排序、栈、队列

逆波兰表达式求值

```
stack=[]
for t in s:
    if t in '+-*/':
        b,a=stack.pop(),stack.pop()
        stack.append(str(eval(a+t+b)))
    else:
        stack.append(t)
print(f'{float(stack[0]):.6f}')
```

中序表达式转后序表达式

```
pre={'+':1,'-':1,'*':2,'/':2}
for _ in range(int(input())):
    expr=input()
    ans=[]; ops=[]
    for char in expr:
        if char.isdigit() or char=='.':
            ans.append(char)
        elif char=='(':
            ops.append(char)
        elif char==')':
            while ops and ops[-1]!='(':
                ans.append(ops.pop())
            ops.pop()
        else:
            while ops and ops[-1]!='(' and pre[ops[-1]]>=pre[char]:
                ans.append(ops.pop())
            ops.append(char)
    while ops:
        ans.append(ops.pop())
    print(''.join(ans))
```

stack

```python
s = input()
stack = []
for c in s:
    if c == "U":
        if len(stack) >= 2 and stack[-1] == "K" and stack[-2] == "P":
            stack.pop()
            stack.pop()
        else:
            stack.append(c)
    else:
        stack.append(c)
print("".join(stack))
```

```python
def check_brackets(s):
    stack = []
    nested = False
    pairs = {')': '(', ']': '[', '}': '{'}
    for ch in s:
        if ch in pairs.values():
            stack.append(ch)
        elif ch in pairs.keys():
            if not stack or stack.pop() != pairs[ch]:
                return "ERROR"
            if stack:
                nested = True
    if stack:
        return "ERROR"
    return "YES" if nested else "NO"

s = input()
print(check_brackets(s))
```

嵌套

```python
def max_cheese_score(n, reward1, reward2, k):
    # 计算每块奶酪的得分差值
    differences = [(reward1[i] - reward2[i], i) for i in range(n)]

    # 按得分差值从大到小排序
    differences.sort(reverse=True, key=lambda x: x[0])

    # 初始化总得分
    total_score = 0

    # 选择前k块奶酪给第一只老鼠
    for i in range(k):
        index = differences[i][1]
        total_score += reward1[index]

    # 选择剩余的奶酪给第二只老鼠
    for i in range(k, n):
        index = differences[i][1]
        total_score += reward2[index]

    return total_score

# 输入处理
n = int(input())
reward1 = list(map(int, input().split()))
reward2 = list(map(int, input().split()))
k = int(input())

# 计算并输出结果
print(max_cheese_score(n, reward1, reward2, k))
```

贪心

5、18182:打怪兽（2h50min）

```python
from collections import deque

t = int(input())
for _ in range(t):
    n = int(input())
    a = deque(map(int, input().split()))
    b = deque(map(int, input().split()))

    cnt = 0
    i = 0
    while i < n:
        if a[i] < b[i]:
            cnt += 1
            a.popleft()
            a.append(b[-1] + 1)
            i = 0
            continue

        i += 1

    print(cnt)
```

队列

```
xbag={}
ncase=int(input())
for i in range(ncase):
    xbag[i]={}
    n,m,b = map(int,input().split())
    xbag[i][0]=(m,b)
    for ii in range(n):
        t,x = map(int,input().split())
        if t not in xbag[i].keys():
            xbag[i][t]=[x]
        else:
            xbag[i][t].append(x)
            xbag[i][t]=sorted(xbag[i][t], reverse=True)
for i in range(ncase):
    sorted_key=sorted(xbag[i].keys())
    m,b=xbag[i][0]
    for k in sorted_key[1:]:
        b-=sum(xbag[i][k][:m])
        if b<=0:
            print(k)
            break
    if b>0:
        print("alive")
```

1、05902:双端队列（1h30min）

```
from collections import deque
for _ in range(int(input())):
    n=int(input())
    q=deque([])
    for i in range(n):
        a, b=map(int,input().split())
        if a==1:
            q.append(b)
        else:
            if b==0:
                q.popleft()
            else:
                q.pop()
    if q:
        print(*q)
    else:
        print('NULL')
```

双端队列

2、02694:波兰表达式（3h）

```
expression = input().split()
stack = []
while expression:
    a = expression.pop(-1)
    if a in ['+', '-', '*', '/']:
        c = stack.pop(-1)
        d = stack.pop(-1)
        if a == '+':
            stack.append(c + d)
        elif a == '-':
            stack.append(c - d)
        elif a == '*':
            stack.append(c * d)
        else:
            stack.append(c / d)
    else:
        stack.append(float(a))

print("{:.6f}".format(stack[0]))
```

求逆序对数

```
from bisect import *
a=[]
rev=0
for _ in range(n):
    num=int(input())
    rev+=bisect_left(a,num)
    insort_left(a,num)
ans=n*(n-1)//2-rev
```

```
def merge_sort(a):
    if len(a)<=1:
        return a,0
    mid=len(a)//2
    l,l_cnt=merge_sort(a[:mid])
    r,r_cnt=merge_sort(a[mid:])
    merged,merge_cnt=merge(l,r)
    return merged,l_cnt+r_cnt+merge_cnt
def merge(l,r):
    merged=[]
    l_idx,r_idx=0,0
    inverse_cnt=0
    while l_idx<len(l) and r_idx<len(r):
        if l[l_idx]<=r[r_idx]:
            merged.append(l[l_idx])
            l_idx+=1
        else:
            merged.append(r[r_idx])
            r_idx+=1
            inverse_cnt+=len(l)-l_idx
    merged.extend(l[l_idx:])
    merged.extend(r[r_idx:])
    return merged,inverse_cnt
```

4、22068:合法出栈序列（2h）

```
origin = input()
while True:
    try:
        outout = input()
        stack, bank = [], list(origin)
        l = len(origin)
        flag = False
        if len(outout) == l:
            for i in range(l):
                if bank and not stack:
                    stack.append(bank.pop(0))
                while bank and stack[-1] != outout[i]:
                    stack.append(bank.pop(0))
                if stack.pop() != outout[i]:
                    print('NO')
                    flag = True
                    break
            if not flag:
                print('YES')
        else:
            print('NO')
    except EOFError:
        break
```

最大全0子矩阵

```
for row in ma:
    stack=[]
    for i in range(n):
        h[i]=h[i]+1 if row[i]==0 else 0
        while stack and h[stack[-1]]>h[i]:
            y=h[stack.pop()]
            w=i if not stack else i-stack[-1]-1
            ans=max(ans,y*w)
        stack.append(i)
    while stack:
        y=h[stack.pop()]
        w=n if not stack else n-stack[-1]-1
        ans=max(ans,y*w)
print(ans)
```

层次遍历

```
from collections import deque
def levelorder(root):
    if not root:
        return ""
    q=deque([root])
    res=""
    while q:
        node=q.popleft()
        res+=node.val
        if node.left:
            q.append(node.left)
        if node.right:
            q.append(node.right)
    return res
```

并查集

```
class disj_set:
    def __init__(self,n):
        self.rank = [1 for i in range(n)]
        self.parent = [i for i in range(n)]

    def find(self,x):
        if self.parent[x] != x:
            self.parent[x] = self.find(self.parent[x])
        return self.parent[x]


    def union(self,x,y):
        x_root = self.find(x)
        y_root = self.find(y)

        if x_root == y_root:
            return

        if self.rank[x_root] > self.rank[y_root]:
            self.parent[y_root] = x_root
        elif self.rank[y_root] < self.rank[x_root]:
            self.parent[x_root] = y_root
        else:
            self.parent[y_root] = x_root
            self.rank[x_root] += 1


##计算父亲节点个数
count = 0
for x in range(1,n+1):
    if D.parent[x-1] == x - 1:
        count += 1
```

最大上升子序列

```python
input()
b = [int(x) for x in input().split()]

n = len(b)
dp = [0]*n

for i in range(n):
    dp[i] = b[i]
    for j in range(i):
        if b[j]<b[i]:
            dp[i] = max(dp[j]+b[i], dp[i])

print(max(dp))
```