常用算法

for <variable变量> in <sequence序列>:

    <statements陈述>

python range() 函数可创建一个整数列表，一般用在 for 循环中。

range(start, stop[, step])

参数说明：

•start: 计数从 start 开始。默认是从 0 开始。例如range（5）等价于range（0， 5）;

•stop: 计数到 stop 结束，但不包括 stop。例如：range（0， 5） 是[0, 1, 2, 3, 4]没有5

•step：步长，默认为1。例如：range（0， 5） 等价于 range(0, 5, 1)

while <表达式判断>:

    <statements陈述>

一、解析法

二、枚举法

三、迭代法

四、二分查找法

五、直接插入排序法

六、递归法

一、解析法（analysis algorithm）是指用解析的方法找出表示问题的前提条件与结果之间关系的数学表达式，并通过表达式的计算来实现问题求解。

用解析法解决问题的关键就是要找出描述求解问题的解析表达式。

求三角形面积

第四种：海伦(*Heran*)公式，已知ΔABC中，$AB=c, BC=a, CA=b$，

$p=\frac{1}{2}(a+b+c)$，则三角形面积：

$$S=\sqrt{p(p-a)(p-b)(p-c)}$$

```
import math

a = float(input('请输入三角形第一边长: a = '))
b = float(input('请输入三角形第二边长: b = '))
c = float(input('请输入三角形第三边长: c = '))
if a + b > c and a + c > b and b + c > a:
    print('周长: %.0f' % (a + b + c))
    p = (a + b + c) / 2
    area = math.sqrt(p * (p - a) * (p - b) * (p - c))
    print('面积: %.2f' % (area))
else:
        print('不能构成三角形')
```

二、枚举法　破解密码

三元不定方程组

$$x+y+z=100$$
$$5x+3y+z/3=100$$

穷举法也叫枚举法或列举法。
在研究对象是由有限个元素构成的集合时，把所有对象一一列举出来，再对其一一进行研究。带入实际，一个个检验是否符合，穷举完所有对象问题将最终得以解决。

穷举法的思路与解题步骤

穷举法的基本思路：

把问题涉及的可能情况**逐个罗列**出来，并且根据题目的条件和实际背景逐个作出判断，从中挑选出符合条件的解答。

保留符合题意的，舍弃不符合的，既不重也不漏。
最大的缺点是运算量比较大

穷举法的解题步骤：

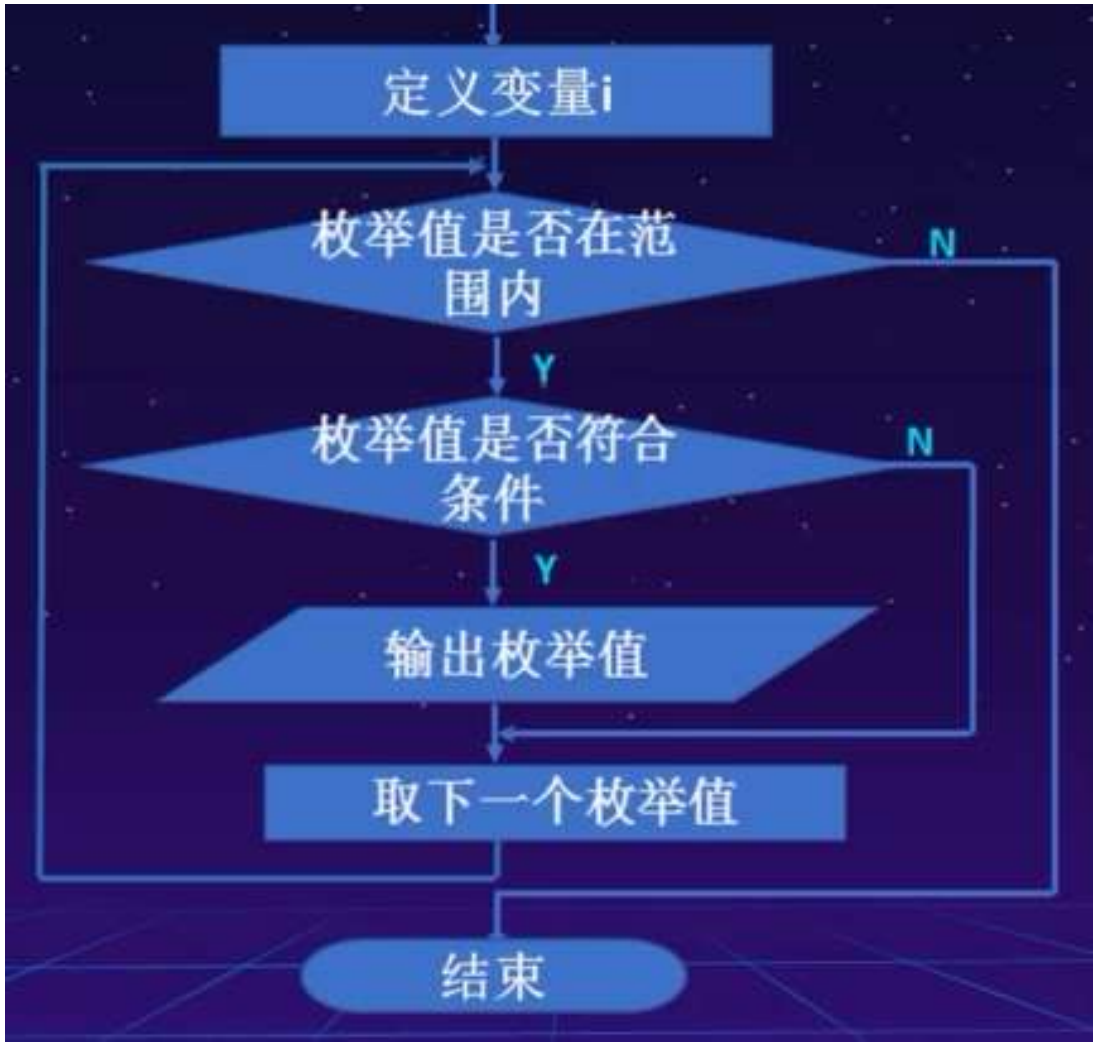
1）、分析问题，找出条件与未知数，确定变量；

2）、列举出变量所有可能的情况，用循环或循环的嵌套实现；

3）、写出符合条件的判断语句，用选择语句实现；

4）、输出符合条件的情况；

5）、优化程序。

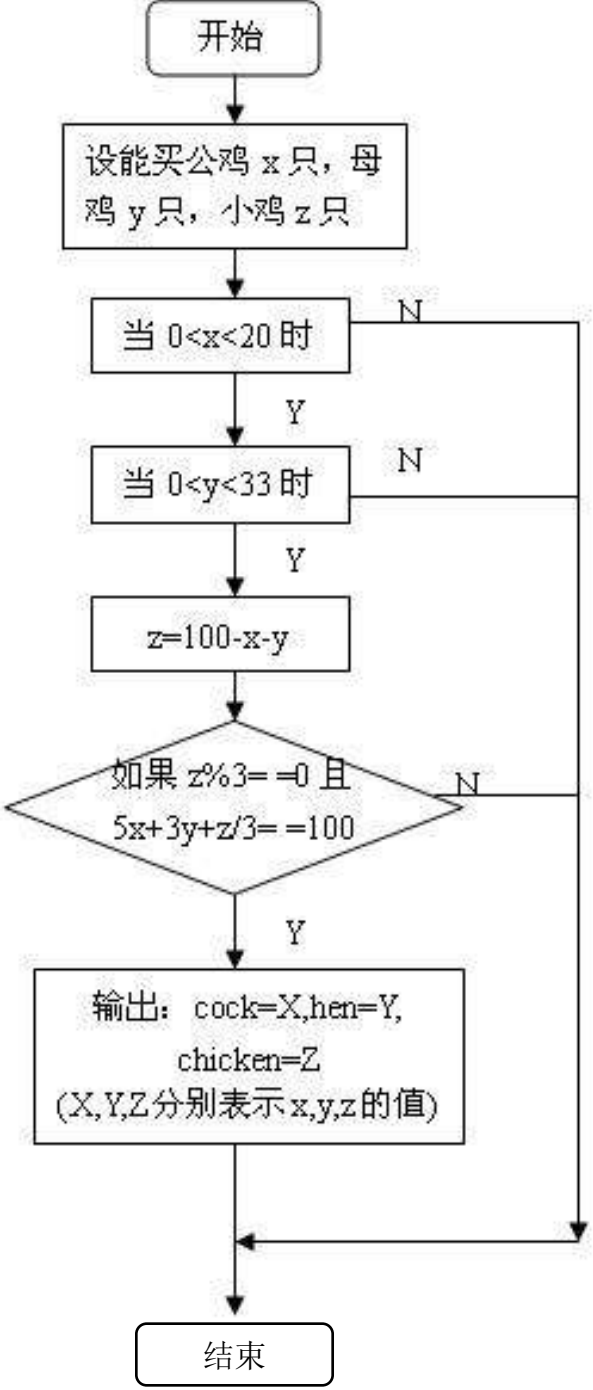


百钱买百鸡问题

用100元钱买100只鸡，公鸡每只5元，母鸡每只3元，小鸡3只1元，要求每种都买，问能买公鸡、母鸡、小鸡各买几只？

算法分析：

设x为公鸡数量，取值从1只到20只，y为母鸡的数量，取值从1只到33只，z为小鸡的数量，判断5元*x+3元*y+z/3是否等于100元，输出每次符合条件的x,y,z即为所求。


开始
设能买公鸡 x 只，母鸡 y 只，小鸡 z 只
当 0<x<20 时
N
Y
当 0<y<33 时
N
Y
z=100-x-y
如果 z%3= =0 且 5x+3y+z/3= =100
N
Y
输出： cock=X,hen=Y, chicken=Z (X,Y,Z分别表示x,y,z的值)
结束

```
#百钱买百鸡的问题算是一套非常经典的不定方程的问题，题目很简单：
#公鸡5文钱一只，
#母鸡3文钱一只，
#小鸡3只一文钱.

#coding=utf-8
chickNum=[['公鸡','母鸡','小鸡']]

#公鸡5元一只，数量x不大于20
for x in range(1,20) :

#    母鸡3元一只，数量y不大于30
    for y in range(1,30) :

#    共买100只鸡
        z=100-x-y
#        小鸡数量z为3的整数，且5x+3y+z/3=100
        if z%3==0 and 5*x+3*y+z/3==100  :

#            将符合条件的答案添加到列表
            chickNum.append([x,y,z])


#打印解题结果
print('\'百元买百鸡\'共有%d种方案：'%(len(chickNum)-1))
print()
for i in chickNum :
    for j in i :
        print(j,end='\t')
    print()
```

三、迭代法也称辗转法，是一种不断用变量的旧值推出新值的过程。S=S+i

迭代算法的基本思想是：
为求一个问题的解x，可由给定的一个初值x_0，根据某一迭代公式得到一个新的值x_1，这个新值x_1比初值x_0更接近要求的值x；再以新值作为初值，即：$x_1 \rightarrow x_0$，重新按原来的方法求x_1，重复这一过程直到$|x_1-x_0|<\varepsilon$(某一给定的精度)。此时可将x_1作为问题的解x。

应用：求根，棋盘放麦粒

四、二分查找

又称折半查找，它是一种效率较高的查找方法。

【二分查找要求】：

1.必须采用顺序存储结构

2.必须按关键字大小有序排列。

【算法思想】

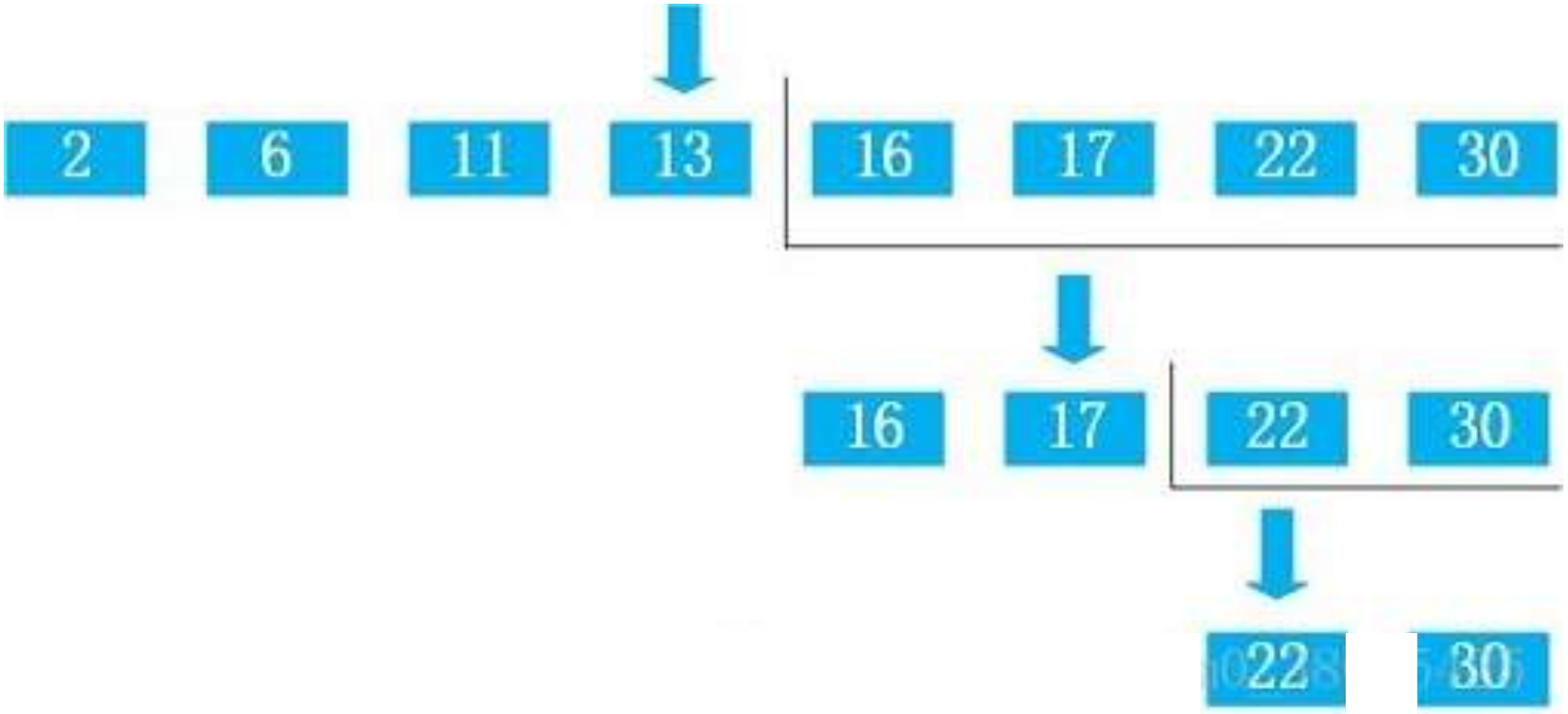
首先，将表中间位置记录的关键字与查找关键字比较，如果两者相等，则查找成功；

否则利用中间位置记录将表分成前、后两个子表，如果中间位置记录的关键字大于查找关键字，则进一步查找前一子表，否则进一步查找后一子表。

重复以上过程，直到找到满足条件的记录，使查找成功，或直到子表不存在为止，此时查找不成功。

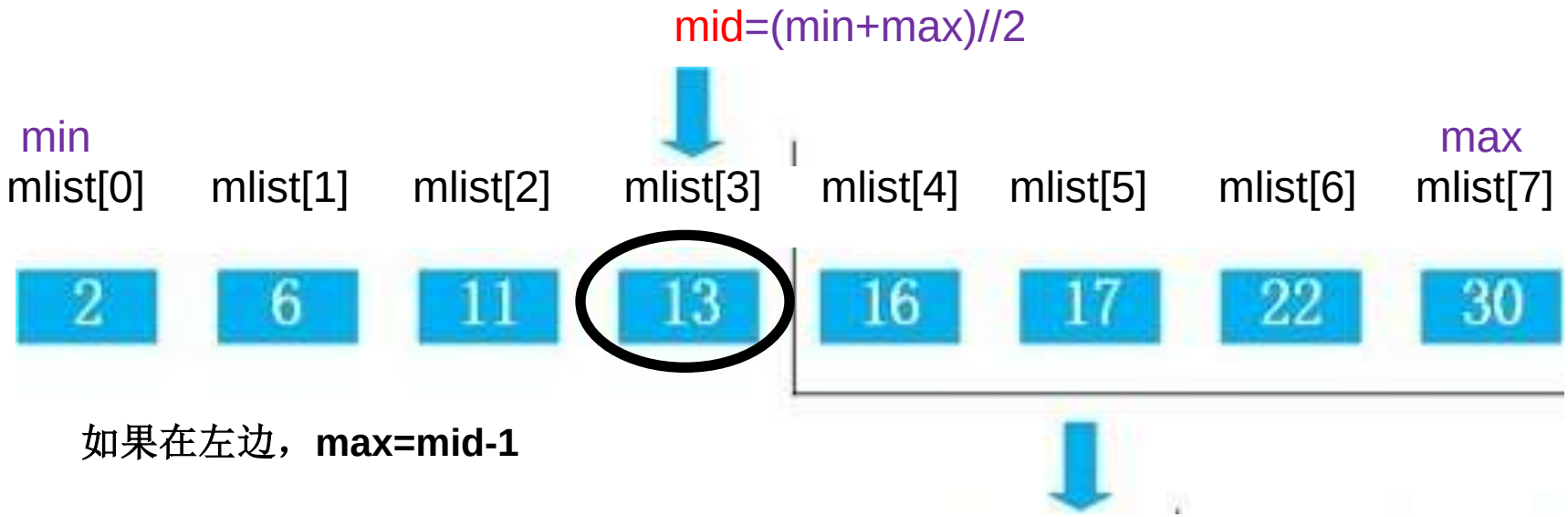
mlist=[2, 6,11,13,16,17, 22,30]

找22的位置



找22的位置

第一次



如果在左边，**max=mid-1**

如果在右边，**min=mid+1**

恰好相等

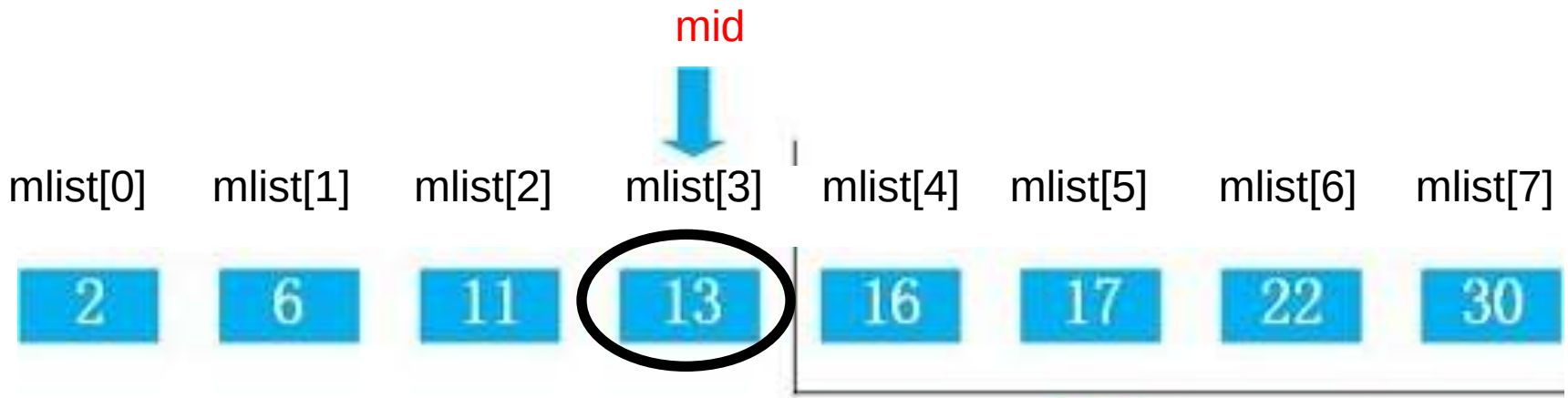
找22的位置

第一次

第二次

找22的位置

第一次



第二次

第三次

```python
def BSearch(lista, key):
    # 记录列表的最高位和最低位
    min = 0
    max = len(lista) - 1

    if key in lista:
        # 建立一个死循环，直到找到key
        while True:
            # 得到中位数
            mid = (min + max) // 2
            # key在列表左边
            if lista[mid] > key:
                max = mid - 1
            # key在列表右边
            elif lista[mid] < key:
                min = mid + 1
            # key在列表中间
            elif lista[mid] == key:

                print("你要查找的数字:",key,"在列表里面的第(%d)个位置"%(mid+1))
                return lista[mid]
    else:
        print("没有该数字!")


if __name__ == "__main__":
    arr=[1,6,9,15,26,38,49,57,63,77,81,93]
    print("已排序的一组数字是: ",arr)
    while True:
        key = input("请输入你要查找的数字: ")
        if key == "":
            print("谢谢使用！")
            break
        else:
            BSearch(arr, int(key))
```

【优缺点】折半查找法的优点是比较次数少，查找速度快，平均性能好;其缺点是要求待查表为有序表，且插入删除困难。因此，折半查找方法适用于不经常变动而查找频繁的有序列表。

应用：调监控录像

五、插入排序（理牌）

选择排序算法是要等数据全部输入到数组以后才开始进行排序操作的，而数据输入往往是一个很慢的过程，这样，计算机就要浪费大量时间等待。插入排序的思路是每一个数据进入时，在它前面输入的数据已经按顺序排好，它只需要插入到一个排好顺序的队列之中。

每次取出一个跟其左边那个元素比较，比左边邻居大说明不用动，比其小，就需要将该元素取出，然后遍历左半边有序序列部分，通过比较大小找出该插入的位置插之

六、递归算法

把问题分解成规模更小、与原问题相同解法的小问题。在函数实现时，因为解决大问题的方法和解决小问题的方法往往是同一个方法，所以就产生了函数调用它自身的情况，我们称之为递归法。

例如：斐波那契的兔子问题

著名的意大利数学家斐波那契在他的著作《算盘书》中提出了一个“兔子问题”：兔子在出生两个月后，就有繁殖能力，一对兔子每个月能生出一对小兔子来。如果年初养了一对小兔子，问到年底时将有多少对兔子？（当然得假设兔子没有死亡而且严格按照上述规律长大与繁殖）

分析问题：

	1月	2月	3月	4月	5月	6月	7月	8月	9月	10月	11月	12月
小兔	1	0	1	1	2	3	5	8	13	21	34	**55**
大兔	0	1	1	2	3	5	8	13	21	34	55	**89**
合计	**1**	**1**	**2**	**3**	**5**	**8**	**13**	**21**	**34**	**55**	**89**	**144**

设计算法：
假设第N个月的兔子数目是F(N)，我们有F(1)=F(2)=1

F(N)=F(N-1)+F(N-2)(当N>=3)

递归程序的特点是独立写出一个函数（或过程），而这个函数只对极简单的几种情况直接给出解答，而在其余情况下通过反复的调用自身而把问题归结到最简单的情况而得到解答。

汉诺塔游戏的目的是将所有的圆盘从塔座from移动到塔座to;
塔座by用来临时圆盘,

1、一次只能移动一个圆盘

2、任何时候都不能将一个较大的圆盘压在较小的圆盘上面.

```
def hanoi(n, f, b, t):
    if n == 1:
        print(f, '-->', t)
    else:
        hanoi(n - 1, f, t, b)
        print(f, '-->', t)
        hanoi(n - 1, b, f, t)

# 调用
hanoi(4, 'X柱', 'Y柱', 'Z柱')
```