Codeforces Contest 23 Problem C. Oranges and Apples
time limit per test
2 seconds
memory limit per test
256 megabytes
input
standard input
output
standard output

In $2N - 1$ boxes there are apples and oranges. Your task is to choose N boxes so, that they will contain not less than half of all the apples and not less than half of all the oranges.

Input

The first input line contains one number T — amount of tests. The description of each test starts with a natural number N — amount of boxes. Each of the following $2N - 1$ lines contains numbers a_i and o_i — amount of apples and oranges in the i-th box ($0 \le a_i, o_i \le 10^9$). The sum of N in all the tests in the input doesn't exceed 10^5. All the input numbers are integer.

Output

For each test output two lines. In the first line output YES, if it's possible to choose N boxes, or NO otherwise. If the answer is positive output in the second line N numbers — indexes of the chosen boxes. Boxes are numbered from 1 in the input order. Otherwise leave the second line empty. Separate the numbers with one space.

Sample test(s)

Input

```
2
2
10 15
5 7
20 18
1
0 0
```

Output

```
YES
1 3
YES
1
```

SGU 138. Games of Chess

time limit per test: 0.50 sec.
memory limit per test: 4096 KB

N friends gathered in order to play chess, according to the following rules. In the first game, two of the **N** friends will play. In the second game, the winner of the first game will play against another friend (maybe even the same friend who lost the first game). In the third game, the winner of the second game will play against someone else and so on.. No game will end as a draw (tie). Given the number of games each of the **N** friends played, find a schedule for the games, so that the above rules are obeyed.

Input

The first line contains the number of friends **N** (**2<=N<=100**). The second line contains **N** integers, separated by blanks, representing the number of games each friend played. The first number represents the number of games played by the first friend, the second number represents the number of games played by the second friend and so on..

Output

The first line should contain the number of games played by all the friends (it will be an integer between **1** and **10 000**, for every test case). Let's suppose this number is **G**. Then, **G** lines follow, each of them containing two integers, describing the games. The first line contains the numbers of the two friends who played the first game. The friend printed first is considered to be the winner. Each of the next **G-1** lines contain the integers **a** and **b**, where **a<>b** and **a** or **b** is the winner of the previous game. The friend printed first on the line is considered to be the winner of the game.
It is guaranteed that for every test case there will be at least one possible scheduling of the games.

Sample Input

```
4
2 4 1 5
```

Sample Output

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

Task: BIR
Birthday

Official English Version

Day 2. Source file `bir.*` *Monday, 22–08–2005*

Available memory: 32 MB. Maximum running time: 2 s.

It is Byteman's birthday today. There are n children at his birthday party (including Byteman). The children are numbered from 1 to n. Byteman's parents have prepared a big round table and they have placed n chairs around the table. When the children arrive, they take seats. The child number 1 takes one of the seats. Then the child number 2 takes the seat on the left. Then the child number 3 takes the next seat on the left, and so on. Finally the child number n takes the last free seat, between the children number 1 and $n-1$.

Byteman's parents know the children very well and they know that some of the children will be noisy, if they sit too close to each other. Therefore the parents are going to reseat the children in a specific order. Such an order can be described by a permutation $p_1, p_2, \ldots, p_n$ ($p_1, p_2, \ldots, p_n$ are distinct integers from 1 to n) — child p_1 should sit between p_n and p_2, child p_i (for $i = 2, 3, \ldots, n-1$) should sit between p_{i-1} and p_{i+1}, and child p_n should sit between p_{n-1} and p_1. Please note, that child p_1 can sit on the left or on the right from child p_n.

To seat all the children in the given order, the parents must move each child around the table to the left or to the right some number of seats. For each child, they must decide how the child will move — that is, they must choose a direction of movement (left or right) and distance (number of seats). On the given signal, all the children stand up at once, move to the proper places and sit down.

The reseating procedure throws the birthday party into a mess. The mess is equal to the largest distance any child moves. The children can be reseated in many ways. The parents choose one with minimum mess. Help them to find such a way to reseat the children.

Task

Your task is to write a program that:

- reads from the standard input the number of the children and the permutation describing the desired order of the children,
- determines the minimum possible mess,
- writes the result to the standard output.

Input

The first line of standard input contains one integer n ($1 \leq n \leq 1\,000\,000$). The second line contains n integers $p_1, p_2, \ldots, p_n$, separated by single spaces. Numbers $p_1, p_2, \ldots, p_n$ form a permutation of the set $\{1, 2, \ldots, n\}$ describing the desired order of the children. Additionally, in 50% of the test cases, n will not exceed 1 000.

Output

The first and the only line of standard output should contain one integer: the minimum possible mess.

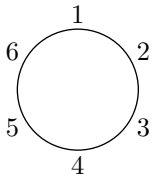
Example

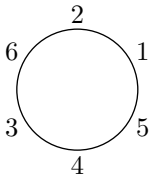
For the input data:

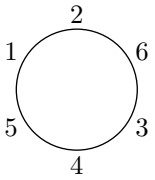
```
6
3 4 5 1 2 6
```

the correct result is:

```
2
```







The left figure shows the initial arrangement of the children. The middle figure shows the result of the following reseating: children number 1 and 2 move one place, children number 3 and 5 move two places, and children number 4 and 6 do not change places. The conditions of arrangement are fulfilled, since 3 sits between 6 and 4, 4 sits between 3 and 5, 5 sits between 4 and 1, 1 sits between 5 and 2, 2 sits between 1 and 6, and 6 sits between 2 and 3. There exists another possible final arrangement of children, depicted in the right figure. In both cases no child moves more than two seats.

Task: SKA
Rock Garden

Stage II. Day 1. Source file `ska.*` *07.02.2007*

Available memory: 32 MB.

Vicomte de Bajteaux is the owner of a renowned collection of boulders. Up to now, he has kept it in the cellars of his palace, but recently, he has decided to display the collection in his vast gardens.

The gardens have a shape of rectangle, whose sides are 1 000 000 000 units long and are parallel to east-west and north-south geographical directions. For each boulder, the vicomte has determined coordinates of the point, which he would like it to be placed in (the coordinates are simply distances to the southern and western side of the garden.), and gave them to his servants. Unfortunately he has forgot to tell them the order of the coordinates (i.e. for some of the boulders the first coordinate of a point is the so called "y coordinate", i.e. the ordinate, while for others the so called "x coordinate", i.e. the abscissa). The servants, unaware of this fact, have placed the boulders assuming customary coordinate ordering (as in standard Cartesian coordinates: the abscissa, commonly known as "x coordinate", first).

To protect his collection, the vicomte has decided to surround it with a fence. For aesthetic reasons the fence has to be a rectangle, with sides parallel to the sides of the garden. The garden layout has been planned, so that the total length of the fence be minimal (i.e. in the space of all coordinate orderings, the original ordering of vicomte requires the minimal length of the fence - we assume that the rectangle may have sides of zero length).

The servants have to move the boulders so that the length of the fence required is minimal lest their mistake become obvious. Each boulder may only be moved in a way that preserves the coordinate set: by interchanging its coordinates. As the boulders are heavy, the servants would like to minimize their effort, by minimizing the weight of the boulders to be moved.

Task

Write a programme which:

- reads the present positions of the boulders in the gardens and their respective weights,
- determines a sequence of moves, which minimizes the length of the fence required to protect the boulders and also minimizes the weight of the boulders to be moved,
- writes the outcome to the standard output.

Input

The first line of the standard input contains a single integer n ($2 \leq n \leq 1\,000\,000$), denoting the number of boulders in the collection The following n lines contain three integers x_i, y_i i m_i each ($0 \leq x_i, y_i \leq 1\,000\,000\,000$, $1 \leq m_i \leq 2\,000$), separated by single spaces, denoting the present coordinates and the weight of i-th boulder. No unordered pair of coordinates will appear in the input more than once.

Output

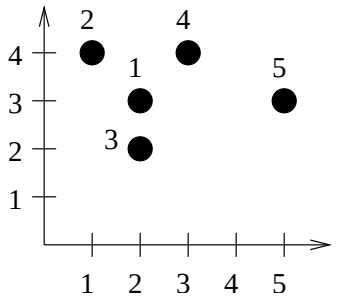
The first line of the standard output should contain two integers, separated by a single space - the minimal length of fence possible and the minimal weight of the boulders to be moved in order to obtain such a length.

The second line should contain a sequence of *n* zeros and/or ones - *i*-th element of the sequence should be a one if in the optimal solution the *i*-th boulder is to be moved and zero otherwise. Should more than one correct solutions exist, your programme is to write out any one of them.

Example

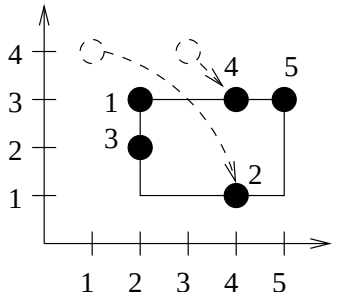
For the input data:

```
5
2 3 400
1 4 100
2 2 655
3 4 100
5 3 277
```



the correct result is:

```
10 200
01010
```



SGU 121. Bridges painting

time limit per test: 0.50 sec.
memory limit per test: 4096 KB

New Berland consists of N ($1 \le N \le 100$) islands, some of them are connected by bridges. There can be no more than one bridge between any pair of islands. Mr. President issued a law to paint all bridges. A bridge can be painted white or black. Any island must have at least one white bridge and at least one black (of course if an island has more than one bridge).

Input

There is *N* on the fisrt line of input. Next N lines contain a list of islands connected with given island. Every list is finished by *0*.

Output

If needed painting exists then write *N* lines. Write "1" and "2" in each line. Write "1" if bridge is painted white and "2" in other case. Write 0 at the end of any list. If needed painting does not exist then write "`No solution`".

Sample Input

```
6
2 3 0
1 3 0
1 2 5 0
5 0
4 6 3 0
5 0
```

Sample Output

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

122. The book

time limit per test: 0.50 sec.
memory limit per test: 4096 KB

There is a group of N (2<=N<=1000) people which are numbered 1 through N, and everyone of them has not less than [(N+1) / 2] friends. A man with number 1 has the book, which others want to read. Write the program which finds a way of transferring the book so that it will visit every man only once, passing from the friend to the friend, and, at last, has come back to the owner. ***Note***: if A is a friend of B then B is a friend of A.

Input

First line of input contains number N. Next N lines contain information about friendships. (i+1)-th line of input contains a list of friends of i-th man.

Output

If there is no solution then your program must output 'No solution'. Else your program must output exactly N+1 number: this sequence should begin and should come to end by number 1, any two neighbours in it should be friends, and any two elements in it, except for the first and last, should not repeat.

Sample Input

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

Sample Output

```
1 3 4 2 1
```