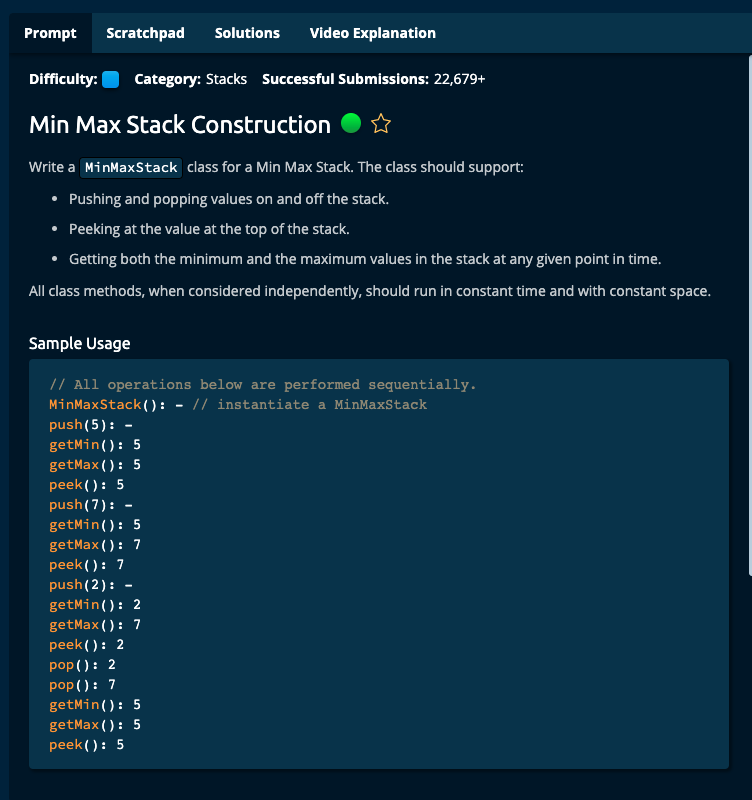
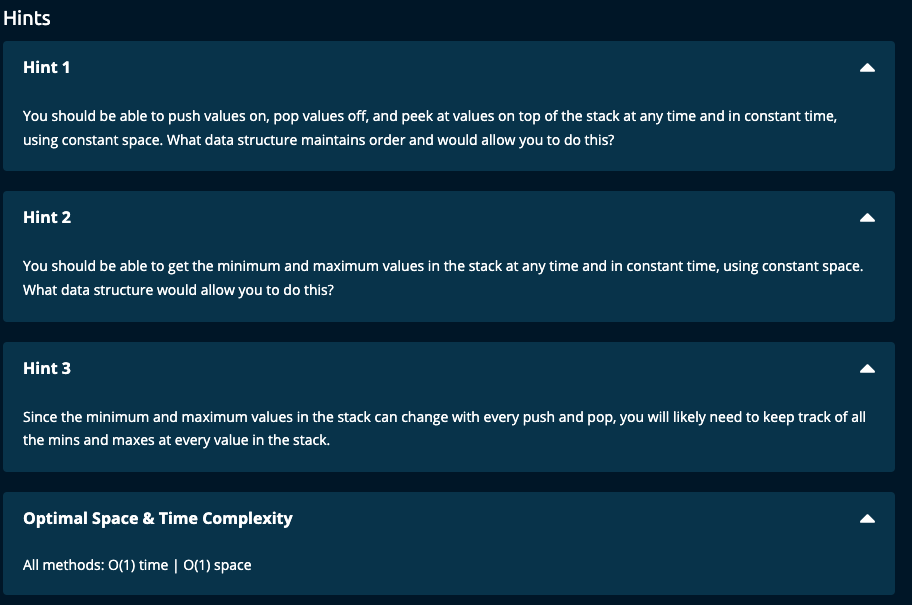
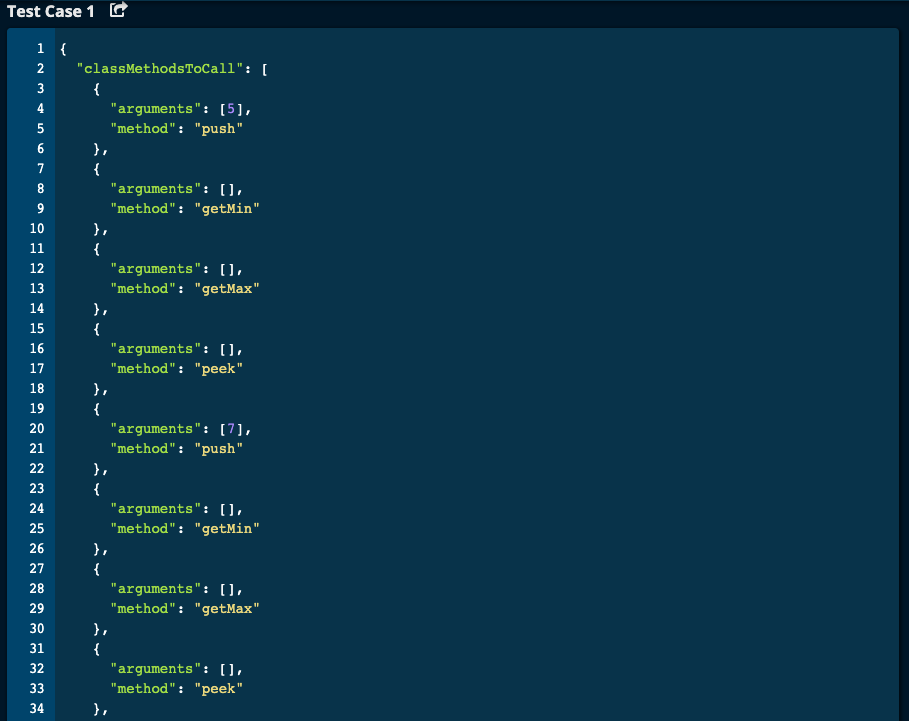
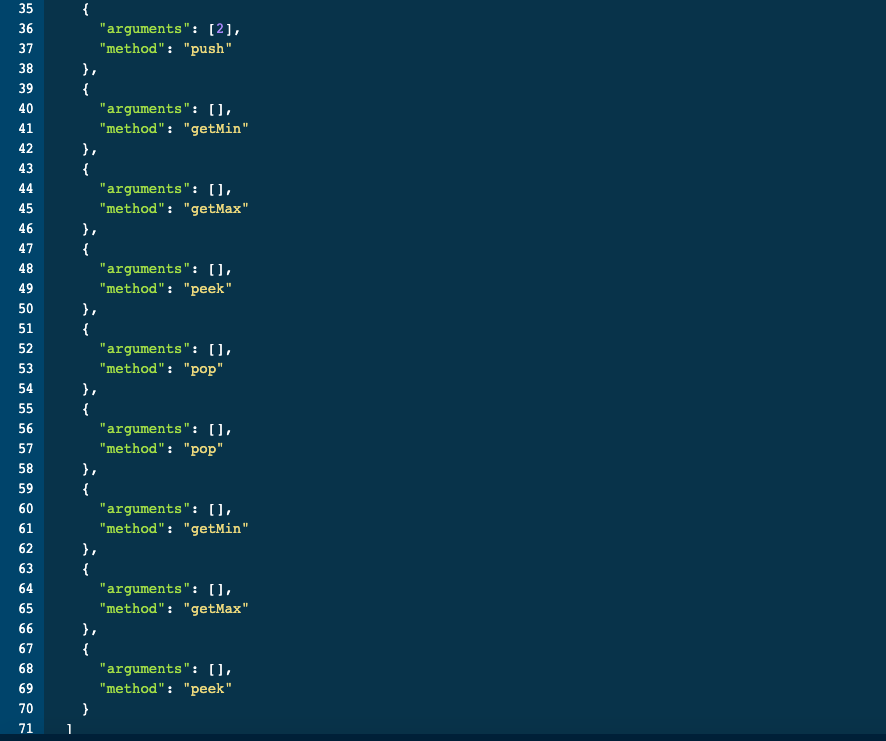
Min Max Stack Construction (Medium)









}

My Solution: My getMin() and getMax() are O(n) time. See Algoexpert Solution for O(1) time

# Feel free to add new properties and methods to the class.

class MinMaxStack:

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

self.stack = []

self.size = 0

def peek(self):

return self.stack[-1]

def pop(self):

if self.size > 0:

element = self.stack[-1]

self.stack = self.stack[ : -1]

self.size -= 1

return element

return None

def push(self, number):

self.stack.append(number)

self.size += 1

def getMin(self):

minEle = self.stack[0]

for i in range(1, self.size):

if self.stack[i] < minEle:

minEle = self.stack[i]

return minEle

def getMax(self):

maxEle = self.stack[0]

for i in range(1, self.size):

if self.stack[i] > maxEle:

maxEle = self.stack[i]

return maxEle

Algoexpert Solution:

# Algo expert Solution -- all methods are O(1) Time | O(1) Space

class MinMaxStack:

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

self.stack = []

self.minMaxStack = []

def peek(self):

return self.stack[len(self.stack) - 1]

def pop(self):

self.minMaxStack.pop()

return self.stack.pop()

def push(self, number):

newMinMax = {"min": number, "max": number}

if len(self.minMaxStack):

lastMinMax = self.minMaxStack[len(self.minMaxStack) - 1]

newMinMax["min"] = min(number, lastMinMax["min"])

newMinMax["max"] = max(number, lastMinMax["max"])

self.minMaxStack.append(newMinMax)

self.stack.append(number)

def getMin(self):

return self.minMaxStack[len(self.minMaxStack) - 1]["min"]

def getMax(self):

return self.minMaxStack[len(self.minMaxStack) - 1]["max"]

======================