[Submit](#)**Time limit:** 1.0 / **Memory limit:** 256M**Point:** 100

Tiếp tục câu chuyện về chiều cao tại **Vương Quốc 28Tech**, hôm nay **Quốc Vương** tại **28Tech** muốn **N** cư dân của mình xếp thành 1 hàng dài và đánh số cho họ từ 1 tới **N**. **Quốc Vương** sẽ yêu cầu mỗi người tìm ra vị trí của người đứng trước gần họ nhất mà có chiều cao thấp hơn họ. Tất nhiên sẽ có những người không thể tìm được người thấp hơn mình. Nhiệm vụ của bạn rất đơn giản, hãy tìm vị trí của người đứng trước gần nhất của mọi cư dân trong **Vương Quốc** thịnh vượng này.

## Đầu vào

- Dòng 1 là **N** số lượng cư dân
- Dòng 2 là chiều cao của cư dân thứ 1 tới thứ **N**

## Giới hạn

$$1 \leq N \leq 10^6$$

Chiều cao của cư dân thuộc đoạn [1, 999999999]

## Đầu ra

In ra vị trí của người thấp hơn gần nhất với mỗi cư dân trong Vương Quốc, nếu không thể tìm được vị trí hợp lý này thì in ra 0.

## Ví dụ :

### Input 01

[Copy](#)



```
1 3 7 2 4 6
```

## Output 01

```
0 1 2 1 4 5
```

[Copy](#)

# [Sắp Xếp - Tìm Kiếm]. Bài 55.Vương Quốc 28Tech

[Submit](#)

**Time limit:** 1.0 / **Memory limit:** 256M

**Point:** 100

Tại **Vương Quốc 28Tech** có **N** cư dân, mỗi cư dân có một chiều cao . Vì yêu thích sự công bằng nên **Quốc Vương** của **28Tech** muốn tất cả mọi người đều có phải có cùng chiều cao với nhau để tránh tình trạng body shaming giữa các cư dân.

Bây giờ Quốc Vương sẽ đi tìm 1 người bất kỳ mà ông ta thích trong **N** người đó và chọn làm hạt giống, người này và những người có cùng chiều cao với hạt giống kia thực sự may mắn bởi vì Quốc Vương có một ý tưởng thực sự rất đáng sợ. Ông ta sẽ cắt bớt chân của những người cao hơn người hạt giống hoặc kéo chân những người thấp hơn người hạt giống sao cho tất cả mọi người có cùng chiều cao với người hạt giống. Tuy nhiên ý tưởng của **Quốc Vương** bị phản đối bởi cư dân trong **Vương Quốc** của mình thành ra **Quốc Vương** phải đi tìm một hạt giống sao cho việc cắt bớt chân và kéo dài chân gây ít đau đớn nhất. Quốc Vương nhờ bạn tìm giúp hạt giống này !

Biết rằng khi chiều cao của 1 dân cư là **X** cao(thấp) hơn người hạt giống có chiều cao là **Y** thì sự đau đớn khi cắt (kéo dài) chân sẽ là **|X - Y|** (Đây là ý nghĩa của giá trị tuyệt đối).

Bạn hãy xác định xem với hạt giống tối ưu thì tổng sự đau đớn của mọi dân cư trong **Vương Quốc 28Tech** sẽ là bao nhiêu để tất cả mọi người có cùng chiều cao với người hạt giống đó.



---

### Đầu vào

- Dòng 1 là **N** : số lượng dân cư
- Dòng 2 gồm **N** số là chiều cao của các cư dân

---

### Giới hạn

- Vương Quốc có không quá 1 triệu cư dân
- Chiều cao của cư dân thuộc đoạn [1, 999999999]

---

### Đầu ra

- In ra tổng số đầu đờn của mọi cư dân được coi là phương án tối ưu

---

### Ví dụ :

#### Input 01

Copy



Hello, Nguyễn Năng Anh.

3 9 10 1 8

### Output 01

15

Copy

Được phát triển bởi **28TECH.COM.VN**