LINKED LISTS -1

Question: Given a non-empty, singly linked list with head node head, return a middle node of linked list. In single pass.

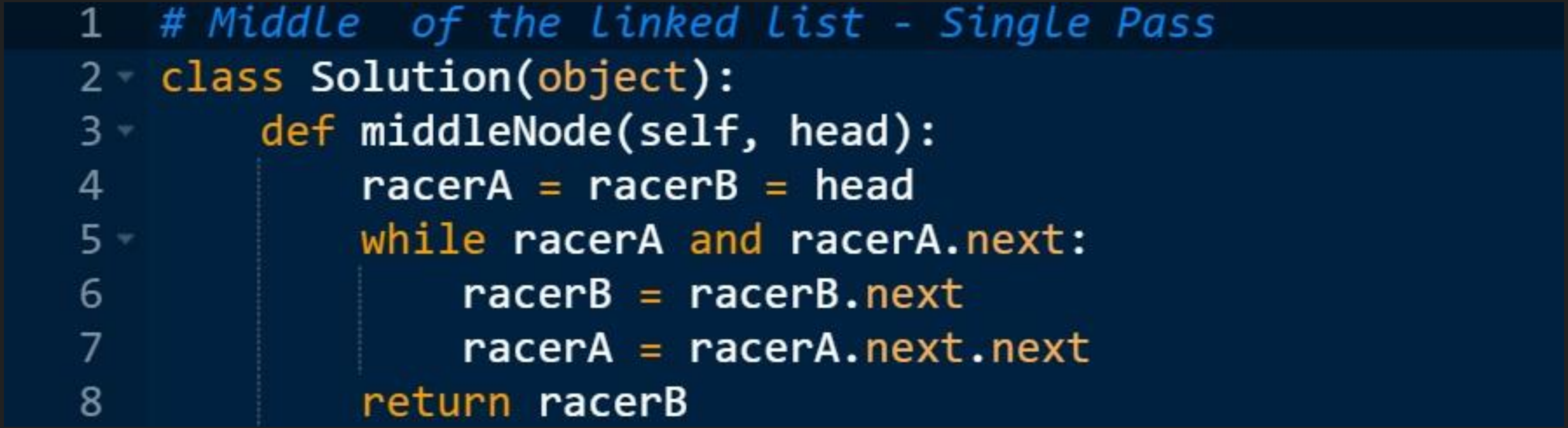
YOU HAVE 15 MINUTES

LOOK UP :

Link: https://leetcode.com/problems/middle-of-the-linked-list/

INTUITION

Two racers start from the same start position on a race track on length 2 miles. Racer A runs at 2 miles per hour. Racer B runs at 1 mile per hour. After 1 hour, where will Racer A ?



```python
# Middle  of the linked list - Single Pass
class Solution(object):
    def middleNode(self, head):
        racerA = racerB = head
        while racerA and racerA.next:
            racerB = racerB.next
            racerA = racerA.next.next
        return racerB
```

TIME COMPLEXITY - O(N), SPACE COMPLEXITY - O(1)

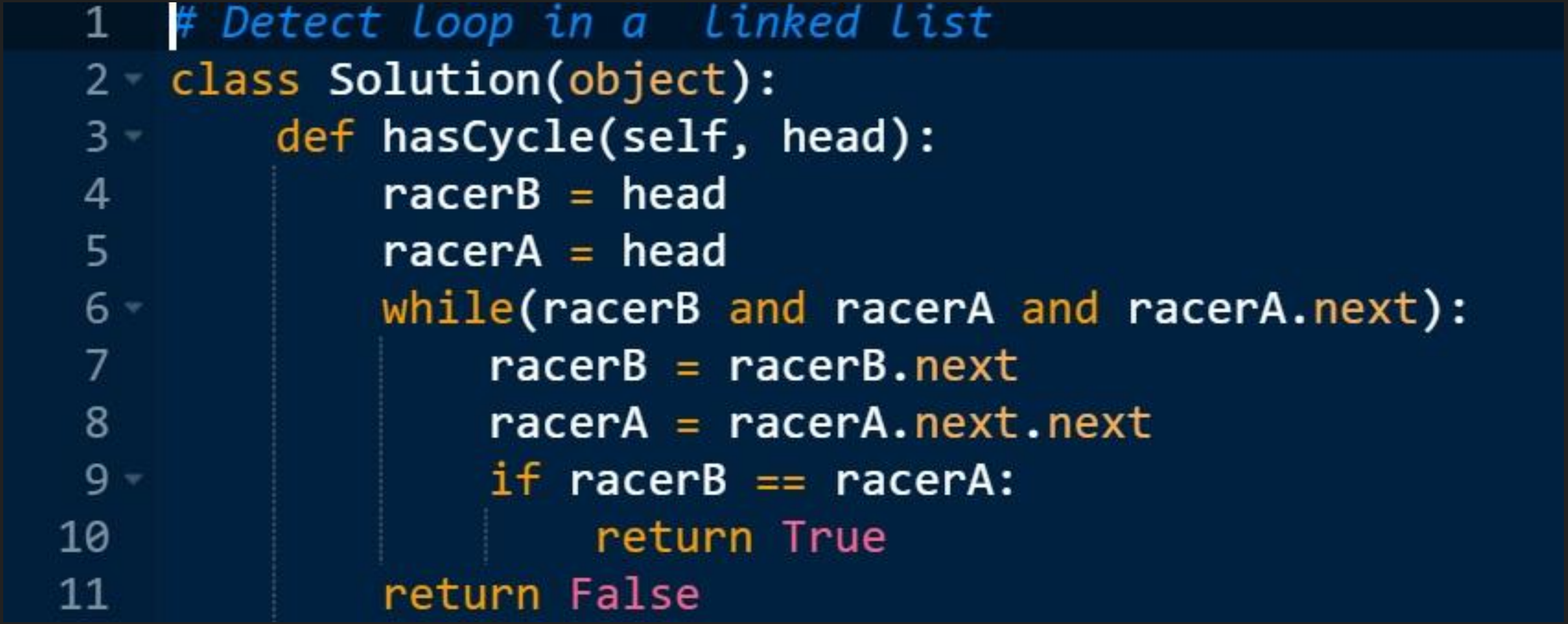
Question: Given a linked list, determine if it has a cycle in it.

YOU HAVE 15 MINUTES

https://leetcode.com/problems/linked-list-cycle/

INTUITION

Two racers start from the same start position on a circular race track on length 2 miles. Racer A runs at 2 miles per hour. Racer B runs at 1 mile per hour. Will they ever meet ?



```python
# Detect loop in a  linked list
class Solution(object):
    def hasCycle(self, head):
        racerB = head
        racerA = head
        while(racerB and racerA and racerA.next):
            racerB = racerB.next
            racerA = racerA.next.next
            if racerB == racerA:
                return True
        return False
```

TIME COMPLEXITY - O(N), SPACE COMPLEXITY - O(1)

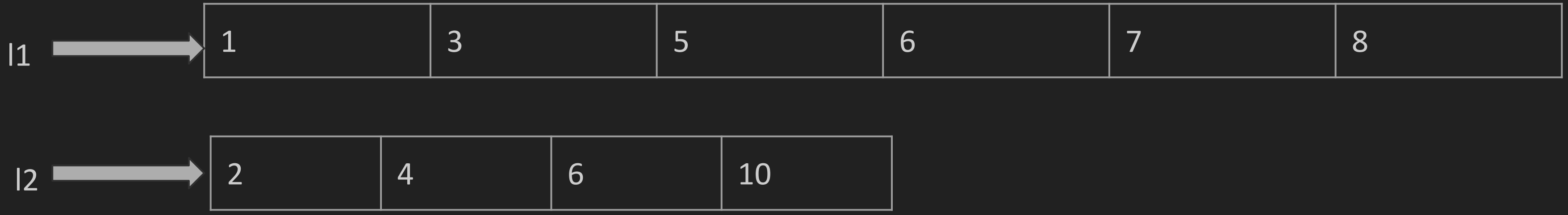
Question: Merge two sorted linked lists and return it as a new list. The new list should be made by splicing together the nodes of the first two lists.   YOU HAVE 15 MINUTES

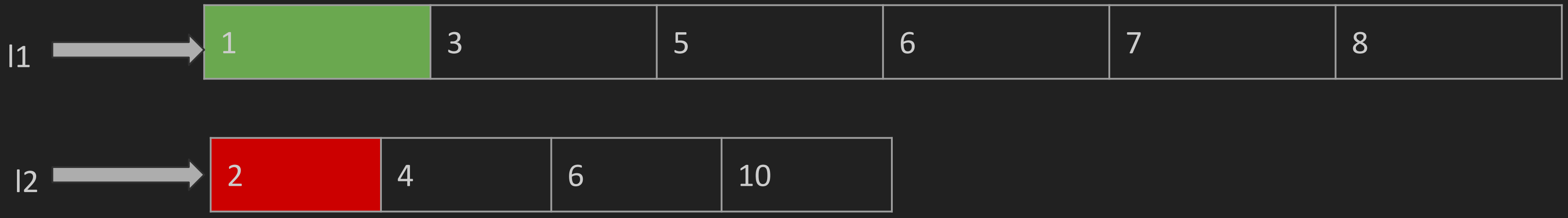
EXAMPLE:

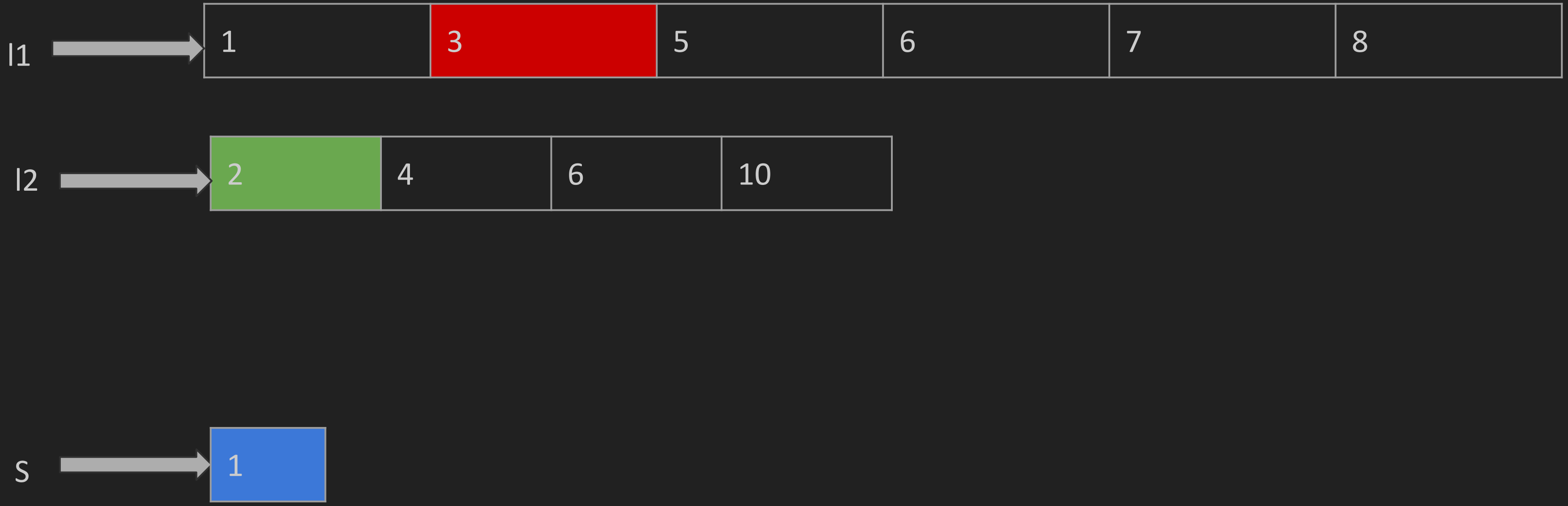
INPUT: 1->2->4, 1->3->4

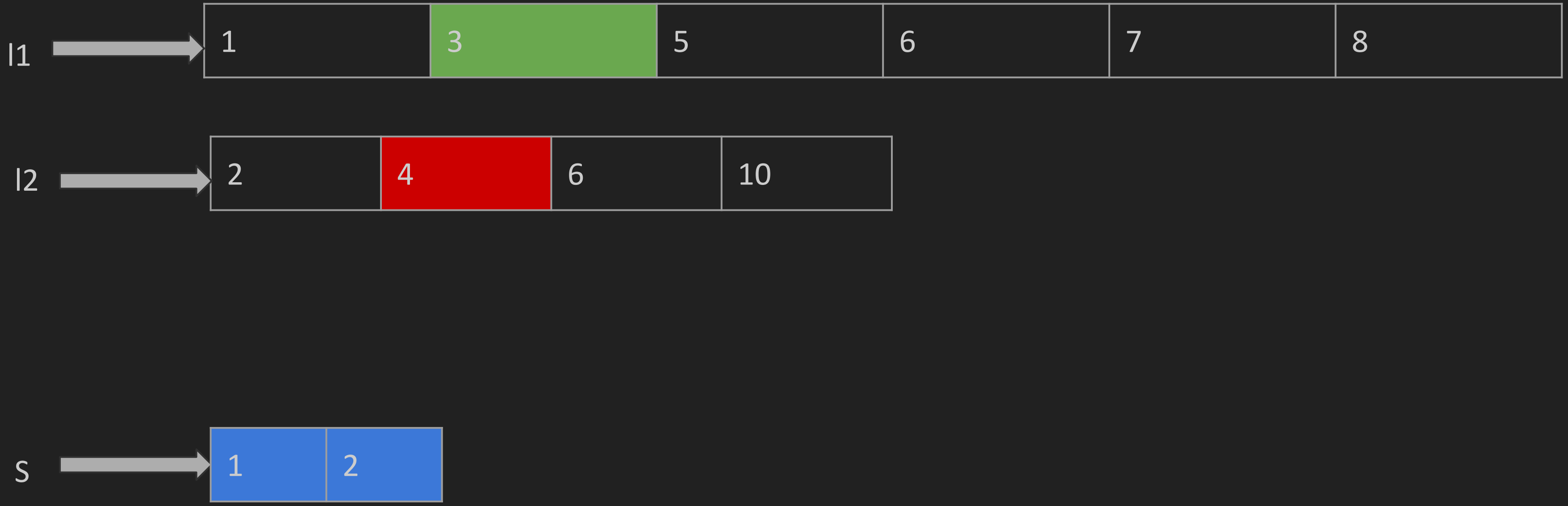
OUTPUT: 1->1->2->3->4->4

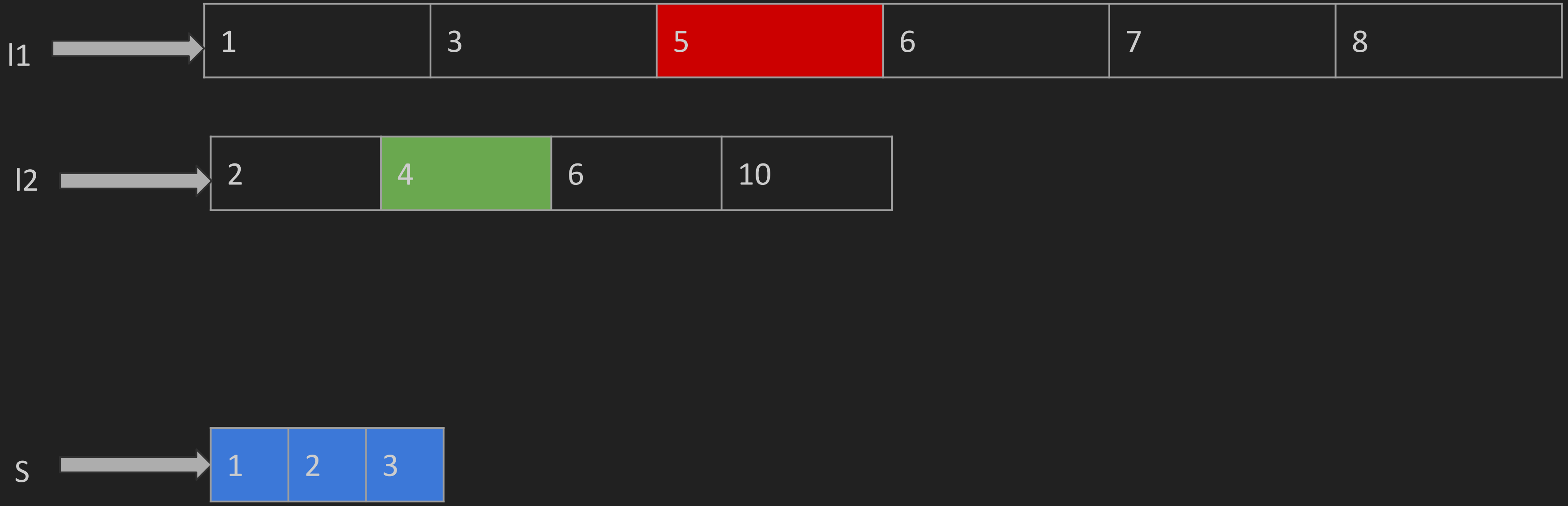
Link: https://leetcode.com/problems/merge-two-sorted-lists/

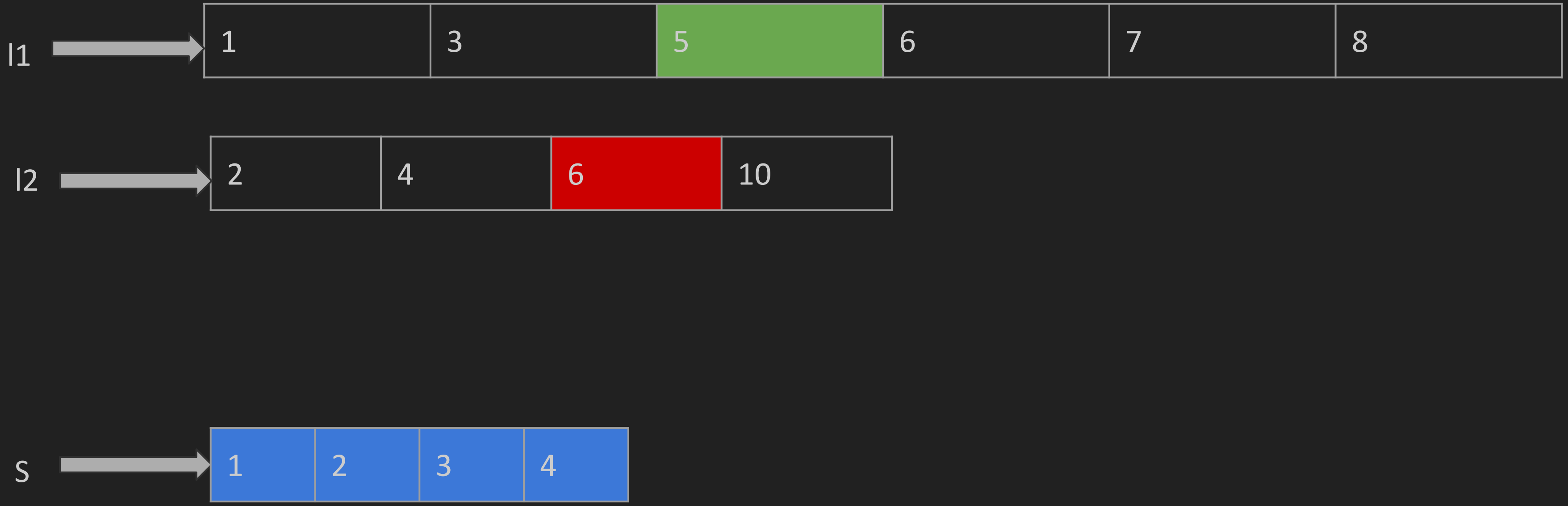

l1
1
3
5
6
7
8
l2
2
4
6
10

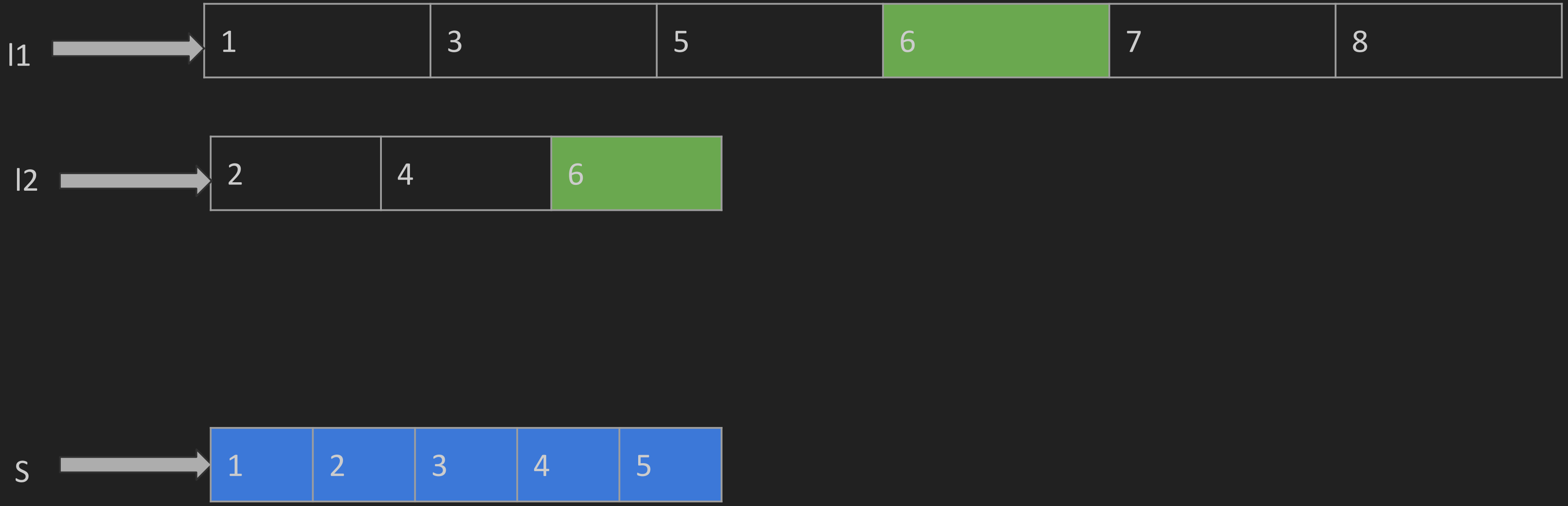

l1
1
3
5
6
7
8
l2
2
4
6
10


l1
1
3
5
6
7
8
l2
2
4
6
10
S
1

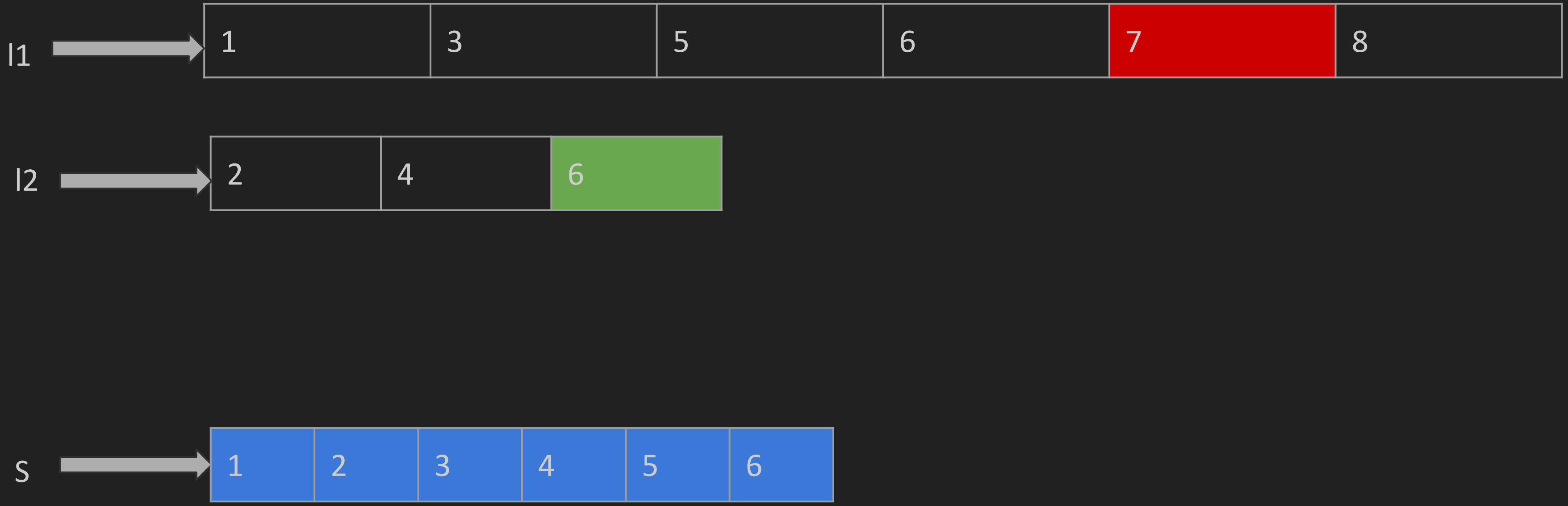

l1
1
3
5
6
7
8
l2
2
4
6
10
s
1
2

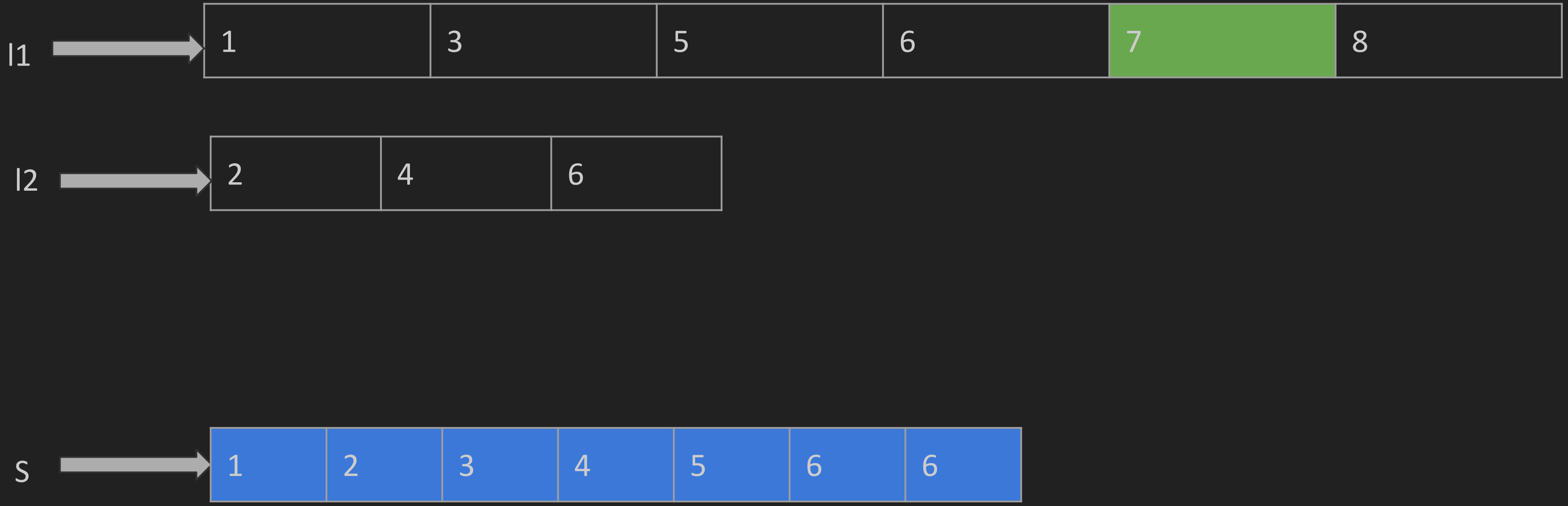

l1
1
3
5
6
7
8
l2
2
4
6
10
S
1
2
3

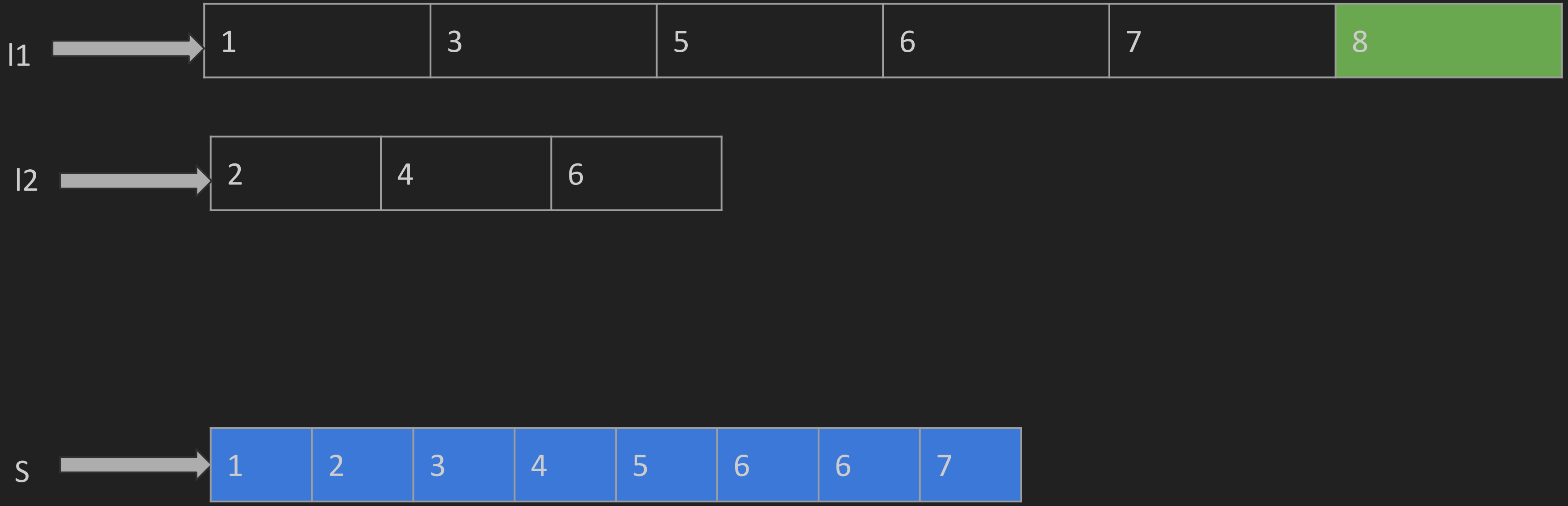

l1
1
3
5
6
7
8
l2
2
4
6
10
S
1
2
3
4

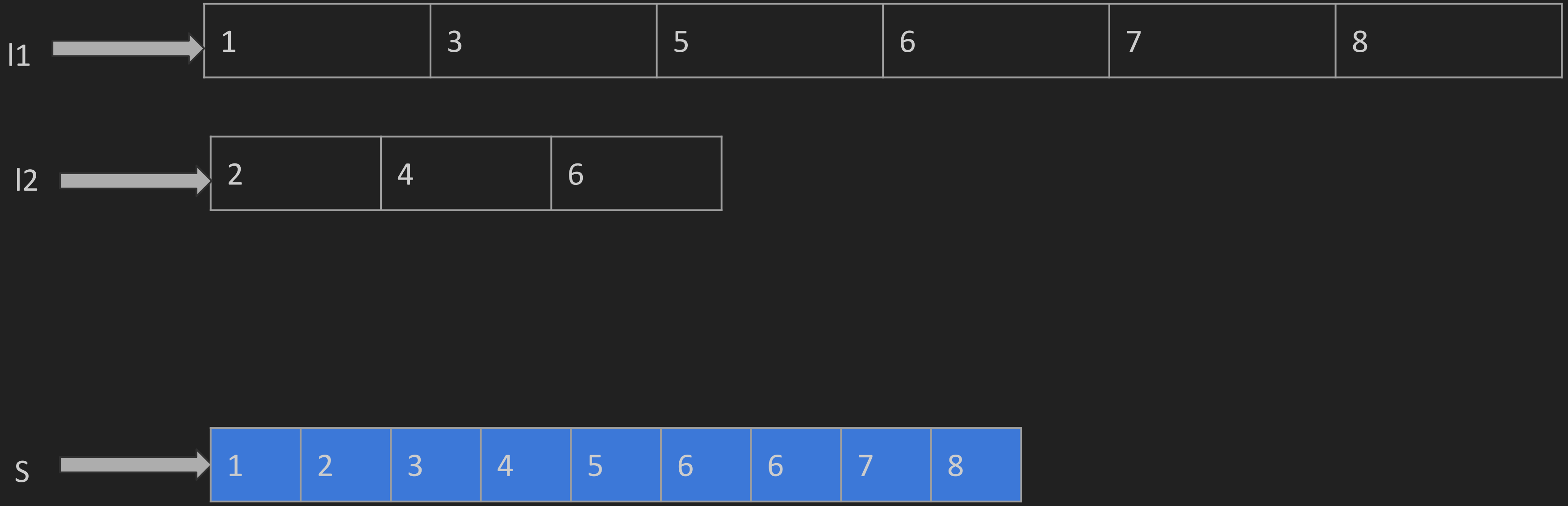


l1
1
3
5
6
7
8
l2
2
4
6
S
1
2
3
4
5

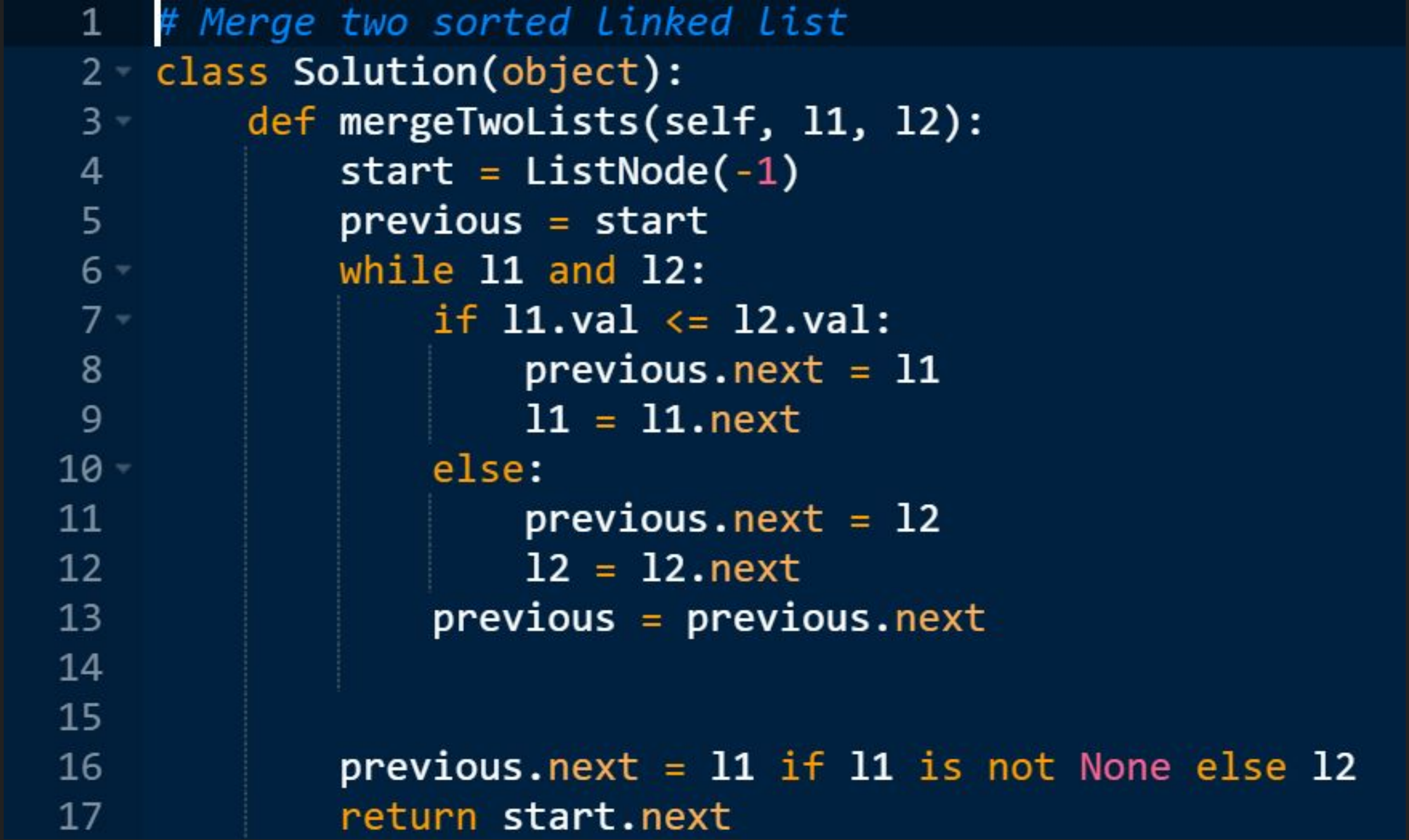

l1
1
3
5
6
7
8
l2
2
4
6
S
1
2
3
4
5
6


l1
1
3
5
6
7
8
l2
2
4
6
S
1
2
3
4
5
6
6


l1
1
3
5
6
7
8
l2
2
4
6
s
1
2
3
4
5
6
6
7



DONE



```python
# Merge two sorted linked list
class Solution(object):
    def mergeTwoLists(self, l1, l2):
        start = ListNode(-1)
        previous = start
        while l1 and l2:
            if l1.val <= l2.val:
                previous.next = l1
                l1 = l1.next
            else:
                previous.next = l2
                l2 = l2.next
            previous = previous.next


        previous.next = l1 if l1 is not None else l2
        return start.next
```

TIME COMPLEXITY - O(M+N)

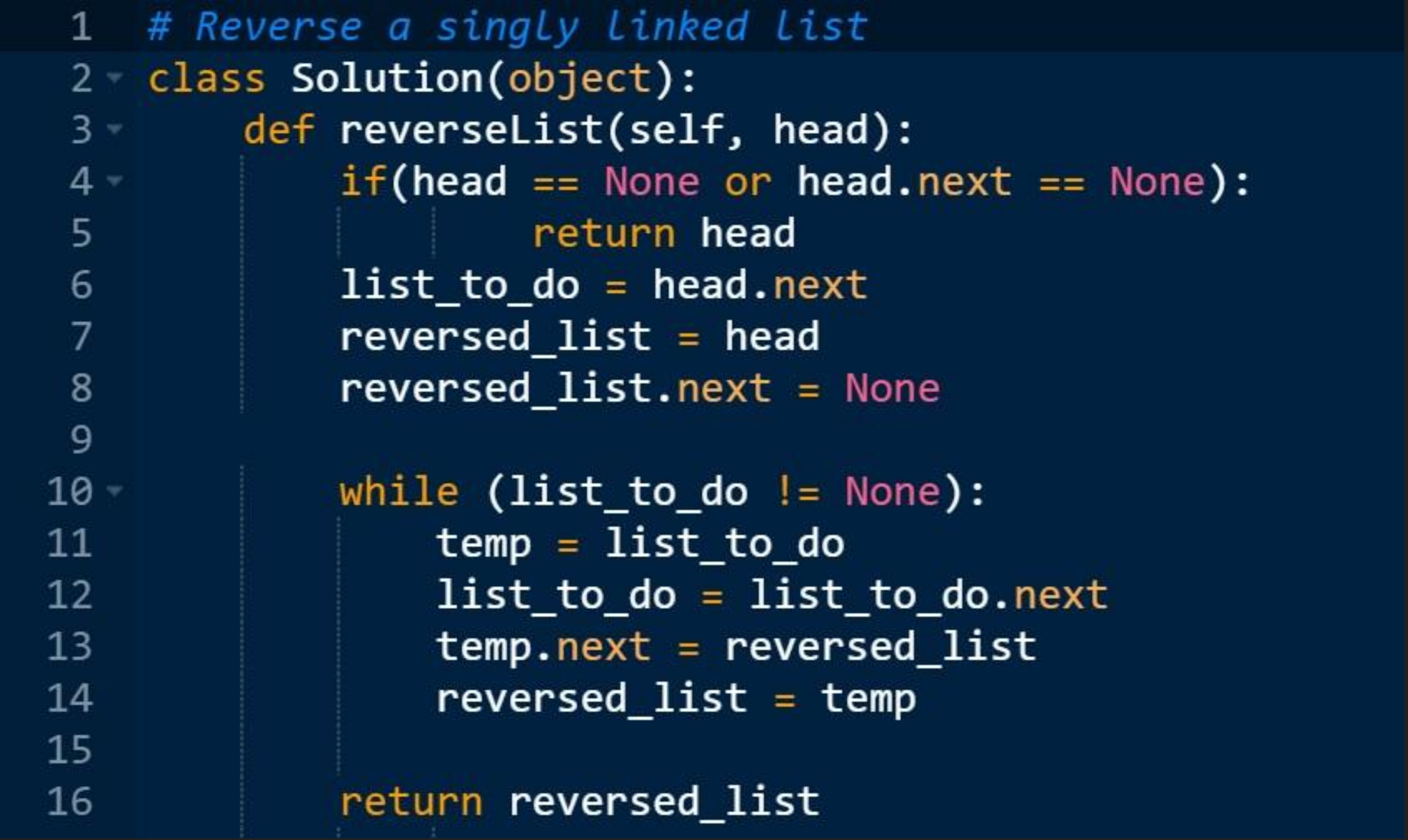
Question: Reverse a singly linked list.

YOU HAVE 15 MINUTES

https://leetcode.com/problems/reverse-linked-list/

INTUITION

https://www.educative.io/courses/coderust-hacking-the-coding-interview/lq2j



```python
# Reverse a singly linked list
class Solution(object):
    def reverseList(self, head):
        if(head == None or head.next == None):
                return head
        list_to_do = head.next
        reversed_list = head
        reversed_list.next = None

        while (list_to_do != None):
            temp = list_to_do
            list_to_do = list_to_do.next
            temp.next = reversed_list
            reversed_list = temp

        return reversed_list
```

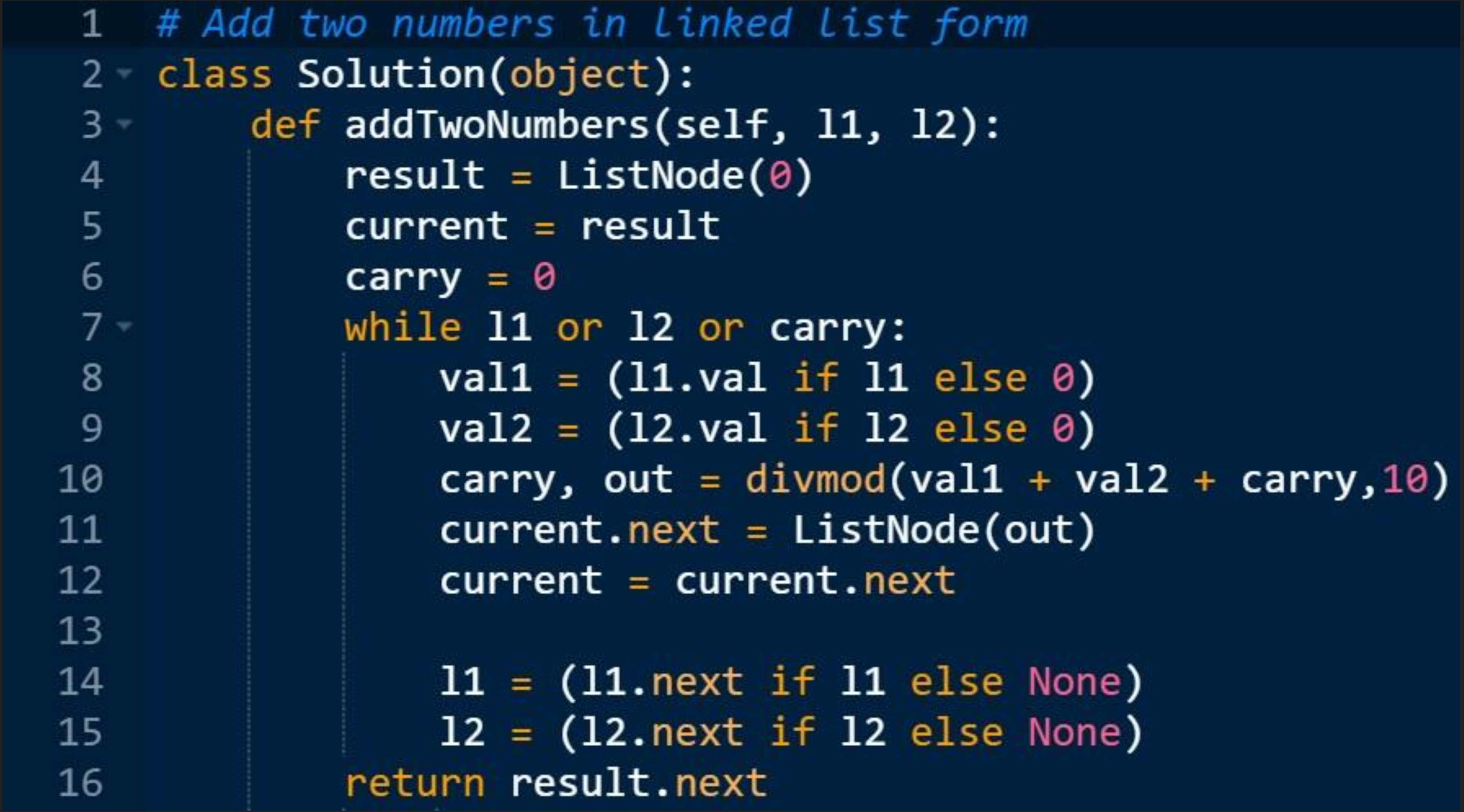
TIME COMPLEXITY - O(N), SPACE COMPLEXITY - O(1)

Question: You are given two non-empty linked lists representing two non-negative integers. The digits are stored in reverse order and each of their nodes contain a single digit. Add the two numbers and return it as a linked list.

YOU HAVE 15 MINUTES

LOOK UP :

Link: https://leetcode.com/problems/add-two-numbers/



```python
# Add two numbers in linked list form
class Solution(object):
    def addTwoNumbers(self, l1, l2):
        result = ListNode(0)
        current = result
        carry = 0
        while l1 or l2 or carry:
            val1 = (l1.val if l1 else 0)
            val2 = (l2.val if l2 else 0)
            carry, out = divmod(val1 + val2 + carry,10)
            current.next = ListNode(out)
            current = current.next

            l1 = (l1.next if l1 else None)
            l2 = (l2.next if l2 else None)
        return result.next
```