基础语法

可访问https://docs.python.org/zh-cn/3/

字符串和编码

对于单个字符的编码，Python提供了ord()函数获取字符的整数表示，chr()函数把编码转换为对应的字符_A:65 _Z:90_a:97_z:122 一种格式化字符串的方法是使用以f开头的字符串，称之为f-string，字符串如果包含{xxx}，就会以对应的变量替换：

```
f'The area of a circle with radius {r} is {s:.2f}'
```

并且:后面的.2f指定了格式化参数（即保留两位小数）

list、tuple、dict、set

- list：append、pop(索引)、remove(元素)、insert(插入位置,元素)、extend
  - count 函数是字符串、列表和元组等序列对象的方法，用于统计某个元素或子字符串在对象中出现的次数。
- tuple和list非常类似，但是tuple一旦初始化就不能修改
- dic字典：d = {'Michael': 95, 'Bob': 75, 'Tracy': 85}
  - 通过in判断key是否存在，或通过dict提供的get()方法，如果key不存在，可以返回None
  - 要删除一个key，用pop(key)方法，对应的value也会从dict中删除
  - dict的key必须是不可变对象
  - 3.7以上版本字典的遍历顺序是基于插入顺序的，使用 sorted() 函数按字典的键进行排序，并生成一个新的字典。
  - sorted() 的 key 参数使用一个匿名函数（lambda），它从每个键值对中取出值进行排序。如：dict(sorted(my_dict.items(), key=lambda item: item[1]))
  - dic.items()获取所有键值对，.key()获取所有键.value()获取所有值
- set：set和dict类似，{}表示，也是一组key的集合，但不存储value。由于key不能重复，所以，在set中，没有重复的key。
  - set的原理和dict一样，所以，同样不可以放入可变对象可以放入int、float、str 元组（只要元组中的所有元素也是不可变的）布尔值，不可以放入列表（list）集合(set)字典（dict）
  - 交&，并|

深浅拷贝

- 浅拷贝新对象和原始对象的嵌套对象共享相同的引用，故修改嵌套对象会影响原始对象。
- 深拷贝会递归地拷贝所有嵌套对象，保证新对象与原始对象完全独立。 深拷贝使用 copy 模块中的 deepcopy() 函数实现：

```
import copy

original_list = [1, 2, [3, 4]]
deep_copy = copy.deepcopy(original_list)
```

生成式

- 列表生成式

```
[x for x in range(1, 11) if x % 2 == 0]
[x if x % 2 == 0 else -x for x in range(1, 11)]
[x for i in range(100) for x in (i, -i)]#（生成交替元素）
```

- 字典生成式

```
{key: value for key, value in zip(l, n)}
```

双端队列

```
from collections import deque
dq = deque([1, 2, 3])  # 使用列表初始化
dq.append(4)       # 在右侧添加
dq.appendleft(0)   # 在左侧添加
dq.pop()        # 移除右侧元素
dq.popleft()    # 移除左侧元素
```

最小堆（可实现优先队列）

```
import heapq

heap = [] # 创建一个空堆
heapq.heappush(heap, item)#将元素 item 插入堆中。
heapq.heappop(heap)#弹出并返回堆中的最小元素。
heapq.heapify(iterable)#将可迭代对象转为堆结构。
```

特殊读取

```
while True:
    try:
        n=input()
    except EOFError:
        break
```

enumerate()

返回一个迭代器，可以将其转换为列表，其中每个元素是 (index, value)

排序

reverse=True倒序

代码示范

递归（recursion）

贪心（greedy）

dp

- 最大整数：假设有n个正整数，要求从中选取几个组成一个位数不超过m的新正整数，现需要求出最大可能数值是多少。第一行m表示新数的最大位数。第二行n表示整数的数量。接下来一行有n个整数。输出为一行，为这n个正整数能组成的不超过m位的最大整数值。

```
m=int(input())
n=int(input())
l=list(map(lambda x:list(map(int,list(x))),input().split()))
def q(li):
    return li*(m//len(li))+li[:(m%len(li))]
l.sort(reverse=True,key=lambda x:q(x))
dp=[[0]*(m+1) for _ in range(n+1)]
for i in range(1,n+1):
    for j in range(1,m+1):
        dp[i][j]=max(dp[i-1][j],dp[i][j])
        if len(l[i-1])<=j:
            dp[i][j-len(l[i-1])]=max(dp[i-1][j]*(10**len(l[i-
1]))+int(''.join(map(str,l[i-1]))),dp[i-1][j-len(l[i-1])])
print(max(dp[-1]))
```

dfs

- 矩阵最大权值路径：现有一个n*m大小的矩阵，矩阵中的每个元素表示该位置的权值。现需要从矩阵左上角出发到达右下角，每次移动只能向上下左右移动一格（不允许移动到曾经经过的位置）。求最后到达右下角时路径上所有位置的权值之和的最大值。

```
n,m=map(int,input().split())
l=[[-1]*(m+2)]+[[-1]+list(map(int,input().split()))+[-1] for _ in range(n)]+[[-1]*
(m+2)]
def dps(visited,x, y,con):
    if x==n and y==m:
        return [con+l[x][y]]+visited+[(n,m)]
    else:
        neighbor=[(x+1,y),(x-1,y),(x,y+1),(x,y-1)]
        return max((dps(visited+[(x,y)],i[0],i[1],con+l[x][y]) for i in neighbor
if i not in visited and 0<i[0]<n+1 and 0<i[1]<m+1),key=lambda x: x[0],default=[-
float('inf'),[]])
```

```
for i in dps([],1,1,0)[1:]:
    print(' '.join(map(str,i)))
```

- 受祝福的平方

```
A=int(input())
def dps(x):
    s=str(x)
    if x**0.5==int(x**0.5):
        return True
    else:
        return any(dps(int(s[:i])) for i in range(len(s)) if
int(s[i:])**0.5==int(int(s[i:])**0.5) and int(s[i:])!=0)
if dps(A):
    print('Yes')
else:
    print('No')
```

bfs

- 体育游戏跳房子：将房子抽象为x轴上的连续的正整数坐标点，第i个房子的坐标为i，并假设房子个数无限。规则如下：操作记为H，则我们可以跳到当前坐标的3倍坐标位置。操作记为O，则我们需跳回当前坐标折半并向下取整的坐标位置。 请给出从第n个房子到第m个房子所需要的最少跳跃次数k和操作方法。 若最少跳跃次数下存在多种方法，则选取字典序最小的。输入包含多组。每组是两个正整数n和m。以0 0作为输入的结束。

```
from collections import deque

def bfs():
    re=0
    min=0
    visited=set()
    while l:
        do=l.pop()
        if do[0] in visited:
            continue
        else:
            visited.add(do[0])
        if re==1 and len(do)>min:
            break
        if do[0]==m:
            re=1
            min=len(do)
            note.append(do)
        if re==0:
            l.appendleft([do[0]*3]+do[1:]+['H'])
            l.appendleft([do[0]//2]+do[1:]+['O'])
while True:
    n,m=map(int,input().split())
```

```
    if n==0 and m==0:
        break
    l=deque([[n]])
    note=[]
    bfs()
    note.sort()
    print(len(note[0][1:]))
    print(''.join(note[0][1:]))
```

Dijkstra

- 走山路：想从一个地方走到另一个地方，并且希望能尽量走平路。 现有一个m*n的地形图，图上是数字代表该位置的高度，"#"代表该位置不可以经过。 该同学每一次只能向上下左右移动，每次移动消耗的体力为移动前后该同学所处高度的差的绝对值。现在给出该同学出发的地点和目的地，需要你求出他最少要消耗多少体力。 第一行是整数 m,n,p，m是行数，n是列数，p是测试数据组数。 0 <= m,n,p <= 100 接下来m行是地形图再接下来p行每行前两个数是出发点坐标（前面是行，后面是列）

```
import heapq
m,n,p=map(int,input().split())
l=[]
def mov(tl):
    return [(tl[0]+1,tl[1]),(tl[0]-1,tl[1]),(tl[0],tl[1]+1),(tl[0],tl[1]-1)]
for _ in range(m):
    l.append(input().split())
for _ in range(p):
    inp=list(map(int,input().split()))
    st=(inp[0],inp[1])
    en=(inp[2],inp[3])
    g=[[float('inf')]*n for _ in range(m)]
    tdl=[(0,st)]
    g[st[0]][st[1]]=0
    while tdl:
        if l[st[0]][st[1]]=='#' or l[en[0]][en[1]]=='#':
            print('NO')
            break
        i=heapq.heappop(tdl)[1]
        if i==en:
            print(g[i[0]][i[1]])
            break
        for t in mov(i):
            if 0<=t[0]<m and 0<=t[1]<n and l[t[0]][t[1]]!='#' and g[t[0]]
[t[1]]>g[i[0]][i[1]]+abs(int(l[i[0]][i[1]])-int(l[t[0]][t[1]])):
                g[t[0]][t[1]]=g[i[0]][i[1]] + abs(int(l[i[0]][i[1]]) - int(l[t[0]]
[t[1]]))
                heapq.heappush(tdl,(g[t[0]][t[1]],t))
    else:
        print('NO')
```

辅助栈

- 快速堆猪：有三种输入1)push n n是整数(0<=0 <=20000)，表示叠上一头重量是n斤的新猪2)pop表示将猪堆顶的猪赶走。如果猪堆没猪，就啥也不干3)min表示问现在猪堆里最轻的猪多重。如果猪堆没猪，就啥也不干

```
l,m=[],[]
while True:
    try:
        inp=input().split()
        if inp[0]=='push':
            l.append(int(inp[1]))
            if len(m)==0 or m[-1]>=int(inp[1]):
                m.append(int(inp[1]))
        elif inp[0]=='pop' and l:
            if l.pop()==m[-1]:
                m.pop()
        elif inp[0]=='min' and l:
            print(m[-1])
    except EOFError:
        break
```

二分查找

- 河中跳房子：在起点和终点之间，有N个岩石，每个岩石与起点的距离分别为Di。他计划移走一些岩石，使得从起点到终点的过程中，最短的跳跃距离最长。他可以移走除起点和终点外的至多M个岩石。确定移走这些岩石后，最长可能的最短跳跃距离是多少?

```
L,n,m=map(int,input().split())
l=[0]+[int(input()) for _ in range(n)]+[L]
cn=L
c=0
do=(c+cn)/2
while int(cn)-int(c)!=1:
    #print(c,cn)
    do=(c+cn)/2
    rm=0
    now=0
    for i in l[1:]:
        if i-now<do:
            rm+=1
        else:
            now=i
        if rm>m:
            cn=do
            break
    else:
        c=do
print(int(cn))
```

双指针

综合杂乱代码

- 滑雪

```
r,c=map(int,input().split())
l=[[float('inf')]*(c+2)]+[[float('inf')]+list(map(int,input().split()))+
[float('inf')] for _ in range(r)]+[[float('inf')]*(c+2)]
cl=[[0]*(c+2) for i in range(r+2)]
tdl=[]
def mov(tl):
    return [(tl[0]+1,tl[1]),(tl[0]-1,tl[1]),(tl[0],tl[1]+1),(tl[0],tl[1]-1)]
for i in range(1,r+1):
    for j in range(1,c+1):
        if l[i-1][j]>=l[i][j] and l[i+1][j]>=l[i][j] and l[i][j-1]>=l[i][j] and
l[i][j+1]>=l[i][j]:
            tdl.append((i,j))
            cl[i][j]=1
while tdl:
    td=tdl.pop()
    for i in mov(td):
        if 0<i[0]<r+1 and 0<i[1]<c+1 and l[i[0]][i[1]]>l[td[0]][td[1]] and
1+cl[td[0]][td[1]]>cl[i[0]][i[1]]:
            cl[i[0]][i[1]]=1+cl[td[0]][td[1]]
            tdl.append(i)
ma=0
for i in cl:
    ma=max(ma,max(i))
print(max(ma,1))
```

- 螃蟹采蘑菇

```
from collections import deque

def mov(li):
    return [((li[0][0]+1,li[0][1]),(li[1][0]+1,li[1][1])),((li[0][0]-1,li[0][1]),
(li[1][0]-1,li[1][1])),((li[0][0],li[0][1]+1),(li[1][0],li[1][1]+1)),((li[0]
[0],li[0][1]-1),(li[1][0],li[1][1]-1))]
n=int(input())
st=[]
l=[]
for _ in range(n):
    lil=list(map(int,input().split()))
    while 5 in lil:
        st.append((_,lil.index(5)))
        lil[lil.index(5)]=0
    if 9 in lil:
        ed=(_,lil.index(9))
    l.append(lil)
```

```
st=tuple(st)
tdl=deque()
tdl.append(st)
visited={st}
#print(visited)
ki=0
while tdl:
    td=tdl.pop()
    for i in mov(td):
        if i not in visited and 0<=i[0][0]<n and 0<=i[0][1]<n and 0<=i[1][0]<n and
0<=i[1][1]<n and l[i[0][0]][i[0][1]]!=1 and l[i[1][0]][i[1][1]]!=1:
            tdl.appendleft(i)
            visited.add(i)
            if i[0] == ed or i[1] == ed:
                print('yes')
                ki = 1
                break
    if ki==1:
        break
else:
    print('no')
```

- 两座孤岛最短距离

```
from collections import deque

n=int(input())
l=[]
for _ in range(n):
    l.append(list(map(int,list(input()))))
for i in range(n):
    start=0
    for j in range(n):
        if l[i][j]==1:
            start=(i,j)
            break
    if start!=0:
        break
tdl=deque([[start,0]])
# print(tdl[0])
visited=set([start])
move=[(0,1),(0,-1),(1,0),(-1,0)]
while True:
    td=tdl.pop()
    if td[1]!=0 and l[td[0][0]][td[0][1]]==1:
        print(td[1])
        break
    for i in move:
        vis=(td[0][0]+i[0],td[0][1]+i[1])
        if 0<=vis[0]<n and 0<=vis[1]<n and vis not in visited:
            visited.add(vis)
            if l[vis[0]][vis[1]]==1:
```

```
                tdl.append([vis,td[1]])
            else:
                tdl.appendleft([vis,td[1]+1])
```

- 积木

```
n=int(input())
l=[list(input()) for i in range(4)]
def dfs(visited,i):
    if i>=len(cl):
        return True
    return any([dfs(visited|{j},i+1) for j in range(len(cl[i])) if cl[i][j]==1 and
j not in visited])
for _ in range(n):
    nee=list(input())
    cl=[[1 if s in li else 0 for li in l] for s in nee]
    # print(cl)
    if dfs(set(),0):
        print('YES')
    else:
        print('NO')
```

- 小偷背包

```
N,B=map(int,input().split())
vl=list(map(int,input().split()))
wl=list(map(int,input().split()))
l=[[0,v,w] for v,w in zip(vl,wl)]
l.sort(reverse=True)
def tr(r,li):
    if len(li)==1:
        if li[0][2]<=r:
            return li[0][1]
        else:
            return 0
    if li[0][2]<=r:
        return max(tr(r,li[1:]),li[0][1]+tr(r-li[0][2],li[1:]))
    else:
        return tr(r,li[1:])
print(tr(B,l))
```

- 水淹七军

```
k=int(input())
def g(x):
    if hl[x[1]-1][x[2]]==-1 or hl[x[1]+1][x[2]]==-1 or hl[x[1]][x[2]-1]==-1 or
hl[x[1]][x[2]+1]==-1:
```

```
            return True
        else:
            return False
    for _ in range(k):
        if _!=0:
            stup=input()
        cont=1
        m,n=map(int,input().split())
        hl=[[False]*(n+2)]+[[False]+[-1]*n+[False] for _ in range(m)]+[[False]*(n+2)]
        l=[[False]*(n+2)]+[[False]+list(map(int,input().split()))+[False] for i in
    range(m)]+[[False]*(n+2)]
        gen_x,gen_y=map(int,input().split())
        p=int(input())
        need=[list(map(int,input().split())) for i in range(p)]
        high=l[gen_x][gen_y]+1
        hl[gen_x][gen_y]=True
        todolist = [(gen_x, gen_y)]
        wait = []
        while cont<m*n:
            while todolist:
                todo=todolist.pop()
                for i in [(todo[0]+1,todo[1]),(todo[0]-1,todo[1]),(todo[0],todo[1]+1),
    (todo[0],todo[1]-1)]:
                    if hl[i[0]][i[1]]==-1:
                        if l[i[0]][i[1]]<high:
                            hl[i[0]][i[1]]=False
                            todolist.append(i)
                        else:
                            hl[i[0]][i[1]]=True
                            wait.append((l[i[0]][i[1]],i[0],i[1]))
                        cont=cont+1
            wait=[i for i in wait if g(i)]
            wait.sort(reverse=True,key=lambda x: x[0])
            if len(wait)>0:
                high=wait[-1][0]+1
                todolist=[(wait[-1][1],wait[-1][2])]
                wait.pop()
        if any(hl[i[0]][i[1]] for i in need):
            print('Yes')
        else:
            print('No')
```