Assignment #6: 回溯、树、双向链表和哈希表

Updated 1526 GMT+8 Mar 22, 2025

2025 spring, Complied by 蔡沐轩 数学科学学院

> **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

LC46.全排列

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

思路：

DFS回溯枚举每个位置的下标，记录当前已确定的位置，保证不重复即可。约5min。

代码：

```
class Solution:
    def permute(self, nums: List[int]) -> List[List[int]]:
        n=len(nums)
        visited=[0]*n
        ans,cur=[],[]

        def dfs(m:int):
            if m==n:
                ans.append(cur.copy())
                return
            for i in range(n):
                if not visited[i]:
                    visited[i]=1
                    cur.append(nums[i])
                    dfs(m+1)
```

```
                visited[i]=0
                cur.pop()

        dfs(0)
        return ans
```

代码运行截图（至少包含有"Accepted"）

代码 | Python3

```python
class Solution:
    def permute(self, nums: List[int]) -> List[List[int]]:
        n=len(nums)
        visited=[0]*n
        ans,cur=[],[]

        def dfs(m:int):
            if m==n:
                ans.append(cur.copy())
                return
            for i in range(n):
                if not visited[i]:
                    visited[i]=1
                    cur.append(nums[i])
                    dfs(m+1)
                    visited[i]=0
                    cur.pop()

        dfs(0)
        return ans
```

收起

LC79: 单词搜索

backtracking, https://leetcode.cn/problems/word-search/

思路：

用DFS回溯记录当前格子和`word`已经匹配位置，直到全部匹配返回即可。约10min。

代码：

```
class Solution:
    def exist(self, board: List[List[str]], word: str) -> bool:
        m,n=len(board),len(board[0])
        directions=[(1,0),(-1,0),(0,1),(0,-1)]
        visited=[[0]*n for _ in range(m)]
        def dfs(x:int,y:int,i:int):
            if board[x][y]!=word[i] or visited[x][y]:return False
            if i==len(word)-1:return True
            visited[x][y]=1
            b = False
            for direction in directions:
                x1,y1=x+direction[0],y+direction[1]
                if 0 <= x1 < m and 0 <= y1 < n and dfs(x1, y1, i + 1): b =True
            visited[x][y]=0
            return b
        for x in range(m):
            for y in range(n):
                if dfs(x,y,0):return True
        return False
```

代码运行截图（至少包含有"Accepted"）

通过 87 / 87 个通过的测试用例

Hyperalgebra 提交于 2025.03.07 10:39

官方题解 写题解

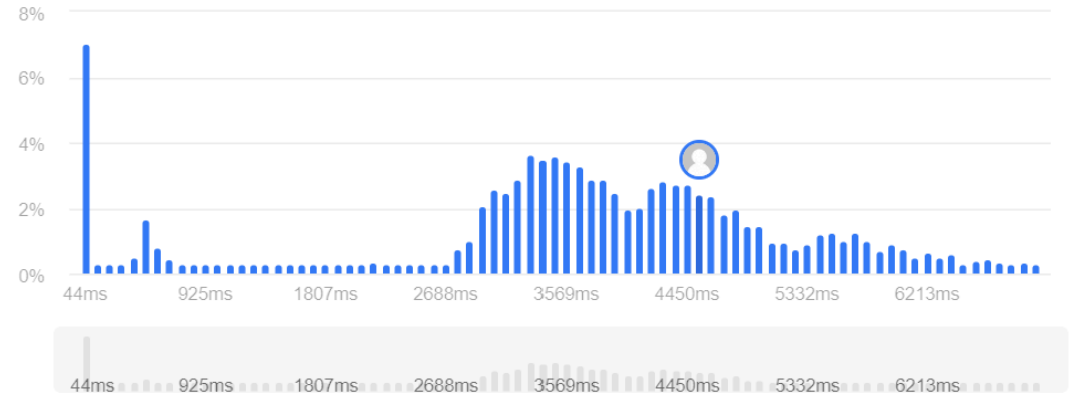
执行用时分布

4574 ms | 击败 30.93%

复杂度分析

消耗内存分布

17.86 MB | 击败 7.40%



代码 | Python3

```python
class Solution:
    def exist(self, board: List[List[str]], word: str) -> bool:
        m,n=len(board),len(board[0])
        directions=[(1,0),(-1,0),(0,1),(0,-1)]
        visited=[[0]*n for _ in range(m)]
        def dfs(x:int,y:int,i:int):
            if board[x][y]!=word[i] or visited[x][y]:return False
            if i==len(word)-1:return True
            visited[x][y]=1
            b = False
            for direction in directions:
                x1,y1=x+direction[0],y+direction[1]
                if 0 <= x1 < m and 0 <= y1 < n and dfs(x1, y1, i + 1): b =True
            visited[x][y]=0
            return b
        for x in range(m):
            for y in range(n):
                if dfs(x,y,0):return True
        return False
```

收起

LC94.二叉树的中序遍历

dfs, https://leetcode.cn/problems/binary-tree-inorder-traversal/

思路：

直接递归。约2min。

代码：

```python
class Solution:
    def inorderTraversal(self, root: Optional[TreeNode]) -> List[int]:
        if not root:return []
        return self.inorderTraversal(root.left) + [root.val] +
self.inorderTraversal(root.right)
```

代码运行截图（至少包含有"Accepted"）

LC102.二叉树的层序遍历

bfs, https://leetcode.cn/problems/binary-tree-level-order-traversal/

思路：

用队列存储节点和深度，BFS即可。约5min。

代码：

```python
class Solution:
    def levelOrder(self, root: Optional[TreeNode]) -> List[List[int]]:
        if not root:return []
        ans=[]
        q=deque([(root,0)])
        while q:
            x,d=q.popleft()
            if d>=len(ans):ans.append([])
            ans[d].append(x.val)
            if x.left:q.append((x.left,d+1))
            if x.right:q.append((x.right,d+1))
        return ans
```

代码运行截图 （至少包含有"Accepted"）

代码 | Python3

```python
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution:
    def levelOrder(self, root: Optional[TreeNode]) -> List[List[int]]:
        if not root:return []
        ans=[]
        q=deque([(root,0)])
        while q:
            x,d=q.popleft()
            if d>=len(ans):ans.append([])
            ans[d].append(x.val)
            if x.left:q.append((x.left,d+1))
            if x.right:q.append((x.right,d+1))
        return ans
```

收起

LC131.分割回文串

dp, backtracking, https://leetcode.cn/problems/palindrome-partitioning/

思路：

使用`@cache`相当于DP，`part(i)`函数相当于`self.partition(s[i:])`，可通过枚举第一个回文串加上后面字符串的回文串分割实现递推，通过DP避免重复计算加快速度。约10min。

代码：

```
class Solution:
    def partition(self, s: str) -> List[List[str]]:
        n=len(s)

        @cache
        def part(i:int):
            if i==n:return [[]]
            ans=[]
            for j in range(i+1,n+1):
                p=s[i:j]
                if p==p[::-1]:
                    p=[p]
                    ans+=map(lambda x:p+x,part(j))
            return ans

        return part(0)
```

代码运行截图（至少包含有"Accepted"）

通过 32 / 32 个通过的测试用例

Hyperalgebra 提交于 2025.03.01 09:22

官方题解 写题解

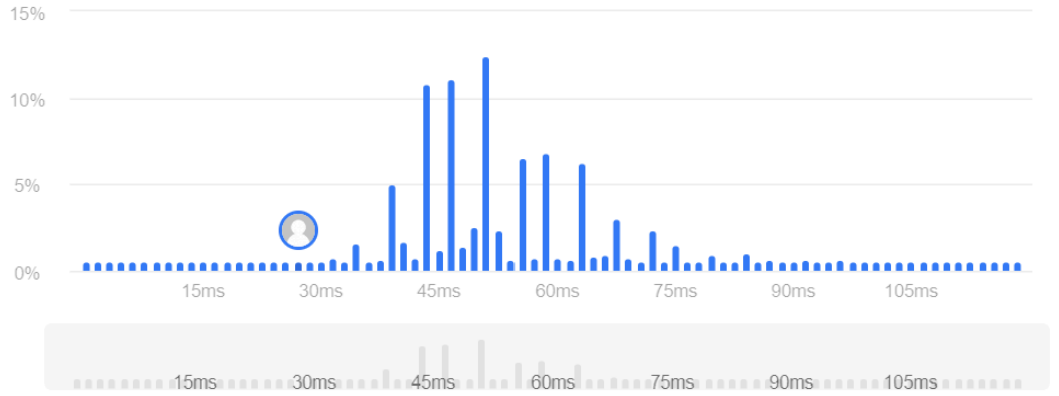
执行用时分布

27 ms | 击败 99.32%

复杂度分析

消耗内存分布

33.21 MB | 击败 78.81%



代码 | Python3

```python
class Solution:
    def partition(self, s: str) -> List[List[str]]:
        n=len(s)

        @cache
        def part(i:int):
            if i==n:return [[]]
            ans=[]
            for j in range(i+1,n+1):
                p=s[i:j]
                if p==p[::-1]:
                    p=[p]
                    ans+=map(lambda x:p+x,part(j))
            return ans

        return part(0)
```

收起

LC146.LRU缓存

hash table, doubly-linked list, https://leetcode.cn/problems/lru-cache/

思路:

使用双向链表按最后被访问的次序存储每一个键值对，用字典存储每个键对应的链表节点，每次更新时将访问的节点移到尾部即可。实际编程时由于忘记修改部分节点的链接情况，导致出错了好几次。约30min。

代码:

```python
class Node:
    def __init__(self,key=0, val=0, pre=None,next=None):
        self.key=key
        self.val = val
```

```
        self.pre = pre
        self.next = next

class LRUCache:

    def __init__(self, capacity: int):
        self.head,self.tail=Node(-1,-1),Node(-1,-1)
        self.tail.pre=self.head
        self.head.next=self.tail
        self.dic={}
        self.cap=capacity

    def visit(self,key:int):
        temp=self.dic[key]
        temp.pre.next=temp.next
        temp.next.pre=temp.pre
        self.tail.pre.next = temp
        temp.pre=self.tail.pre
        self.tail.pre=temp
        temp.next=self.tail

    def get(self, key: int) -> int:
        if key in self.dic:
            self.visit(key)
            return self.dic[key].val
        else:return -1

    def put(self, key: int, value: int) -> None:
        if key in self.dic:
            self.visit(key)
            self.dic[key].val=value
        elif self.cap:
            self.dic[key]=Node(key,value,self.tail.pre,self.tail)
            self.tail.pre.next=self.dic[key]
            self.tail.pre=self.dic[key]
            self.cap-=1
        else:
            self.dic.pop(self.head.next.key)
            self.head=self.head.next
            self.dic[key]=Node(key,value,self.tail.pre,self.tail)
            self.tail.pre.next=self.dic[key]
            self.tail.pre=self.dic[key]
```

代码运行截图 （至少包含有"Accepted"）

通过 23 / 23 个通过的测试用例

Hyperalgebra 提交于 2025.03.23 09:49

官方题解 写题解

面向在校学生的专享特惠

完成认证享 7 折 Plus 会员，享受更多学业及职业成长帮助

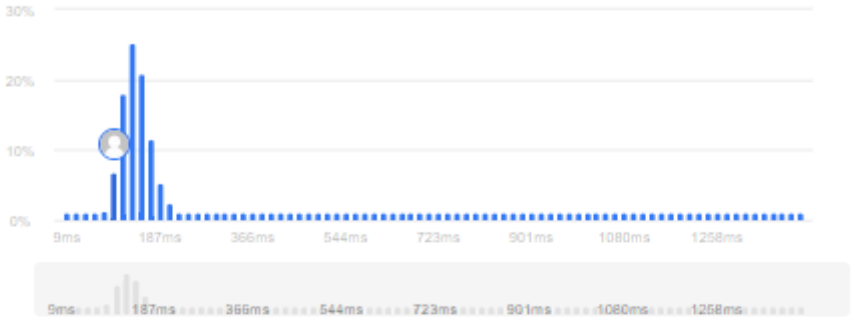
执行用时分布

109 ms | 击败 90.08%

复杂度分析

消耗内存分布

78.70 MB | 击败 5.19%



代码 | Python3

```python
class Node:
    def __init__(self,key=0, val=0, pre=None,next=None):
        self.key=key
        self.val = val
        self.pre = pre
        self.next = next

class LRUCache:

    def __init__(self, capacity: int):
        self.head,self.tail=Node(-1,-1),Node(-1,-1)
        self.tail.pre=self.head
        self.head.next=self.tail
        self.dic={}
        self.cap=capacity

    def visit(self,key:int):
        temp=self.dic[key]
        temp.pre.next=temp.next
        temp.next.pre=temp.pre
        self.tail.pre.next = temp
        temp.pre=self.tail.pre
        self.tail.pre=temp
        temp.next=self.tail

    def get(self, key: int) -> int:
        if key in self.dic:
            self.visit(key)
            return self.dic[key].val
        else:return -1

    def put(self, key: int, value: int) -> None:
        if key in self.dic:
            self.visit(key)
            self.dic[key].val=value
        elif self.cap:
            self.dic[key]=Node(key,value,self.tail.pre,self.tail)
            self.tail.pre.next=self.dic[key]
            self.tail.pre=self.dic[key]
            self.cap-=1
        else:
            self.dic.pop(self.head.next.key)
            self.head=self.head.next
            self.dic[key]=Node(key,value,self.tail.pre,self.tail)
            self.tail.pre.next=self.dic[key]
            self.tail.pre=self.dic[key]
# Your LRUCache object will be instantiated and called as such:
# obj = LRUCache(capacity)
# param_1 = obj.get(key)
# obj.put(key,value)
```

更多挑战

2. 学习总结和收获

如果发现作业题目相对简单，有否寻找额外的练习题目，如“数算2025spring每日选做”、LeetCode、Codeforces、洛谷等网站上的题目。

这次LeetCode周赛题目出得简单了，虽然AK了，但是完成得太慢，一些细节上调了好几回，排名甚至还没有之前比赛高 。最近写每日选做里面的DFS、BFS搜索题也经常在一些小地方出bug，每回都得调试半天，感觉自己的做题熟练度还是有待提高。