Assignment #4: 位操作、栈、链表、堆和NN

Updated 1203 GMT+8 Mar 10, 2025

2025 spring, Complied by **袁奕 2400010766 数院**

> **说明：**
>
> 1. **解题与记录：**
>
>   对于每一个题目，请提供其解题思路（可选），并附上使用Python或C++编写的源代码（确保已在OpenJudge， Codeforces，LeetCode等平台上获得Accepted）。请将这些信息连同显示“Accepted”的截图一起填写到下方的作业模板中。（推荐使用Typora https://typoraio.cn 进行编辑，当然你也可以选择Word。）无论题目是否已通过，请标明每个题目大致花费的时间。
>
> 2. **提交安排：**提交时，请首先上传PDF格式的文件，并将.md或.doc格式的文件作为附件上传至右侧的“作业评论”区。确保你的Canvas账户有一个清晰可见的头像，提交的文件为PDF格式，并且“作业评论”区包含上传的.md或.doc附件。
>
> 3. **延迟提交：**如果你预计无法在截止日期前提交作业，请提前告知具体原因。这有助于我们了解情况并可能为你提供适当的延期或其他帮助。
>
> 请按照上述指导认真准备和提交作业，以保证顺利完成课程要求。

1. 题目

136.只出现一次的数字

bit manipulation, https://leetcode.cn/problems/single-number/

请用位操作来实现，并且只使用常量额外空间。

想到一个类似的很经典的问题 : 1024瓶酒, 1瓶有毒. 可以用10只小鼠找到.

方法为给第 i 只鼠鼠喂二进制中第 i 位为 1 的所有酒, 然后设被毒死的小鼠为 $a_1, a_2, \cdots, a_k$,

那么毒酒的编号恰好二进制中 $a_1, a_2, \cdots, a_k$ 位为 1, 其他位为 0

代码：

```
class Solution(object):
    def singleNumber(self, nums):
        sum = 0
        for n in nums:
            sum = sum ^ n
        return sum
```

20140:今日化学论文

stack, http://cs101.openjudge.cn/practice/20140/

代码:

```
def trans(s : str) -> str:
    st, res = [], ""
    for i, c in enumerate(s):
        if len(st) == 0 and c not in {"[", "]"}:
            res += c
        if c == "[":
            st.append(i)
        if c == "]":
            start = st.pop() + 1
            if len(st) == 0:
                nums = ""
                while s[start].isdigit():
                    nums += s[start]
                    start += 1
                res += trans(s[start : i]) * int(nums)
    return res
```

```
17
18 s = input()
19 print(*trans(s), sep = "")
```

状态：Accepted

源代码

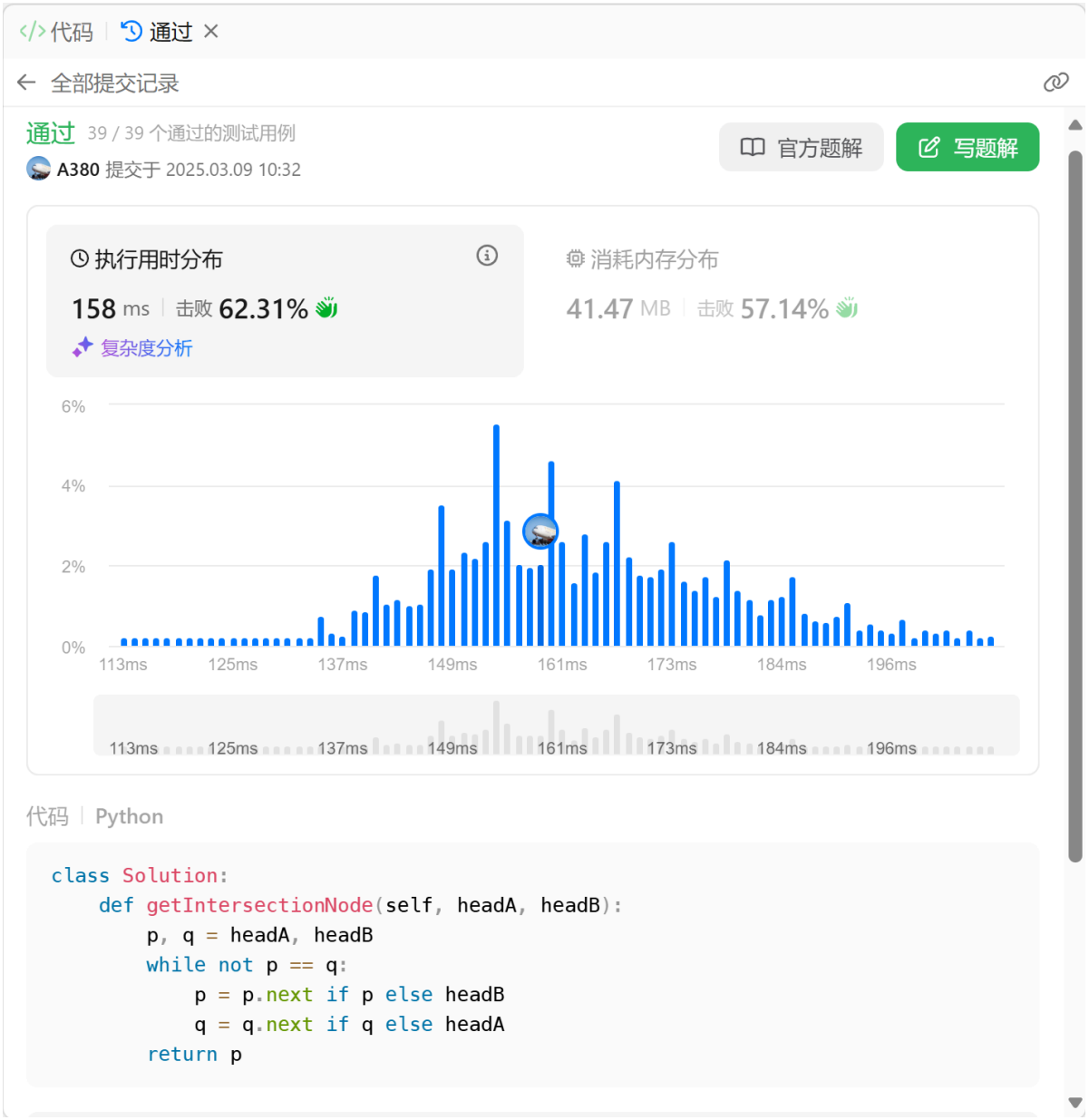
```
def trans(s : str) -> str:
    st, res = [], ""

    for i, c in enumerate(s):
        if len(st) == 0 and c not in {"[", "]"}:
            res += c
        if c == "[":
            st.append(i)
        if c == "]":
            start = st.pop() + 1
            if len(st) == 0:
                nums = ""
                while s[start].isdigit():
                    nums += s[start]
                    start += 1
                res += trans(s[start : i]) * int(nums)
    return res

s = input()
print(*trans(s), sep = "")
```

160.相交链表

linked list, https://leetcode.cn/problems/intersection-of-two-linked-lists/

代码：

```
1 class Solution:
2     def getIntersectionNode(self, headA, headB):
3         p, q = headA, headB
4         while not p == q:
5             p = p.next if p else headB
6             q = q.next if q else headA
7         return p
```

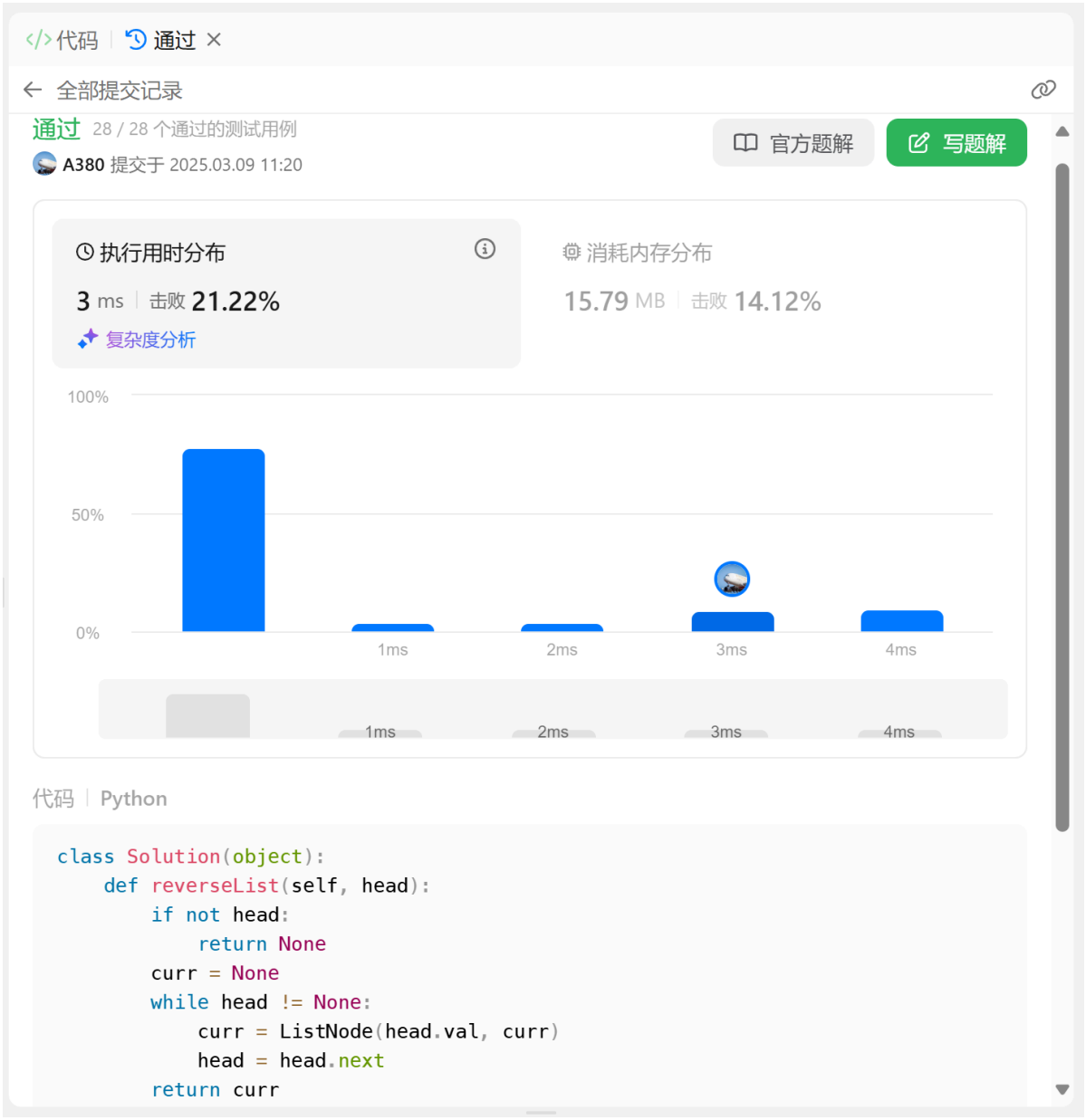


206.反转链表

linked list, https://leetcode.cn/problems/reverse-linked-list/

代码:

```
class Solution(object):
    def reverseList(self, head):
        if not head:
            return None
        curr = None
        while head != None:
            curr = ListNode(head.val, curr)
            head = head.next
        return curr
```



3478.选出和最大的K个元素

heap, https://leetcode.cn/problems/choose-k-elements-with-maximum-sum/

有一个错误的思路, 只能处理 **恰好** 为 k 个数的情形

```python
from functools import reduce
import heapq

class MAXHeap:
    def __init__(self, nums):
        self.nums = [(-x, y, z) for (x, y, z) in nums]
        heapq.heapify(self.nums)
    def push(self, ele):
        (x, y, z) = ele
        heapq.heappush(self.nums, (-x, y, z))
    def pop(self):
```

```
        (x, y, z) = heapq.heappop(self.nums)
        return -x, y, z
    def peek(self):
        x, y, z = self.nums[0]
        return -x, y, z

class Solution(object):
    def simp_num1(self, nums1):
        pair = sorted((x, i) for i, x in enumerate(nums1))
        pair = {id : i for i, (_, id) in enumerate(pair)}
        return [pair[i] for i in range(len(nums1))]
    def findMaxSum(self, nums1, nums2, k):
        n = len(nums1)
        res = [0] * n
        nums1 = self.simp_num1(nums1)
        pair = sorted([(x, y, i) for i, (x, y) in enumerate(zip(nums1, nums2))],
                      key = lambda x: x[1], reverse = True)
        hp = MAXHeap(pair[:k])
        sum = reduce(lambda x, y: x[1] + y[1], pair[:k])
        for i in range(k, n):
            (x, y, _) = hp.pop()
            x += 1
            while x in range(n) and res[x] == 0:
                res[x] = sum
                x += 1
            hp.push(pair[i])
            sum += pair[i][1] - y
        return [res[nums1[i]] for i in range(n)]

print(Solution().findMaxSum(nums1 = [4,2,1,5,3], nums2 = [10,20,30,40,50], k = 2))
```

我的思路是 greedy, 先选 `num2` 最大的 `k` 个元素, 每次去除其中 `num1` 最大的元素, 加入没选过的 `num2` 最大的元素. 每次一进一出, 总数固定为 `k

学习了解答, 感觉以 `num1` 为基准的 `heap` 维护更加灵活, 直到堆满才 `heappop`

```
from heapq import heappush, heappop

class Solution:
    def findMaxSum(self, nums1, nums2, k):
        n = len(nums1)
        res = [0] * n
        pair = sorted((x, y, i) for i, (x, y) in enumerate(zip(nums1, nums2)))
        hp = []
        sum = 0
        for i, (x, y, idx) in enumerate(pair):
            res[idx] = res[pair[i - 1][2]] if i and x == pair[i - 1][0] else sum
            sum += y
            heappush(hp, y)
            if len(hp) > k:
                sum -= heappop(hp)
```

```
        return res
```

代码

Python 智能模式

```python
from heapq import heappush, heappop

class Solution:
    def findMaxSum(self, nums1, nums2, k):
        n = len(nums1)
        pair = sorted((x, y, i) for i, (x, y) in enumerate(zip(nums1, nums2)))
        res = [0] * n
        h = []
        sum = 0
        for i, (x, y, idx) in enumerate(pair):
            res[idx] = res[pair[i - 1][2]] if i and x == pair[i - 1][0] else sum
            sum += y
            heappush(h, y)
            if len(h) > k:
                sum -= heappop(h)
        return res
```

已存储 行 7，列 22

测试用例 测试结果

通过 执行用时: 0 ms

• Case 1 • Case 2

输入

nums1 =

[4,2,1,5,3]

nums2 =

[10,20,30,40,50]

Q6.交互可视化neural network

https://developers.google.com/machine-learning/crash-course/neural-networks/interactive-exercises

Your task: configure a neural network that can separate the orange dots from the blue dots in the diagram, achieving a loss of less than 0.2 on both the training and test data.

Instructions:

In the interactive widget:

1. Modify the neural network hyperparameters by experimenting with some of the following config settings:
    - Add or remove hidden layers by clicking the **+** and **-** buttons to the left of the **HIDDEN LAYERS** heading in the network diagram.

    - Add or remove neurons from a hidden layer by clicking the **+** and **-** buttons above a hidden-layer column.
    - Change the learning rate by choosing a new value from the **Learning rate** drop-down above the diagram.
    - Change the activation function by choosing a new value from the **Activation** drop-down above the diagram.
2. Click the Play button above the diagram to train the neural network model using the specified parameters.
3. Observe the visualization of the model fitting the data as training progresses, as well as the **Test loss** and **Training loss** values in the **Output** section.
4. If the model does not achieve loss below 0.2 on the test and training data, click reset, and repeat steps 1–3 with a different set of configuration settings. Repeat this process until you achieve the preferred results.

给出满足约束条件的截图，并说明学习到的概念和原理。

2. 学习总结和收获

---------->

其中 146. LRU 缓存 - 力扣（LeetCode） 花费很多经历.

1. 链表插入和删除需要注意讨论链表为空, 删除在头或尾等 trivial case.
2. 尽量不要用一个 `dict[key_type, tuple[pointer_type, value_type]]` 来同时描述一个 `key` 在链表中的地址和对应的 `value`, 这样很容易混乱. 可以分成两个 `dict` 分别存储.

```python
class ListNode:
    def __init__(self, val, prev=None, next=None):
        self.val = val
        self.prev = prev
        self.next = next

class LinkedList:
    def __init__(self):
        self.head = None
        self.tail = None
    def head_push(self, val):
        new_node = ListNode(val)
        if not self.head:
            self.head, self.tail = new_node, new_node
        else:
            self.head.prev = new_node
            new_node.next = self.head
            self.head = new_node
        return val
    def delete(self, node):
        if self.head == self.tail:
            self.head, self.end = None, None
        elif node == self.head:
            node.next.prev = None
            self.head = node.next
        elif node == self.tail:
            node.prev.next = None
            self.tail = node.prev
        else:
            node.prev.next = node.next
            node.next.prev = node.prev
    def put_head(self, node):
        self.delete(node)
        self.head_push(node.val)


class LRUCache(object):
    def __init__(self, capacity):
        self.cache = dict() # key : node (type node : ListNode)
        self.pair = dict()  # key : value
        self.capacity = capacity
        self.linked_list = LinkedList() # (val : key)
    def get(self, key):
        if key not in self.cache:
            return -1
        node = self.cache[key]
        self.linked_list.put_head(node)
        self.cache[key] = self.linked_list.head
        return self.pair[key]
    def put(self, key, value):
        if key in self.cache:
            node = self.cache[key]
            self.linked_list.put_head(node)
            self.cache[key] = self.linked_list.head
            self.pair[key] = value
```

```
        else:
            self.linked_list.head_push(key)
            self.cache[key] = self.linked_list.head
            self.pair[key] = value
            if len(self.cache) > self.capacity:
                tail_key = self.linked_list.tail.val
                self.cache.pop(tail_key)
                self.pair.pop(tail_key)
                self.linked_list.delete(self.linked_list.tail)
```