数据结构与算法B 作业3

1078: Bigram分词

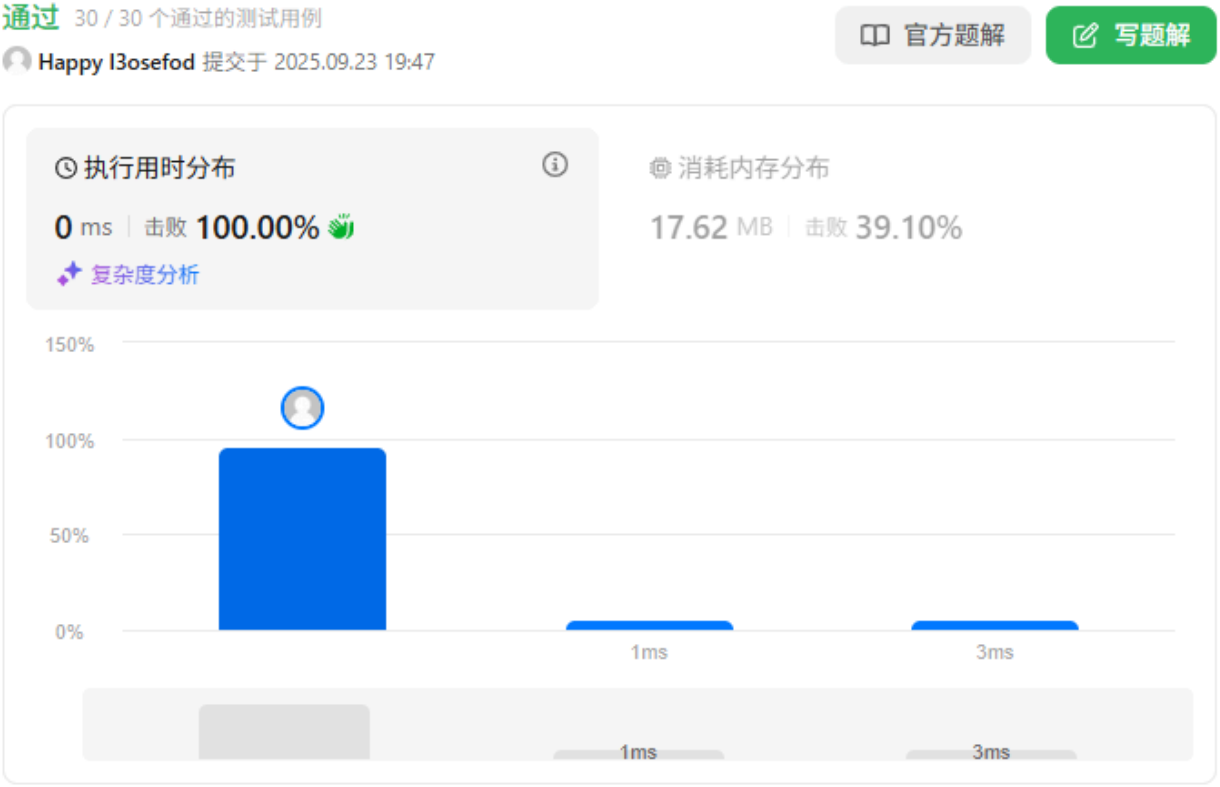
用时：20 min

https://leetcode.cn/problems/occurrences-after-bigram/

思路：使用一次遍历，判断i和i+1是否为第一个词和第二个词，若是，将第三个词加入即可。

代码：

```
class Solution:
    def findOcurrences(self, text: str, first: str, second: str) -> list[str]:
        word_list = text.split()
        length = len(word_list)
        ans = []
        for i in range(length-2):
            if word_list[i] == first:
                if word_list[i+1] == second:
                    ans.append(word_list[i+2])
        return ans
```



代码 | Python3

```python
class Solution:
    def findOcurrences(self, text: str, first: str, second: str) -> list[str]:
        word_list = text.split()
        length = len(word_list)
        ans = []
        for i in range(length-2):
            if word_list[i] == first:
                if word_list[i+1] == second:
```

查看更多

283.移动零

用时：20 min（倒着遍历）+ 30 min （双指针）

stack, two pinters, https://leetcode.cn/problems/move-zeroes/

思路：一开始使用倒着遍历的方式，避免pop和append元素的时候影响到index的位次。后来发现根本没用到双指针，同时pop和append的速度可能也导致了速度较慢。后来也是看了题解，重新学习了一下双指针的思路，重新写了一遍双指针的思路。

代码：

```python
class Solution:
    # 非双指针
```

```python
    def moveZeroes(self, nums: List[int]) -> None:
        """
        Do not return anything, modify nums in-place instead.
        """
        for i in range(len(nums)-1,-1,-1):
            if nums[i] == 0:
                nums.pop(i)
                nums.append(0)
```

通过 75 / 75 个通过的测试用例

Happy l3osefod 提交于 2025.09.23 19:59

官方题解 写题解

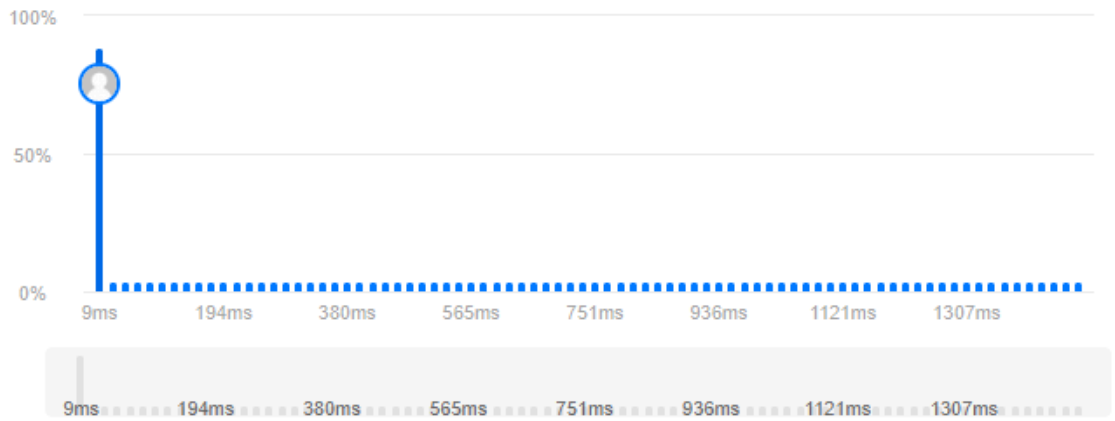
执行用时分布

15 ms | 击败 13.57%

复杂度分析

消耗内存分布

18.37 MB | 击败 97.06%



代码 | Python3

```python
class Solution:
    def moveZeroes(self, nums: List[int]) -> None:
        """
        Do not return anything, modify nums in-place instead.
        """
        for i in range(len(nums)-1,-1,-1):
            if nums[i] == 0:
                nums.pop(i)
                nums.append(0)
```

```python
class Solution:
    # 双指针
    def moveZeroes(self, nums: list[int]) -> None:
        i = 0
        j = 0
        while i < len(nums):
            if nums[i] == 0:
```

```
            while j < len(nums):
                # 将第二个指针j向右移动至非0处
                if nums[j] == 0:
                    j += 1
                elif j < i:
                    j += 1
                else:
                    nums[i], nums[j] = nums[j], nums[i]
                    break
        i += 1
```

通过 75 / 75 个通过的测试用例

Happy l3osefod 提交于 2025.09.23 20:18

官方题解 写题解

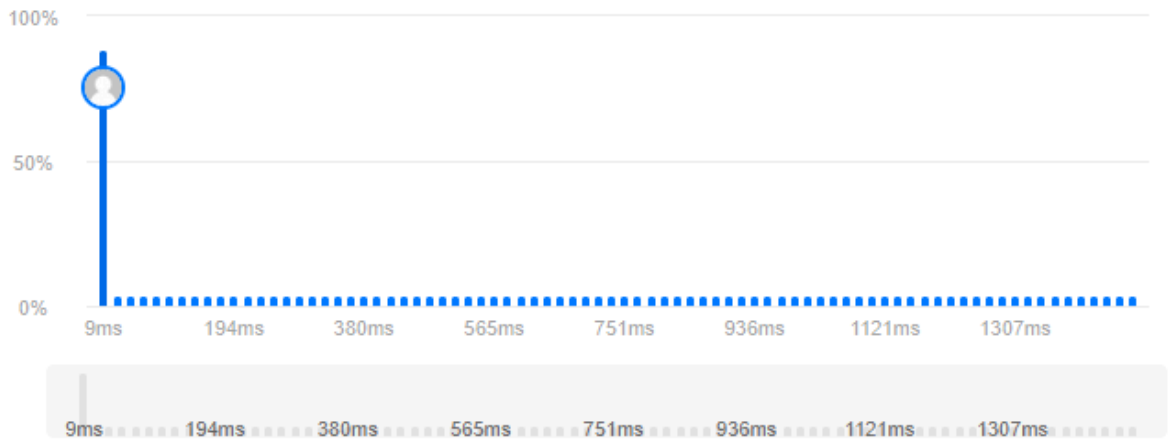
执行用时分布

7 ms | 击败 46.76%

复杂度分析

消耗内存分布

18.76 MB | 击败 15.08%



代码 | Python3

```python
class Solution:
    def moveZeroes(self, nums: list[int]) -> None:
        i = 0
        j = 0
        while i < len(nums):
            if nums[i] == 0:
                while j < len(nums):
                    if nums[j] == 0:
                        j += 1
                    elif j < i:
                        j += 1
                    else:
                        nums[i], nums[j] = nums[j], nums[i]
                        break
            i += 1
```

20.有效的括号

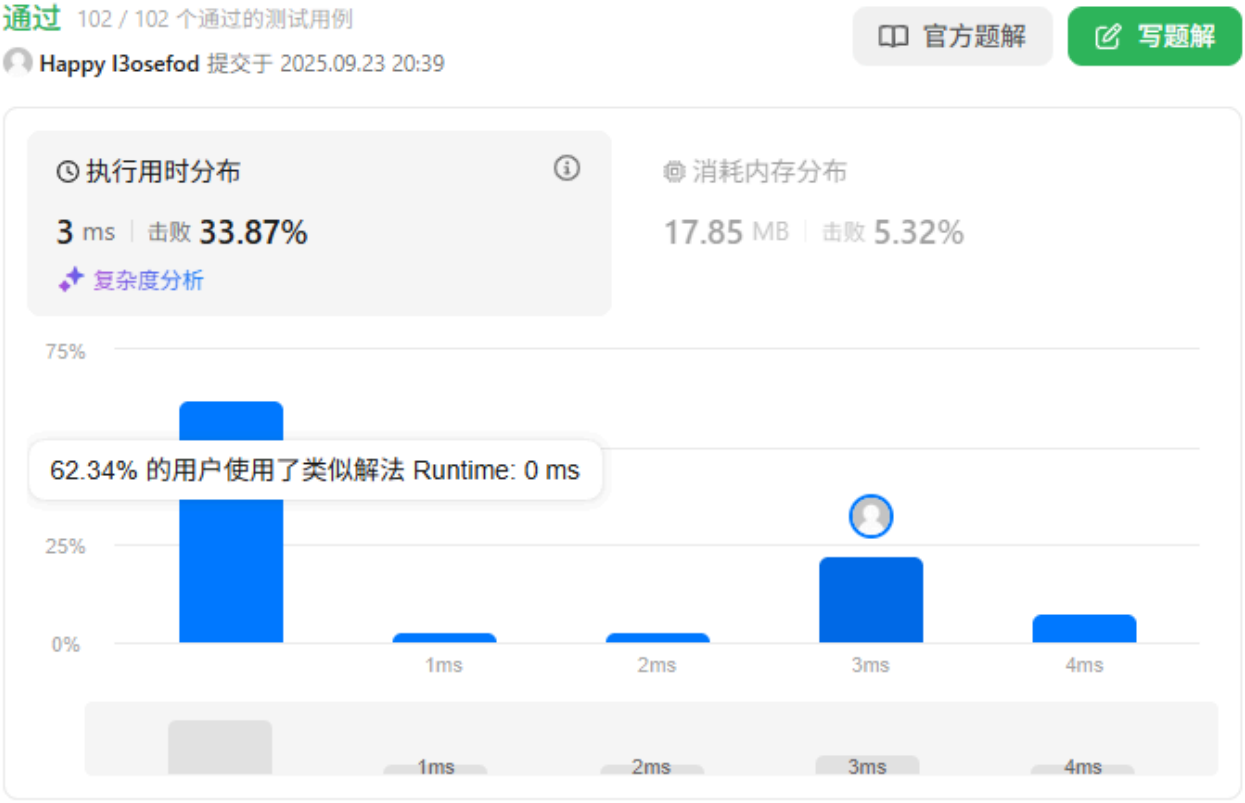
用时：20 min（基础框架）+10 min（debug左右括号数目不等的情况）

stack, https://leetcode.cn/problems/valid-parentheses/

思路：比较简单的栈的题目。总体就是将左括号添加至栈中，对新出现的右括号，就判断其与栈的最后一个左括号是否可以正确组合。除此之外，注意给出的文段会存在左右括号数目不等的问题即可。

代码：

```
class Solution:
    def isValid(self, s: str) -> bool:
        stack = []
        left = ['[', '{', '(']
        Valid = ['[]', '{}', '()']
        res = ''
        for i in s:
            if i in left:
                stack.append(i)
            else:
                if stack == []:
                    return False
                res = stack.pop() + i
                if res in Valid:
                    continue
                else:
                    return False
        if stack == []:
            return True
        else:
            return False
```



代码 | Python3

```python
class Solution:
    def isValid(self, s: str) -> bool:
        stack = []
        left = ['[', '{', '(']
        Valid = ['[]', '{}', '()']
        res = ''
        for i in s:
            if i in left:
                stack.append(i)
            else:
                if stack == []:
                    return False
                res = stack.pop() + i
                if res in Valid:
                    continue
                else:
```

118.杨辉三角

30 min

dp, https://leetcode.cn/problems/pascals-triangle/

思路：比较简单的DP题，题中给出的思路也很清晰，注意新行的每一个位置（除了首尾）都是上一行的[i-1][j-1]+[i-1][j] 即可。

代码：

```python
class Solution:
    def generate(self, numRows: int) -> list[list[int]]:
        ans = []
        first_two = [[1],[1, 1]]
        for i in range(min(numRows, 2)):
            ans.append(first_two[i])
        for i in range(2, max(2, numRows)):
            Row = [1]
            for j in range(1, i):
                Row.append(ans[i-1][j-1]+ans[i-1][j])
            Row.append(1)
            ans.append(Row)
        return ans
```

代码 | Python3

```python
class Solution:
    def generate(self, numRows: int) -> list[list[int]]:
        ans = []
        first_two = [[1],[1, 1]]
        for i in range(min(numRows, 2)):
            ans.append(first_two[i])
        for i in range(2, max(2, numRows)):
            Row = [1]
            for j in range(1, i):
                Row.append(ans[i-1][j-1]+ans[i-1][j])
            Row.append(1)
            ans.append(Row)
        return ans
```

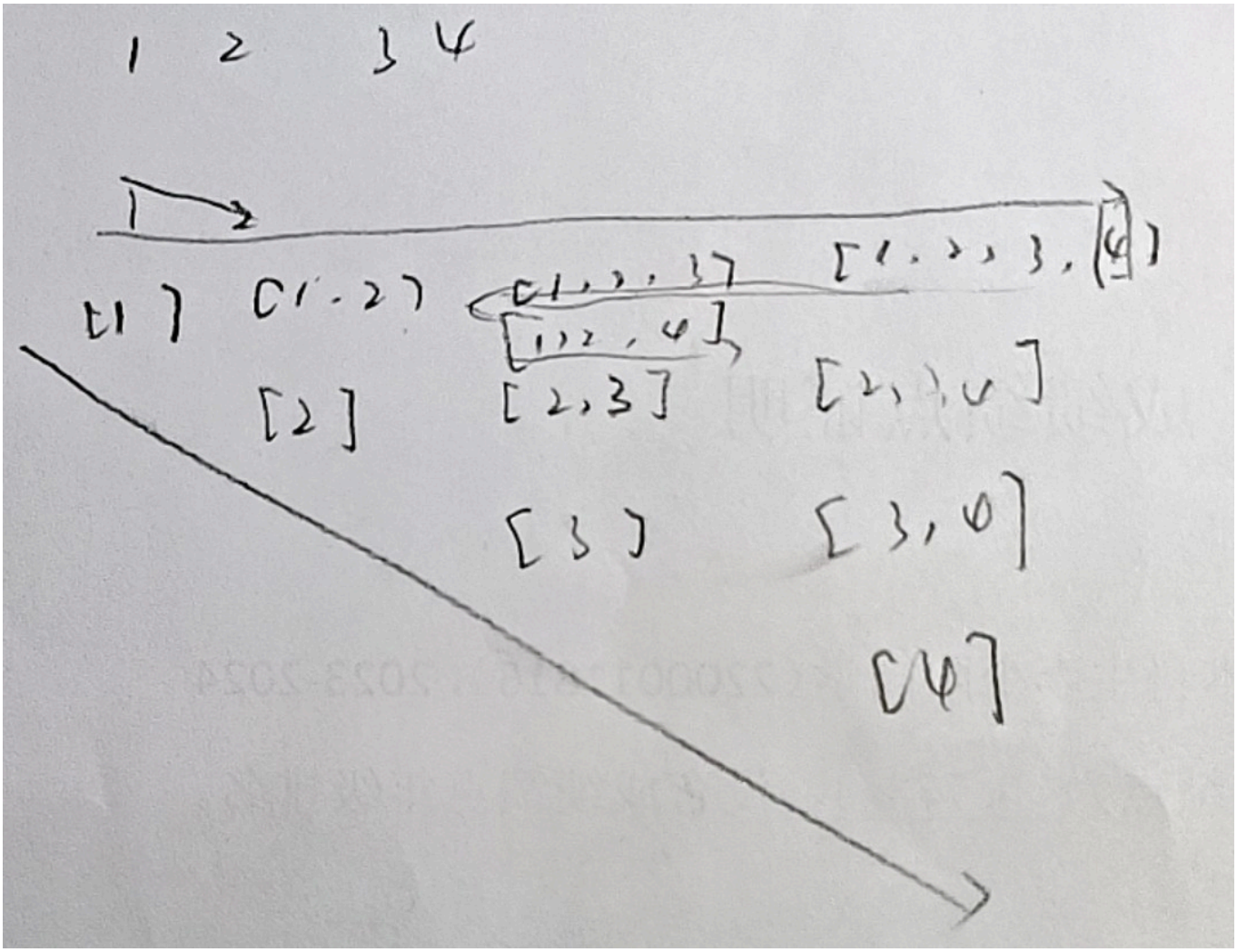
46.全排列

4 h（自己想怎么解）+ 1 h（看题解并写代码解题）

backtracking, https://leetcode.cn/problems/permutations/

思路：这题知道用递归去写，也善用了搜索，但是最后确实是思路一片混乱，写不出来，只能无奈地看了题解，在纸上照着题解找一个例子“运行”了一下代码流程，学会了之后再去把代码写出来。后面也是发现先在纸上画一下递归的路径对解题很有帮助（例如下图。不过只找到了写“子

集”这道题目的时候画的图）。



代码：

```python
class Solution:
    def permute(self, nums: list[int]) -> list[list[int]]:
        n = len(nums)
        permuted_array = []
        def backtrack(index):
            if index == n:
                permuted_array.append(nums[:])
            else:
                for i in range(index, n):
                    nums[index], nums[i] = nums[i], nums[index]
                    backtrack(index + 1)
                    nums[index], nums[i] = nums[i], nums[index]
        backtrack(index=0)
        return permuted_array
```

通过 26 / 26 个通过的测试用例

Happy l3osefod 提交于 2025.09.25 17:33

官方题解 写题解

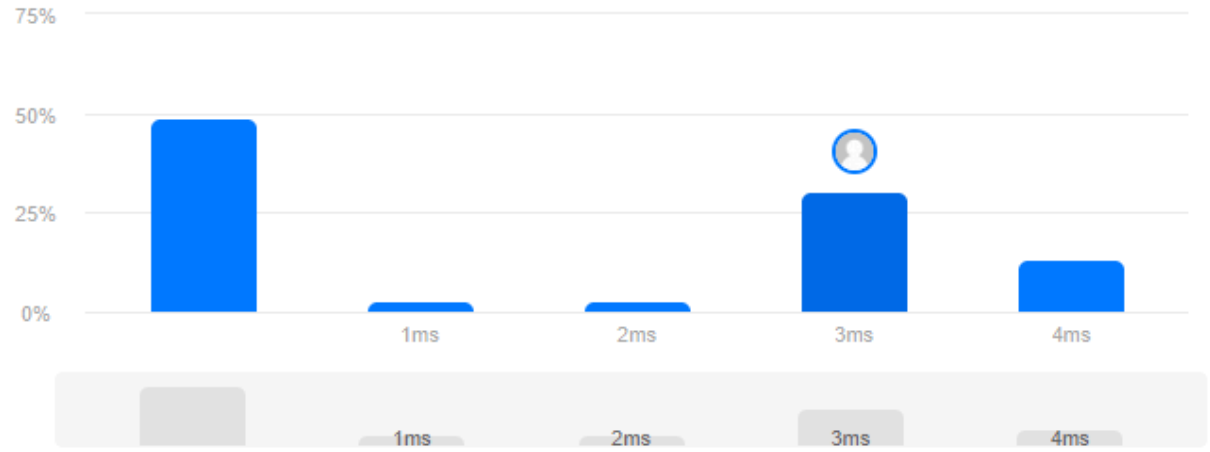
执行用时分布

3 ms | 击败 46.30%

复杂度分析

消耗内存分布

17.71 MB | 击败 53.32%



代码 | Python3

```python
class Solution:
    def permute(self, nums: list[int]) -> list[list[int]]:
        n = len(nums)
        permuted_array = []
        def backtrack(index):
            if index == n:
                permuted_array.append(nums[:])
            else:
                for i in range(index, n):
                    nums[index], nums[i] = nums[i], nums[index]
                    backtrack(index + 1)
                    nums[index], nums[i] = nums[i], nums[index]
        backtrack(index=0)
        return permuted_array
```

收起

78.子集

1 h

backtracking, https://leetcode.cn/problems/subsets/

思路：和全排列类似的递归思路。由于没有相同元素，只需将元素append进当前的子集（'current'）中，再通过递归的方式添加下一个元素（index为i+1的数），并在退出递归的时候删去添加进current中的元素即可。

代码：

```python
class Solution:
    def subsets(self, nums: list[int]) -> list[list[int]]:
        n = len(nums)
        subset_array = [[]]
        def backtrack(index, current):
            if index != n:
                for i in range(index, n):
                    current.append(nums[i])
                    subset_array.append(current[:])
                    backtrack(i + 1, current)
                    current.pop()
        backtrack(index=0, current=[])
        return subset_array
```

代码 | Python3

```python
class Solution:
    def subsets(self, nums: list[int]) -> list[list[int]]:
        n = len(nums)
        subset_array = [[]]
        def backtrack(index, current):
            if index != n:
                for i in range(index, n):
                    current.append(nums[i])
                    subset_array.append(current[:])
                    backtrack(i + 1, current)
                    current.pop()
        backtrack(index=0, current=[])
        return subset_array
```

收起

2. 学习总结和个人收获：

前面的4道题由于难度确实不高，完成的也相对轻松。但是后面出现递归的题目之后，发现自己也确实对递归不够熟悉，之前计概学的有个印象，但是几乎忘掉了怎么做的了，所以，花费了大量的时间去完成这两道题目，但是由于这周比较地忙，也没有时间去加练了。提交作业的时候是9月27日中午，后续（周日、国庆）计划也会对递归进行更多的题目练习。