Problem A

Tour de BINUS

BINUS University has N rooms to be used as lecture classes. These N rooms are numbered from 1 to N and arranged in a straight line, where the 1^{st} room is the left-most room, and the N^{th} room is the right-most room.

On one fine afternoon, Ayu and Budi, two BINUS' alumni, visit their lovely alma mater and walk around the campus. At that particular time, there might be a lecture in each class where A_i students attend the lecture in the i^{th} room.

Ayu starts her walk from the x^{th} room and walk in one direction to the first or last room (depends on the direction). While she walks, she counts the total number of students in all the rooms she passed. For example, if she starts from the 5^{th} room and going left, then she will count the total number of students in the 5^{th}, 4^{th}, 3^{rd}, 2^{nd}, and 1^{st} room. If she is going right, then she will count the total number of students from the 5^{th} room up to the N^{th} room. Budi also does a similar thing as Ayu, however, instead of counting the total number of students, Budi only counts the number of rooms which are not being used (empty rooms) at that time, i.e. when $A_i = 0$.

Given array A (representing the number of students in each room), Ayu's starting point (x_1), Ayu's direction, Budi's starting point (x_2), and Budi's direction, compute the total number of students in Ayu's walk and the total number of empty rooms in Budi's walk.

Input

Input begins with an integer N ($1 \leq N \leq 100$) representing the number of rooms. The second line contains N integers: A_i ($0 \leq A_i \leq 60$) representing the number of students in the i^{th} room. If $A_i = 0$, then it means the room is not being used (empty). The third line contains an integer x_1 ($1 \leq x_1 \leq N$) and a string d_1, representing Ayu's walk from the x_1^{th} room in d_1's direction. The fourth line contains an integer x_2 ($1 \leq x_2 \leq N$) and a string d_2, representing Budi's walk from the x_2^{th} room in d_2's direction. It is guaranteed that d_1 and d_2 will be either "`left`" or "`right`" (without quotes).

Output

Output in a line two integers (separated by a single space) representing the total number of students in Ayu's walk and the total number of empty rooms in Budi's walk, respectively.

Sample Input #1

```
7
10 0 43 21 0 15 0
4 right
5 left
```

Sample Output #1

```
36 2
```

Explanation for the sample input/output #1

Ayu starts from the 4^{th} room and going right, so she will pass the 4^{th}, 5^{th}, 6^{th}, and 7^{th} room. The number of students in each of these rooms are 21, 0, 15, 0, with a total of $21 + 0 + 15 + 0 = 36$.

Budi starts from the 5^{th} room and going left, so he will pass the 5^{th}, 4^{th}, 3^{rd}, 2^{nd}, and 1^{st} room. The rooms which are empty among these are the 5^{th} and 2^{nd} room (2 rooms).

Sample Input #2

```
5
60 0 20 60 60
1 left
1 right
```

Sample Output #2

```
60 1
```

Sample Input #3

```
10
0 0 0 45 15 0 20 60 30 0
3 left
6 left
```

Sample Output #3

```
0 4
```

Problem B
Linked List

A linked list is one basic linear data structure which is usually taught in any *Data Structure* course in college. Despite its practicality (or impracticality) for real-life applications, it is often used to demonstrate a way to store data in a non-continuous storage.

Three common operations in a linked list are insertion, deletion, and searching. In this problem, you are going to deal with the fourth operation which may find its place in real-life applications, i.e. sliding. Supposed you are given a linked list with N integers which are sequentially linked from 1 to N ($1 \rightarrow 2 \rightarrow 3 \rightarrow \cdots \rightarrow N$). A slide operation involves two integers a and b, i.e. $\texttt{slide}(a, b)$, and it moves integer a from its position to the (immediate) right of b's position.

For example, let $N = 5$, thus, the initial linked list is $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$. The operation $\texttt{slide}(4, 1)$ will change the linked list into $1 \rightarrow 4 \rightarrow 2 \rightarrow 3 \rightarrow 5$, i.e. 4 is moved from its position to the immediate right of 1's position. In this case, 4 is moved *two* positions to the *left*. If another operation, $\texttt{slide}(1, 5)$, is performed, then the linked list becomes $4 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 1$, i.e. 1 is moved from its position to the immediate right of 5's position. In this case, 1 is moved *four* positions to the *right*.

Given a linked list with N integers from 1 to N (all integers are linked from 1 to N sequentially) and Q $\texttt{slide}$ operations. For each $\texttt{slide}(a, b)$ operation, you should perform the sliding operation and output how far a is moved in the linked list. If a is moved to the left, then output it as the negative value (e.g., -2 in the above example); otherwise, output it as the non-negative value (e.g., 4 in the above example).

After all of the operations have been performed, you also need to output the final linked list.

Input

Input begins with two integers: N Q ($1 \leq N, Q \leq 100000$), representing the number of integers in the linked list and the number of operations, respectively. The next Q lines, each contains two integers: a b ($1 \leq a, b \leq N$; $a \neq b$), representing a $\texttt{slide}(a, b)$ operation to be performed.

Output

For each operation, output in a line how far the respected integer is moved. If the respected integer is moved to the left, output the negative value; otherwise, output the non-negative value. The last line of the output contains N integers (each separated by a single space) representing the linked list data after all operations have been performed.

Sample Input #1

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

Sample Output #1

```
-2
4
0
4 2 3 5 1
```

Explanation for the sample input/output #1

- initial: $1 \to 2 \to 3 \to 4 \to 5$.
- `slide`$(4, 1)$: $1 \to 4 \to 2 \to 3 \to 5$. The integer 4 is moved two positions to the left (output -2).
- `slide`$(1, 5)$: $4 \to 2 \to 3 \to 5 \to 1$. The integer 1 is moved four positions to the right (output 4).
- `slide`$(3, 2)$: $4 \to 2 \to 3 \to 5 \to 1$. The integer 3 is moved zero position (output 0).

Sample Input #2

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

Sample Output #2

```
1
2
0
1
3 1 2
```

Sample Input #3

```
10 2
2 7
10 7
```

Sample Output #3

```
5
-3
1 3 4 5 6 7 10 2 8 9
```

Problem C
Hierarchical Structure

Andi works in ACM (Association for Cool Magics). This company specializes in generating fun content for the internet, e.g., memes, jokes, games, etc. Although the working environment in ACM is fun and all employees are treated equal, a hierarchical structure of all the employees is still needed to keep the company run smoothly.

There are N employees in ACM, each with a unique ID ranging from 1 to N. Each employee has exactly one direct supervisor, except one person (usually referred as the "big boss" in ACM) who has no supervisor, thus, this hierarchical structure resembles a tree (in graph theory) rooted at the big boss, the highest level among all employees.

ACM has a rather interesting way to assign projects to its employees. When a project is said to be assigned to two employees a and b, then all the employees in the *path* between a and b in the tree hierarchical structure are **also** involved in this project. These employees along with a and b are known as the member of the project. It is common in any other company for the highest level member to lead the discussion in the group, but not within ACM. As ACM focuses on generating fun and creative ideas, the discussion should be led by someone who can understand any joke which pops up during the discussion, thus, they agree that the discussion is to be led by someone whose age is between l and r (inclusive). Andi is wondering, among the employees who might lead the discussion, what is the sum of the ages? It is also possible that no employee might be able to lead the discussion, in which the sum is 0.

Given the hierarchical structure of ACM and Q queries. Each query is in the form of a, b, l, and r representing two employees to whom a project is assigned and the age range. Your task for each query is to find the sum of the ages of the employees involved in the project whose age is between l and r (inclusive).

Input

Input begins with two integers: N Q ($1 \leq N \leq 100000$; $1 \leq Q \leq 100000$) representing the number of employees and the number of queries, respectively. The next line contains N integers: A_i ($1 \leq A_i \leq 10^9$) representing the age of the i^{th} employee. The next $N-1$ lines, each contains two integers: x y ($1 \leq x, y \leq N$; $x \neq y$) representing that x is the direct supervisor of y. It is guaranteed that each employee has at most one direct supervisor. The next Q lines, each contains four integers: a b l r ($1 \leq a, b \leq N$; $0 \leq l \leq r \leq 10^9$) representing the query in which you should answer.

Output

For each query, output in a line the sum of the ages of the employees involved in the project whose age is between l and r (inclusive).

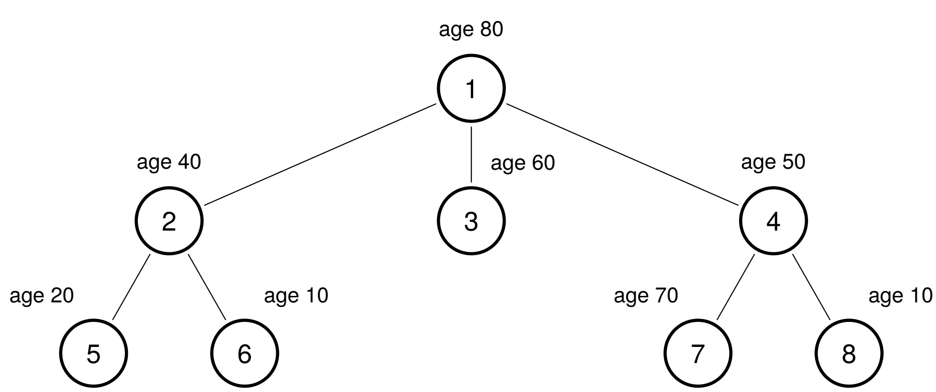
Sample Input

```
8 5
80 40 60 50 20 10 70 10
1 2
1 3
1 4
2 5
2 6
4 7
4 8
5 7 30 60
6 8 20 30
8 6 10 100
1 3 70 70
7 3 30 70
```

Sample Output

```
90
0
190
0
180
```

Explanation for the sample input/output



In the first query, the members of the project are $1, 2, 4, 5, 7$, and their ages are $(80, 40, 50, 20, 70)$. The sum of the ages of all employees whose age is between 30 and 60 is $40 + 50 = 90$.

In the second query, the members of the project are $1, 2, 4, 6, 8$, and their ages are $(80, 40, 50, 10, 10)$. There is no employee whose age is between 20 and 30, thus, the sum is 0.

In the third query, the members of the project are $1, 2, 4, 6, 8$, and their ages are $(80, 40, 50, 10, 10)$. The sum of the ages of all employees whose age is between 10 and 100 is $80 + 40 + 50 + 10 + 10 = 190$.

Problem D
Substring Permutation

Given two strings S and P, there are several ways to find whether P appears as a substring of S. The simplest one would be directly checking whether P is equal to any substring of S. As there can be $O(|S|)$ substring of S of length $|P|$, this approach has $O(|S| \times |P|)$ time complexity. There is also a more sophisticated way by using *knuth-morris-pratt* (KMP) algorithm to solve this problem in $O(|S| + |P|)$.

In this problem, you are challenged to a similar problem.

Given two strings S and P. Let $\Pi(S)$ be the set containing all strings which are permutations of S, and $\Pi(P)$ be the set containing all strings which are permutations of P. Determine whether there exists a string $p \in \Pi(P)$ and a string $s \in \Pi(S)$, such that p appears as a substring of s.

For example, let S = `guru` and P = `rug`. Then, $\Pi(S)$ = {`gruu`, `guru`, `guur`, `rguu`, `rugu`, `ruug`, `ugru`,`ugur`, `urgu`, `urug`, `uugr`, `uurg`}, and $\Pi(P)$ = {`gru`, `gur`, `rgu`, `rug`, `ugr`, `urg`}. Observe that the string `rug` in $\Pi(P)$ appears as a substring of the string `rugu` in $\Pi(S)$, i.e. `[rug]u`. In this example, you can also find another $\langle p, s\rangle$ which satisfies the requirement, e.g., $\langle \texttt{gru}, \texttt{gruu}\rangle$, $\langle \texttt{gru}, \texttt{ugru}\rangle$, $\langle \texttt{urg}, \texttt{uurg}\rangle$, $\langle \texttt{gur}, \texttt{guru}\rangle$, etc.

Input

Input contains two lines. The first line contains a string S $(1 \leq |S| \leq 100000)$. The second line contains a string P $(1 \leq |P| \leq |S| \leq 100000)$. Both S and P contains only lowercase alphabetical character (`a`-`z`).

Output

Output in a line "`YES`" (without quotes) if there exists a string $p \in \Pi(P)$ and a string $s \in \Pi(S)$, such that p appears as a substring of s; otherwise, output "`NO`" (without quotes).

Sample Input #1

```
guru
rug
```

Sample Output #1

```
YES
```

Sample Input #2

```
icpc
inc
```

Sample Output #2

```
NO
```

Sample Input #3

```
yesorno
sore
```

Sample Output #3

```
YES
```

Sample Input #4

```
indonesia
icpcasia
```

Sample Output #4

```
NO
```

Problem E

The Good, the Great, and the Superb

In this problem, we consider three types of integer sequences: the Good, the Great, and the Superb.

A sequence is considered as **Superb** if it contains at least 3 elements, and all the elements are of the same value. For example, $(1, 1, 1)$, $(4, 4, 4, 4)$, and $(9, 9, 9, 9, 9, 9)$ are Superb sequences.

A sequence is considered as **Great** if it contains at least 3 elements, and the difference between any successive elements is at most 1. By definition, all Superb sequences are also Great sequences. For example, $(1, 2, 3, 4)$, $(4, 4, 3, 2, 3)$, $(5, 5, 5)$ are Great sequences.

A sequence is considered as **Good** if it is constructed from concatenation of any Great or Superb sequences. By definition, all Great or Superb sequences are also Good sequences. For example,

- $(2, 2, 3, 6, 6, 6)$ from $(2, 2, 3)$ concatenated with $(6, 6, 6)$.
- $(5, 5, 5, 5)$ from $(5, 5, 5, 5)$.
- $(4, 3, 4, 7, 7, 8, 9, 2, 1, 0)$ from $(4, 3, 4)$ concatenated with $(7, 7, 8, 9)$ and $(2, 1, 0)$.

You are given a sequence of N integers S, your task is to find three integers: a, b, and c, which represent the minimum number of elements you need to modify such that S become a Good sequence, a Great sequence, and a Superb sequence, respectively.

Input

Input begins with an integer N ($3 \leq N \leq 100000$) representing the number of integers in S. The next line contains N integers: S_i ($0 \leq S_i \leq 9$) representing the given integer sequences.

Output

Output in a line three integers a b c (each separated by a single space) representing the minimum number of elements you need to modify from S such that it becomes a Good sequence, a Great sequence, and a Superb sequence, respectively.

Sample Input #1

```
4
1 2 3 4
```

Sample Output #1

```
0 0 3
```

Sample Input #2

```
7
2 8 0 2 3 7 4
```

Sample Output #2

```
2 3 5
```

Explanation for the sample input/output #2

Following are some example sequences (the underlined elements are modified):

2 8 0 2 3 7 4 : Original sequence.
2 $\underline{1}$ 0 2 3 $\underline{4}$ 4 : Good sequence (2 modifications).
2 $\underline{2}$ $\underline{2}$ 2 3 $\underline{4}$ 4 : Great sequence (3 modifications).
2 $\underline{2}$ $\underline{2}$ 2 $\underline{2}$ $\underline{2}$ $\underline{2}$: Superb sequence (5 modifications).

Sample Input #3

```
7
1 2 4 4 6 7 8
```

Sample Output #3

```
1 3 5
```

Explanation for the sample input/output #3

Following are some example sequences (the underlined elements are modified):

1 2 4 4 6 7 8 : Original sequence.
1 2 $\underline{3}$ 4 6 7 8 : Good sequence (1 modification).
$\underline{2}$ $\underline{3}$ 4 $\underline{5}$ 6 7 8 : Great sequence (3 modifications).
$\underline{4}$ $\underline{4}$ 4 4 $\underline{4}$ $\underline{4}$ $\underline{4}$: Superb sequence (5 modifications).

Problem F
KMP

KMP is a string algorithm that searches for occurrences of a string as substrings on another string. The name of the algorithm comes from the first character of the last names of the authors, namely Donald Knuth, James Hiram Morris, and Vaughan Pratt.

This problem has nothing to do with the algorithm, rather this problem is about the naming itself. There are N computer scientists in this world numbered from 1 to N. The i^{th} computer scientist has a name A_i. A name consists of one or more word, separated by a single space. A word consists of one or more characters. The first character is an uppercase alphabetical character (`A`-`Z`), while the rest of the characters are lowercase alphabetical characters (`a`-`z`).

There are Q queries, each consists of a string S of uppercase alphabetical characters. You are wondering whether an algorithm S can be authored by a subset of these N computer scientists. Let say Donald Knuth, James Hiram Morris, and Vaughan Pratt invent another algorithm. The first characters of any word in each computer scientist's name are $\{\texttt{D}, \texttt{K}\}$ in Donald Knuth, $\{\texttt{J}, \texttt{H}, \texttt{M}\}$ in James Hiram Morris, and $\{\texttt{V}, \texttt{P}\}$ in Vaughan Pratt. Then the algorithm can be named by taking exactly one of those first characters from each name (in the subset of the N computer scientists), e.g., `DJV`, `DHP`, `KHV`, `KMP`, `KJP`, etc. Note that the order does not matter, so algorithm names like `PKH` or `VHK` are also valid. However, `KKMP` or `LHO` are not valid in this example.

More formally, you are wondering whether there is a sequence of integers $X_1, X_2, \cdots, X_{|S|}$ such that:

- $1 \leq X_1, X_2, \cdots, X_{|S|} \leq N$
- $X_i \neq X_j$ for $i \neq j$
- One of the word in A_{X_i} starts with the character S_i

Input

Input begins with two integers: N Q ($1 \leq N \leq 50$; $1 \leq Q \leq 1000$) representing the number of computer scientists and the number of queries, respectively. The next line contains N strings: A_i representing the name of the i^{th} computer scientist. It is guaranteed that the sum of the length of all A_i is not more than 10^6. The next Q lines contains a string S representing the query which you should answer. It is guaranteed that the sum of the length of all S is not more than 10^6. It is also guaranteed that A_i and S satisfy the format given in the problem statement.

Output

For each query, output "`YES`" in a line if an algorithm of name S can be authored by a subset of the N computer scientists; otherwise, output "`NO`".

Sample Input #1

```
3 3
Donald Knuth
Vaughan Pratt
James Hiram Morris
KMP
DVJ
LHO
```

Sample Output #1

```
YES
YES
NO
```

Explanation for the sample input/output #1

The sample input/output illustrates the example given in the problem statement above.

Sample Input #2

```
3 3
Donald Knuth
Vaughan Pratt
James Hiram Morris
D
KP
KKMP
```

Sample Output #2

```
YES
YES
NO
```

Explanation for the sample input/output #2

Note that you can choose a subset of the N computer scientists.

Problem G
Discs

There are N discs on a two dimensional field, and all of them are centered on different lattice points. These discs have an interesting arrangement, i.e. for every two different discs A and B, then either A is inside B or B is inside A (they may touch each other). Note that the shape of each disc is a circle.

Now, forget the discs, and you will see the truth! As you step forward into a new dimension, an astonishing and utterly new world arises as only the N center points remain. The time has come for you to reconstruct the discs in a better way: Each disc mush be centered at one of those center points, and each center point must be the center of exactly one disc. Every two different discs must satisfy the previous arrangement (i.e. either A is inside B or B is inside A). Finally, the sum of all discs' radii is minimum. Note that a disc's radius can be 0, which is essentially a dot.

Given N different lattice points, find the minimum possible sum for such discs' radii. Your answer will be considered as correct if it has an absolute or relative error less than 10^{-6}.

Input

Input begins with an integer: N $(2 \leq N \leq 16)$ representing the number of center points. The next N lines, each contains two integers: x_i y_i $(0 \leq x_i, y_i \leq 10^9)$ representing a center point at coordinate (x_i, y_i). You may safely assume that all the given center points are unique.

Output

Output in a line the minimum possible sum of the discs' radii which satisfies the requirement given in the problem statement.

Sample Input #1

```
3
1 1
2 1
3 1
```

Sample Output #1

```
3.000000000
```

Explanation for the sample input/output #1

Let the radii of the discs centered at $(1, 1)$ be 0, at $(2, 1)$ be 1, and at $(3, 1)$ be 2.

Sample Input #2

```
3
5 5
0 0
10 0
```

Sample Output #2

```
21.213203436
```

Explanation for the sample input/output #2

Let the radii of the discs centered at $(5, 5)$ be $5\sqrt{2}$, at $(0, 0)$ be 0, and at $(10, 0)$ be $10\sqrt{2}$.

Sample Input #3

```
4
5 1
5 5
5 6
5 8
```

Sample Output #3

```
12.000000000
```

Explanation for the sample input/output #3

Let the radii of the discs centered at $(5, 1)$ be 7, at $(5, 5)$ be 3, at $(5, 6)$ be 2, and at $(5, 8)$ be 0.

Problem H
Plate Parity

Odd-even rationing is a common method to restrict resource consumption to half population on any given day. For example, a private vehicle is allowed to drive, park, or buy gasoline only on alternating days depends on whether its license plate is odd or even. This policy is enforced in Jakarta to reduce traffic jam during the Asian Games 2018, i.e. odd plate private vehicles are allowed to drive on some restricted roads and certain hours only on odd dates; similarly, even plate private vehicles are allowed only on even dates.

Despite its effectiveness, some people are unaware of the fact that zero (0) is an even number, and this might cause confusion. That's why we need to investigate a variant of odd-even rationing.

In this problem, whether a license plate is even or odd is determined by its **rightmost non-zero digit**: If it is odd, then the license plate is considered odd; otherwise, the license plate is considered even. For example:

- License plate of `701038` is even because `8` is even.
- License plate of `701803` is odd because `3` is odd.
- License plate of `801350` is odd because `5` is odd.
- License plate of `3800` is even because `8` is even.

Notice that the smallest license plate number is `1`, e.g. in our presidential plate, "RI 1".

Your task is to investigate how many integers between A and B (inclusive) are odd plates, and how many of them are even plates.

Input

Input contains two integers: A B $(1 \leq A \leq B \leq 10^{16})$ representing the interval $[A, B]$ in which plates you should investigate.

Output

Output in a line two integers: O E (separated by a single space) representing the number of odd plates and the number of even plates, respectively.

Sample Input #1

```
1 10
```

Sample Output #1

```
6 4
```

Explanation for the sample input/output #1

The odd plates are: `1`, `3`, `5`, `7`, `9`, and `10`.

The even plates are: `2`, `4`, `6`, and `8`.

Sample Input #2

```
296 311
```

Sample Output #2

```
10 6
```

Explanation for the sample input/output #2

The odd plates are: `297`, `299`, `300`, `301`, `303`, `305`, `307`, `309`, `310`, and `311`.

The even plates are: `296`, `298`, `302`, `304`, `306`, and `308`.

Sample Input #3

```
946 1073
```

Sample Output #3

```
72 56
```

Problem I
Expected Value of a Permutation

You have an array of N integers $A = [A_1, A_2, \cdots, A_N]$. Summing all integers in A is boring, so you decided to take it to the next level. You have a permutation P of 1 to N generated randomly. Each permutation from 1 to N has an equal probability to be chosen as P.

You also want to define arrays $X_0, X_1, X_2, ..., X_N$ and an integer Y as follows:

- $X_0 = A$
- X_i for $1 \leq i \leq N$ is defined as X_{i-1} but all integers whose indices are multiples of i are changed to 0.
- $Y = sum(X_1) + sum(X_2) + \cdots + sum(X_N)$, where $sum(X_i)$ is the sum of all integers in the array X_i.

For example, if $A = [4, 1, 2, 3, 4]$ and $P = [3, 2, 4, 1, 5]$, then:

- $X_0 = [4, 1, 2, 3, 4]$
- $X_1 = [4, 1, 0, 3, 4] \leftarrow P_1 = 3$, so, the 3^{rd} element of X_1 is changed to 0.
- $X_2 = [4, 0, 0, 0, 4] \leftarrow P_2 = 2$, so, the 2^{nd} and 4^{th} elements of X_2 are changed to 0.
- $X_3 = [4, 0, 0, 0, 4] \leftarrow P_3 = 4$, so, the 4^{th} element of X_3 is changed to 0.
- $X_4 = [0, 0, 0, 0, 0] \leftarrow P_4 = 1$, so, all elements of X_4 are changed to 0.
- $X_5 = [0, 0, 0, 0, 0] \leftarrow P_5 = 5$, so, the 5^{th} element of X_5 is changed to 0.

Therefore, $Y = 12 + 8 + 8 + 0 + 0 = 28$ in this case.

Since P is generated randomly, you are wondering the expected value of Y. Let $\frac{C}{D}$ be the expected value of Y where C and D are relatively prime non-negative integers. Print the value of $(C \times D^{-1}) \mod 1000000007$. In other words, you must print the value of the unique integer K $(0 \leq K < 1000000007)$ satisfying $C \equiv DK \pmod{1000000007}$.

Input

Input begins with an integer N $(1 \leq N \leq 100000)$ representing the number of integers in A. The second line contains N integers: A_i $(0 \leq A_i \leq 10^9)$ representing the array A.

Output

Output in a line the expected value of Y using the format specified in the problem description.

Sample Input

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

Sample Output

```
500000020
```

Explanation for the sample input/output

There are $5! = 120$ possible permutations for the value of P.

- When the value of $P = [3, 2, 4, 1, 5]$, the value of $Y = 28$ as described in the problem statement above.
- When the value of $P = [2, 1, 3, 4, 5]$, the value of $Y = 10$.
- ...

The sum of Y for all possible values of P is 1980. Therefore, the expected value of Y is $\frac{1980}{120} = \frac{33}{2}$. Since $33 \equiv 2 \times 500000020 \pmod{1000000007}$, you must print 500000020 for this sample case.

Problem J

A Study on Groups

You are given a bag of N integers, and you are to put these integers into M distinct groups. When you distribute the integers, you have to make sure that each integer belongs to exactly one group and the size of any two different groups differs by at most 1.

The cost of a group is equal to the lowest integer in that group, while the total cost of M groups simply equals to the summation of the cost of each group. Note that the cost of an empty group is zero.

Now, your task in this problem is to find a way to put N given integers (not necessarily unique) into M distinct groups such that the total cost of those groups is minimum. Also, find a way such that the total cost is maximum. Output only the total costs.

Input

Input begins with two integers: N M ($1 \leq N, M \leq 100000$) representing the number of given integers and the number of groups to be made, respectively. The next line contains N integers: A_i ($0 \leq A_i \leq 1000000$) representing the given integers.

Output

Output in a line two integers (separated by a single space) representing the minimum total cost and the maximum total cost, respectively.

Sample Input #1

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

Sample Output #1

```
6 11
```

Explanation for the sample input/output #1

The minimum total cost can be obtained from $\{(6,3,5),(4,1),(2,7)\}$ with a total cost of $3+1+2=6$.

The maximum total cost can be obtained from $\{(4,5),(7,6),(2,1,3)\}$ with a total cost of $4+6+1=11$.

Sample Input #2

```
5 1
10 15 17 4 8
```

Sample Output #2

```
4 4
```

Sample Input #3

```
3 2
470 105 222
```

Sample Output #3

```
327 575
```

Problem K

Living Subgraph

In this problem, you are given a simple undirected graph $G = (V, E)$ of $|V| = N$ nodes and $|E| = M$ edges. An **induced** subgraph of G is defined as a subset of G's nodes together with any edges whose endpoints are both in the subset.

Let $W \subseteq V$ and $G[W]$ be an induced subgraph of G with W as its nodes. $G[W]$ is a **living** subgraph if and only if: (1) It contains at least 3 nodes; (2) It is connected; (3) $G[W \setminus u]$ is connected for all $u \in W$. A graph is *connected* if and only if all nodes are reachable from any node in the graph. $W \setminus u$ denotes a set in which u is removed from W.

Your task is to find a set W with the minimum cardinality such that $G[W]$ is a living subgraph; output only the number of nodes. If G does not contain any living subgraph, then output -1.

Input

Input begins with two integers: N M $(1 \leq N \leq 20000; 0 \leq M \leq 20000)$ representing the number of nodes and edges in the given graph, respectively. The next M lines, each contains two integers: u v $(1 \leq u < v \leq N)$ representing an edge connecting node u and v. You may safely assume that each edge appears at most once in the given list.

Output

Output in a line an integer representing the minimum number of nodes in which the induced subgraph is a living subgraph. Output -1 if the given graph contains no living subgraph.

Sample Input #1

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

Sample Output #1

```
3
```

Explanation for the sample input/output #1

The induced subgraph $G[W]$ with the set of nodes $W = \{1, 2, 3\}$ is a living subgraph, and it has the minimum number of nodes.

Sample Input #2

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

Sample Output #2

```
-1
```

Explanation for the sample input/output #2

The given graph does not contain any living subgraph.

Sample Input #3

```
7 8
1 2
1 3
1 6
2 4
2 7
3 5
4 5
5 7
```

Sample Output #3

```
4
```

Explanation for the sample input/output #3

The induced subgraph $G[W]$ with the set of nodes $W = \{2, 4, 5, 7\}$ is a living subgraph, and it has the minimum number of nodes.

Problem L
Fair Tournament

Ayu was participating in Asian Championship for Mind Sports (ACMS) 2018, a competition where the (supposedly) best mind triumphs. This competition uses a knockout-tournament system with 2^N players. A match is played by two opposing players and its result cannot be a tie (i.e. there's always a winner). In the first round, the 1^{st} player will compete against the 2^{nd} player, the 3^{rd} player will compete against the 4^{th} player, the 5^{th} player will compete against the 6^{th} player, and so on. On the second round, the winner of the first match (1^{st} or 2^{nd} player) will compete against the winner of the second match (3^{rd} or 4^{th} player), and so on. At the last/final round, there will be only two players competing against each other, and there will be one champion.

Needless to say, the ACMS 2018's champion is Ayu!

Budi, who knows Ayu pretty well, doesn't believe this fact. He suspects the competition was rigged. Budi also knows all other contestants, and he's pretty sure these contestants will not cheat or fix any match. So, the only possible set-up is on the organizer side.

Budi has computed a matrix A of $2^N \times 2^N$. If A_{ij} is positive, then the i^{th} player will win against the j^{th} with an effort of A_{ij}, if they have a match. On the other hand, if A_{ij} is negative, then the i^{th} player will lose against the j^{th} player in a match; in this case, the absolute value means nothing. Of course, A_{ii} will be zero while all other elements are non-zero. It is guaranteed that exactly one of A_{ij} and A_{ji} will be positive for $i \neq j$. The total effort of a player to win the tournament is defined as the sum of all efforts he/she spent in all of his/her matches.

Budi believes that the organizer fixed the initial players/matches arrangement for the first round such that Ayu will win the tournament with the minimum total effort spent. Notice that for a tournament with M player, there are $M!$ arrangements can be made for the first round.

Given the matrix A, your task is to find the players arrangement such that Ayu (the 1^{st} player) wins the tournament with the minimum possible total effort. Output only the total effort. If it's not possible for Ayu to win the tournament, output -1 instead.

Input

Input begins with an integer N ($1 \leq N \leq 4$) in a line. The next 2^N lines, each contains 2^N integers: A_{ij} ($-1 \leq A_{ij} \leq 10^6$; $A_{ii} = 0$; $i \neq j \implies A_{ij} \neq 0$). It is guaranteed that exactly one of A_{ij} and A_{ji} will be positive for $i \neq j$. The entries for A_1 correspond to Ayu.

Output

Output in a line the minimum total effort needed by Ayu to win the tournament.

Sample Input #1

```
2
0 1 7 2
-1 0 4 2
-1 -1 0 3
-1 -1 -1 0
```

Sample Output #1

```
3
```

Explanation for the sample input/output #1

There are $4! = 24$ initial arrangements can be made for the first round. Some of them are:

- (1, 2, 3, 4). Ayu wins against the 2^{nd} player with an effort of 1. The 3^{rd} player wins against the 4^{th} player with an effort of 3. On the final round, Ayu wins against the 3^{rd} player with an effort of 7. The total effort spent by Ayu is $1 + 7 = 8$.
- (3, 1, 2, 4). Ayu wins against the 3^{rd} player with an effort of 7. The 2^{nd} player wins against the 4^{th} player with an effort of 2. On the final round, Ayu wins against the 2^{nd} player with an effort of 1. The total effort spent by Ayu is $7 + 1 = 8$.
- (2, 3, 1, 4). The 2^{nd} player wins against the 3^{rd} player with an effort of 4. Ayu wins against the 4^{th} player with an effort of 2. On the final round, Ayu wins against the 2^{nd} player with an effort of 1. The total effort spent by Ayu is $2 + 1 = 3$.
- ...

Among all possible permutations for the first round, the minimum possible for Ayu's total effort to win the tournament is 3.

Sample Input #2

```
1
0 -1
3 0
```

Sample Output #2

```
-1
```

Explanation for the sample input/output #2

There is no way Ayu can win the tournament.

Problem M
Moving Around

There are N points of interest in Jakarta, positioned west to east on a straight line. These points are numbered from 1 to N, where the 1^{st} point is the west-most point, and the N^{th} point is the east-most point.

There is a bus stop at each of the point. At the i^{th} point, you can buy a westbound bus ticket for W_i rupiahs. With this ticket, you can go to and visit any point to the west of point i. Similarly, you can also buy an eastbound bus ticket for E_i rupiahs. With this ticket, you can go to and visit any point to the east of point i.

You are at point S and want to visit the rest $N - 1$ points one by one for your trip. Each point must be visited exactly once. You can choose any point to be your last point. However, at the last point, you must buy either an eastbound or westbound bus ticket to go out from that point. In other words, if your last point is x, you must pay additional $\min(W_x, E_x)$ for your trip.

Determine the order of points visited to minimize the total cost for your trip. If there are more than one order of points with minimum total cost, you can choose any one of them.

Input

Input begins with two integers: N S ($2 \leq N \leq 100000$; $1 \leq S \leq N$) representing the number of points of interest and your initial location, respectively. The next N lines, each contains two integers: W_i E_i ($0 \leq W_i, E_i \leq 10^9$) representing the cost of the westbound and eastbound bus ticket from point i, respectively.

Output

Output in a line N integers (each separated by a single space) representing the order of points visited to minimize the total cost for your trip.

Sample Input #1

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

Sample Output #1

```
2 1 3 4
```

Explanation for the sample input/output #1

If you visit the points with the order described in the sample output, the following is the costs for your trip:

- You pay 3 rupiahs to go from point 2 to point 1.

- You pay 2 rupiahs to go from point 1 to point 3.
- You pay 10 rupiahs to go from point 3 to point 4.
- You pay $\min(4, 1) = 1$ rupiah to go out from point 4.

Therefore, you pay a total of 16 rupiahs. There is no order of points visited such that the total cost for your trip is less than 16.

Sample Input #2

```
5 5
4 2
1 10
9 8
3 7
6 5
```

Sample Output #2

```
5 4 2 1 3
```