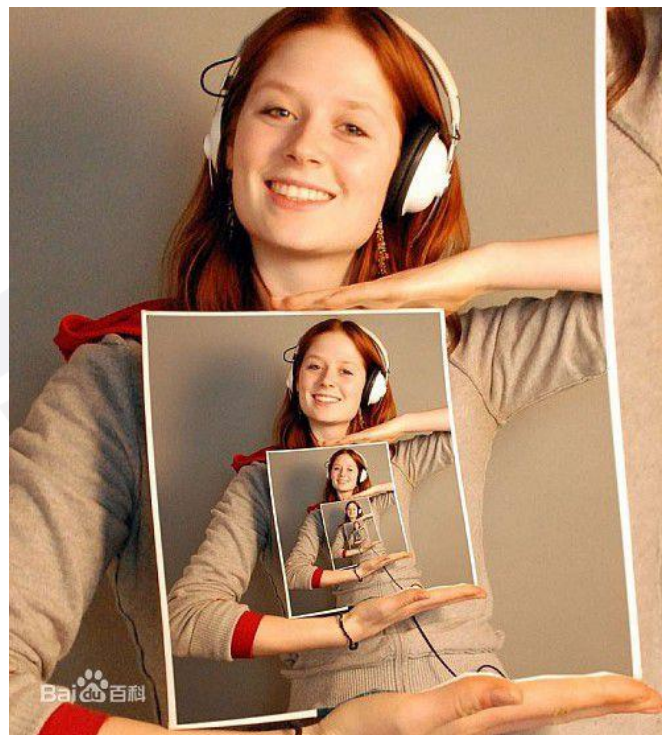


下面来看大家都熟悉的一个民间故事:

从前有座山，山上有座庙，庙里有一个老和尚在给小和尚讲故事：
从前有座山，山上有座庙，庙里有一个老和尚在给小和尚讲故事：
从前有座山，山上有座庙，庙里有一个老和尚在给小和尚讲故事：
从前有座山，山上有座庙，庙里有一个老和尚在给小和尚讲故事：
...

故事里包含了它本身



图片的内容包含了本身

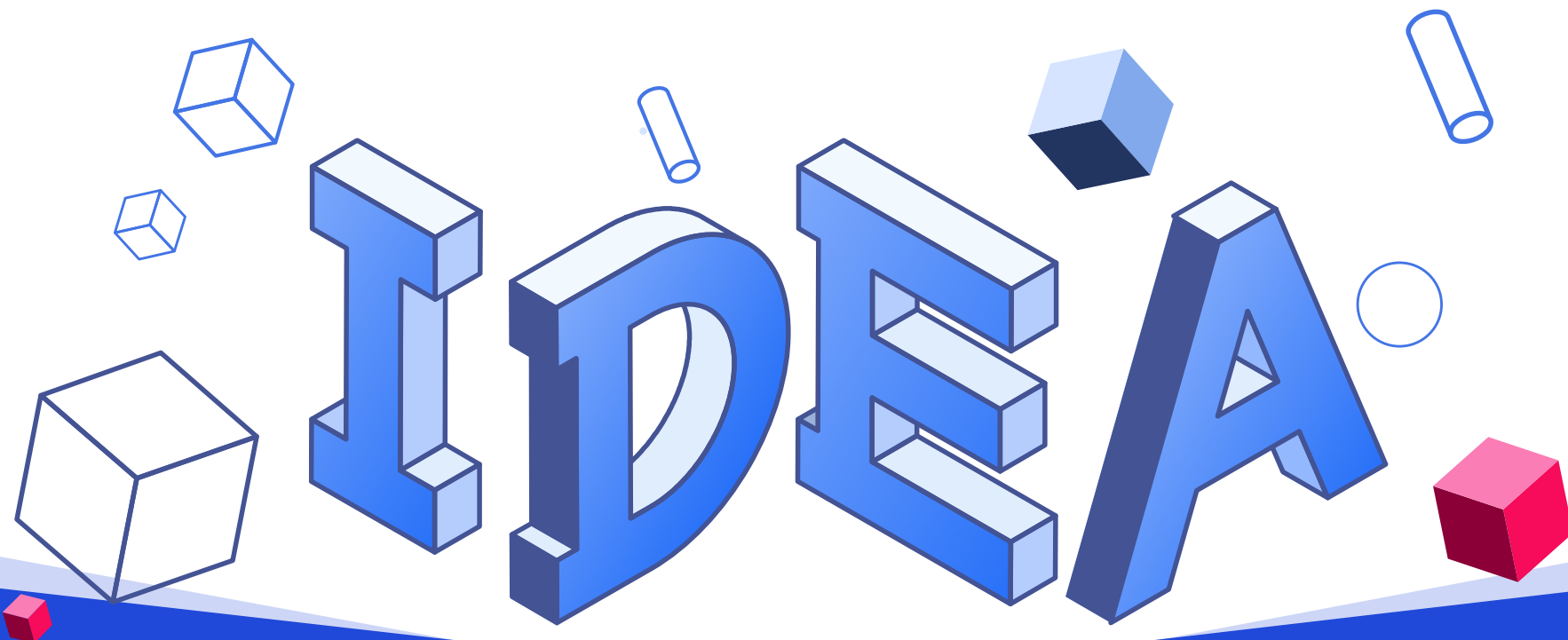


数学上的分形图



数学上的分形图

这些都有**递归**思想



第十九节课 初识递归

西南大学附属中学校
信息奥赛教练组



递归的定义参见递归的定义

在计算机科学领域

一个过程或函数在其定义中有直接或间接**调用自身**的一种方法



基本思想：把问题分解成规模更小，但和原问题有着相同步骤解法的问题，即子问题。

思考：实际所求的问题能否无限的分解下去？

必须存在一个能让递归调用退出的简单出口(称为边界条件)，
即递归不能无休止的进行下去。

1.递归的思想：将大问题转化为小问题求解。

对问题分解，将一个问题分解为规模较小的问题并用相同的步骤去解决。

2.求解步骤

(1).**确定递归公式**,即该问题的解能如何通过调用本身得到，一般通过数学推导、总结规律等方法得来

(2).**确定边界条件**(递归终止条件),一般是我们已知的条件



例1: 1~n求和



西南大学附属中学
High School Affiliated to Southwest University

计算 $1+2+3+\dots n$ 的值

若定义一个函数 $f(n)$,代表求1~n之间数的和

通过题目分析, 我们可以得到这样的一个关系式:

$$f(n)=f(n-1)+n \rightarrow \text{递归式}$$

$f(i)$ 称为一个递归函数

问题定义: $f(n)$

$$\begin{cases} 1 & n=1 \\ f(n-1)+n & n \geq 2 \end{cases}$$



$$f(n) \begin{cases} 1 & n=1 \\ f(n-1)+n & n \geq 2 \end{cases}$$

递归的边界条件,如何结束递归?

已知条件: $f(1)=1$



递归过程



西南大学附属中学
High School Affiliated to Southwest University

第一层递归状态

$$f(5)=f(4)+5$$

第二层递归状态

$$f(4)=f(3)+4$$

第三层递归状态

$$f(3)=f(2)+3$$

第四层递归状态

$$f(2)=f(1)+2$$

边界条件: $f(1)=1$

$$f(5)=f(4)+5=15$$

$$f(4)=f(3)+4=10$$

$$f(3)=f(2)+3=6$$

$$f(2)=f(1)+2=3$$

$$f(1)=1$$

递：从未知到已知

归：从已知回推到未知



递归过程



西南大学附属中学
High School Affiliated to Southwest University

第一层递归状态

$$f(5)=f(4)+5$$

第二层递归状态

$$f(4)=f(3)+4$$

第三层递归状态

$$f(3)=f(2)+3$$

第四层递归状态

$$f(2)=f(1)+2$$

边界条件: $f(1)=1$

$$f(5)=f(4)+5=15$$

$$f(4)=f(3)+4=10$$

$$f(3)=f(2)+3=6$$

$$f(2)=f(1)+2=3$$

$$f(1)=1$$

递：从未知到已知

归：从已知回推到未知

Q: 下一层状态得到的结果如何返回给上一层?

return语句



递归定义式: $f(n) \begin{cases} 1 & n=1 \\ f(n-1)+n & n \geq 2 \end{cases}$

递归条件: $f(1)=1$

递归代码:

```
int f(int n){  
    if(n==1) return 1; //边界条件  
    else return f(n-1)+n; //递归执行f(n-1)+n  
}
```



例2：求阶乘n!



西南大学附属中学
High School Affiliated to Southwest University

$$f(n) \begin{cases} 1 & n=0 \\ f(n-1)*n & n>0 \end{cases}$$

若定义一个函数f(n)来计算n!的阶乘

可得一个计算n! 的关系式： **$f(n)=f(n-1)*n$**

边界条件? **$n=0$ 时, $f(0)=1$;**



```
int f(int n){  
    if(n==0) return 1; //边界条件  
    else return f(n-1)*n; //递归执行f(n-1)*n  
}
```



例2、走楼梯



西南大学附属中学
High School Affiliated to Southwest University

题目描述

楼梯有 n ($n \leq 20$) 级台阶，上楼一步可以上1级，也可以一步上2级。编写程序计算共有多少种不同的走法。

输入

一行：n

输出

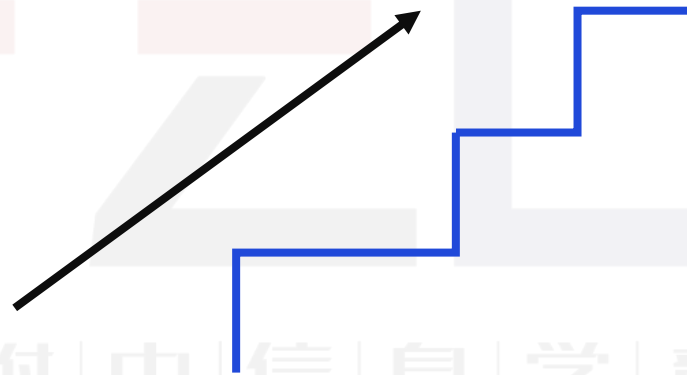
一行：有多少种走法

样例输入

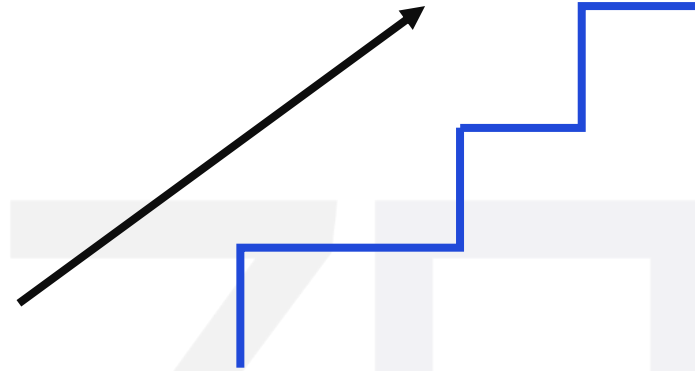
2

样例输出

2



西 | 大 | 附 | 中 | 信 | 息 | 学 | 竞 | 赛 |
High School Affiliated to Southwest University



若定义函数 $f(n)$ 代表着， n 阶楼梯的走法总数：

首先我们考虑最简单的情况：如果只有1级台阶，那显然只有一种跳法， $f(1)=1$ ；如果有2级台阶，那就有两种跳的方法了：一种是分两次跳，每次跳1级；另外一种就是一次跳2级， $f(2)=2$ 。


现在我们来讨论一般情况：

我们把n级台阶时的跳法看成是n的函数，记为 $f(n)$ ，当 $n > 2$ 时：
第一次跳的时候就有两种不同的选择：一是第一次只跳1级，此时跳法数目等于后面剩下的 $n-1$ 级台阶的跳法数目，即为 $f(n-1)$ ；
另外一种选择是第一次跳2级，此时跳法数目等于后面剩下的 $n-2$ 级台阶的跳法数目，即为 $f(n-2)$ 。

因此n级台阶时的不同跳法的总数： $f(n) = f(n-1) + f(n-2)$ 。

边界条件： $f(1)=1$ ， $f(2)=2$  递归的边界条件可能不止一个



问题定义式: $f(n) \begin{cases} 1 & n=1 \\ 2 & n=2 \\ f(n-1)+f(n-2) & n>2 \end{cases}$  斐波拉契数列

递归代码:

```
int stair(int n){  
    if(n<=2) return n;  
    else return stair(n-1) + stair(n-2);  
}
```



例2：汉诺塔(hanio)



西南大学附属中学
High School Affiliated to Southwest University

设a, b, c是三个塔座, 开始时, 在塔座a上有一叠共n个圆盘, 这些圆盘自下而上, 由大到小地叠放在一起, 各圆盘从小到大编号为1, 2, 3, ..., n。现要求将**塔座a上的一叠圆盘移到塔座c**上, 并仍按同样顺序叠置。

在移动圆盘是应遵守以下移动规则:

- (1) 每次只能移动一个圆盘;
- (2) 任何时刻都不允许将较大的圆盘压在较小的圆盘之上;
- (3) 在满足移动规则 (1) 和 (2) 的前提下, 可以将圆盘移至a, b, c中任一塔座上。

要求打印出若干行, 每行表示盘子的一次移动

如: 1 a->c 表示将1号圆盘从a塔座移到c塔座

输入

一行: n表示表示初始时a塔有n个圆盘

输出

未知行数

每行表示一次移动:

格式: 圆盘编号 塔的编号->塔的编号

如: 1 a->c 表示将1号圆盘从a塔座移到c塔座

样例输入

2

样例输出

1 A->B

2 A->C

1 B->C

西 | 大 | 附 | 中 | 信 | 息 | 学 | 竞 | 赛
High School Affiliated to Southwest University



只有2个盘的情况



西南大学附属中学
High School Affiliated to Southwest University



a



b



c

移动过程:

第一步: 第一个盘从a移动到b

第二步: 第二个盘从a移动到c

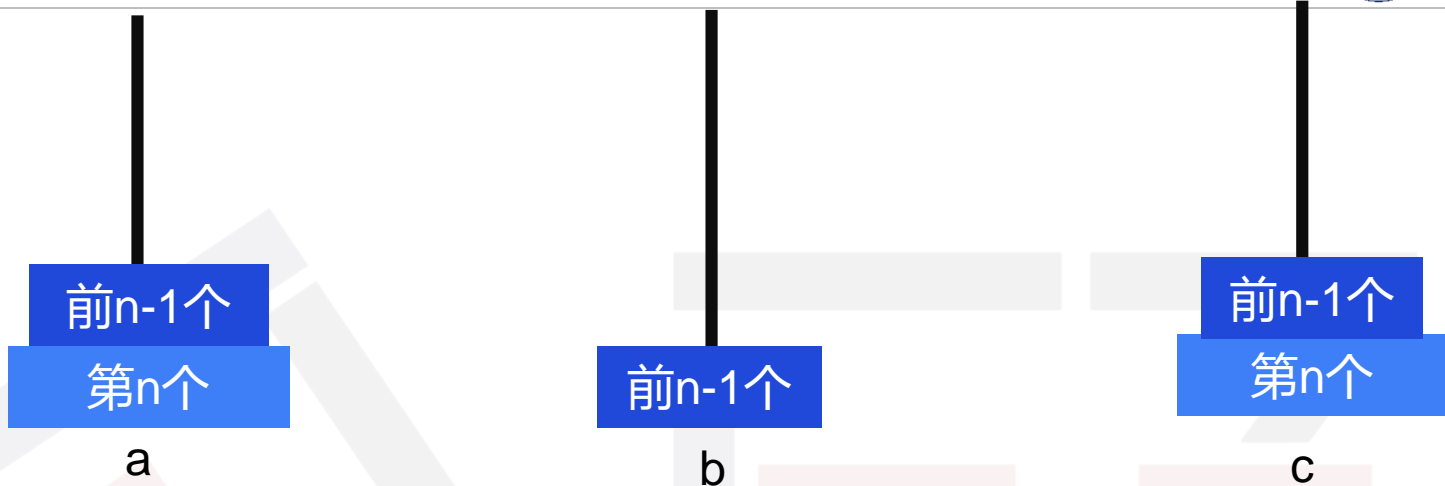
第三步: 第一个盘从b移动到c, 完成



更一般的情况



西南大学附属中学
High School Affiliated to Southwest University



第一步：前 $n-1$ 个盘从a移动到b

第二步：第 n 个盘从a移动到c

第三步：前 $n-1$ 个盘从b移动到c，完成

我们总结可以发现：

- 对于前 $n-1$ 个盘子，是从a借助了b，移动到了c
- 对于第 n 个盘子，是从a没有借助其他柱子，直接移动到了c



问题分解：n的规模不断减小

初步确定递归函数的参数之一：**n**



考虑更具体的移动情况



西南大学附属中学
High School Affiliated to Southwest University

值得注意的是：

- 在前面，我们是把前 $n-1$ 个盘子当作一个盘子看待，没有去关心它内部的转移情况
- 虽然前 $n-1$ 个盘子最终转移的效果是从 a 借助 b 到 c ，但内部的转移时可能还有其他的转移方式，才能有这样的最终效果。
- 当有多个盘子转移的时候，它们都会有一个**起点柱、终点柱**以及**帮助转移的柱子**



递归函数的另外的参数：起点柱a,帮助转移的柱子b,终点柱c

定义递归函数： `Hanoi(int n,char a,char b,char c)`

整个解决流程：

第一步：前n-1个盘从a移动到b,借助c

`Hanoi(n-1,a,c,b)`

第二步：第n个盘直接从a移动到c

输出移动情况

第三步：前n-1个盘从b移动到c,借助a，完成

`Hanoi(n-1,b,a,c)`

边界条件是什么？

当n=1,只有一个盘需要移动的时候，问题不能再分解，直接输出。

参数定义的关键：如何准确地描述问题的转化



n=2的情况:

最初的调用: Hanoi(2,'A','B','C');

函数执行:

- 1.Hanoi(n-1,a,c,b); Hanoi(1,'A','C','B'); 此时n=1 输出1 A->B, 结束调用
- 2.printf("%d %c->%c\n",n,a,c); 此时n=2 输出2 A->C
- 3.Hanoi(n-1,b,a,c); Hanoi(1,'B','A','C'); 此时n=1 输出1 B->C, 结束调用



```
void Hanoi(int n,char a,char b,char c) {  
    if (n==1)printf("%c -> %c\n",a,c); //只有一个盘子, 直接将其移到c  
    else  
    {  
        Hanoi(n-1,a,c,b); //第一步, A借助C, 将n-1个盘子移到B;  
        printf("%c -> %c\n",a,c); //第二步, 将A上剩余的一个盘移到C, 输出移动情况;  
        Hanoi(n-1,b,a,c); //第三步, 将B上的n-1个盘子移到C。  
    }  
}
```

如何实现盘子编号的输出?



递归函数



西南大学附属中学
High School Affiliated to Southwest University

```
void Hanoi(int n,char a,char b,char c) {  
    if (n==1)printf("%d %c->%c\n",n,a,c);  
    else  
    {  
        Hanoi(n-1,a,c,b);  
        printf("%d %c->%c\n",n,a,c);  
        Hanoi(n-1,b,a,c);  
    }  
}
```

还欠缺一句话，请思考

Hanoi(n-1,a,c,b); //第一步，A借助C，将n-1个盘子移到B；

printf("%d %c->%c\n",n,a,c); //第二步，将A上剩余的一个盘移到C，输出移动情况；

Hanoi(n-1,b,a,c); //第三步，将B上的n-1个盘子移到C。

Q：为什么输出n就可以了？

Q：n是不是在递归时，永远都是我们最初传入的值？

变量n是一个局部变量，递归时每次函数调用执行时都会生成一个新的、独立的n。

Thanks

For Your Watching



练手题:1212 走楼梯

1392 数组求和

1393 二分查找

1213 汉诺塔(hanio)

1477 求最大公约数

进阶题:1216 数的组合

1217 自然数拆分

1220 选数

1401 逆波兰表达式

1395 集合的划分

1440 分解因数

1222 2的幂次方