A *binary gap* within a positive integer N is any maximal sequence of consecutive zeros that is surrounded by ones at both ends in the binary representation of N.

For example, number 9 has binary representation 1001 and contains a binary gap of length 2. The number 529 has binary representation 1000010001 and contains two binary gaps: one of length 4 and one of length 3. The number 20 has binary representation 10100 and contains one binary gap of length 1. The number 15 has binary representation 1111 and has no binary gaps. The number 32 has binary representation 100000 and has no binary gaps.

Write a function:

class Solution { public int solution(int N); }

that, given a positive integer N, returns the length of its longest binary gap. The function should return 0 if N doesn't contain a binary gap.

For example, given N = 1041 the function should return 5, because N has binary representation 10000010001 and so its longest binary gap is of length 5. Given N = 32 the function should return 0, because N has binary representation '100000' and thus no binary gaps.

Write an **efficient** algorithm for the following assumptions:

* N is an integer within the range [1..2,147,483,647].

def solution(N):

bin1 = str( bin(N) ) [2:] # to remove “0b” part

bin\_group = False

high\_zero = 0

zero\_counter = 0

for i in bin1:

if i == "1":

if high\_zero < zero\_counter:

high\_zero = zero\_counter

bin\_group = True

zero\_counter = 0

elif bin\_group:

zero\_counter += 1

return high\_zero

deger = int(1041)

car\_1 = solution(deger)

print(car\_1)

deger2 = int(5)

car\_2 = solution(deger2)

print(car\_2)