[Python] 4 solutions, space complexity O(n) O(n^(1/t)) O(lgn) O(1)

migeater ★ 81 Last Edit: April 26, 2020 12:10 AM 844 VIEWS

26

Welcome to add more combinations of time and space complexity.

n is length of linked list.

time	space	algorithm
O(n)	O(n)	recursion or use stack
O(n)	O(n^(1/t) + t) for an arbitrary positive integer t	sqrt decomposition
O(n lg n)	O(lg n)	divide and conquer
O(n^2)	O(1)	load a magazine?

recursion or use stack

```
class Solution:
    def printLinkedListInReverse(self, head: 'ImmutableListNode') -> None:
        if head:
            self.printLinkedListInReverse(head.getNext());
            head.printValue();
```

sqrt decomposition

1. Divide the linked list into n^(1/2) blocks of size n^(1/2);
2. Then the start-nodes of these blocks are stored in a stack of length n^(1/2).
3. Finally, take blocks from the stack and *print it recursively* with time O(n^(1/2)) and space O(n^(1/2)).

The total time complexity is O(n) and space O(n^(1/2)).

```
class Solution:
    def printLinkedListInReverseDirect(self, head, size):
        if size and head:
            self.printLinkedListInReverseDirect(head.getNext(),size-1)
            head.printValue()

    def printLinkedListInReverse(self, head: 'ImmutableListNode') -> None:
        def getLinkedListSize(head):
            size=0
            while head!=None:
                size+=1
                head=head.getNext()
            return size

        size=getLinkedListSize(head)

        num_blocks = math.ceil(math.sqrt(size))
        block_size = math.ceil(size/num_blocks)

        blocks = [] # using List as Stack
        head_cpy,cur = head,0
        for cur in range(size):
            if cur%block_size==0:
                blocks.append(head_cpy)
            head_cpy=head_cpy.getNext()

        for i in range(num_blocks-1,0 -1,-1):
            self.printLinkedListInReverseDirect(blocks[i], block_size)
```

You can cost less space if you use such decomposition technique to print blocks, see miracle173's answer

```
            head.printValue()
    def printLinkedListInReverse(self, head: 'ImmutableListNode') -> None:
        def getLinkedListSize(head):
            size=0
            while head!=None:
                size+=1
                head=head.getNext()
            return size

        size=getLinkedListSize(head)
        self.printLinkedListInReverseDecomposition(head,size,t=3)
    def printLinkedListInReverseDecomposition(self, head, size, t):
        if size==1:
            head.printValue()
            return
        num_blocks = math.ceil(size**(1/t)) # at least two blocks.
        block_size = math.ceil(size/num_blocks)

        blocks = [] # using List as Stack

        head_cpy,cur = head,0

        for cur in range(size):
            if cur%block_size==0:
                blocks.append(head_cpy)
            head_cpy=head_cpy.getNext()

        for i in range(num_blocks-1,0 -1,-1):
            nxt_size = block_size
            if i==num_blocks-1:
                nxt_size = cur%block_size+1

            if t==2:
                self.printLinkedListInReverseDirect(blocks[i], nxt_size)
            else:
                self.printLinkedListInReverseDecomposition(blocks[i], nxt_size,t-1)
```

divide and conquer

Divide into two parts print the right part first using this algorithm again.

```
class Solution:
    def printLinkedListInReverse(self, head: 'ImmutableListNode') -> None:
        def getLinkedListSize(head):
            size=0
            while head!=None:
                size+=1
                head=head.getNext()
```

```
                half=head
                for _ in range(n//2):
                    half=half.getNext()
                helper(half,n-n//2)
                helper(head,n//2)
            elif n!=0:
                head.printValue()

        size=getLinkedListSize(head)
        helper(head,size)
```

T(n)=2T(n/2)+n/2
use master theorem
$$ f(n)=\Theta(n^1log^0(n)) $$
$$ T(n)=\Theta(n^1log^1(n)) $$

load a magazine?

```
class Solution:
    def printLinkedListInReverse(self, head: 'ImmutableListNode') -> None:
        tail=None
        while tail!=head:
            curr=head
            while curr.getNext()!=tail:
                curr=curr.getNext()
            tail=curr
            tail.printValue()
```

divide and conquer recusion sqrt decomposition

Comments: 3 Best Most Votes Newest to Oldest Oldest to Newest

Type comment here... (Markdown is supported)

Post

hmz1991 ★ -5 Last Edit: March 30, 2020 2:15 PM

Another O(N logN) time and O(log(N)) space complexity:

```
def printLinkedListInReverse(self, head: 'ImmutableListNode') -> None:

        def help(head,next_head):

            if head==None:
                return

            # base case
            if head.getNext()==None or head.getNext()==
```

Read More

0 Show 2 replies Reply

dstark85 ★ 0 February 27, 2020 7:13 AM

Fantastic! Do you happen to know of a good resource for sqrt decomposition? That blew my mind.

0 Show 1 reply Reply

shankla2 ★ 35 January 26, 2020 3:14 AM

Great Summary!

0 Reply