3-3. Lokahian Relics

Ancient Lokahian Civilization was the only known human civilization that used magic in real life. You can still find the traces of the Lokahian Civilization at the Alchemia Plain. There are N Lokahian ruins and $N-1$ roads left in the Alchemia Plain.

The ruins are numbered 0 through $N-1$, and the roads are numbered 0 through $N-2$. For each $0 \le i \le N-2$, road i connects ruin $S[i]$ and ruin $E[i]$, and it is possible to move in both directions of each road.

All ruins are connected by the roads. In other words, it is possible to move from any ruin to any other ruin by passing a finite sequence of roads.

Jaeun, who loves traveling, found an altar at a field while traveling through the traces of the Lokahian Civilization. Being amazed, he stepped his feet on the altar, which immediately made him cursed by the altar. Now, he can move between ruins only through roads because of the curse. Jaeun tries to collect relics from the ruins to break the curse.

There is exactly one relic in each ruin. The only relic in ruin i is numbered by i. Jaeun can take the relic in the ruin if he is in that ruin. Since there are N relics, Jaeun can take up to N relics.

For any r, if Jaeun collects r relics, his magical power becomes exactly r. Weirdly, there is an unknown fog in each road, so for each i ($0 \le i \le N-2$), Jaeun can pass road i (in any direction) if and only if his magical power is at least $M[i]$.

Jaeun can choose one ruin on his own and start traveling the ruins from there. Jaeun wants to collect more than $\lfloor N/2 \rfloor$ relics during his travel, so he should carefully decide where he should begin his trip. Fortunately, he can check whether he can take relic $R1$ and relic $R2$ simultaneously by traveling the same way, with his own choice of $R1$ and $R2$, multiple times.

Help Jaeun and write a program that does the prior check and chooses the ruin to begin his trip according to the results of the prior check.

Implementation details

You should implement the following function.

```
int FindBase(int N)
```

- N: the number of relics
- This function should return the number of the ruin Jaeun should begin his adventure to collect strictly more than $\lfloor N/2 \rfloor$ relics.
  - If there are multiple such ruins, this function should return any one of them.
  - If it is impossible to collect more than $\lfloor N/2 \rfloor$ relics, this function should return -1 instead.
- This function is called exactly once for each test case.

The function `FindBase` can call the following grader function:

```
int CollectRelics(int R1, int R2)
```

- $R1$: the number of the first relic. ($0 \leq R1 \leq N-1$).
- $R2$: the number of the second relic. ($0 \leq R2 \leq N-1$, $R1 \neq R2$).
- This function returns the number of the ruin that Jaeun should begin his adventure to collect both relic $R1$ and relic $R2$.
  - If there are multiple such ruins, this function returns the minimum number.
  - If it is impossible to collect both relics, this function returns -1 instead.
- This function can be called up to 600 times for each test case.

If some of the above conditions are not satisfied, your program is judged as `Wrong Answer`. Otherwise, your program is judged as `Accepted`.

Constraints

- $1 \leq N \leq 200$
- For each $0 \leq i \leq N-2$,
  - $0 \leq S[i] \leq N-1$
  - $0 \leq E[i] \leq N-1$
  - $S[i] \neq E[i]$
  - $1 \leq M[i] \leq N$
- All ruins are connected by roads.

In this problem, the grader is NOT adaptive. This means that S, E, and M are fixed at the beginning of the running of the grader and they do not depend on the queries asked by your solution.

Subtasks

1. (100 points) No additional constraints.

Assume your program is judged as `Accepted`, and makes C calls to `CollectRelics`. Then your score P for the test case is calculated as follows:

- If $C \leq 300$, $P = 100$.
- If $301 \leq C \leq 400$, $P = 77$.
- If $401 \leq C \leq 500$, $P = 51$.
- If $501 \leq C \leq 600$, $P = 39$.
- If $C > 600$, $P = 0$.

Your score is the minimum of the scores for the test cases.

Example

Suppose the structure of relics is as below:

```
0-1: 2
1-3: 1
1-2: 4
2-4: 1
```

The grader makes the following function call:

```
FindBase(5)
```

Let us consider the following calls to the function `CollectRelics`:

- `CollectRelics(0, 2)`: The function returns -1.
- `CollectRelics(1, 3)`: The function returns 1.
- `CollectRelics(2, 4)`: The function returns 2.

The answer is either 1 or 3.

Sample grader

You can download the sample grader package on the same page you downloaded the problem statement. (scroll down if you don't see the attachment)

If you use IDEs like Visual Studio, Eclipse or Code::Blocks, then import `lokahia.cpp`, `lokahia.h` and `grader.cpp` into one project and you will be able to compile all these files at once.

If you want to compile by yourself, refer to the compilation commands in the statement page.

You should submit only `lokahia.cpp`.

Input format

- line 1: N
- line $2 + i$ ($0 \leq i \leq M - 2$): $S[i]$ $E[i]$ $M[i]$

Output format

If your program is judged as `Accepted`, the sample grader prints `Correct` in the first line and q in the second line, with q the number of calls to `CollectRelics`.

If your program is judged as `Wrong Answer`, the sample grader prints the error message in the first line.