6.2 Проверка свойства дерева поиска

Проверка свойства дерева поиска
Проверить, является ли данное двоичное дерево деревом поиска.

> **Вход.** Двоичное дерево.
> **Выход.** Проверить, является ли оно корректным деревом поиска: верно ли, что для любой вершины дерева её ключ больше всех ключей в левом поддереве данной вершины и меньше всех ключей в правом поддереве.

Вы тестируете реализацию двоичного дерева поиска. У вас уже написан код, который ищет, добавляет и удаляет ключи, а также выводит внутреннее состояние структуры данных после каждой операции. Вам осталось проверить, что в каждый момент дерево остаётся корректным деревом поиска. Другими словами, вы хотите проверить, что для дерева корректно работает поиск, если ключ есть в дереве, то процедура поиска его обязательно найдёт, если ключа нет — то не найдёт.

Формат входа. Первая строка содержит число вершин n. Вершины дерева пронумерованы числами от 0 до $n-1$. Вершина 0 является корнем. Каждая из следующих n строк содержит информацию о вершинах $0, 1, \ldots, n-1$: i-я строка задаёт числа key_i, $left_i$ и $right_i$, где key_i — ключ вершины i, $left_i$ — индекс левого сына вершины i, а $right_i$ — индекс правого сына вершины i. Если у вершины i нет одного или обоих сыновей, соответствующее значение равно -1.

Формат выхода. Выведите «CORRECT», если дерево является корректным деревом поиска, и «INCORRECT» в противном случае.

Ограничения. $0 \le n \le 10^5$; $-2^{31} < key_i < 2^{31} - 1$; $-1 \le left_i, right_i \le n - 1$. Гарантируется, что вход задаёт корректное двоичное дерево: в частности, если $left_i \ne -1$ и $right_i \ne -1$, то $left_i \ne right_i$; никакая вершина не является сыном двух вершин; каждая вершина является потомком корня.

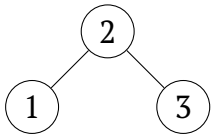
Пример.

Вход:

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

Выход:

```
CORRECT
```



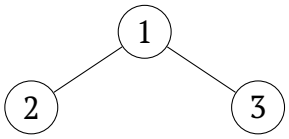
Пример.

Вход:

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

Выход:

```
INCORRECT
```



Пример.

Вход:

```
0
```

Выход:

```
CORRECT
```

Пустое дерево считается корректным.

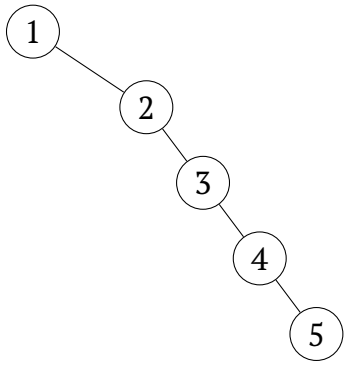
Пример.

Вход:

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

Выход:

```
CORRECT
```



Дерево не обязано быть сбалансированным.

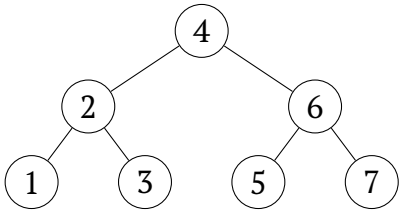
Пример.

Вход:

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

Выход:

```
CORRECT
```



Пример.

Вход:

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

Выход:

```
INCORRECT
```