Problem #284: Peeking Iterator (Medium)

https://leetcode.com/problems/peeking-iterator/

My Solution:

Runtime faster than 88.51%

# Below is the interface for Iterator, which is already defined for you.

#

# class Iterator:

# def \_\_init\_\_(self, nums):

# """

# Initializes an iterator object to the beginning of a list.

# :type nums: List[int]

# """

#

# def hasNext(self):

# """

# Returns true if the iteration has more elements.

# :rtype: bool

# """

#

# def next(self):

# """

# Returns the next element in the iteration.

# :rtype: int

# """

class PeekingIterator:

def \_\_init\_\_(self, iterator):

"""

Initialize your data structure here.

:type iterator: Iterator

"""

self.iterator = iterator

if self.iterator.hasNext():

self.next\_ele = self.iterator.next()

else:

self.next\_ele = None

def peek(self):

"""

Returns the next element in the iteration without advancing the iterator.

:rtype: int

"""

return self.next\_ele

def next(self):

"""

:rtype: int

"""

temp = self.next\_ele

if self.iterator.hasNext():

self.next\_ele = self.iterator.next()

else:

self.next\_ele = None

return temp

def hasNext(self):

"""

:rtype: bool

"""

return self.next\_ele != None

# Your PeekingIterator object will be instantiated and called as such:

# iter = PeekingIterator(Iterator(nums))

# while iter.hasNext():

# val = iter.peek() # Get the next element but not advance the iterator.

# iter.next() # Should return the same value as [val].