1. 快速判断算法时间复杂度

**简单情况**

1. 确定问题规模n

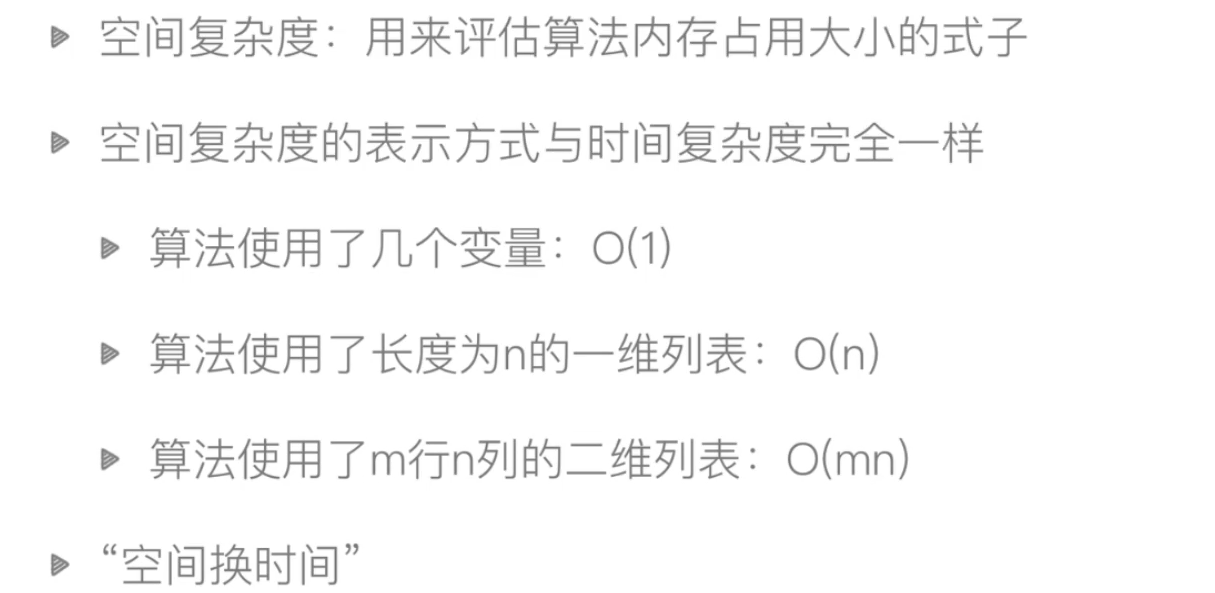
2. 循环减半过程——>logn

3. k曾关于n的循环——>n的k次方

**复杂情况：**

根据算法执行过程

2. 空间复杂度

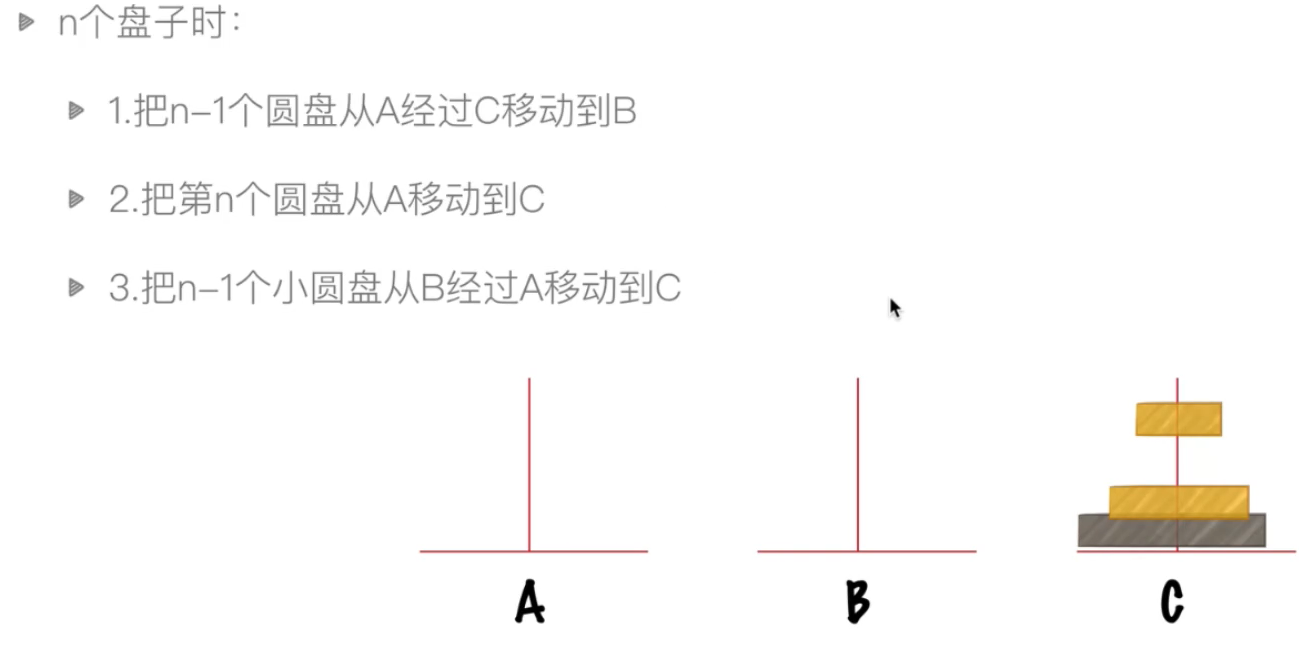


3. 递归

1. 循环自身

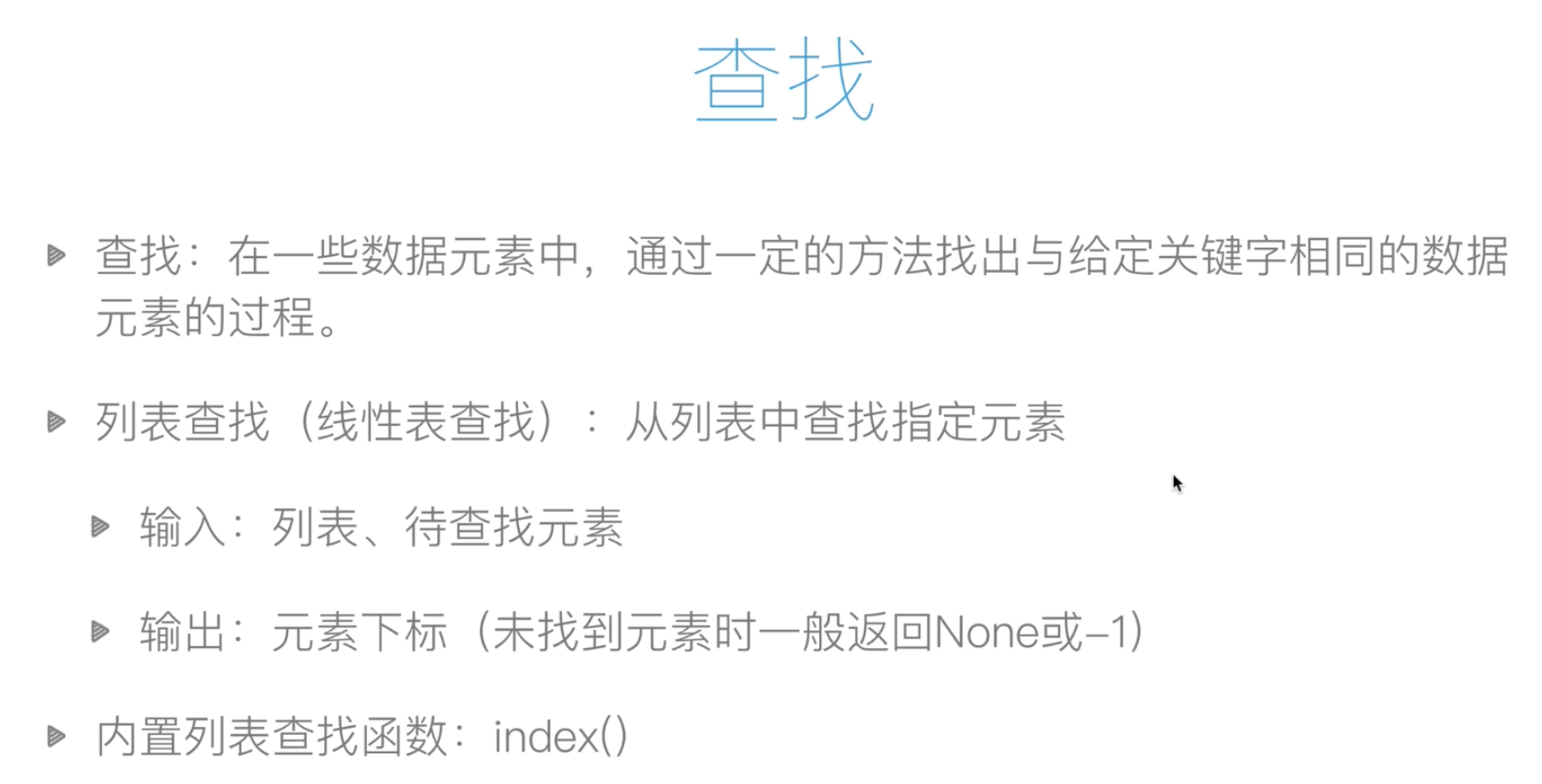
2. 结束条件

汉诺塔问题：



4. 查找

**1.顺序查找**



def search\_val(lis, v):

for index, val in enumerate(lis):

if val == v:

return index

else:

return None

a = [1, 2, 3, 4, 5]

b = search\_val(a, 5)

print(b)

# 列表顺序查找enumerate(lis， i) lis为可迭代对象，i表示下标开始位置默认为0

# dic = {

# "k1":"v1",

# "k2":"v2",

# "k3":"v3",

# }

# for i, key in enumerate(dic, 1):

# print(i,"\t",key)

**1.二分查找**

列表内置函数index()是顺序查找，因为二分查找需要有序

def binary\_search(lis, v):

left = 0

right = len(a)-1

while left <= right:

mid = (left+right)//2

# 表示

if v > lis[mid]:

left = mid + 1

if v < lis[mid]:

right = right - 1

if v == lis[mid]:

return v

else:

return None

c =binary\_search(a, 1)

print(c)