4354 Déjà vu

An antique machine with $\binom{N}{3}$ switches capable of processing integers in the range $0..2^N - 1$ has just been discovered. Each switch is associated to a distinct integer in $0..2^N - 1$ with exactly three ones in its binary representation. By setting switches associated with number $X_0, X_1, \ldots, X_{M-1}$ to **on**, any integer Y passing through the machine will render a result of $Y \oplus X_0 \oplus X_1 \oplus \ldots \oplus X_{M-1}$ (here "$\oplus$" stands for bitwise-XOR).

Further inspections reveal that contrary to what we assumed in problem *Binary Integer*, some of the switches on the machine are damaged due to their old age. We are interested in whether a configuration transforming integer S into T still exists, and if so, the minimum number of switches that have to be set to **on** to make it possible.

WARNING: a naïve algorithm might not be sufficient to solve this problem.

Input

There are multiple test cases in the input file.

Each test case starts with two integers, N and M ($1 \leq N \leq 20$), representing the number of bits and the number of functioning switches, respectively. Two integers, S and T ($0 \leq S, T < 2^N$), come in the next line, followed by another M lines, the i-th one describing the value V_i associated to the i-th switch ($0 \leq V_i < 2^N$).

Two successive test cases are separated by a blank line. A case with $N = 0$ and $M = 0$ indicates the end of the input file, and should not be processed by your program.

Output

For each test case, please print a single integer, the minimum number of operations, or '`Impossible`' (without quotes) if no feasible sequence exists.

Sample Input

```
6 7
55 21
11
22
25
56
41
49
28

5 2
0 21
22
28

0 0
```

Sample Output

```
Case #1: 2
Case #2: Impossible
```