**基本语法**

chr()字符串转ascii码

ord()ascii码转字符串

math.ceil() 向上取 1.5 - 2

math.floot() 向下取 1.5 - 2

math.sqrt(x)

math.pos(a,b,c) #return a \*\* b

math.log(x,base)

list.index()

list.extend()

str.find()

zip(list1,list2) 返回tuple

map(int,input().split())

Import sys

sys.setrecursionlimit(times)

from functools import lru\_cache

@ lru\_cache(maxsize = None(具体数))

lru\_cache的用法

heapq

itertools

product([迭代元素]，repeat = x)输出的组合长度

for \_ in list(itertools.permutations([1,2,3],3)):

print(\_)

**手动排序算法**

**归并排序**

#思路：把无序数组不断二分直到每个数组都只有一个元素，然后再将数组合并为小的有序数组，不断合并，最后产生的最长有序数组就排序完成了

nums = [5,3,6,4,1,2,8,7]

def MergeSort(num):

**if(len(num)<=1):** #递归边界条件

**return num**  #到达边界时返回当前的子数组

mid = int(len(num)/2) #求出数组的中位数

l\_list,r\_list = MergeSort(num[:mid]),MergeSort(num[mid:]) #不断二分

result = []

i,j = 0,0

while i < len(l\_list) and j < len(r\_list):

if r\_list[j] < l\_list[i]:

result.append(r\_list[j])

j += 1

else:

result.append(l\_list[i])

i += 1

result += l\_list[i:] + r\_list[j:]

return result

print(MergeSort(nums))

**快速排序**

def QuickSort(num):

**if len(num) <= 1:**

**return num**

key = num[0]

l\_list,r\_list,m\_list = [],[],[key]

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

if num[i] > key:

r\_list.append(num[i])

elif num[i] < key:

l\_list.append(num[i])

else:

m\_list.append(num[i])

return **QuickSort(l\_list) + m\_list + QuickSort(r\_list)**

print(QuickSort(nums))

**数论**

**欧拉筛**

def euler\_sieve(n):

primes = []

is\_prime = [True] \* (n + 1)

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

if is\_prime[i]:

primes.append(i)

for j in range(len(primes)):

if i \* primes[j] > n:

break

is\_prime[i \* primes[j]] = False

if i % primes[j] == 0:

break

return primes

x = int(input())

print(euler\_sieve(x))

**欧几里得取余**

def Euclidean(a,b):

if (b == 0): #边界条件

return a

else:

return Euclidean(b,a % b)

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

print(Euclidean(a,b))

**阶乘**

def fact(x):

if x == 1:

return 1

else :

y = x\*fact(x-1)

return y

**二分法**

Bisect

bisect(l,x). = bisect\_right)

bisect\_left(l,x). 插入x之后x应有的坐标，返回最左边的索引

bisect\_right(l,x). 返回最右边的索引 + 1

**标准格式**

注意找最大的最小还是最小的最大

l, r = 0, l

mid = (l + r) // 2

while l <**=** r:

if remove <= m:

l = mid **+ 1**

else:

r = mid **- 1**

mid = (l + r) // 2

有时也可能会有1的调整

**月度开销**

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

budget = [0]

add\_up = 0

mx = 0

for i in range(n):

t = int(input())

mx = max(t,mx)

add\_up += t

budget.append(add\_up)

l,r = mx,budget[-1]

mid = (l + r **+ 1**) // 2

while l <= r:

check = 0

total = 0

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

if budget[i] - check > mid:

check = budget[i - 1]

total += 1

#print(check)

if i == n and budget[i] - check <= mid:

total += 1

if total <= m:

r = mid - 1

mid = (l + r **+ 1**) // 2

else:

l = mid + 1

mid = (l + r **+ 1**) // 2

print(mid)

**放弃考试（分数规划）**

def can\_achieve(target,a,b,k):

diffs=[a[i]-target\*b[i] for i in range(len(a))]

diffs.sort()

#放弃k场考试后可以达到target

return sum(diffs[k:])>=0

def max\_avg\_score(k,a,b):

l,r=0,100

while r-l > 1e-5: #非整数二分

m=(l+r)/2

if can\_achieve(m,a,b,k):

l = m

else:

r = m

return m\*100

while True:

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

if n==0 and k==0:

break

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

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

print(f'{max\_avg\_score(k,a,b):.0f}')

**二分查找**

#必须从小到大排序

arr.sort()

#构建左右双指针

l,r = 0,len(arr) - 1

n = int(input())

#当左指针大于右指针时循环停止

while l <= r:

mid = (l+r) // 2

if arr[mid] < n: #左指针移到mid+1处

l = mid + 1

elif arr[mid] > n:

r = mid - 1 #右指针移到mid-1处

else: #如果mid指向元素与key相等，则直接跳出循环

print(mid)

break

#while循环自然结束，说明没有找到与关键字相等的元素

print(-1)

**约瑟夫环**

def circle(arr,start,num):

total\_list = []

for i in range(start,start + arr):

if i % arr == 0:

i = arr

else:

i = i % arr

total\_list.append(i)

selected = []

cnt = 0

while len(total\_list) > 1:

front = total\_list.pop(0)

cnt += 1

if cnt == num:

selected.append(front)

cnt = 0

else:

total\_list.append(front)

selected.append(total\_list[0])

return ','.join([str(x) for x in selected])

while True:

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

if n == p == m == 0:

break

else:

print(circle(n,p,m))

**动态规划**

**最大连续子序列**

l = int(input())

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

dp = [0] \* l

dp[0] = num[0]

for i in range(l - 1):

if dp[i] <= 0:

dp[i + 1] = num[i + 1]

else:

dp[i + 1] = dp[i] +num[i + 1]

print(dp)

**最长上升子序列**

def longest\_ones(num):

l = len(num)

if l <= 1:

return l

dp = [1] \* l

for i in range(1,l):

for j in range(i):

if num[i] > num[j]:

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

return max(dp)

num = list(map(int,input().split()))

print(longest\_ones(num))

**最长公共子序列**

str1,str2 = input().split()

a,b = len(str1),len(str2)

dp = [[0]\*(b + 1) for i in range(a + 1)]

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

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

if str1[i - 1] == str2[j - 1]:

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

elif str1[i - 1] != str2[j - 1]:

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

print(dp[a][b])

**滑雪**

#dfs+lru\_cache(dp)，最长路

from functools import lru\_cache

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

mat=[]

mat.append([10001]\*(c+2))

for i in range(r):

mat.append([10001]+list(map(int,input().split()))+[10001])

mat.append([10001]\*(c+2))

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

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

#print(mat)

@lru\_cache(maxsize=1024)

def dp(x,y):

ans=1

for i in range(4):

nx=x+dx[i]

ny=y+dy[i]

if mat[x][y]>mat[nx][ny]:

ans=max(ans,1+dp(nx,ny))

return ans

ans=0

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

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

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

#print(i,j,dp(i,j))

print(ans)

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

def is\_valid(x,y):

if 0 <= x < r and 0 <= y < c:

return True

return False

def dfs(x,y):

global res

if res[x][y] > 0:

return res[x][y]

for i in range(4):

nx = x + dx[i]

ny = y + dy[i]

if is\_valid(nx,ny):

if altitude[nx][ny] > altitude[x][y]:

res[x][y] = max(res[x][y],dfs(nx,ny) + 1)

return res[x][y]

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

altitude = [[int(x) for x in input().split()] for i in range(r)]

res = [[0]\*c for i in range(r)]

ans = 0

for i in range(r):

for j in range(c):

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

print(ans + 1)

**核电站**

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

dp = [0] \* 51

dp[0] = 1

dp[1] = 2

for i in range(2,51):

if i < m:

dp[i] = 2 \* dp[i - 1]

elif i == m:

dp[i] = 2 \* dp[i - 1] - 1

elif i > m:

**dp[i] = 2 \* dp[i - 1] - dp[i - m - 1]**

print(dp[n])

**放苹果**

def counter(m,n):

if m <= 1 or n == 1:

return 1

elif m < n:

return counter(m,m)

else:

return **counter(m,n - 1) + counter(m - n,n)**

cases= int(input())

for i in range(cases):

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

print(counter(m,n))

**股票买卖**

cases = int(input())

for i in range(cases):

n = int(input())

price\_list = [int(x) for x in input().split()]

x,y = price\_list[0],price\_list[-1]

l,r = [x]\*n,[y]\*n

pl,pr = [0]\*n,[0]\*n

for i in range(1,n):

l[i] = min(l[i - 1],price\_list[i])

#第i天前卖最大收益

pl[i] = max(pl[i - 1],price\_list[i] - l[i])

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

r[i] = max(r[i + 1],price\_list[i])

#第i天后卖最大收益

pr[i] = max(pr[i + 1],r[i] - price\_list[i])

ans = 0

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

ans = max(ans,pl[i] + pr[i + 1])

print(ans)

**通配符匹配**

n = int(input())

def isMatch(s, p):

m, n = len(s), len(p)

dp = [[False] \* (n + 1) for \_ in range(m + 1)]

**dp[0][0] = True**

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

if p[j-1] == '\*':

dp[0][j] = dp[0][j-1]

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

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

if p[j-1] in {s[i-1], '?'}:

dp[i][j] = dp[i-1][j-1]

elif p[j-1] == '\*':

dp[i][j] = dp[i][j-1] or dp[i-1][j]

return dp[m][n]

for i in range(n):

s = input()

p = input()

print('yes' if isMatch(s,p) else 'no')

**Sumsets(找规律)**

n = int(input())

def search():

global result

result = [1] \* (n+1)

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

if i % 2 == 0:

result[i] = (result[i - 1] + result[i // 2])%10\*\*9

else:

result[i] = result[i - 1]%10\*\*9

search()

print(result[n])

**分解因数**

def change(x,t):

if x == 1:

return 1

cnt = 0

for i in range(t,x + 1):

if x % i == 0:

cnt += change(x // i,i)

return cnt

n = int(input())

for i in range(n):

print(change(int(input()),2))

**健身房**

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

**dp = [float('-inf')]\*(n + 1)**

t,w = [],[]

for i in range(m):

x,y = map(int,input().split())

t += [x]

w += [y]

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

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

**if j == t[i - 1]:**

dp[j] = max(dp[j],dp[j - t[i - 1]] + w[i - 1],**w[i - 1]**)

elif j > t[i - 1]:

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

print(dp[-1] if dp[-1] > 0 else "-1")

**最大公共子序列**

s1,s2 = input().split()

dp = [[0]\*(len(s2) + 1) for i in range(len(s1) + 1)]

for i in range(1,len(s1) + 1):

for j in range(1,len(s2) + 1):

if s1[i - 1] == s2[j - 1]:

**dp[i][j] = dp[i - 1][j - 1] + 1**

else:

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

print(dp[len(s1)][len(s2)])

**神奇的口袋**

n = int(input())

a = [int(input()) for i in range(n)]

dp = [[0]\*41 for i in range(n + 1)]

for i in range(n + 1):

dp[i][0] = 1

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

for j in range(1,41):

dp[i][j] = dp[i - 1][j]

if j >= a[i - 1]:

dp[i][j] += dp[i - 1][j - a[i - 1]]

print(dp[-1][-1])

**Apple catching**

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

info = [int(input()) for i in range(t)]

dp = [[0]\*(w + 1) for i in range(t + 1)]

if info[0] == 1:

dp[1][0] = 1

else:

dp[1][1] = 1

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

dp[i][0] = dp[i - 1][0] + info[i - 1] % 2

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

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

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

if j % 2 + 1 == info[i - 1]:

dp[i][j] += 1

ans = 0

for j in range(w + 1):

ans = max(ans,dp[-1][j])

print(ans)

**宠物小精灵之收服**

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

L = [**[-1]**\*(M+1) for i in range(K+1)]

**L[0][M] = N**

for i in range(K):

cost, dmg = map(int, input().split())

for p in range(M):

for q in range(i+1, 0, -1):

if p+dmg <= M and L[q-1][p+dmg] >= 0:

L[q][p] = max(L[q][p], L[q-1][p+dmg]-cost)

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

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

if L[i][j] != -1:

print(i,j)

exit()

**最佳凑单**

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

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

l = sum(p)

if l < t:

print("0")

else:

p.sort()

dp = [[0]\*(l + 1) for i in range(n + 1)]

res = l

for i in range(n + 1):

dp[i][0] = 1

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

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

dp[i][j] = dp[i - 1][j]

if j == p[i - 1]:

dp[i][j] = 1

elif j > p[i - 1]:

if dp[i - 1][j] != 0 or dp[i - 1][j - p[i - 1]] != 0:

dp[i][j] = 1

if j >= t and dp[i][j]:

res = min(res,j)

break

print(res)

**完美的爱**

length = int(input())

value\_list = [0] \* (length + 1)

put\_in = list(map(int,input().split()))

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

value\_list[i] = put\_in[i - 1]

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

value\_list[i] += value\_list[i - 1]

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

value\_list[i] -= 520 \* i

position\_dict = {}

for i in range(length + 1):

if value\_list[i] not in position\_dict:

position\_dict[value\_list[i]] = [i]

else:

position\_dict[value\_list[i]].append(i)

res = 0

for keys,values in position\_dict.items():

if len(values) >= 2:

if values[-1] - values[0] > res:

res = values[-1] - values[0]

print(res \* 520)

**贪心**

**摆动序列**

**def sgn(x):**

**if x == 0:**

**return 0**

**elif x > 0:**

**return 1**

**elif x < 0:**

**return -1**

n = int(input())

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

delta = [sgn(num[i + 1] - num[i]) for i in range(n - 1)]

res = 1

sign = 0

for i in range(n - 1):

**if delta[i] \* sign < 0 or (sign == 0 and delta[i] != 0)**:

res += 1

sign = delta[i]

print(res)

**jumping cows**

p = int(input())

h = [int(input()) for i in range(p)]

res = 0

k = 1

for i in range(p - 1):

if **(h[i + 1] - h[i])\*k** < 0:

res += k \* h[i]

k = -k

if k == 1:

res += h[-1]

print(res)

**背包问题**

**def process2(value, weight, N, V):**

# dp[0~V]

dp = [0] \* (V + 1)

for i in range(1, N + 1): # 仍然需要N层循环，只能节省空间，不能节省时间

for j in range**(V, -1, -1):**  # 需求方向从0开始，所以反方向填补

ans1 = dp[j]

ans2 = 0

if j - weight[i - 1] >= 0:

# 在这里可以把判断放到j循环里面， 写为for j in range(V,weigth[i-1]-1,-1)， 可以节省一点时间，但是为了方便理解暂时牺牲一点点时间

ans2 = dp[j - weight[i - 1]] + value[i - 1]

dp[j] = max(ans1, ans2)

return dp[V]

**def process3(value, weight, N, V):**

# dp[0~V]

dp = [0] \* (V + 1)

for i in range(1, N + 1): # 仍然需要N层循环，只能节省空间，不能节省时间

for j in range**(V + 1):**

ans1 = dp[j]

ans2 = 0

if j - weight[i - 1] >= 0:

# 在这里可以把判断放到j循环里面， 写为for j in range(weigth[i-1],V+1)， 可以节省一点时间，但是为了方便理解暂时牺牲一点点时间

ans2 = dp[j - weight[i - 1]] + value[i - 1]

dp[j] = max(ans1, ans2)

return dp[V]

**def process4(value, weight, N, V, S):**

# dp[0~V]

dp = [0] \* (V + 1)

for i in range(1, N + 1): # 仍然需要N层循环，只能节省空间，不能节省时间

for j in range**(V, -1, -1):** # 需求方向从0开始，所以反方向填补

**for s in range(S + 1):**

# 增加一个循环就行，因为需要掌控数量，所以从j从大到小循环

if j - s \* weight[i - 1] < 0:

break

dp[j] = max(dp[j], dp[j - s \* weight[i - 1]] + s \* value[i - 1]) # dp[j - 0 \* weight[i - 1]] + 0 \* value[i - 1] == dp[j]

return dp[V]

多个dp数组（往往从左到右考虑上一步或下一步）

**红蓝玫瑰**

s = input()

n = len(s)

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

if s[0] == "R":

b[1] = 1

else:

r[1] = 1

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

if s[i - 1] == "R":

r[i] = r[i - 1]

b[i] = min(b[i - 1],r[i - 1]) + 1

else:

b[i] = b[i - 1]

r[i] = min(r[i - 1],b[i - 1]) + 1

print(r[-1])

**CF篮球选人**

n = int(input())

h1 = [int(x) for x in input().split()]

h2 = [int(x) for x in input().split()]

dp = [[0]\*3 for i in range(n + 1)]

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

**dp[i][0] = max(dp[i - 1][1],dp[i - 1][2])**

**dp[i][1] = max(dp[i - 1][0],dp[i - 1][2]) + h1[i - 1]**

**dp[i][2] = max(dp[i - 1][0],dp[i - 1][1]) + h2[i - 1]**

print(max(dp[n]))

**上机**

n = int(input())

a = [0] + [int(x) for x in input().split()]

b = [0] + [int(x) for x in input().split()]

c = [0] + [int(x) for x in input().split()]

c[1] = 0

dp = [[0]\*2 for i in range(n + 1)]

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

dp[i][0] = max(dp[i - 1][0] + b[i],dp[i - 1][1] + a[i])

if i == 1:

dp[i][0] = a[i]

dp[i][1] = max(dp[i - 1][1] + b[i],dp[i - 1][0] + c[i])

print(dp[n][0])

**螺旋矩阵**

**螺旋读出**

n = int(input())

matrix = []

t = n \*\* 2

for i in range(n):

matrix.append(list(map(int,input().split())))

ans = ""

visit\_list = [[1] \* (n + 2) for i in range(n + 2)]

for i in range(n):

for j in range(n):

visit\_list[i + 1][j + 1] = 0

times = 0

x,y = 0,0

move = [[0,1],[1,0],[0,-1],[-1,0]]

k = 0

visit\_list[1][1] = 1

ans += str(matrix[0][0])

while True:

x += move[k][0]

y += move[k][1]

if not visit\_list[x + 1][y + 1]:

ans += str(matrix[x][y])

visit\_list[x + 1][y + 1] = 1

else:

x -= move[k][0]

y -= move[k][1]

k += 1

k = k % 4

x += move[k][0]

y += move[k][1]

ans += str(matrix[x][y])

visit\_list[x + 1][y + 1] = 1

times += 1

if times == t - 1:

break

print(ans)

**区间覆盖问题**

**世界杯只因**

n = int(input())

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

lst = []

for i in range(n) :

lst.append([i-arr[i]-0.5,i+arr[i]+0.5])

lst.sort(key=lambda x:x[0])

cnt = ans = mx = l = 0

while l < n - 1:

mx = 0

while cnt < n and lst[cnt][0] <= l:

mx = max(mx,lst[cnt][1])

cnt += 1

ans += 1

l = mx

print(ans)

**建筑修建**

def generate\_intervals(pos,width):

tmp = []

for start in range(max(0,pos - width + 1),min(m,x + 1)):

**end = start + width**

**if end <= m:**

**tmp.append((start,end))**

return tmp

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

mx = m

plan,info = [],[]

for i in range(n):

plan += [tuple(map(int,input().split()))]

for x,y in plan:

info.extend(generate\_intervals(x,y))

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

cnt = 0

end = 0

for s,e in info:

if s >= end:

end = e

cnt += 1

print(cnt)

**畜栏保留**

from heapq import heappop,heappush

n = int(input())

info = [list(map(int,input().split())) + [i] for i in range(n)]

info.sort()

column = []

res = [1]\*n

cnt = 1

heappush(column,[info[0][1],1])

for i in range(1,n):

end,pos = heappop(column)

if info[i][0] > end:

**res[info[i][2]]** = pos

heappush(column,[info[i][1],pos])

else:

**heappush(column,[end,pos])**

cnt += 1

**res[info[i][2]]** = cnt

heappush(column,[info[i][1],cnt])

print(len(column))

for i in range(n):

print(res[i])

**加油问题**

import heapq

n = int(input())

stop = []

for i in range(n):

pos,fuel = map(int,input().split())

stop.append((pos,fuel))

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

stop.append((0,0))

stop.sort(reverse = True)

cur = p

now = l

cnt = 0

que = []

for i in range(n + 1):

di = now - stop[i][0]

while cur < di:

if not que:

print(-1)

exit()

cur -= heapq.heappop(que)

cnt += 1

cur -= di

now = stop[i][0]

heapq.heappush(que,-stop[i][1])

print(cnt)

**充实的假期生活**

n = int(input())

plan\_dic = {}

for i in range(n):

s,e = map(int,input().split())

plan\_dic[e] = max(plan\_dic.get(e,0),s)

plan\_list = []

for key,value in plan\_dic.items():

plan\_list.append([value,key])

plan\_list.sort(key = lambda x:x[1])

ans = 0

end = -1

for \_ in plan\_list:

if \_[0] > end:

ans += 1

end = \_[1]

print(ans)

**幸福的寒假生活**

def change\_date(x):

t = x.split('.')

m,d = int(t[0]),int(t[1])

return 31\*(m - 1) + d - 7

info = []

n = int(input())

for i in range(n):

s,e,v = input().split()

s = change\_date(s)

e = change\_date(e)

v = int(v)

if s >= 0 and e <= 44:

info.append((s,e,v))

info.sort()

dp = [0]\*45

**for i in range(1,46):**

dp[i - 1] = dp[i - 2]

**for start,end,value in info:**

if end == i - 1:

**dp[i - 1] = max(dp[i - 1],dp[start - 1] + value)**

print(dp[-1])

**栈结构**

**今日化学论文**

i = 0

def decode():

global i

res = []

while i < len(string):

chr = string[i]

i += 1

if chr == "[":

num = []

while string[i] in "0123456789":

num.append(string[i])

i += 1

res.extend(int(''.join(num)) \* decode())

elif chr == "]":

return res

else:

res.append(chr)

return res

string = input()

print(''.join(decode()))

**合法出入问题**

def is\_valid\_pop\_sequence(n, pop\_sequence):

stack = []

now\_max = 0

for x in pop\_sequence:

x = int(x)

if x > now\_max:

for j in range(now\_max + 1, x + 1):

stack.append(j)

now\_max = x

if stack[-1] != x:

return "No"

else:

stack.pop()

return "Yes"

# 读取输入

n = int(input())

pop\_sequence = list(map(int, input().split()))

# 调用函数进行判断

result = is\_valid\_pop\_sequence(n, pop\_sequence)

print(result)

**波兰表达式和逆波兰表达式**

s = input().split()

stack = []

for i in range(len(s) - 1,-1,-1):

if s[i] in "+-\*/":

a = stack.pop()

b = stack.pop()

if s[i] == "+":

stack.append(a + b)

elif s[i] == "-":

stack.append(a - b)

elif s[i] == "\*":

stack.append(a \* b)

elif s[i] == "/":

stack.append(a / b)

else:

stack.append(float(s[i]))

print(f"{stack[-1]:.6f}")

**前缀和思想**

**护林员盖房子**

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

a = []

a.append([0]\*(n+2))

for i in range(m):

a.append([0] + [int(x) for x in input().split()] + [0])

a.append([0]\*(n+2))

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

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

if a[i][j] == 0:

a[i][j] = a[i][j-1] + 1 #宽度累加

else:

a[i][j] = 0 #种树了的地方因为没办法算数字,记为0

# 遍历每一个位置，向上回溯的方式，求以该位置为最右下⻆的树群的最大值

ans = 0

width = 0

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

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

if a[i][j] == 0:

continue

width = a[i][j] # 设置为档期的宽度

ans = max(ans, width\*1);

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

if a[k][j] == 0:

break # 如果搜到树，则说明不用再往上了

else:

width = min(a[k][j], width) # 更新可以盖房子的宽度

ans = max(ans, width \* (i - k + 1))

print(ans)

**递归**

**八皇后问题**

res = []

def recurse(**path = [],i = 0,col = [],diag\_1 = set(),diag\_2 = set()**):

if i == 8:

res.append(path)

return

for j in range(8):

if j not in col and **i - j** not in diag\_1 and **i + j** not in diag\_2:

recurse(**path + [j + 1],i + 1,col + [j],diag\_1|{i - j},diag\_2|{i + j}**)

recurse()

n = int(input())

for i in range(n):

dex = int(input())

print(''.join([str(x) for x in res[dex - 1]]))

**最小奖金方案**

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

award = [100]\*n

result = [[[],[]] for i in range(n)]

for i in range(turns):

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

result[w][0] += [l]

result[l][1] += [w]

def change(res):

while True:

flag = 0

for i in range(n):

if len(res[i][0]) == 0 and len(res[i][1]) != 0:

flag = 1

while res[i][1]:

team = res[i][1].pop()

res[team][0].remove(i)

award[team] = max(award[team],award[i] + 1)

if flag == 0:

return sum(award)

print(change(result))

**BFS**

**抓住那头牛**

from collections import deque

in\_queue = [False]\***100010**

def get\_step(x,y):

step = 0

q = deque()

q.append(x)

while True:

cnt = len(q)

for i in range(cnt):

front = q.popleft()

if front == y:

return step

in\_queue[front] = True

if front - 1 **>= 0** and not in\_queue[front - 1]:

q.append(front - 1)

if front + 1 <= 100000 and not in\_queue[front + 1]:

q.append(front + 1)

if 2 \* front <= 100000 and not in\_queue[2\*front]:

q.append(2\*front)

step += 1

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

print(get\_step(n,k))

**迷宫最短路径**

from collections import deque

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

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

def is\_valid(x,y):

if 0 <= x < n and 0 <= y < m and maze[x][y] == 0 and not in\_queue[x][y]:

return True

return False

def bfs(x,y):

global step,flag

q = deque([(x,y)])

path = deque([[(x,y)]])

while q:

cnt = len(q)

for i in range(cnt):

front = **q.popleft()**

fpath = **path.popleft()**

for i in range(4):

nx = front[0] + dx[i]

ny = front[1] + dy[i]

npath = fpath + [(nx,ny)]

if nx == n - 1 and ny == m - 1:

return npath

if is\_valid(nx,ny):

in\_queue[nx][ny] = True

**q.append((nx,ny))**

**path.append(npath)**

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

maze = [list(map(int,input().split())) for i in range(n)]

in\_queue = [[False]\*m for i in range(n)]

in\_queue[0][0] = True

step,flag = 0,0

ans = bfs(0,0)

for \_ in ans:

print(\_[0] + 1,\_[1] + 1)

**DFS**

**最大权值路径**

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

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

temppath,optpath = [[1,1]],[]

def dfs(maze,x,y,nowvalue):

global maxvalue,temppath,optpath

for i in range(4):

nx = x + dx[i]

ny = y + dy[i]

if nx == n and ny == m:

nextvalue = nowvalue + maze[nx][ny]

temppath.append([nx,ny])

if nextvalue > maxvalue:

maxvalue = nextvalue

optpath = temppath.copy()

temppath.pop()

if maze[nx][ny] >= -100:

nextvalue = nowvalue + maze[nx][ny]

**tmp = maze[x][y]**

**maze[x][y] = -9999**

temppath.append([nx,ny])

dfs(maze,nx,ny,nextvalue)

**maze[x][y] = tmp**

temppath.pop()

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

maze = [[-9999] \* (m + 2)]

for i in range(n):

maze.append([-9999] + [int(x) for x in input().split()] + [-9999])

maze.append([-9999] \* (m + 2))

maxvalue = -9999

dfs(maze,1,1,maze[1][1])

for \_ in optpath:

print(\*\_)

**迷宫可行路径数**

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

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

def dfs(maze, x, y):

global cnt

for i in range(4):

nx = x + dx[i]

ny = y + dy[i]

if maze[nx][ny] == 'e':

cnt += 1

continue

if maze[nx][ny] == 0:

maze[nx][ny] = 1

dfs(maze, nx, ny)

**maze[nx][ny] = 0**

return

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

maze = []

maze.append([-1 for x in range(m+2)])

for \_ in range(n):

maze.append([-1] + [int(\_) for \_ in input().split()] + [-1])

maze.append([-1 for x in range(m+2)])

maze[1][1] = 's'

maze[n][m] = 'e'

cnt = 0

dfs(maze, 1, 1)

print(cnt)

**全排列**

Import itertools

temp = itertools.permutations(str,len)输入字符串和要全排列的长度

def dfs(step):

if step == n + 1:

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

print(a[i], end="")

print()

return

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

if book[i] == 0:

a[step] = i

book[i] = 1

dfs(step + 1)

book[i] = 0

n = int(input())

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

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

dfs(1)

最大乘积

def removal(num):

changed\_num = sorted(num)

i,j,k = 0,0,0

for \_ in num:

if \_ < 0:

i += 1

elif \_ == 0:

j += 1

elif \_ > 0:

k += 1

if j >= 2:

return num[0]

if j == 1:

if k > 0:

if i % 2 != 0:

if num[0] != 0:

return num[0]

else:

return num[1]

elif i % 2 == 0:

return 0

elif k == 0:

if i % 2 != 0:

if num[0] != 0:

return num[0]

else:

return num[1]

elif i % 2 == 0:

return 0

if j == 0:

if k > 0:

if i % 2 != 0:

return changed\_num[i - 1]

elif i % 2 == 0:

return changed\_num[i]

elif k == 0:

if i % 2 != 0:

return changed\_num[i - 1]

elif i % 2 == 0:

return changed\_num[0]

cases = int(input())

for i in range(cases):

length = int(input())

num\_list = list(map(int,input().split()))

res = removal(num\_list)

print(res)

注意入队标记和出队标记！