常用代码总结

-----------------数据结构与算法----------------

【栈】

List.append()

List.pop()

【队列】

from collections import deque

queue = deque()

入队： `queue.append()`

出队： `v = queue.popleft()`

【二分查找】

def bsearch(nums, target):

left = 0

right = len(nums) - 1

while left <= right:

mid = (right + left) // 2

if nums[mid] == target:

return mid

elif nums[mid] < target:

left = mid + 1

else:

right = mid - 1

return -1 – left

【快速排序】

def qsort(nums: [int]) -> [int]:

def recursion(begin, end):

if end - begin <= 1:

return

mid = (begin + end) // 2

pivot = nums[mid]

nums[begin], nums[mid] = nums[mid], nums[begin]

i = begin + 1

j = end - 1

while i <= j:

while i < end and nums[i] < pivot:

i += 1

while j >= begin + 1 and nums[j] >= pivot:

j -= 1

if i < j:

nums[i], nums[j] = nums[j], nums[i]

nums[j], nums[begin] = nums[begin], nums[j]

recursion(begin, j)

recursion(j + 1, end)

recursion(0, len(nums))

return nums

//另一个版本快排

def qsort(nums: [int]) -> [int]:

if len(nums)==0 or len(nums)==1:

return nums

else:

qsortHelper(nums,0,len(nums)-1)

return nums

def qsortHelper(nums,first,last):

if first<last:

splitpoint=partition(nums,first,last)

qsortHelper(nums,first,splitpoint-1)

qsortHelper(nums,splitpoint+1,last)

def partition(nums,first,last):

median=(first+last+1)//2

if nums[median]<nums[first]:

temp=nums[median]

nums[median]=nums[first]

nums[first]=temp

if nums[first]>nums[last]:

temp=nums[last]

nums[last]=nums[first]

nums[first]=temp

if nums[median]>nums[last]:

temp=nums[last]

nums[last]=nums[first+1]

nums[first+1]=temp

temp=nums[first+1]

nums[first+1]=nums[median]

nums[median]=temp

pivotvalue=nums[first+1]

leftmark=first+2

rightmark=last

done=False

while not done:

while leftmark<=rightmark and nums[leftmark]<=pivotvalue:

leftmark+=1

while rightmark>=leftmark and nums[rightmark]>=pivotvalue:

rightmark-=1

if rightmark<leftmark:

done=True

else:

temp=nums[leftmark]

nums[leftmark]=nums[rightmark]

nums[rightmark]=temp

nums[first+1]=nums[rightmark]

nums[rightmark]=pivotvalue

return rightmark

【归并排序】

def merge\_sort(data\_list):

if len(data\_list)<=1:

return data\_list

middle=int(len(data\_list)/2)

left=merge\_sort(data\_list[:middle])

right=merge\_sort(data\_list[middle:])

merged=[]

while left and right:

merged.append(left.pop(0) if left[0]<=right[0] else right.pop(0))

merged.extend(right if right else left)

return merged

data\_list=[random.randint(1,100) for \_ in range(50)]

print(merge\_sort(data\_list))

【桶排序】

def radixsort(sortlist):

l=[]

main=sortlist

j=0

for i in range(10):

l.append(Queue())

remain=[]

while len(remain)<len(main):

remain=[]

for i in main:

if i>=(10\*\*j):

l[i//(10\*\*j)%10].enqueue(i)

else:

remain.append(i)

main=remain[:]

for i in range(10):

while not l[i].isEmpty():

main.append(l[i].dequeue())

j+=1

return [str(x) for x in main]

print(' '.join(radixsort([int(x) for x in input().split()]

【中缀转后缀】

prec = {"\*":3,"/":3,"+":2,"-":2}

def infixToPostfix(infixexpr):

opStack = Stack()

postfixList = []

tokenList = infixexpr.split()

for token in tokenList:

if token in "ABCDEFGHIJKLMNOPQRSTUVWXYZ" \

or token in "0123456789": #操作数的处理

postfixList.append(token)

elif token == '(': #标记子表达式开始

opStack.push(token)

elif token == ')': #子表达式结束

while opStack.peek() != '(':

postfixList.append(opStack.pop())

else:

opStack.pop() #弹出'('

else: #操作符

while (not opStack.isEmpty()

and opStack.peek() != '('

and prec[opStack.peek()] >= prec[token]):

postfixList.append(opStack.pop())

opStack.push(token) #所有操作符都必须进栈等待

while not opStack.isEmpty():

postfixList.append(opStack.pop())

return " ".join(postfixList)

print(infixToPostfix("A + B \* 5"))

print(infixToPostfix("( A + B ) \* 5"))

【中缀转前缀】

def infixToPrefix(infixexpr):

prec={'\*':3,'/':3,'+':2,'-':2,')':1}

opStack=Stack()

postfixList=[]

tokenList=infixexpr.split()

retokenList=reversed(tokenList)

for token in retokenList:

if token in 'ABCDEFGHIJKLMNOPQRSTUVWXYZ' or token in '0123456789':

postfixList.append(token)

elif token==')':

opStack.push(token)

elif token=='(':

topToken=opStack.pop()

while topToken!=')':

postfixList.append(topToken)

topToken=opStack.pop()

else:

while (not opStack.isEmpty()) and (prec[opStack.peek()]>prec[token]):

postfixList.append(opStack.pop())

opStack.push(token)

while not opStack.isEmpty():

postfixList.append(opStack.pop())

return ' '.join(reversed(postfixList))

print(infixToPrefix(input()))

【热土豆（约瑟夫）】

﻿def hotPotato(namelist, num):

que =Queue()

for name in namelist:

que.enqueue(name)

while que.size() > 1:

for i in range(num-1):

que.enqueue(que.dequeue())

print(que.dequeue()) #杀掉一个

return que.dequeue()

----------------------输入--------------------

【矩阵输入】

﻿rowA,colA=map(int,input().split())

A=[[int(x) for x in input().split()] for i in range(rowA)]

A=[[0]\*n for i in range(n)]

【矩阵输入（加保护圈）】

﻿n,m=map(int,input().split())

board=[[0]\*(m+2)]+[[0]+[int(x) for x in input().split()]+[0] for j in range(n)]+[[0]\*(m+2)]

【矩阵输出】

﻿for y in range(n):

print(' '.join([str(x) for x in output[y]]))

【矩阵提取个别行列】

D=[A[r][j:j+q] for r in range(i,i+p)]

【小数格式输出】

﻿print('{:.2f}'.format(x))

【定义四周的函数】

dx=[0,0,1,-1]

dy=[1,-1,0,0]

﻿dx=[0,0,1,1,1,-1,-1,-1]

dy=[1,-1,1,0,-1,1,0,-1]

﻿def turn(x,y):

for i in range(len(dx)):

if board[x+dx[i]][y+dy[i]]==0:

board[x+dx[i]][y+dy[i]]=1

elif board[x+dx[i]][y+dy[i]]==1:

board[x+dx[i]][y+dy[i]]=0

return

【异常细胞】

﻿N=int(input())

A=[[int(x) for x in input().split()] for i in range(N)]

dx=[0,0,1,-1]

dy=[1,-1,0,0]

def abnormal(x,y):

count=0

for i in range(4):

if A[x+dx[i]][y+dy[i]]>=A[x][y]+50:

count+=1

else:

return False

if count==4:

return True

ans=0

for i in range(1,N-1):

for j in range(1,N-1):

if abnormal(i,j):

ans+=1

print(ans)

【捕获结束】

﻿l=input()

while l!='0 0 0 0 0 0':

l=input()

【不知道何时结束程序】

﻿while True:

try:

n=int(input())

except EOFError:

break

【排序】

l.sort(key=lambda x:(x[0],-x[1]))

l.sort(key=lambda x: -x[0]/x[1])

【深拷贝】

﻿import copy

﻿board=copy.deepcopy(mat)

【字典】

﻿按照值大小返回键：d\_key = sorted(d, key=lambda k: d[k])

d\_key = max (d, key=lambda k: d[k])

指定值返回键：list(s.keys())[list(s.values()).index(value)]

Zip实现值和键的翻转，注意内容只能使用一次：zip(prices.values(), prices.keys())

【矩阵】

【初始化】

slots=[[] for \_ in len(self.table\_size)]

【矩阵】上机考试

﻿def same(l,board,x,y):

if l[board[x][y]]==l[board[x-1][y]] or l[board[x][y]]==l[board[x][y-1]] or l[board[x][y]]==l[board[x][y+1]] \

or l[board[x][y]]==l[board[x+1][y]]:

return 1

else:

return 0

m,n=map(int,input().split())

board=[[-1]\*(n+2)]+[[-1]+[int(x) for x in input().split()]+[-1] for j in range(m)]+[[-1]\*(n+2)]

l=[]

score=[]

for i in range(m\*n):

l1=[int(x) for x in input().split()]

l.append(l1)

score.append(sum(l1))

l.append([])

s=0

for i in range(m):

for j in range(n):

s+=same(l,board,i+1,j+1)

score.sort(reverse=True)

t=0

for i in range(m\*n\*2//5):

if score[i]>score[m\*n\*2//5]:

t+=1

print(s,t,sep=' ')

【矩阵】图的拉普拉斯

﻿n,m=map(int,input().split())

D=[[0]\*n for i in range(n)]

A=[[0]\*n for i in range(n)]

L=[[0]\*n for i in range(n)]

for i in range(m):

a,b=map(int,input().split())

D[a][a]+=1

D[b][b]+=1

A[a][b]=A[b][a]=1

for i in range(n):

for j in range(n):

L[i][j]=D[i][j]-A[i][j]

for i in range(n):

print(' '.join([str(x) for x in L[i]]))

【矩阵】卷积

﻿n,m=map(int,input().split())

D=[[0]\*n for i in range(n)]

A=[[0]\*n for i in range(n)]

L=[[0]\*n for i in range(n)]

for i in range(m):

a,b=map(int,input().split())

D[a][a]+=1

D[b][b]+=1

A[a][b]=A[b][a]=1

for i in range(n):

for j in range(n):

L[i][j]=D[i][j]-A[i][j]

for i in range(n):

print(' '.join([str(x) for x in L[i]]))

【螺旋矩阵】

﻿n=int(input())

l=[[-1]\*(n+2)]+[[-1]+[0]\*n+[-1] for j in range(n)]+[[-1]\*(n+2)]

turn=[[0,1],[1,0],[0,-1],[-1,0]]

i=1

j=1

s=1

t=0

while s<=n\*\*2:

l[i][j]=s

s+=1

if l[i+turn[t][0]][j+turn[t][1]]!=0:

t=(t+1)%4

i+=turn[t][0]

j+=turn[t][1]

for i in range(1,n+1):

print(' '.join([str(x) for x in l[i][1:n+1]]))

【线性代数运算】

﻿rowA,colA=map(int,input().split())

A=[[int(x) for x in input().split()] for i in range(rowA)]

rowB,colB=map(int,input().split())

B=[[int(x) for x in input().split()] for i in range(rowB)]

rowC,colC=map(int,input().split())

C=[[int(x) for x in input().split()] for i in range(rowC)]

if colA!=rowB or rowA!=rowC or colB!=colC:

print('Error!')

else:

ans=[[0]\*colB for i in range(rowA)]

for i in range(rowA):

for j in range(colB):

for k in range(colA):

ans[i][j]+=A[i][k]\*B[k][j]

ans[i][j]+=C[i][j]

for i in range(rowC):

print(' '.join([str(x) for x in ans[i]]))

Greedy

贪心算法很多的思路在于要先排序！

【装箱问题】greedy

﻿l=input()

while l!='0 0 0 0 0 0':

a,b,c,d,e,f=map(int,l.split())

d1={0:0,1:5,2:3,3:1}

s=d+e+f-(-c)//4

b1=max(0,b-5\*d-d1[c%4])

s=s-(-b1)//9-min((-4\*b-9\*c-16\*d-25\*e-36\*f+36\*(s-(-b1)//9)-a),0)//36

print(s)

l=input()

【greedy】打怪兽-注意边界处理！！！

﻿cases=int(input())

for i in range(cases):

n,m,b=map(int,input().split())

l=[]

for j in range(n):

l.append([int(x) for x in input().split()])

l.sort(key=lambda x:(x[0],-x[1]))

k=1

b-=l[0][1]

for j in range(1,n):

if b<=0:

break

if l[j][0]==l[j-1][0]:

k+=1

else:

k=1

if k<=m:

b-=l[j][1]

if j==n-1 and b<=0:

j+=1

print(l[j-1][0] if b<=0 else 'alive')

【greedy】充实的寒假生活

﻿n=int(input())

l=[]

for i in range(n):

l.append([int(x) for x in input().split()])

l.sort(key=lambda x:x[1])

s=0

if l[0][1]<=60:

s=1

t2=l[0][1]

for i in range(n):

if l[i][0]>t2:

s+=1

t2=l[i][1]

if l[i][1]>60:

break

print(s)

﻿n=int(input())

l=[-1 for i in range(61)]

for i in range(n):

a,b=map(int,input().split())

if l[b]<a:

l[b]=a

s=0

m=-1

for i in range(61):

if l[i]>m:

s+=1

m=i

print(s)

【greedy】送外卖

﻿n=int(input())

output=[]

for i in range(n):

m=int(input())

a=[int(x) for x in input().split()]

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

l=[([0]\*2) for i in range(m)]

for j in range(m):

l[j][0]=a[j]

l[j][1]=b[j]

l.sort(key=lambda x:x[0],reverse=True)

t=l[0][0]

s=0

for j in range(m-1):

s+=l[j][1]

t=min(t,max(s,l[j+1][0]))

output.append(str(min(t,s+l[m-1][1])))

print('\n'.join(output))

【greedy】熄灯（注意保存列表）

﻿n,m=map(int,input().split())

l=[int(x) for x in input().split()]

j=0

b=[0]\*(n+1)

b[0]=l[0]

for i in range(0,n-1):

b[i+1]=b[i]+(-1+2\*(i%2))\*(l[i+1]-l[i])

b[-1]=b[n-1]+(1-2\*(n%2))\*(m-l[-1])

print(max((b[-1]+m)//2,(m+2\*max(b)-b[-1]-1)//2))

【greedy，Huffman】剪绳子

﻿#CS101 18164

N=int(input())

l=sorted([int(x) for x in input().split()])

ans=0

for i in range(N-1):

remain=l[0]+l[1]

ans+=remain

l.remove(l[0])

l.remove(l[0])

l.append(remain)

l.sort()

print(ans)

【greedy，双指针】军备竞赛

p = int(input())

n = [int(x) for x in input().split()]

n.sort()

cnt = 0

left = 0

right = len(n) - 1

while left<=right:

if n[left]<=p:

cnt += 1

p -= n[left] l

eft += 1

else:

if right==left:

break

p += n[right]

cnt -= 1

if cnt<0:

cnt=0

break

right -= 1

print(cnt)

p=int(input())

l=[int(x) for x in input().split()]

l.sort()

j=1

n=len(l)

s=0

for i in range(n):

if i+j>n:

break

else:

if p>=l[i]:

s+=1

p-=l[i]

else:

p+=l[-j]

s-=1

if s<0:

s=0

break

else:

p-=l[i]

j+=1

s+=1

print(s)

【双指针】number of ways

﻿n=int(input())

l=[int(x) for x in input().split()]

m=0

s=0

k=sum(l)

if k%3!=0:

print(0)

else:

j=0

for i in range(n-1):

s+=l[i]

if s==k//3:

j+=1

if s==(k//3)\*2:

m+=j

print(m)

【双指针】饼干分配

﻿g=sorted([int(x) for x in input().split()])

s=sorted([int(x) for x in input().split()])

ans=0

t=0

i=0

while t<len(g):

if g[t]<=s[i]:

ans+=1

t+=1

i+=1

else:

i+=1

if i>=len(s):

break

print(ans)

【双指针策略】田忌赛马

﻿n=int(input())

while n!=0:

l1=[int(x) for x in input().split()]

l1.sort(reverse=True)

l2=[int(x) for x in input().split()]

l2.sort(reverse=True)

ans=0

for \_ in range(n):

if l2[0]<l1[0]:

ans+=1

del l2[0]

del l1[0]

else:

if l2[-1]<l1[-1]:

ans+=1

del l2[-1]

del l1[-1]

else:

if l2[0]>l1[-1]:

ans-=1

del l2[0]

del l1[-1]

print(200\*ans)

n=int(input())

【暴力brute force】完美立方

﻿N=int(input())

l=[0]\*(N+2)

ans=[]

for i in range(N+1):

l[i]=i\*\*3

for a in range(6,N+1):

for b in range(2,a):

for c in range(b,a):

for d in range(c,a):

if l[a]==l[b]+l[c]+l[d]:

ans.append([a,b,c,d])

for i in range(len(ans)):

print('Cube = ',ans[i][0],', Triple = (',ans[i][1],',',ans[i][2],',',ans[i][3],')',sep="")

【约瑟夫问题】

﻿n,m=map(int,input().split())

while m+n!=0:

l=[1]\*n

i=0

j=m

s=n

while s!=1:

if l[i]==1:

j-=1

if j==0:

l[i]=0

s-=1

j=m

i+=1

i=i%n

else:

i+=1

i=i%n

print(l.index(1)+1)

n,m=map(int,input().split())

【汉诺塔】递归

def move(n, a, b, c):

if(n == 1):

print(a,"->",c)

return

move(n-1, a, c, b)

move(1, a, b, c)

move(n-1, b, a, c)

move(3, "a", "b", "c")

【质数筛法】

﻿l=[True]\*1000001

l[0]=l[1]=False

for i in range(1001):

if l[i]==True:

for j in range(2\*i,1000001,i):

l[j]=False

n=int(input())

k=[int(x) for x in input().split()]

for i in range(n):

if k[i]\*\*0.5!=int(k[i]\*\*0.5):

print('NO')

else:

print('YES' if l[int(k[i]\*\*0.5)]==True else 'NO')

【二分查找】

n = int(input())

shop = sorted(list(map(int,input().split())))

m = int(input())

for i in range(m):

a = int(input()) l=0

r = n-1

while l<=r:

mid = (l+r)//2

if a < shop[mid]:

r = mid-1

elif a >= shop[mid]:

l = mid + 1

print(r+1)

【DP】

【构造列表式】Sereja and Suffixes

n,m=map(int,input().split())  
l=[int(x) for x in input().split()]

l.reverse()

s={l[0]}  
l2=[1]\*n  
for i in range(n-1):

if l[i+1] not in s:

s.add(l[i+1])

l2[i+1]=l2[i]+1

else:

l2[i+1]=l2[i]

for i in range(m):

print(l2[n-int(input())])

【构造列表式】Ilya and Queries

s=input()

n=len(s)

l1=[0]\*n

for i in range(n-1):

if s[i]==s[i+1]:

l1[i+1]+=1

for i in range(n-2):

l1[i+2]+=l1[i+1]

m=int(input())

for i in range(m):

l,r=map(int,input().split())

print(l1[r-1]-l1[l-1])

【剪丝带】完全背包

inf = 1e9 + 7

n,a,b,c = map(int,input().split()) dp = [0]+[-inf]\*n

for i in range(1,n+1):

for j in (a,b,c):

if i >= j:

#dp[i] = max(dp[i-j], dp[i-j] + 1, dp[i]) dp[i] = max(dp[i-j] + 1, dp[i])

print(dp[n])

【boredom】

n=int(input())

l1=[int(x) for x in input().split()]

l2=[0]\*(max(l1)+1)

for i in l1:

l2[i]+=1 f=[0]\*(max(l1)+1)

for i in range(max(l1)+1):

f[i]=max(f[i-1],f[i-2]+i\*l2[i])

print(f[max(l1)])

【滑雪】

﻿r,c=map(int,input().split())

l=[[10001]\*(c+2)]+[[10001]+[int(x) for x in input().split()]+[10001] for i in range(r)]+[[10001]\*(c+2)]

output=[[0]\*(c+2) for i in range(r+2)]

dx=[0,0,-1,1]

dy=[1,-1,0,0]

def dp(i,j):

if output[i][j]>0:

return output[i][j]

for s in range(4):

if l[i][j]>l[i+dx[s]][j+dy[s]]:

output[i][j]=max(output[i][j],dp(i+dx[s],j+dy[s])+1)

return output[i][j]

ans=0

for i in range(1,r+1):

for j in range(1,c+1):

ans=max(ans,dp(i,j))

print(ans+1)

【最大上升子序列和】

﻿n=int(input())

l=[int(x) for x in input().split()]

s=l[:]

for i in range(1,n):

for j in range(i):

if l[i]>l[j]:

s[i]=max(s[j]+l[i],s[i])

print(max(s))

【最长上升子序列】

n=int(input())

l=[int(x) for x in input().split()]

s=[1]\*n

for i in range(1,n):

for j in range(i):

if l[i]>l[j]:

s[i]=max(s[j]+1,s[i])

print(max(s))

【三角形】

﻿n=int(input())

l1=[int(x) for x in input().split()]

for i in range(n-1):

l2=[int(x) for x in input().split()]

l3=[l2[0]+l1[0]]+[max(l1[i],l1[i+1])+l2[i+1] for i in range(len(l2)-2)]+[l2[-1]+l1[-1]]

l1=l3

print(max(l1))

【组合乘积】

﻿T=int(input())

l=[int(x) for x in input().split()]

copyl=l[:]

for i in copyl:

if T%i!=0 or i==1:

l.remove(i)

ans=set()

for i in l:

ans.add(i)

copyans=list(ans)

for j in copyans:

ans.add(i\*j)

print('YES' if T in ans else 'NO')

【合唱团】（最大上升子序列变体）

n=int(input())

performance=list(input().split())

kd=list(input().split())

k=int(kd[0])

d=int(kd[1])

dp=[[0]\*n for \_ in range(k)]

dp1=[[0]\*n for \_ in range(k)]

for j in range(n):

performance[j]=int(performance[j])

dp[0][j]=performance[j]

dp1[0][j]=performance[j]

for i in range(1,k):

for j in range(n):

for k in range(max(j-d,0),j):

dp[i][j]=max(dp[i][j],dp1[i-1][k]\*performance[j],dp[i-1][k]\*performance[j])

dp1[i][j]=min(dp1[i][j],dp1[i-1][k]\*performance[j],dp[i-1][k]\*performance[j])

print(max(dp[-1]))

【最长公共子序列】

def LCS(string1,string2):

len1 = len(string1)

len2 = len(string2)

res = [[0 for i in range(len1+1)] for j in range(len2+1)]

for i in range(1,len2+1):

for j in range(1,len1+1):

if string2[i-1] == string1[j-1]:

res[i][j] = res[i-1][j-1]+1

else:

res[i][j] = max(res[i-1][j],res[i][j-1])

return res,res[-1][-1]

print(LCS("helloworld","loop"))

【最长公共子串】

def LCstring(string1,string2):

len1 = len(string1)

len2 = len(string2)

res = [[0 for i in range(len1+1)] for j in range(len2+1)]

result = 0

for i in range(1,len2+1):

for j in range(1,len1+1):

if string2[i-1] == string1[j-1]:

res[i][j] = res[i-1][j-1]+1

result = max(result,res[i][j])

return result

print(LCstring("helloworld","loop"))

【简单的整数划分】

def GPC3(n):

if n < 0:

return 0

dp = [1] + [0]\*n

for num in range(1,n+1):

for i in range(num,n+1):

dp[i] += dp[i-num]

return dp[-1]

【01背包问题】

﻿T,M=map(int,input().split())

l=[0]\*(T+1)

copyl=l[:]

for i in range(M):

t,m=map(int,input().split())

if t<=T:

for j in range(t,T+1):

l[j]=max(copyl[j-t]+m,l[j])

copyl=l[:]

print(l[T])

【矩阵转移】最小距离

l1='algorithm'

l2='alligator'

m=len(l1)

n=len(l2)

edit=[list(range(0,(n+1)\*20,20))]+[[(i+1)\*20]+[0]\*(n) for i in range(m)]

editProcedure=[['']+[('replicate '+l2[i]) for i in range(n)]]+[['replicate '+l1[i]]+[' ']\*(n) for i in range(m)]

for i in range(2,m+1):

editProcedure[i][0]=editProcedure[i-1][0]+' '+editProcedure[i][0]

for i in range(2,n+1):

editProcedure[0][i]=editProcedure[0][i-1]+' '+editProcedure[0][i]

for i in range(1,m+1):

for j in range(1,n+1):

if l1[i-1]==l2[j-1]:

edit[i][j]=edit[i-1][j-1]+5

editProcedure[i][j]=editProcedure[i-1][j-1]+' replicate '+l1[i-1]

else:

edit[i][j]=min(edit[i-1][j]+20,edit[i][j-1]+20)

if edit[i-1][j]<=edit[i][j-1]:

editProcedure[i][j]=editProcedure[i-1][j]+' delete '+l1[i-1]

else:

editProcedure[i][j]=editProcedure[i][j-1]+' insert '+l2[j-1]

print('最小编辑距离得分为',edit[i][j])

print('编辑过程为',editProcedure[i][j])

【开餐馆】

变形01背包，状态方程

f[i] = max(f[i],f[j] + c[i])，其中j与i距离大于临界值，对于j要遍历

【dfs-八皇后】

﻿def queen(A,cur=0):

if cur==len(A):

l.append(A[:])

return 0

for i in range(len(A)):

A[cur],flag=i,True

for j in range(cur):

if A[j]==A[cur] or abs(A[j]-A[cur])==cur-j:

flag=False

break

if flag:

queen(A,cur+1)

l=[]

queen([None]\*8)

n=int(input())

for i in range(n):

print(''.join([str(x+1) for x in l[int(input())-1]]))

【dfs-池塘】

﻿dx=[0,0,1,1,1,-1,-1,-1]

dy=[1,-1,1,0,-1,1,0,-1]

count=0

def dfs(i,j):

global count

if l[i][j]!='W':

return

l[i][j]='M'

count+=1

for s in range(8):

dfs(i+dx[s],j+dy[s])

T=int(input())

for i in range(T):

N,M=map(int,input().split())

l=[[0]\*(M+2)]+[[0]+list(input())+[0] for i in range(N)]+[[0]\*(M+2)]

ans=0

for i in range(1,N+1):

for j in range(1,M+1):

count=0

if l[i][j]=='W':

dfs(i,j)

ans=max(ans,count)

print(ans)

【冒泡排序】

﻿def bubble\_sort(array):

for i in range(1, len(array)):

for j in range(0, len(array)-i):

if array[j]+array[j+1] > array[j+1]+array[j]:

array[j], array[j+1] = array[j+1], array[j]

return array

﻿n=int(input())

l=input().split()

sortl=bubble\_sort(l)

print(''.join([str(x) for x in reversed(sortl)]),''.join([str(x) for x in sortl]))

【最大联通子图】

﻿def dfs(graph,node,visited):

if node!=-1 and node not in visited:

visited.append(node)

if node not in graph:

return visited

for nei in graph[node]:

dfs(graph,nei,visited)

return visited

graph={}

ids=set()

n=int(input())

for i in range(n):

l=input().split(' : ')

ids.add(int(l[0]))

if int(l[0]) not in graph:

graph[int(l[0])]=[int(x) for x in l[1].split()]

maxp=0

for i in ids:

dfs\_path=dfs(graph,i,[])

maxp=max(maxp,len(dfs\_path))

print(maxp)