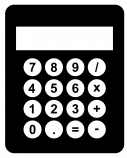
2 Primitive Calculator

Problem Introduction



You are given a primitive calculator that can perform the following three operations with the current number x: multiply x by 2, multiply x by 3, or add 1 to x. Your goal is given a positive integer n, find the minimum number of operations needed to obtain the number n starting from the number 1.

Problem Description

Task. Given an integer n, compute the minimum number of operations needed to obtain the number n starting from the number 1.

Input Format. The input consists of a single integer $1 \leq n \leq 10^6$.

Output Format. In the first line, output the minimum number k of operations needed to get n from 1. In the second line output a sequence of intermediate numbers. That is, the second line should contain positive integers $a_0, a_2, \ldots, a_{k-1}$ such that $a_0 = 1$, $a_{k-1} = n$ and for all $0 \leq i < k - 1$, a_{i+1} is equal to either $a_i + 1$, $2a_i$, or $3a_i$. If there are many such sequences, output any one of them.

Sample 1.

Input:

```
1
```

Output:

```
0
1
```

Sample 2.

Input:

```
5
```

Output:

```
3
1 2 4 5
```

Here, we first multiply 1 by 2 two times and then add 1. Another possibility is to first multiply by 3 and then add 1 two times. Hence "1 3 4 5" is also a valid output in this case.

Sample 3.

Input:

```
96234
```

Output:

```
14
1 3 9 10 11 22 66 198 594 1782 5346 16038 16039 32078 96234
```

Again, another valid output in this case is "1 3 9 10 11 33 99 297 891 2673 8019 16038 16039 48117 96234".

Starter Files

Going from 1 to n is the same as going from n to 1, each time either dividing the current number by 2 or 3 or subtracting 1 from it. Since we would like to go from n to 1 as fast as possible it is natural to repeatedly reduce n as much as possible. That is, at each step we replace n by $\min\{n/3, n/2, n-1\}$ (the terms $n/3$ and $n/2$ are used only when n is divisible by 3 and 2, respectively). We do this until we reach 1. This gives rise to the following algorithm and it is implemented in the starter files:

```
GreedyCalculator(n):
numOperations ← 0
while n > 1:
  numOperations ← numOperations + 1
  if n mod 3 = 0:
    n ← n/3
  else if n mod 2 = 0:
    n ← n/2
  else:
    n ← n − 1
return numOperations
```

This seemingly correct algorithm is in fact incorrect. You may want to submit one of the starter files to ensure this. Hence in this case moving from n to $\min\{n/3, n/2, n-1\}$ is not *safe*.

Need Help?

Ask a question or see the questions asked by other learners at this forum thread.