

Assignment #A: 递归回溯、 (3/4)

Updated 2203 GMT+8 Nov 3, 2025

2025 fall, Complied by 同学的姓名、院系

1.1. 题目

1.1 T51.N皇后

backtracking, <https://leetcode.cn/problems/n-queens/>

思路：

- dfs

代码：

```
1 class Solution:
2     def solveNQueens(self, n: int) -> List[List[str]]:
3
4
5         def dfs(deep,n,res,huanghou):
6             if deep==n:
7                 res.append(huanghou[:])
8                 return
9             for i in range(n):
10                 if i in huanghou:
11                     continue
12                 judge=0
13                 for j in range(deep):
14                     dx=deep-j
15                     dy=i-huanghou[j]
16                     if dx+dy==0 or dx==dy:
17                         judge=1
18                         break
19                 if judge==1:
20                     continue
21                 huanghou.append(i)
22                 dfs(deep+1,n,res,huanghou)
23                 huanghou.pop()
24
25         res=[]
26         huanghou=[]
27         dfs(0,n,res,huanghou)
28         ans=[]
29         for i in range(len(res)):
30             qipan=[]
31             for j in range(n):
32                 row='.'*n
33                 row[res[i][j]]='Q'
34                 astr=''.join(row)
35                 qipan.append(astr)
36         return qipan
```

```

34         qipan.append(astr)
35         ans.append(qipan)
36
37
38     return ans

```

Fence 1

代码运行截图 (至少包含有"Accepted")

通过 9 / 9 个通过的测试用例

AND-Y 提交于 2025.08.31 16:47

官方题解

写题解

① 执行用时分布

15 ms | 击败 50.95%

复杂度分析

②

消耗内存分布

17.89 MB | 击败 75.19%

20%

Figure 1

1.2 M22275: 二叉搜索树的遍历

<http://cs101.openjudge.cn/practice/22275/>

思路：

- 可以直接给前序序列排序得到中序序列，再构造BST
- 或者利用stack和BST的有序性来构造BST

代码：

```

1 class Treenode:
2     def __init__(self, val, left=None, right=None):
3         self.val = val
4         self.left = left
5         self.right = right
6
7     def post_order(root):
8         if root:
9             post_order(root.left)
10            post_order(root.right)
11            res.append(root.val)
12
13
14
15
16 n=int(input())
17 preorder_list=list(map(int,input().split()))
18 root = Treenode(preorder_list[0])

```

```

19  stack=[root]
20  pr=root
21  for i in preorder_list[1:]:
22      if pr.val>i:
23          pr.left=Treenode(i)
24          stack.append(pr.left)
25          pr=pr.left
26      else:
27          while len(stack)>1 and stack[-2].val<i:
28              stack.pop()
29              pr=stack.pop()
30              pr.right=Treenode(i)
31              stack.append(pr.right)
32              pr=pr.right
33
34  res=[]
35  post_order(root)
36  print(*res)

```

Fence 2

代码运行截图 (至少包含有"Accepted")

#50763294提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```

class Treenode:
    def __init__(self, val, left=None, right=None):
        self.val=val
        self.left=left
        self.right=right

def post_order(root):

```

基本信息

#: 50763294
题目: 22275
提交人: 25n2400011769
内存: 4012kB
时间: 25ms
语言: Python3
提交时间: 2025-11-09 11:31:50

Figure 2

1.3 M25145: 猜二叉树 (按层次遍历)

<http://cs101.openjudge.cn/practice/25145/>

思路:

- 略

代码:

```

1  from collections import deque
2  class TreeNode:
3      def __init__(self, val, left=None, right=None):
4          self.val=val
5          self.left=left
6          self.right=right
7
8  def build_tree(begin, end):

```

```

9     if begin>end:
10        return None
11    root=TreeNode(post_order.pop())
12    for index in range(begin,end+1):
13        if in_order[index]==root.val:
14            break
15    root.right=build_tree(index+1,end)
16    root.left=build_tree(begin,index-1)
17    return root
18
19 def cengxu(root):
20     alist=deque([root])
21     res=[]
22     while alist:
23         t=alist.popleft()
24         res.append(t.val)
25         if t.left:
26             alist.append(t.left)
27         if t.right:
28             alist.append(t.right)
29     return res
30
31 n=int(input())
32 for i in range(n):
33     in_order=list(input())
34     post_order=list(input())
35     root=build_tree(0,len(in_order)-1)
36     res=cengxu(root)
37     print(''.join(res))

```

Fence 3

代码运行截图 (至少包含有"Accepted")

#50763523提交状态

状态: Accepted 源代码	基本信息 #: 50763523 题目: 25145 提交人: 25n2400011769 内存: 3652kB 时间: 23ms 语言: Python3 提交时间: 2025-11-09 11:42:39
----------------------------	---

```

from collections import deque
class TreeNode:
    def __init__(self, val, left=None, right=None):
        self.val=val
        self.left=left
        self.right=right

def build_tree(begin end):

```

Figure 3

1.4 T20576: printExp (逆波兰表达式建树)

<http://cs101.openjudge.cn/practice/20576/>

思路：

- 中序转后序
- 后序构造树
- 运算等级确定是否填括号

代码

```

1  class TreeNode:
2      def __init__(self, x=None):
3          self.val = x
4          self.left = None
5          self.right = None
6
7      def build_postorder():
8          str_stack=[]
9          postlist=[]
10         for i in data:
11             if i=='(':
12                 str_stack.append(i)
13             elif i==')':
14                 t=str_stack.pop()
15                 while t!='(':
16                     postlist.append(t)
17                     t=str_stack.pop()
18             elif i=='True' or i=='False':
19                 postlist.append(i)
20             else:
21                 while str_stack and dic[str_stack[-1]]>=dic[i]:
22                     t=str_stack.pop()
23                     postlist.append(t)
24                     str_stack.append(i)
25         while str_stack:
26             postlist.append(str_stack.pop())
27         return postlist
28
29     def build_tree():
30         postlist=build_postorder()
31         stack=[]
32         for val in postlist:
33             pr=TreeNode(val)
34             if val=='True' or val=='False':
35                 stack.append(pr)
36             elif val=='not':
37                 pr.left=stack.pop()
38                 stack.append(pr)
39             else:

```

```

40         pr.right=stack.pop()
41         pr.left=stack.pop()
42         stack.append(pr)
43     return pr
44
45 def build_res(root):
46     if dic[root.val]==-1:
47         return [root.val]
48     elif root.val=='not':
49         if dic[root.left.val]==-1:
50             return ['not']+build_res(root.left)
51         else:
52             return ['not','(']+build_res(root.left)+[')']
53     elif root.val=='and':
54         if dic[root.left.val]==1:
55             left=['(']+build_res(root.left)+[')']
56         else:
57             left=build_res(root.left)
58         if dic[root.right.val]==1:
59             right=['(']+build_res(root.right)+[')']
60         else:
61             right=build_res(root.right)
62         return left+['and']+right
63     else:
64         return build_res(root.left)+[root.val]+build_res(root.right)
65
66
67 data=list(input().split())
68 dic={'not':3,'and':2,'or':1,'(':0,'True':-1,'False':-1}
69
70 root=build_tree()
71 res=build_res(root)
72 print(*res)

```

Fence 4

代码运行截图 (至少包含有"Accepted")

#50736572提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

状态: Accepted

源代码

```

class TreeNode:
    def __init__(self, x=None):
        self.val = x
        self.left = None
        self.right = None

def build_postorder():
    str stack=[]

```

基本信息

#: 50736572
 题目: 20576
 提交人: 25n2400011769
 内存: 3724kB
 时间: 22ms
 语言: Python3
 提交时间: 2025-11-07 13:04:31

Figure 4

1.5 T04080:Huffman编码树

greedy, <http://cs101.openjudge.cn/practice/04080/>

思路：

- 由下至上构造二叉树，再递归求路径和

代码

```

1 import heapq
2 import itertools
3 class TreeNode:
4     def __init__(self, x, left=None, right=None):
5         self.val = x
6         self.left = left
7         self.right = right
8
9     def buildTree():
10        heap = []
11        count=itertools.count()
12        for i in num_list:
13            pr = TreeNode(i)
14            heapq.heappush(heap, (i, next(count),pr))
15        while len(heap) > 1:
16            left_val , cnt , left = heapq.heappop(heap)
17            right_val , cnt , right = heapq.heappop(heap)
18            pr = TreeNode(left_val + right_val, left, right)
19            heapq.heappush(heap, (pr.val, next(count),pr))
20        return heap[0][2]
21
22    def cal_min_length(root,deep):
23        if not root.left and not root.right:
24            return deep*root.val
25        return cal_min_length(root.left,deep+1) +
26        cal_min_length(root.right,deep+1)
27
28    n=int(input())
29    num_list=list(map(int,input().split()))
30    root=buildTree()
31    print(cal_min_length(root,0))

```

Fence 5

代码运行截图 (至少包含有"Accepted")

#50833013提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```

import heapq
import itertools
class TreeNode:
    def __init__(self, x, left=None, right=None):
        self.val = x
        self.left = left
        self.right = right

```

基本信息

#: 50833013
题目: 04080
提交人: 25n2400011769
内存: 3688kB
时间: 21ms
语言: Python3
提交时间: 2025-11-14 10:57:46

Figure 5

1.6 M04078: 实现堆结构

<http://cs101.openjudge.cn/practice/04078/>

要求手搓堆实现。

思路:

- 略

代码:

```

1 class Heap:
2     def __init__(self):
3         self.heap=[]
4     def push(self,val):
5         self.heap.append(val)
6         self.up_balance(len(self.heap)-1)
7     def pop(self):
8         self.heap[0],self.heap[-1]=self.heap[-1],self.heap[0]
9         t=self.heap.pop()
10        self.down_balance(0)
11        return t
12    def up_balance(self,pr):
13        if pr==0:
14            return
15        parent_index=(pr-1)//2
16        if self.heap[parent_index]>self.heap[pr]:
17
18            self.heap[parent_index],self.heap[pr]=self.heap[pr],self.heap[parent_index]
19            self.up_balance(parent_index)
20        else:
21            return
22    def down_balance(self,pr):
23        left=pr*2+1
24        right=pr*2+2
25        if left>len(self.heap)-1:
26            return
27        elif left==len(self.heap)-1:
28            if self.heap[left]<self.heap[pr]:

```

```

28
29         self.heap[left], self.heap[pr]=self.heap[pr], self.heap[left]
30             return
31     else:
32         if self.heap[pr]<min(self.heap[right],self.heap[left]):
33             return
34         else:
35             if self.heap[left]<self.heap[right]:
36                 self.heap[left], self.heap[pr]=self.heap[pr], self.heap[left]
37                 self.down_balance(left)
38             else:
39                 self.heap[right], self.heap[pr]=self.heap[pr], self.heap[right]
40             self.down_balance(right)
41 n=int(input().strip())
42 heap=Heap()
43 for i in range(n):
44     l=tuple(map(int,input().strip().split()))
45     if l[0]==1:
46         heap.push(l[1])
47     elif l[0]==2:
48         print(heap.pop())

```

Fence 6

代码运行截图 (至少包含有"Accepted")

#50830059提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```

class Heap:
    def __init__(self):
        self.heap=[]
    def push(self,val):
        self.heap.append(val)
        self.up_balance(len(self.heap)-1)
    def pop(self):
        self.heap[0], self.heap[-1]=self.heap[-1], self.heap[0]

```

基本信息

#: 50830059
 题目: 04078
 提交人: 25n2400011769
 内存: 4240kB
 时间: 761ms
 语言: Python3
 提交时间: 2025-11-13 20:59:19

Figure 6

2. 2. 学习总结和个人收获

本周继续练习了一些较为复杂的树的题目。在T04080:Huffman编码树中使用最小堆时，学习了用`itertools.count()`和`next()`实现相同元素的比较。在另一个题目22161:哈夫曼编码树，学习了用`__lt__`方法来构造可比较对象，使得可以直接把树的节点放入最小堆中。