In a parallel universe, there exist a kingdom that is known for its unique way of cooking. In this kingdom, there is a famous chef who is known for her delicious dishes. One day, the chef decided to create a new dish that consists of a sequence of ingredients. Each ingredient has a distinct weight, and the chef wants to choose a subarray of ingredients that have an increasing weight.

The chef wants to know the maximum possible sum of the weights of the ascending subarray she can choose. Can you help the chef by writing a function that returns the maximum possible sum of an ascending subarray in the weights of ingredients?

The array of ingredients is represented by the array **A**. ans % (10^9+7) → integer.

arr[] = {1, 2, 3, 4, 5}

arr[] = $\boxed{\{9, 2, 4, 2\}}$ → 9.

$n \to \frac{n(n+1)}{2}$

$\frac{4(4+1)}{2}$ ⇒ 10.

long.

max Ans

2

10^{14}

long.

$n = 10^5$

$O(n^2)$

$10^7 - 10^8$ iterations

1 <= |A| <= 10^5

1 <= A[i] <= 10^9 → entire array.

$O(n\log n)$, $O(n)$

10^9

10^5

$a_0 < a_1 < a_2 > a_3 < a_4 \quad a_5$

ans?

currsum = .

T.C → O(n), S.C → O(1).

dish

unique

greedy

```
ans = A[0];      currSum = A[0];
for (i=1; i<n; i++) {
    if (A[i] > A[i-1]) {
        currSum += A[i]
    }
    else {
        currSum = A[i];
    }
    ans = Max(ans, currSum);
}
```

Ques) Benjamin & xor.

	0	1	2	3
$A \rightarrow$	2	4	7	11

Queries → 3, 2, 1, 0.

Pairs ↓ 0 or 1 → 1

Pair

	3	2	1	0
2 →	0	0	1	0
4 →	0	1	0	0
7 →	0	1	1	1
11 →	1	0	1	1

2nd bit.

set: (4, 7) — 2

unset: (2, 11) — 2

$2 \times 2 = 4$ pairs.

3rd bit

set: 11 — 1

unset: (2, 4, 7) — 3

$1 \times 3 \Rightarrow 3$ pairs.

n

ith set

set (x)

unset $(n-x)$

T.C → O(N)
S.C → O(1).

how many pairs will have j th bit set :-

```
int count = 0;
for (i = 0; i < n; i++) {
    if ( checkbit (arr[i], j) {
        count++
    }
}
print ( count * (n - count) );
```

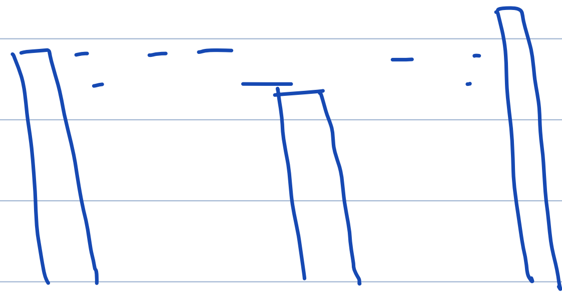
$N \simeq 10^4$
$Q = 100$

$\Rightarrow$ 10^6.

* Rain water trapping Problem. → O(1) soln.

↳ Const → memory.

→ Interview.



lmax, Rmax

(min(lmax, Rmax) - arr[i]) > 0.

$(x \,\&\, \sim(x-1))$

x = 12 => 1100

x-1 => 11 => 1011

~(x-1) => 0100

~(x-1)

x = 10 => 1010

x-1 = 9 => 1001

x-1

→ original

↳ sign

x = 12 => 1100

~(x-1) = 0100

flip back.

$x = 1010$

$(x-1) = 1001$

$-(x-1) \Rightarrow 0110$

flipped

become as it is sign

right

x :

$(x-1) =$ remain flipped

$-(x-1) =$ flipp remain

&

00000 1 0000

A.reverse()

for i in range (1, len (A))

sufMax[i] = max(sufMax[i-1], A[i-1])

suf

2	4	16	8	10	-1

new Array →	-1	10	8	16	4	2
sufmax →	0	0	0	0	0	0
	-1	-1	10	10	16	16

16 16 10 10 -1 0

1 3 2 4

→ 4 4 4 0

→ 4 3 2 3 1

Ans 0 $\cancel{0}$ $\cancel{0}$ 0

4 4 4

$$A + B = (A \wedge B) + 2 * (A \& B);$$

addition without carry

right carry

1 1 1 0

\+ 0 0 1 0

1 0 0 0 0

1 1 1 0

& 0 0 1 0

2 (0 0 1 0)

(0 1 0 0)

1 1 1 0

^ 0 0 1 0

1 1 0 0