Sorting

Alice and Bob are playing a game. Alice has N items numbered from 0 to $N-1$. The value of each item i is a non-negative integer $A[i]$.

Alice knows the values of all items, but Bob only knows the number of items N and the fact that the values are non-negative integers. Bob's goal is to sort the items in non-decreasing order of their values. That is, Bob needs to find an integer array P of length N satisfying the following:

- Each element of P is distinct.
- Every element of P is between 0 and $N-1$ inclusive.
- For every integer i satisfying $0 \le i \le N-2$, $A[P[i]] \le A[P[i+1]]$.

To achieve this, Bob can ask Alice up to $10\,000$ questions. Each question proceeds as follows:

1. Bob places $N-1$ threads, each connecting two distinct items. At this time, the whole set must be connected so that it is possible to travel between any two items along the threads.
2. Alice selects 0 or more items according to the following rules:
    - She cannot select two items directly connected by a thread.
    - While satisfying the condition above, the sum of the values of the selected items must be maximized.

    If there are multiple combinations of items satisfying the conditions, Alice arbitrarily chooses one of them and reveals it.
3. Alice reveals the selected items to Bob, and removes all threads.

You must help Bob win the game by asking as few questions as possible.

Implementation Details

You must implement the following function.

```
vector<int> sorting(int N)
```

- N: The number of items used in the game.
- This function must return an array P of item numbers listed in non-decreasing order of value. If the array P satisfying these conditions is not unique, any such array may be returned.
- This function is called exactly once.

This function can call the following function.

```
vector<int> ask_question(vector<array<int, 2>> threads)
```

- This means Alice and Bob proceed with one question process.
- `threads` : An array of size $N-1$ representing pairs of items directly connected by threads. For each element $[a, b]$ in `threads` , it means a thread is placed connecting item a and item b.
- When items are connected as specified in `threads` , it must be possible to travel between any two items along the threads.
- This function returns an integer array C of length N. For each item i, if Alice selects item i in this question, $C[i] = 1$, otherwise $C[i] = 0$. $(0 \leq i \leq N-1)$
  - If Alice has multiple item combinations satisfying the conditions in this question, the grader selects one of the valid arrays C and returns it. Note that multiple calls to `ask_question` with the same `threads` array in a single test case may return different arrays.
- This function can be called at most $10\,000$ times in a single test case.

Constraints

- $5 \leq N \leq 1\,000$
- $A[i]$ is a non-negative integer. Note that no upper bound for $A[i]$ is given. $(0 \leq i \leq N-1)$
- `ask_question` can be called at most $10\,000$ times in a single test case.
- In this problem, the grader is adaptive. This means the array A is not fixed and may change depending on the calls to `ask_question` . The grader guarantees that there is at least one array A consistent with the results of all previous `ask_question` calls whenever it provides an answer.
- For all test cases, the grader consumes time within 2 seconds and memory within 16 MiB regardless of the calls to `ask_question` .

Subtasks

No.	Points	Constraints
1	7	$N = 5$
2	8	$N \leq 100$
3	10	There is at most one i satisfying $0 \leq i < N$ and $A[i] > 0$.
4	30	For all integers i satisfying $0 \leq i < \frac{N}{2}$, $A[i] = 0$.
5	45	No additional constraints.

Scoring

Subtasks 1, 2

In subtasks 1 and 2, if the return value of the `sorting` function is correct, you get 100% of the subtask points.

Subtasks 3, 4, 5

The score the contestant gets in subtasks 3, 4, and 5 is determined as follows.

If the program terminates abnormally or the return value of the `sorting` function is incorrect, you get 0 points.

Otherwise, the score for that subtask is calculated as follows.

Let Q_{max} be the maximum number of times `ask_question` is called during a single execution of the `sorting` function in that subtask.

Define X based on Q_{max} as follows:

Condition	X
$10\,000 < Q_{max}$	0
$80 < Q_{max} \leq 10\,000$	$90 - 35 \log_{10} (\frac{Q_{max}}{80})$
$70 < Q_{max} \leq 80$	$170 - Q_{max}$
$Q_{max} \leq 70$	100

The contestant receives $X\%$ of the score for that subtask.

Examples

Consider the case where $N = 6$ and the array A representing the values of items Alice has is $[5, 3, 3, 0, 8, 1]$.

The grader initially calls the following function.

```
sorting(6)
```

The contestant's code may interact as follows.

```
ask_question([[0, 1], [1, 2], [2, 3], [3, 4], [4, 5]])
ask_question([[0, 1], [0, 2], [0, 3], [0, 4], [0, 5]])
```

In the case of the first call, if Alice chooses items $0, 2, 4$, the sum of values is $5 + 3 + 8 = 16$, which is maximum. Therefore, this call returns $[1, 0, 1, 0, 1, 0]$.

In the case of the second call, the sets of items Alice can choose while satisfying the conditions are $\{1, 2, 4, 5\}$ and $\{1, 2, 3, 4, 5\}$. Therefore, this call returns either $[0, 1, 1, 0, 1, 1]$ or $[0, 1, 1, 1, 1, 1]$.

Consider the following interaction.

```
ask_question([[0, 1], [2, 3], [4, 5]])
ask_question([[0, 1], [1, 2], [2, 3], [3, 0], [4, 5]])
```

In the case of the first call, it is not a valid call because the size of the `threads` array is not $N - 1$.

In the case of the second call, it is not a valid call because it is not possible to travel from item 0 to item 4 along the threads.

There are two integer arrays P satisfying the conditions:

- $[3, 5, 1, 2, 0, 4]$
- $[3, 5, 2, 1, 0, 4]$

Therefore, the function must return one of these two arrays.

Sample Grader

The input format of the sample grader is as follows.

- line 1: N
- line 2: $A[0]\ A[1] \ldots A[N-1]$

The provided sample grader is guaranteed to work normally only when the input $A[i]$ is an integer between 0 and 10^9 inclusive.

The sample grader prints the array returned by your code in the `sorting` function and the number of times `ask_question` was called in the following format.

- line 1: Assuming the `sorting` function returned an array P of length M, $P[0]\ P[1]\ \cdots P[M-1]$
- line 2: The number of times `ask_question` function was called, Q

Note that the sample grader may differ from the grader used for actual grading.