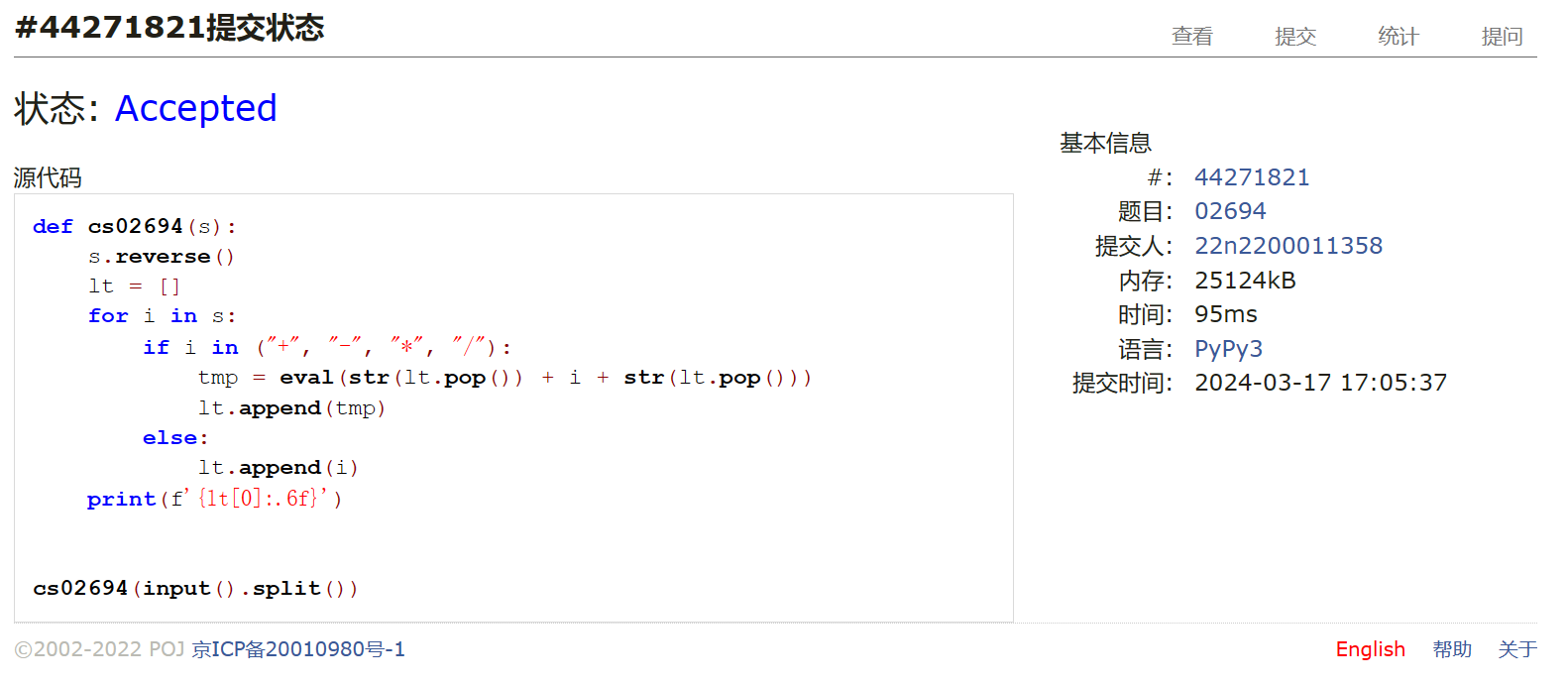
**05902:双端队列**

def cs05902(t):  
 for i in range(t):  
 lt = []  
 for j in range(int(input())):  
 k1, k2 = map(int, input().split())  
 if k1 == 1:  
 lt.append(k2)  
 else:  
 lt.pop(-1 \* k2)  
 if lt:  
 for j in lt[:-1]:  
 print(j, end=" ")  
 print(lt[-1])  
 else:  
 print("NULL")

## 02694:波兰表达式

def cs02694(s):  
 s.reverse()  
 lt = []  
 for i in s:  
 if i in ("+", "-", "\*", "/"):  
 tmp = eval(str(lt.pop()) + i + str(lt.pop()))  
 lt.append(tmp)  
 else:  
 lt.append(i)  
 print(f'{lt[0]:.6f}')

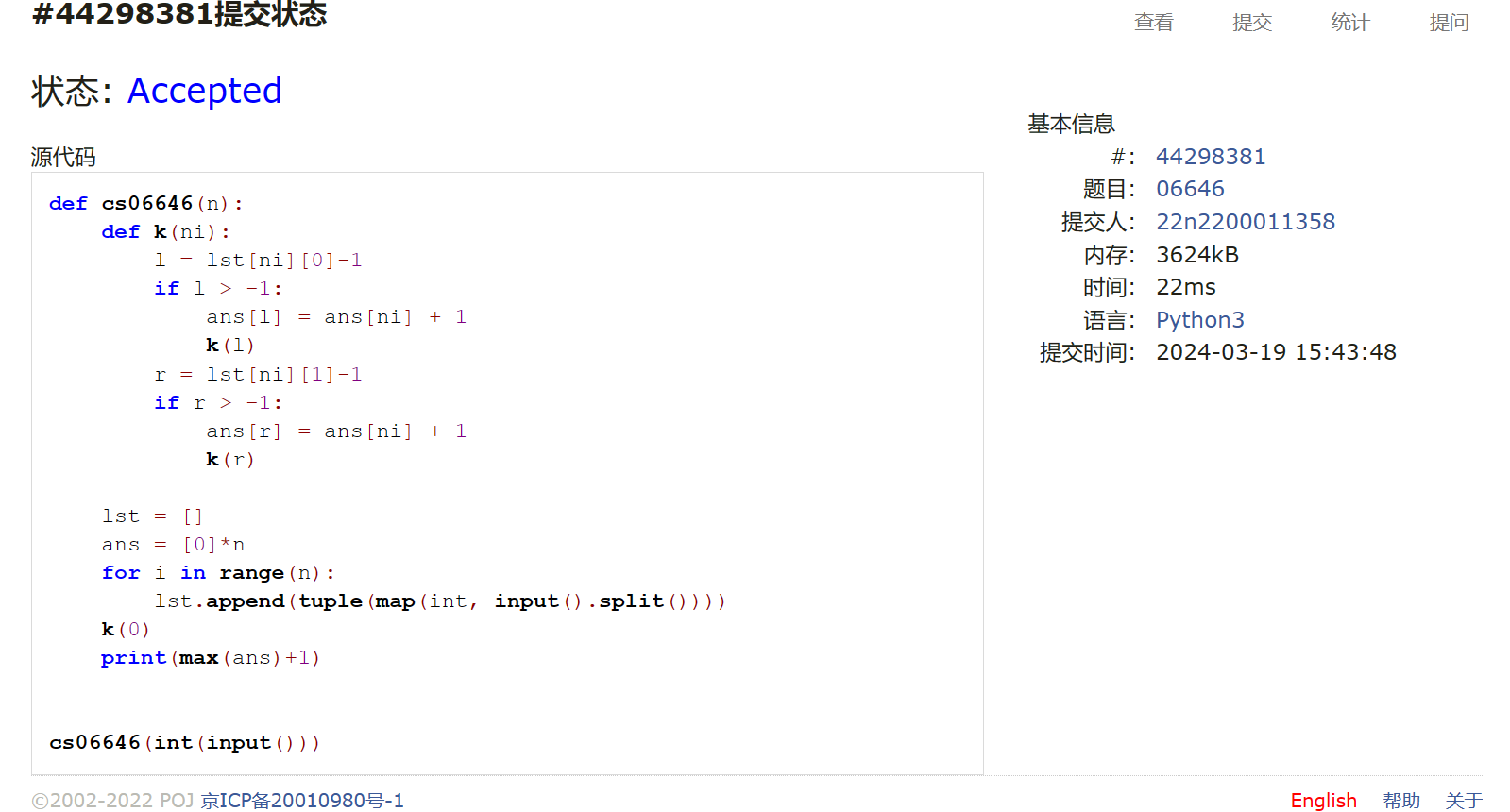
## 24591:中序表达式转后序表达式

def cs24591(n):  
 for i in range(n):  
 s = input()  
 tmp = ""  
 lt, ans = [], []  
 for j in range(len(s)):  
 if s[j] in ("+", "-", "\*", "/", ")", "("):  
 if tmp:  
 ans.append(tmp)  
 tmp = ""  
 else:  
 tmp += s[j]  
 if s[j] == "(":  
 lt.append(s[j])  
 elif s[j] in ("+", "-", "\*", "/"):  
 while lt and not (s[j] in ("\*", "/") and lt[-1] in ("+", "-") or lt[-1] == "("):  
 ans.append(lt.pop())  
 lt.append(s[j])  
 elif s[j] == ")":  
 a = lt.pop()  
 while a != "(":  
 ans.append(a)  
 a = lt.pop()  
 if ans:  
 for j in ans[:-1]:  
 print(j, end=" ")  
 if lt:  
 print(ans[-1], end=" ")  
 if tmp:  
 print(tmp, end=" ")  
 while len(lt) > 1:  
 print(lt.pop(), end=" ")  
 print(lt[0])  
 elif tmp:  
 print(ans[-1], tmp)  
 else:  
 print(ans[-1])   
 else:   
 print(tmp)

## 22068:合法出栈序列

def cs22068(x):  
 try:  
 while True:  
 l = len(x)  
 st = []  
 it = input()  
 if l != len(it):  
 print("NO")  
 else:  
 j, k = 0, 0  
 for i in range(l \* 2):  
 if st == [] or st[-1] != it[j]:  
 if k > l - 1:  
 st = True  
 break  
 st.append(x[k])  
 k += 1  
 else:  
 st.pop()  
 j += 1  
 if j > l - 1:  
 break  
 if st:  
 print("NO")  
 else:  
 print("YES")  
  
 except:  
 exit()

## 06646:二叉树的深度

def cs06646(n):  
 def k(ni):  
 l = lst[ni][0] - 1  
 if l > -1:  
 ans[l] = ans[ni] + 1  
 k(l)  
 r = lst[ni][1] - 1  
 if r > -1:  
 ans[r] = ans[ni] + 1  
 k(r)  
  
 lst = []  
 ans = [0] \* n  
 for i in range(n):  
 lst.append(tuple(map(int, input().split())))  
 k(0)  
 print(max(ans) + 1)

## 02299:Ultra-QuickSort

def cs02299():  
 def merge\_sort(a):  
 if len(a) == 1:  
 return a, 0  
 else:  
 l = len(a) // 2  
 rt, ans = [], 0  
 left, la = merge\_sort(a[:l])  
 right, ra = merge\_sort(a[l:])  
 ans += la + ra  
 left.reverse()  
 right.reverse()  
 for il in range(l):  
 while right and left[-1] > right[-1]:  
 rt.append(right.pop())  
 ans += len(left)  
 else:  
 rt.append(left.pop())  
 for il in range(len(right)):  
 rt.append(right.pop())  
 return rt, ans  
  
 while True:  
 n = int(input())  
 lt = []  
 if n:  
 for i in range(n):  
 lt.append(int(input()))  
 print(merge\_sort(lt)[1])  
 else:  
 break

总结：难度明显提升，归并排序和合法出栈是看了题解思路才会写，程序长了BUG也多；

在做的过程中学了eval等内置函数，学了归并排序算法，递归是用得越来越熟了。