Code Jam 2014 - Round 1C

Analysis: Part Elf

In the first generation (i.e., 40 generations ago) there were only either pure Elf (1/1 Elf) or Human (0/1 Elf). If we enumerate all possible children for the next 3 generations we have:

```
1st gen:  0/1                                    1/1
2nd gen:  0/2               1/2                  2/2
3rd gen:  0/4       1/4     2/4       3/4        4/4
4th gen:  0/8  1/8  2/8  3/8  4/8  5/8  6/8  7/8  8/8
```

It is apparent that at any generation, the denominator is always a power of two. Thus, the answer for the **impossible** case is easy to check: first reduce the given fraction to its lowest terms and check whether the denominator is a power of two. A fraction can be reduced to its lowest terms by dividing both the numerator and denominator by their greatest common divisor.

Now, the remaining question is for a **P**/**Q** Elf, what is the minimum number of generations ago that there could have been a 1/1 Elf?

For a small input, where **Q** is at most 1000, we can generate the first 10 generations (with 2^{10} possible Elf combinations) to cover all possible small input. When we generate the child from the two parent we record their relationship. Then to answer the question, we simply do a breadth-first search (shortest path in unweighted graph) in the relationship graph from the **P**/**Q** Elf and stop whenever we encounter 1/1 Elf and report the length (shortest path length). This algorithm runs in $O(\mathbf{2^N} * \mathbf{2^{2N}})$ to generate the relationship graph of size 1001 x 1001 which is still feasible for a small input but not for the large input.

For the large input, another insight is needed. Letâ€™s do some examples to get the intuition. Suppose Vida is an 3/8 Elf, what are the possible parents? Letâ€™s enumerate them:

- (0/8 + 6/8) / 2 = 3/8
- (1/8 + 5/8) / 2 = 3/8
- (2/8 + 4/8) / 2 = 3/8
- (3/8 + 3/8) / 2 = 3/8

The possible parents of 3/8 are: 0/8, 1/8, 2/8, 3/8, 4/8, 5/8, 6/8 which form a â€œconsecutiveâ€ (for the lack of a better word) fraction from 0/8 to 6/8.

Formally speaking, a **P**/**Q** Elf could have a **Z**/**Q** parent where **Z** is ranged from *max*(0, **P** - (**Q**-**P**)) to *min*(**Q**, **P** * 2). Notice that the denominator **Q** does not change.

With this intuition, it is obvious that to get to the pure 1/1 Elf as fast as possible, we want to greedily generate a parent with numerator as large as possible. In other words, from an **P**/**Q** Elf we would like to generate **Z**/**Q** parent where **Z** is maximized. We continue the process for **Z**/**Q**, picking the parent with the greatest numerator and so on until we get to a 1/1 Elf. This greedy algorithm runs in O(log(**Q**)). Below is a sample implementation in Python 3:

```
from fractions import gcd

def is_power_of_two(x):
  return x & (x - 1) == 0

def min_generation(P, Q):
  g = gcd(P, Q)
```

```
  P = P // g
  Q = Q // g

  if not(is_power_of_two(Q)):
    return "impossible"

  gen = 0
  while P < Q:
    P = P * 2
    gen += 1
  return gen


for tc in range(int(input())):
  print("Case #%d: %s" % (tc+1, \
    min_generation(*map(int, input().split('/')))))
```