Problem PA

Non Classical Problem Strikes Again

Usually, the easiest problem in a contest, especially in the practice session, is to find either the minimum integer, the maximum integer, or the sum of a given multiset of integers. We find that such problem is too classical and too boring.

Since this is an ICPC, we are going to pose you with a more challenging problem! Instead of giving you a multiset of integers, we will give you a multiset of real numbers. The i^{th} real number can be represented as a pair of integers (a_i, b_i) and the value of the number is $\frac{a_i}{b_i}$.

Find the minimum number, maximum number, and the sum of the given real numbers!

Input

The first line contains one integer: N $(1 \leq N \leq 100000)$ in a line denoting the size of the given multiset. The next N following lines, each contains two integers: a_i b_i $(1 \leq a_i, b_i \leq 2000000000)$ denoting the real numbers.

Output

The output contains three real numbers (each separated by a single space): the minimum number, the maximum number, and the sum of the given numbers, in a line. Your answer will be considered correct if the relative or absolute difference between your answer and judge's answer is not more than 10^{-6}.

Sample Input

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

Sample Output

```
0.20000000000 1.33333333333 3.28333333333
```

This page is intentionally left blank.

Problem PB
Maximum Subset

Let us define the value of a multiset of integers is the minimum difference between any two distinct elements. If a multiset contains two elements with the same value, then the two elements are considered different elements thus the value of the multiset is 0.

Given a multiset of integers A consisting of N elements, we want to find the value of the subset of A consisting of K elements which has the maximum value.

Input

The first line contains two integers: N K ($2 \leq K \leq N \leq 100000$) in a line denoting the number of elements of A and the number of elements of the subset of A we are looking for. The second line contains N integers: $A_1, A_2, \cdots, A_N$ ($0 \leq A_i \leq 1000000000$) representing the elements of set A.

Output

The output contains the value of the subset of A consisting of K elements which has the maximum value, in a line.

Sample Input #1

```
4 2
1 2 4 10
```

Sample Output #1

```
9
```

Explanation for the sample input/output #1

On the first sample, the optimal subset is $\{1, 10\}$. The value is $10 - 1 = 9$.

Sample Input #2

```
4 3
1 2 4 10
```

Sample Output #2

```
3
```

Explanation for the sample input/output #2

On the second sample, the optimal subset is $\{1, 4, 10\}$. The value is $4 - 1 = 3$.

Sample Input #3

```
4 4
1 2 4 10
```

Sample Output #3

```
1
```

Explanation for the sample input/output #3

On the third sample, the optimal subset is $\{1, 2, 4, 10\}$. The value is $2 - 1 = 1$.

Problem PC

Flip and Combos

A binary array is an array which each element can be either 0 or 1. Aleka has a binary array B or length N. The elements of B are indexed from 1 to N.

Aleka will play with her array. She will run Q queries one after another. Each query can be one of the following type :

- `FLIP` L R : Flip all bits of B from index L to R, inclusive. Flipping bit is changing the value of a bit from 0 to 1, or from 1 to 0.
- `COMBO` L R : Let B' be the subarray B of only containing bits which indexed between L to R, inclusive. Find the length of the longest contiguous subarray of B' such that all elements in that subarray have the same value.

All the queries should be executed as in the input order, and for every `COMBO`-type query, output the answer for that query.

Input

The first line contains two integers: N Q $(1 \leq N, Q \leq 100000)$ in a line denoting the length of the array and the number of queries. The second line contains a string of N characters (of '`0`' or '`1`') representing the binary array B. The i^{th} character in the string corresponds to the i^{th} element of the binary array ('`0`' represents 0, while '`1`' represents 1). The next Q lines each contains three integers T L R $(1 \leq T \leq 2; 1 \leq L \leq R \leq N)$ denoting the query. If $T = 1$, then this query is a `FLIP` query, otherwise this query is a `COMBO` query.

Output

For each `COMBO`-type query, print the answer of the query in the same order of the queries running order.

Sample Input

```
5 5
11000
1 2 3
2 1 5
1 4 5
2 1 5
2 1 4
```

Sample Output

```
2
3
2
```

Explanation for the sample input/output

After the first query, B becomes `10100`.

For the second query, $B' =$ `10100`. The longest subarray satisfying the `COMBO` constraint is `00`.

After the third query, B becomes `10111`.

For the fourth query, $B' =$ `10111`. The longest subarray satisfying the `COMBO` constraint is `111`.

For the fifth query, $B' =$ `1011`. The longest subarray satisfying the `COMBO` constraint is `11`.

Problem PD

Inverse Common Superstring

Given a set of string $S = \{S_1, S_2, \cdots, S_n\}$, a common superstring R of the set S is a string such that each string in S appears as a substring in R. For example, let S be `abb`, `baab`, `bbc`, then one possible common superstring R of S is `abbbaabbbc` which has the length of 10 characters. Notice that all strings in S appear as substring in R. To verify: `abb` appears in `[abb]baabbbc`, `baab` appears in `abb[baab]bbc`, and `bbc` appears in `abbbaab[bbc]`. The string `abbbaabbbc` is also a common superstring of S; you can verify it by yourself.

Among all possible common superstrings, usually the shortest common superstring is more attractive. It has many real-world application such as sparse matrix compression, DNA sequencing, and many others. In the example above, the shortest common superstring will be `baabbc` with the length of 6 characters. To verify: `aab` appears in `b[aab]bc`, `baab` in `[baab]bc`, and `bbc` in `baa[bbc]`.

Unfortunately, the problem of finding the shortest common superstring is known to be NP-hard, i.e. up to this moment, there is no known polynomial-time algorithm to solve the problem.

The inverse problem of finding the shortest common superstring will be: given a string R, find the set of string S such that R is the shortest common superstring of S. Of course this inverse problem is very easy and trivial! The set S can simply contains a single string which equals to R (notice that a string is also a substring to the string itself).

Now, you are going to solve a more challenging problem. Given a string R, find the lexicographically (alphabetically) smallest string which does NOT appear as a substring in R. To simplify the problem, a string is defined as a non-empty sequence of only lowercase alphabetical character (`a`-`z`). For example, let the string be `icpc`, then the lexicographically smallest string which does not appear as substring in R is `a`.

String $S = S_1S_2S_3\cdots$ is lexicographically smaller than string $T = T_1T_2T_3\cdots$ if one of the following is true:

- $|S| < |T|$ and $S_i = T_i$ for all $1 \le i \le |S|$, or
- There exists an index i where $S_i < T_i$ and $S_j = T_j$ for all $1 \le j < i$.

Input

The first line contains a string which length between 1 and 1000, inclusive. The given string contains only lowercase alphabetical character (`a`-`z`).

Output

The output contains the smallest lexicographical string which is NOT a substring of the input string, in a line. The output string should contain only lowercase alphabetical character.

Sample Input #1

```
icpc
```

Sample Output #1

```
a
```

Sample Input #2

```
jakarta
```

Sample Output #2

```
aa
```