A. Kevin and Arithmetic

```
Input file:     standard input
Output file:    standard output
Time limit:     1 second
Memory limit:   256 megabytes
```

To train young Kevin's arithmetic skills, his mother devised the following problem.

Given n integers $a_1, a_2, \ldots, a_n$ and a sum s initialized to 0, Kevin performs the following operation for $i = 1, 2, \ldots, n$ in order:

- Add a_i to s. If the resulting s is even, Kevin earns a point and repeatedly divides s by 2 until it becomes odd.

Note that Kevin can earn at most one point per operation, regardless of how many divisions he does.

Since these divisions are considered more beneficial for Kevin's development, his mother wants to rearrange a so that the number of Kevin's total points is maximized. Determine the maximum number of points.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 500$). The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 100$) — the number of integers.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$).

Output

For each test case, output one integer — the maximum number of points.

Standard Input	Standard Output

5
1
1
2
1 2
3
2 4 6
4
1000000000 999999999 999999998 999999997
10
3 1 4 1 5 9 2 6 5 3
	0
	2
	1
	3
	8

Note

In the first test case, the only arrangement of a is $[1]$. s becomes 1. Kevin earns no points.

In the second test case, the only possible arrangement of a is $[2, 1]$. s becomes 1 and 1 successively. Kevin earns points in both operations.

In the third test case, one possible arrangement of a is $[2, 4, 6]$. s becomes 1, 5, and 11 successively. Kevin earns a point in the first operation.

B. Kevin and Geometry

```
Input file:     standard input
Output file:    standard output
Time limit:     1 second
Memory limit:   256 megabytes
```

Kevin has n sticks with length $a_1, a_2, \ldots, a_n$.

Kevin wants to select 4 sticks from these to form an isosceles trapezoid* with a positive area. Note that rectangles and squares are also considered isosceles trapezoids. Help Kevin find a solution. If no solution exists, output -1.

*An isosceles trapezoid is a convex quadrilateral with a line of symmetry bisecting one pair of opposite sides. In any isosceles trapezoid, two opposite sides (the bases) are parallel, and the two other sides (the legs) are of equal length.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains a single integer n ($4 \le n \le 2 \cdot 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^8$).

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output 4 integers — the lengths of sticks. If no solution exists, output -1.

If there are multiple solutions, print any of them.

Standard Input	Standard Output

7
4
5 5 5 10
4
10 5 10 5
4
1 2 3 4
4
1 1 1 3
6
4 2 1 5 7 1
6
10 200 30 300 30 100
4
100000000 100000000 1 2
	5 5 5 10
	5 5 10 10
	-1
	-1
	1 1 4 5
	-1
	100000000 100000000 1 2

Note

In the first test case, you can form an isosceles trapezoid with bases of length 5 and 10, and two legs of length 5.

In the second test case, you can form an isosceles trapezoid with two bases of length 5 and two legs of length 10. A rectangle is considered an isosceles trapezoid here.

In the third test case, there are no sticks with the same length. It's impossible to form an isosceles trapezoid.

In the fourth test case, it's impossible to form an isosceles trapezoid with a positive area.

,

C. Kevin and Puzzle

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

Kevin enjoys logic puzzles.

He played a game with n classmates who stand in a line. The i-th person from the left says that there are a_i liars to their left (not including themselves).

Each classmate is either honest or a liar, with the restriction that **no two liars can stand next to each other**. Honest classmates always say the truth. **Liars can say either the truth or lies**, meaning their statements are considered unreliable.

Kevin wants to determine the number of distinct possible game configurations modulo $998\,244\,353$. Two configurations are considered different if at least one classmate is honest in one configuration and a liar in the other.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains an integer n ($1 \le n \le 2 \cdot 10^5$) — the number of classmates.

The second line contains n integers $a_1, a_2, \ldots, a_n$ $(0 \le a_i \le n)$ — the number of liars to the left of the i-th person they claimed.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output one integer — the number of distinct game configurations modulo $998\,244\,353$.

Standard Input	Standard Output

8
3
0 1 2
5
0 0 0 0 0
5
0 0 1 1 2
5
0 1 2 3 4
5
0 0 1 1 1
5
5 1 5 2 5
1
0
4
	1
	2
	3
	0
	4
	1
	2
	0

2 3 1 1	

Note

We will use red to mark liars and blue to mark honest people.

In the first test case, the only possible way is $(0, 1, 2)$.

In the second test case, two possible ways are $(0, 0, 0, 0, 0)$ and $(0, 0, 0, 0, 0)$.

In the third test case, three possible ways are $(0, 0, 1, 1, 2)$, $(0, 0, 1, 1, 2)$, $(0, 0, 1, 1, 2)$.

,

D. Kevin and Numbers

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

Kevin wrote an integer sequence a of length n on the blackboard.

Kevin can perform the following operation any number of times:

- Select two integers x, y on the blackboard such that $|x - y| \leq 1$, erase them, and then write down an integer $x + y$ instead.

Kevin wants to know if it is possible to transform these integers into an integer sequence b of length m through some sequence of operations.

Two sequences a and b are considered the same if and only if their multisets are identical. In other words, for any number x, the number of times it appears in a must be equal to the number of times it appears in b.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 10^4$). The description of the test cases follows.

The first line of each test case contains two integers n and m ($1 \leq m \leq n \leq 2 \cdot 10^5$) — the length of a and the length of b.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$).

The third line contains m integers $b_1, b_2, \ldots, b_m$ ($1 \leq b_i \leq 10^9$).

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output "`Yes`" if it is possible to transform a into b, and "`No`" otherwise.

You can output the answer in any case (upper or lower). For example, the strings "`yEs`", "`yes`", "`Yes`", and "YES" will be recognized as positive responses.

Standard Input	Standard Output

9
2 1
4 5
9
2 1
3 6
9
4 2
1 2 2 2
3 4
4 2
1 1 3 3
	Yes
	No
	Yes
	Yes
	No
	Yes
	No
	No
	No

```
3 5
4 2
1 2 3 4
3 5
5 5
1 2 3 4 5
5 4 3 2 1
4 2
1 1 1 1
1 1
4 4
1 1 1 1
1 1 1 2
1 1
1
1000000000
```

Note

In the first test case, you can erase $4, 5$, and write down 9.

In the second test case, you can't erase $3, 6$.

In the third test case, one possible way could be:

- Erase $2, 2$, and write down 4. The remaining numbers are $1, 2, 4$ now.
- Erase $1, 2$, and write down 3. The remaining numbers are $3, 4$ now.

In the fourth test case, one possible way could be:

- Erase $1, 1$, and write down 2. The remaining numbers are $2, 3, 3$ now.
- Erase $2, 3$, and write down 5. The remaining numbers are $3, 5$ now.

,

E. Kevin and And

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   512 megabytes
```

Kevin has an integer sequence a of length n. At the same time, Kevin has m types of magic, where the i-th type of magic can be represented by an integer b_i.

Kevin can perform at most k (possibly zero) magic operations. In each operation, Kevin can do the following:

- Choose two indices i $(1 \le i \le n)$ and j $(1 \le j \le m)$, and then update a_i to $a_i \ \& \ b_j$. Here, $\&$ denotes the bitwise AND operation.

Find the minimum possible sum of all numbers in the sequence a after performing at most k operations.

Input

Each test contains multiple test cases. The first line contains the number of test cases t $(1 \le t \le 10^4)$. The description of the test cases follows.

The first line of each test case contains three integers n, m, k $(1 \le n \le 10^5, 1 \le m \le 10,$ $0 \le k \le nm)$ — the length of a, the number of types of magic, and the maximum number of operations.

The second line contains n integers $a_1, a_2, \ldots, a_n$ $(0 \le a_i < 2^{30})$.

The third line contains m integers $b_1, b_2, \ldots, b_m$ $(0 \le b_i < 2^{30})$.

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

For each test case, output one integer — the minimum possible sum of all numbers in the sequence a after performing at most k operations.

Standard Input	Standard Output

```
5
1 3 2
7
5 6 3
2 3 2
5 6
5 6 3
10 2 5
3 1 4 1 5 9 2 6 5 3
7 8
5 1 0
1073741823 1073741823 1073741823 1073741823
1073741823
1073741823
1 1 0
0
0
```

```
1
3
11
5368709115
0
```

Note

In the first test case, one possible way could be:

- Update a_1 to a_1 $\&$ b_1. The sequence will become $[5]$.
- Update a_1 to a_1 $\&$ b_3. The sequence will become $[1]$.

In the second test case, one possible way could be:

- Update a_1 to a_1 $\&$ b_3. The sequence will become $[1, 6]$.
- Update a_2 to a_2 $\&$ b_3. The sequence will become $[1, 2]$.

F1. Kevin and Binary String (Easy Version)

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

This is the easy version of the problem. The difference between the versions is that in this version, string t consists of only '0' and '1'. You can hack only if you solved all versions of this problem.

Kevin has a binary string s of length n. Kevin can perform the following operation:

- Choose two adjacent blocks of s and swap them.

A block is a maximal substring* of identical characters. Formally, denote $s[l, r]$ as the substring $s_l s_{l+1} \ldots s_r$. A block is $s[l, r]$ satisfying:

- $l = 1$ or $s_l \neq s_{l-1}$.
- $s_l = s_{l+1} = \ldots = s_r$.
- $r = n$ or $s_r \neq s_{r+1}$.

Adjacent blocks are two blocks $s[l_1, r_1]$ and $s[l_2, r_2]$ satisfying $r_1 + 1 = l_2$.

For example, if $s = 000\,\mathbf{11}\,\mathbf{00}\,111$, Kevin can choose the two blocks $s[4, 5]$ and $s[6, 7]$ and swap them, transforming s into $000\,\mathbf{00}\,\mathbf{11}\,111$.

Given a string t of length n consisting of '0', '1' and '?', Kevin wants to determine the minimum number of operations required to perform such that for any index i ($1 \le i \le n$), if $t_i \neq$ '?' then $s_i = t_i$. If it is impossible, output -1.

---

* A string a is a substring of a string b if a can be obtained from b by the deletion of several (possibly, zero or all) characters from the beginning and several (possibly, zero or all) characters from the end.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains a string s consisting of '0' and '1'.

The second line of each test case contains a string t consisting of **'0' and '1'**.

It is guaranteed that the lengths of s and t are the same.

It is guaranteed that the sum of the length of s over all test cases will not exceed $4 \cdot 10^5$.

Output

For each test case, output one integer — the minimum number of operations required. If it is impossible, output -1.

Standard Input	Standard Output
6	1
0001100111	3
0000011111	1
010101	-1

111000	-1
0101	-1
0110	
0101	
1010	
011001	
001110	
0	
1	

Note

In the first test case, the possible way is shown in the statement.

In the second test case, one possible way could be:

- Swap blocks $[2, 2]$, $[3, 3]$, s will become `001101`.
- Swap blocks $[3, 4]$, $[5, 5]$, s will become `000111`.
- Swap blocks $[1, 3]$, $[4, 6]$, s will become `111000`.

,

F2. Kevin and Binary String (Hard Version)

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

This is the hard version of the problem. The difference between the versions is that in this version, string t consists of '0', '1' and '?'. You can hack only if you solved all versions of this problem.

Kevin has a binary string s of length n. Kevin can perform the following operation:

- Choose two adjacent blocks of s and swap them.

A block is a maximal substring* of identical characters. Formally, denote $s[l, r]$ as the substring $s_l s_{l+1} \dots s_r$. A block is $s[l, r]$ satisfying:

- $l = 1$ or $s_l \neq s_{l-1}$.
- $s_l = s_{l+1} = \dots = s_r$.
- $r = n$ or $s_r \neq s_{r+1}$.

Adjacent blocks are two blocks $s[l_1, r_1]$ and $s[l_2, r_2]$ satisfying $r_1 + 1 = l_2$.

For example, if $s = 000\,\mathbf{11}\,\mathbf{00}\,111$, Kevin can choose the two blocks $s[4, 5]$ and $s[6, 7]$ and swap them, transforming s into $000\,\mathbf{00}\,\mathbf{11}\,111$.

Given a string t of length n consisting of '0', '1' and '?', Kevin wants to determine the minimum number of operations required to perform such that for any index i ($1 \leq i \leq n$), if $t_i \neq$ '?' then $s_i = t_i$. If it is impossible, output -1.

---

* A string a is a substring of a string b if a can be obtained from b by the deletion of several (possibly, zero or all) characters from the beginning and several (possibly, zero or all) characters from the end.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 10^4$). The description of the test cases follows.

The first line of each test case contains a string s consisting of '0' and '1'.

The second line of each test case contains a string t consisting of **'0', '1' and '?'**.

It is guaranteed that the lengths of s and t are the same.

It is guaranteed that the sum of the length of s over all test cases will not exceed $4 \cdot 10^5$.

Output

For each test case, output one integer — the minimum number of operations required. If it is impossible, output -1.

Standard Input	Standard Output
6 0001100111 0000011111 010101	1 3 1 -1

111000 0101 0110 0101 1010 011001 001110 0 1	-1 -1
6 010101 ?0?0?? 0101 ?0?0 11100101 ???????? 11100101 ???11?1? 1000100011 ?11?000?0? 10101 ?1011	2 -1 0 2 2 -1

Note

In the first test case of the first example, the possible way is shown in the statement.

In the second test case of the first example, one possible way could be:

- Swap blocks $[2, 2]$, $[3, 3]$, s will become $\mathtt{001101}$.
- Swap blocks $[3, 4]$, $[5, 5]$, s will become $\mathtt{000111}$.
- Swap blocks $[1, 3]$, $[4, 6]$, s will become $\mathtt{111000}$.

In the first test case of the second example, one possible way could be:

- Swap blocks $[1, 1]$, $[2, 2]$, s will become $\mathtt{100101}$.
- Swap blocks $[4, 4]$, $[5, 5]$, s will become $\mathtt{100011}$.

,

G. Kevin and Teams

```
Input file:     standard input
Output file:    standard output
Time limit:     4 seconds
Memory limit:   256 megabytes
```

This is an interactive problem.

Kevin has n classmates, numbered $1, 2, \ldots, n$. Any two of them may either be friends or not friends.

Kevin wants to select $2k$ classmates to form k teams, where each team contains exactly 2 people. Each person can belong to at most one team.

Let u_i and v_i be two people in the i-th team. To avoid potential conflicts during team formation, the team members must satisfy one of the following two conditions:

- For all i ($1 \le i \le k$), classmate u_i and v_i are friends.
- For all i ($1 \le i \le k$), classmate u_i and v_i are not friends.

Kevin wants to determine the maximum k such that, regardless of the friendship relationships among the n people, he can always find $2k$ people to form the teams. After that, he needs to form k teams. But asking whether two classmates are friends is awkward, so Kevin wants to achieve this while asking about the friendship status of no more than n pairs of classmates.

The interactor is **adaptive**. It means that the hidden relationship between classmates is not fixed before the interaction and will change during the interaction.

Interaction

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains one positive integer n ($2 \le n \le 10^5$) — the number of classmates.

First, you should output an integer k ($1 \le k \le \frac{n}{2}$) — the maximum number of teams you can form.

You can make queries in the following way — print one line of the form $?\ u\ v$ where $1 \le u \ne v \le n$. After that, read a single integer: 0 or 1 indicating whether they are friends. 1 means they are friends while 0 means not.

If you want to print the answer, output $!\ u_1\ v_1\ u_2\ v_2\ \ldots\ u_k\ v_k$. You should output **exactly** $2k$ distinct numbers. Then, the interaction continues with the next test case.

You can make at most n queries. Printing the answer does **not** count towards the number of queries made.

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

After printing each query do not forget to output the end of line and flush* the output. Otherwise, you will get `Idleness limit exceeded` verdict.

If, at any interaction step, you read -1 instead of valid data, your solution must exit immediately. This means that your solution will reveive `Wrong answer` because of an invalid query or any other mistake. Failing to exit can result in an arbitrary verdict because your solution will continue to read from a closed stream.

Hacks

The interactor for hacks is not adaptive. To make a hack, use the following format.

The first line contains the word "`manual`".

The second line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases.

The first line of each test case contains an integer n ($2 \le n \le 10^5$) — the number of classmates.

The second line of each test case contains a string s consisting of '0' and '1' of length $\frac{n(n-1)}{2}$ — indicating the relationship between $(1, 2)$, $(1, 3)$, $\ldots$, $(1, n)$, $(2, 3)$, $(2, 4)$, $\ldots$, $(2, n)$, $\ldots$, $(n-1, n)$. '1' means being friends and '0' means not being friends.

The following is the input of example.

```
manual
2
3
011
5
1011101011
```

The sum of n over all test cases must not exceed 10^5.

There is an additional constraint for hacks: The sum of $\frac{n(n-1)}{2}$ over all test cases must not exceed 10^7.

* To flush, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `sys.stdout.flush()` in Python;
- see the documentation for other languages.

Standard Input	Standard Output
`2`	
`3`	`1`
	`? 1 2`
`1`	`! 1 2`
`5`	`2`
	`? 1 2`
`1`	`? 3 4`
`0`	`? 3 5`
`1`	`? 1 3`
`0`	`? 2 4`
`0`	`! 1 2 3 5`

Note

In the first test case:

Kevin claims he can form 1 team regardless of the friendship relationships among the 3 people.

Kevin asks about the friendship relationship between people 1 and 2. The jury responds that they are friends.

Kevin answers that he can form a team with people 1 and 2.

In the second test case:

Kevin claims he can form 2 teams regardless of the friendship relationships among the 5 people.

Kevin asks about the friendship relationship between people $(1, 2)$, $(3, 4)$, $(3, 5)$, $(1, 3)$, $(2, 4)$. The jury responds with $1, 0, 1, 0, 0$.

Kevin answers that he can form two teams with people $(1, 2)$ and $(3, 5)$.

It is also possible to form two teams with people $(1, 3)$ and $(2, 4)$, since they are both not friends.

,

H1. Kevin and Stones (Easy Version)

```
Input file:     standard input
Output file:    standard output
Time limit:     4 seconds
Memory limit:   512 megabytes
```

This is the easy version of the problem. The difference between the versions is that in this version, you only need to determine whether a valid sequence of operations exists. You can hack only if you solved all versions of this problem.

Kevin has an undirected graph with n vertices and m edges. Initially, some vertices contain stones, which Kevin wants to move to new positions.

Kevin can perform the following operation:

- For each stone at u_i, select a neighboring vertex v_i. Simultaneously move each stone from u_i to its corresponding v_i.

At any time, each vertex can contain **at most one** stone.

Determine whether a valid sequence of operations exists that moves the stones from the initial state to the target state.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 1000$). The description of the test cases follows.

The first line of each test case contains two integers n and m ($1 \le n \le 2000, 0 \le m \le \min(\frac{n(n-1)}{2}, 10^4)$) — the number of vertices and edges in the graph.

The second line contains a binary string s consisting of '0' and '1'. The i-th bit of s indicates the number of stones on the i-th vertex in the initial state.

The third line contains a binary string t consisting of '0' and '1'. The i-th bit of t indicates the number of stones on the i-th vertex in the target state.

Each of the next m lines contains two integers u and v ($1 \le u, v \le n$) — an undirected edge between the u-th vertex and the v-th vertex.

It is guaranteed that the graph is simple. There are no self-loops and parallel edges in the graph.

It is guaranteed that the numbers of '1' in s and t are the same.

It is guaranteed that the sum of n over all test cases does not exceed 2000.

It is guaranteed that the sum of m over all test cases does not exceed 10^4.

Output

For each test case, on the first line, output "`Yes`" or "`No`" to indicate whether a valid sequence of operations exists.

You can output the answer in any case (upper or lower). For example, the strings "`yEs`", "`yes`", "`Yes`", and "YES" will be recognized as positive responses.

Standard Input	Standard Output
4	Yes
2 1	Yes
10	No
01	Yes
1 2	
11 11	
11011001010	
01101011100	
1 2	
2 3	
3 4	
4 5	
5 6	
6 7	
7 8	
8 9	
9 10	
10 11	
11 1	
3 2	
110	
101	
1 2	
2 3	
3 2	
111	
111	
1 2	
2 3	

,

H2. Kevin and Stones (Hard Version)

```
Input file:     standard input
Output file:    standard output
Time limit:     4 seconds
Memory limit:   512 megabytes
```

This is the hard version of the problem. The difference between the versions is that in this version, you need to output a valid sequence of operations if one exists. You can hack only if you solved all versions of this problem.

Kevin has an undirected graph with n vertices and m edges. Initially, some vertices contain stones, which Kevin wants to move to new positions.

Kevin can perform the following operation:

- For each stone at u_i, select a neighboring vertex v_i. Simultaneously move each stone from u_i to its corresponding v_i.

At any time, each vertex can contain **at most one** stone.

Determine whether a valid sequence of operations exists that moves the stones from the initial state to the target state. Output a valid sequence of operations with no more than $2n$ moves if one exists. It can be proven that if a valid sequence exists, a valid sequence with no more than $2n$ moves exists.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 1000$). The description of the test cases follows.

The first line of each test case contains two integers n and m ($1 \le n \le 2000, 0 \le m \le \min(\frac{n(n-1)}{2}, 10^4)$) — the number of vertices and edges in the graph.

The second line contains a binary string s consisting of '0' and '1'. The i-th bit of s indicates the number of stones on the i-th vertex in the initial state.

The third line contains a binary string t consisting of '0' and '1'. The i-th bit of t indicates the number of stones on the i-th vertex in the target state.

Each of the next m lines contains two integers u and v ($1 \le u, v \le n$) — an undirected edge between the u-th vertex and the v-th vertex.

It is guaranteed that the graph is simple. There are no self-loops and parallel edges in the graph.

It is guaranteed that the numbers of '1' in s and t are the same.

It is guaranteed that the sum of n over all test cases does not exceed 2000.

It is guaranteed that the sum of m over all test cases does not exceed 10^4.

Output

For each test case, on the first line, output "`Yes`" or "`No`" to indicate whether a valid sequence of operations exists.

You can output the answer in any case (upper or lower). For example, the strings "`yEs`", "`yes`", "`Yes`", and "YES" will be recognized as positive responses.

If a valid sequence of operations exists, output a single integer k ($0 \leq k \leq 2n$) on the second line, representing the number of operations. Suppose there are c stones in the initial state. The next $k+1$ lines should each contain distinct c integers, representing the positions of the stones before the operations and after each operation. These positions should satisfy the following:

- The positions of the stones in the first line match the initial state from the input, in any order.
- The positions of the stones in the last line match the target state from the input, in any order.
- For all i ($1 \leq i \leq k$) and j ($1 \leq j \leq c$), ensure that the j-th integer in the i-th line and the j-th integer in the $(i+1)$-th line correspond to adjacent vertices in the graph. In other words, the stone is moved from its previous position to the next.

If there are multiple solutions, print any of them.

Standard Input	Standard Output
`4`	`Yes`
`2 1`	`1`
`10`	`1`
`01`	`2`
`1 2`	`Yes`
`11 11`	`6`
`11011001010`	`1 2 4 5 8 10`
`01101011100`	`2 3 5 6 9 11`
`1 2`	`3 2 6 7 10 1`
`2 3`	`4 3 7 8 11 2`
`3 4`	`5 2 8 9 1 3`
`4 5`	`6 3 7 8 2 4`
`5 6`	`7 2 8 9 3 5`
`6 7`	`No`
`7 8`	`Yes`
`8 9`	`0`
`9 10`	`1 2 3`
`10 11`	
`11 1`	
`3 2`	
`110`	
`101`	
`1 2`	
`2 3`	
`3 2`	
`111`	
`111`	
`1 2`	
`2 3`	

,

I. Kevin and Nivek

```
Input file:     standard input
Output file:    standard output
Time limit:     4 seconds
Memory limit:   512 megabytes
```

Kevin and Nivek are competing for the title of "The Best Kevin". They aim to determine the winner through n matches.

The i-th match can be one of two types:

- **Type 1**: Kevin needs to spend a_i time to defeat Nivek and win the match. If Kevin doesn't spend a_i time on it, Nivek will win the match.
- **Type 2**: The outcome of this match depends on their historical records. If Kevin's number of wins is greater than or equal to Nivek's up to this match, then Kevin wins. Otherwise, Nivek wins.

Kevin wants to know the minimum amount of time he needs to spend to ensure he wins at least k matches.

Output the answers for $k = 0, 1, \ldots, n$.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 3 \cdot 10^5$) — the number of matches.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($-1 \le a_i \le 10^9$).

If $a_i = -1$, the i-th match is of **Type 2**. Otherwise, the i-th match is of **Type 1**, and a_i represents the amount of time Kevin needs to spend to win this match.

It is guaranteed that the sum of n over all test cases does not exceed $3 \cdot 10^5$.

Output

For each test case, output $n + 1$ integers. The i-th integer represents the minimum amount of time to win at least $i - 1$ matches.

Standard Input	Standard Output

3
5
-1 -1 -1 -1 -1
5
3 2 5 4 1
5
100 -1 -1 -1 1
	0 0 0 0 0 0
	0 1 3 6 10 15
	0 1 100 100 100 101

Note

In the first test case, all matches are of Type 2. Kevin can automatically win all matches.

In the second test case, all matches are of Type 1. Kevin can choose matches in increasing order of a_i.

In the third test case:

- If Kevin spends a_1 time on match 1, he can win matches $1, 2, 3, 4$.
- If Kevin spends a_5 time on match 5, he can win match 5.
- If Kevin spends a_1 time on match 1 and a_5 time on match 5, he can win all matches.