Code Jam 2021 - Round 1C

Double or NOTing

Problem

You are given a starting non-negative integer $\mathbf{S}$ and an ending non-negative integer $\mathbf{E}$. Both $\mathbf{S}$ and $\mathbf{E}$ are given by their binary representation (that is, they are given written in base 2). Your goal is to transform $\mathbf{S}$ into $\mathbf{E}$. The following two operations are available to you:

- Double your current value.
- Take the bitwise NOT of your current value. The binary representation of your current value is taken without unnecessary leading zeroes, and any unnecessary leading zeroes produced by the operation are dropped. (The only necessary leading zero is the one in the representation of 0).

For example, by using the double operation, 6 becomes 12, 0 becomes 0, and 10 becomes 20. By using the NOT operation, 0 becomes 1, 1 becomes 0, $3 = 11_2$ becomes 0, $14 = 1110_2$ becomes 1, $10 = 1010_2$ becomes $5 = 101_2$, and $5 = 101_2$ becomes $2 = 10_2$. (X_2 means the integer whose binary representation is X).

You can use these operations as many times as you want in any order. For example, you can transform $\mathbf{S} = 10001_2$ to $\mathbf{E} = 111_2$ using the NOT operation first, then using the double operation twice, and then another NOT operation:

$$10001_2 \stackrel{\text{NOT}}{\Longrightarrow} 1110_2 \stackrel{\times 2}{\Longrightarrow} 11100_2 \stackrel{\times 2}{\Longrightarrow} 111000_2 \stackrel{\text{NOT}}{\Longrightarrow} 111_2.$$

Determine the smallest number of operations needed to complete the transformation, or say it is impossible to do so.

Input

The first line of the input gives the number of test cases, $\mathbf{T}$. $\mathbf{T}$ test cases follow. Each consists of a single line containing two strings $\mathbf{S}$ and $\mathbf{E}$, the binary representations of the starting and ending integers, respectively.

Output

For each test case, output one line containing `Case #`x`: `y, where x is the test case number (starting from 1) and y is `IMPOSSIBLE` if there is no way to transform $\mathbf{S}$ into $\mathbf{E}$ using the two operations. Otherwise, y is the smallest number of operations needed to transform $\mathbf{S}$ into $\mathbf{E}$.

Limits

Time limit: 10 seconds.
Memory limit: 1 GB.
$1 \leq \mathbf{T} \leq 100$.
Each character of $\mathbf{S}$ is either `0` or `1`.
The first digit of $\mathbf{S}$ can be `0` only if the length of $\mathbf{S}$ is 1.
Each character of $\mathbf{E}$ is either `0` or `1`.
The first digit of $\mathbf{E}$ can be `0` only if the length of $\mathbf{E}$ is 1.

Test Set 1 (Visible Verdict)

$1 \leq$ the length of $\mathbf{S} \leq 8$.
$1 \leq$ the length of $\mathbf{E} \leq 8$.

Test Set 2 (Hidden Verdict)

$1 \leq$ the length of $\mathbf{S} \leq 100$.
$1 \leq$ the length of $\mathbf{E} \leq 100$.

Sample

Sample Input

```
6
10001 111
1011 111
1010 1011
0 1
0 101
1101011 1101011
```

Sample Output

```
Case #1: 4
Case #2: 3
Case #3: 2
Case #4: 1
Case #5: IMPOSSIBLE
Case #6: 0
```

Sample Case #1 is the example shown in the main part of the statement.

These are possible optimal ways of solving Sample Cases #2, #3, and #4, respectively:

$$1011_2 \overset{\text{NOT}}{\Longrightarrow} 100_2 \overset{\times 2}{\Longrightarrow} 1000_2 \overset{\text{NOT}}{\Longrightarrow} 111_2,$$

$$1010_2 \overset{\times 2}{\Longrightarrow} 10100_2 \overset{\text{NOT}}{\Longrightarrow} 1011_2, \text{ and}$$

$$0_2 \overset{\text{NOT}}{\Longrightarrow} 1_2.$$

In Sample Case #5, it is not possible to get from 0_2 to 101_2 with any sequence of operations.

In Sample Case #6, we do not need to perform any operations because $\mathbf{S} = \mathbf{E}$.