递归

山东省实验中学

宁华

一个故事


从前有座山，山里有个庙，
庙里有个老和尚
给小和尚讲故事，讲的那是
从前有座山，
山里有个庙，
庙里有个老和尚
给小和尚讲故事
讲的那是
从前有座山...

几张图片——德罗斯特效应

可是到底什么是递归呢？

- 谷歌又调皮了~

知乎~

- https://www.zhihu.com/question/20507130
- 一个洋葱是一个带着一层洋葱皮的洋葱。

知乎大神的回答之一：

- 天下有奇族人姓计，长生不老。一日其孙问其父：吾之18代祖名何？
- 其父不明，父问其父
- 其父不明，父问其父
- 其父不明，父问其父
- …
- 晌后，其18代祖回其子：你猜
- 然其回其子：你猜
- 然其回其子：你猜
- 然其回其子：你猜
- ……
- 终，计姓末代孙知其18代祖名“你猜”

- 此乃，递归。

知乎大神的回答之二：

- 古之欲明明德于天下者，先治其国；欲治其国者，先齐其家；欲齐其家者，先修其身；欲修其身者，先正其心；欲正其心者，先诚其意；欲诚其意者，先致其知，致知在格物。物格而后知至，知至而后意诚，意诚而后心正，心正而后身修，身修而后家齐，家齐而后国治，国治而后天下平。
- 这是一个调用自身的过程，我们把“明德于天下”当作函数本身来理解，每一层调用的参数依次是治国、齐家、修身、正心、诚意、致知、格物。最后在“格物”触发返回条件。

知乎大神的回答之三：

- 假设你在一个电影院，你想知道自己坐在哪一排，但是前面人很多，你懒得去数了，于是你问前一排的人「你坐在哪一排？」，这样前面的人 (代号 A) 回答你以后，你就知道自己在哪一排了——只要把 A 的答案加一，就是自己所在的排了。不料 A 比你还懒，他也不想数，于是他也问他前面的人 B「你坐在哪一排？」，这样 A 可以用和你一模一样的步骤知道自己所在的排。然后 B 也如法炮制。直到他们这一串人问到了最前面的一排，第一排的人告诉第二排问问题的人「我在第一排」，然后第二排的人再告诉第三排的人……。最后大家就都知道自己在哪一排了。

递归

- 递归是一种思想。
- 递归是一种思维方式。
- 递归是一种算法。

- “To Iterate is Human, to Recurse, Divine.”
- “人理解迭代，神理解递归。”
- 难得糊涂

自然数的递归定义

- 0是自然数;
- 如果n是自然数，则n+1也是自然数。

递归算法的实现

- 递归算法，通常是借助于递归函数来实现的。
- 一个函数直接或间接调用自己——递归函数

一个简单的例子

- 计算n的阶乘：n!
- n!=1*2*...*(n-1)*n
- 规定0!=1
- （数据范围0<=n<=20）
- 样例输入：5
- 样例输出：120

分析：

- 1、循环实现

- 小细节：
- 数据范围预估--->数据类型

分析：

- 2、递归实现

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

递归出口

递归式

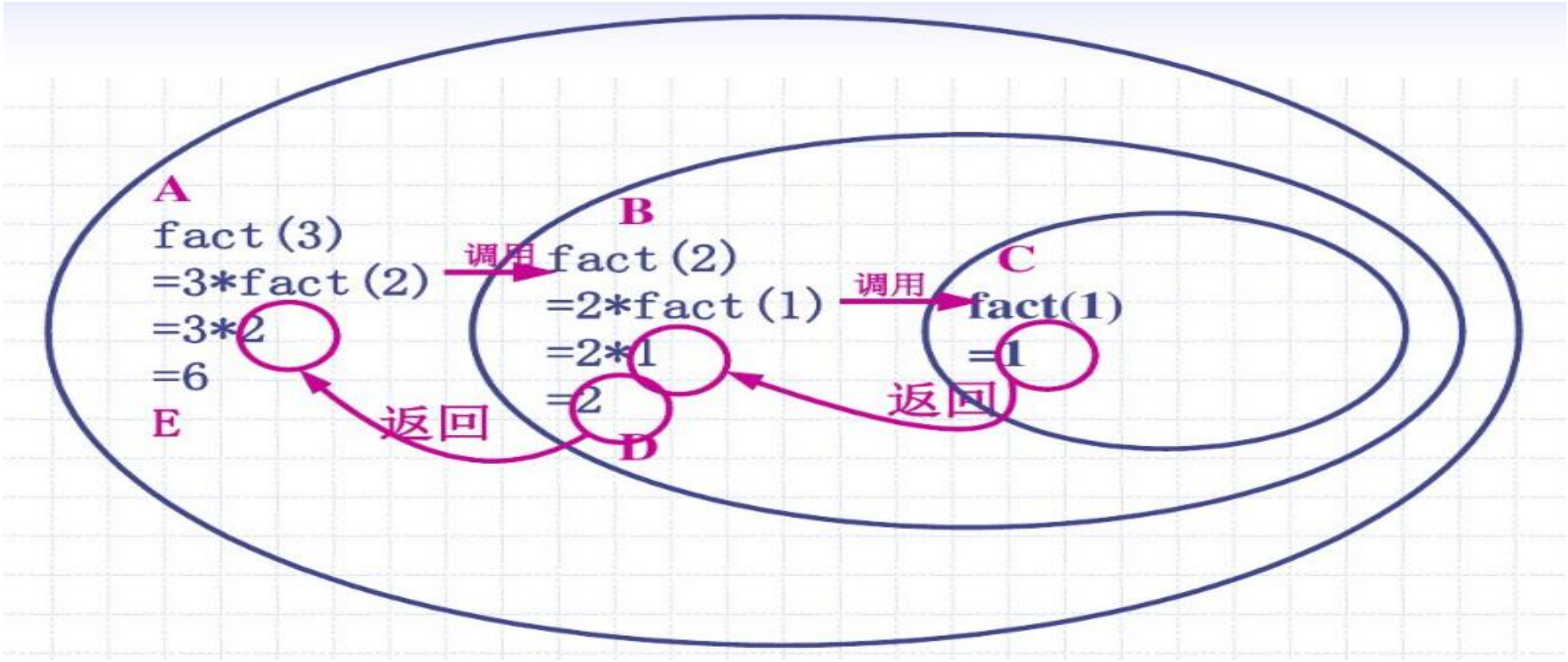
参考代码：

```cpp
#include<bits/stdc++.h>
using namespace std;
#define LL long long
LL f(int n)
{
    if(n==0)return 1;
    return n*f(n-1);
    // return n ? n*f(n-1) : 1;
}
int main()
{
    int n;
    cin>>n;
    cout<<f(n)<<endl;
    return 0;
}
```

递归出口

递归式

递归过程分析



小结：

- 递归过程必须解决两个问题

- 跳进递归——递归式
- 跳出递归——递归出口

小结：

- 递归过程的算法描述框架：

- if (到达递归出口)

  返回递归出口处的函数值;

- else

  递归计算公式并返回结果;

随堂小练习

- 输入a和n的值，求a^n。（a,n<=10）
- 样例输入：2 3
- 样例输出：8

- 请写出求a^n的递归函数，并用代码实现。

练习

- 用递归法求一个整数数组中所有元素的平均值。
- 样例输入：
- 3
- 1 2 3
- 样例输出：
- 2.00

```
float ave(int n)
{
    if(n==1)return a[1];
    return (ave(n-1)*(n-1)+a[n])/n;
}
int main()
{
    int n;
    cin>>n;
    for(int i=1;i<=n;i++)
        cin>>a[i];
    printf("%.2f\n",ave(n));
}
```

练习

- 输入x、n，求f(x,n)。

$$f(x,n)=\sqrt{n+\sqrt{n-1+\sqrt{n-2+\sqrt{\ldots\ldots+\sqrt{1+x}}}}}$$

练习

- 输入x、n，求f(x,n)。

$$f(x,n)=\cfrac{x}{n+\cfrac{x}{n-1+\cfrac{x}{n-2+\ddots+\cfrac{x}{1+x}}}}$$

练习

- 输入x、n，计算勒让德多项式的值。

$$p_n(x)=\begin{cases}1 & n=0\\ x & n=1\\ ((2n-1)p_{n-1}(x)-(n-1)p_{n-2}(x))/n & n>1\end{cases}$$

练习

- 任意输入一串字符，以#号结束。要求逆序输出这串字符（#号不输出）。要求使用递归方法，不使用字符串类型。
- 样例输入：
- good dog#
- 样例输出：
- god doog

```
void rev(char c)
{
        if(c=='#')return;
        char cc=getchar();
        rev(cc);
        cout<<c;
}
int main()
{
        char c;
        c=getchar();
        rev(c);
}
```

例：递归经典问题之——兔子数列

月份	1	2	3	4	5	6	7	8	9	10	11	12
小兔子	1											
中兔子												
大兔子												

斐波那契数列的递推与递归实现分析

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

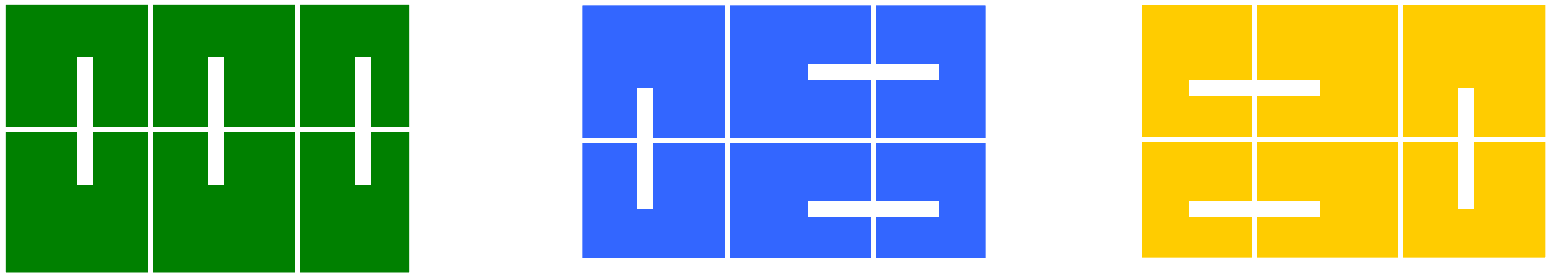
练习：走楼梯

- 一个含有n阶的楼梯，一次可以走1阶或2阶，从底走到顶一共有多少种走法？
- 输入：台阶数n
- 输出：不同走法种数
- 输入样例：3
- 输出样例：3

练习：骨牌覆盖问题

- 在2×n的长方形方格中，用n个1×2的骨牌铺满方格，输入n，输出铺放方案的总数。

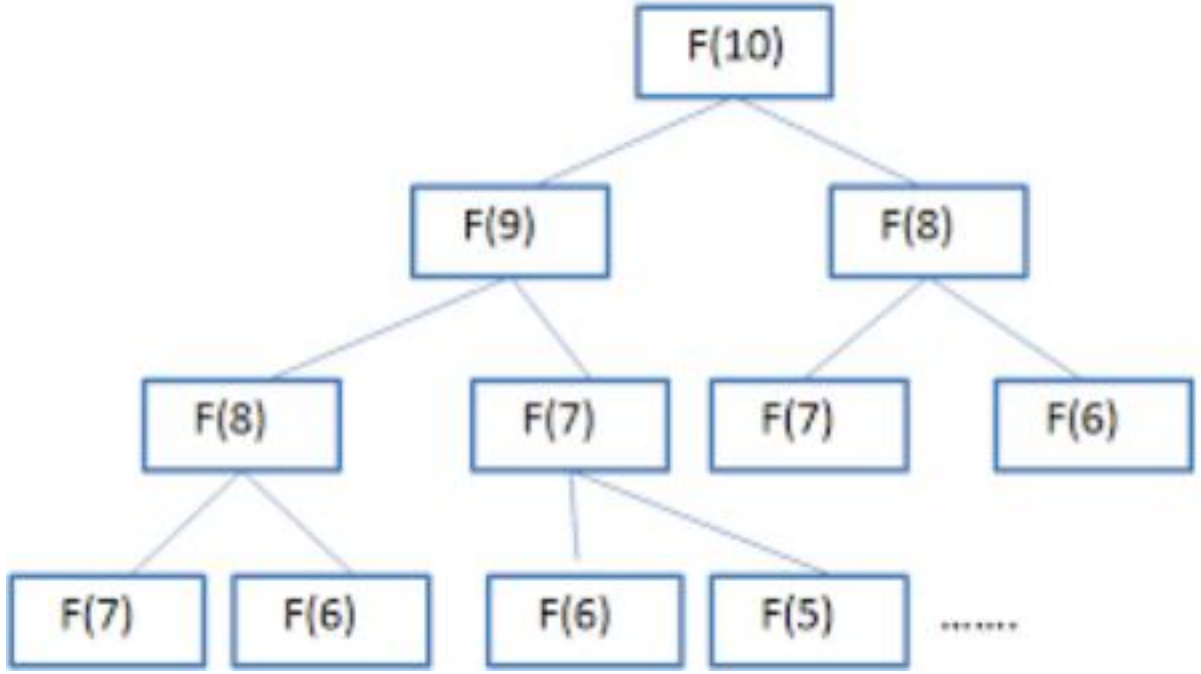
- 例如 n=3时，为2×3方格，骨牌的铺放方案有三种方法，如下图所示：



递归效率分析

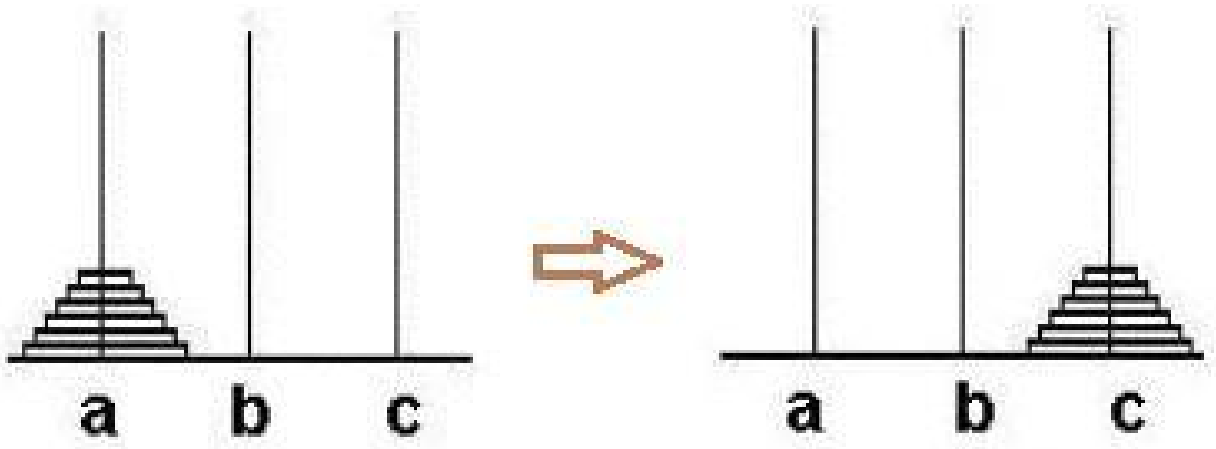
- 递归的优缺点

- 解决效率低的方法：
- 1、记忆化
- 2、转递推



例：递归经典问题之——Hanoi塔问题

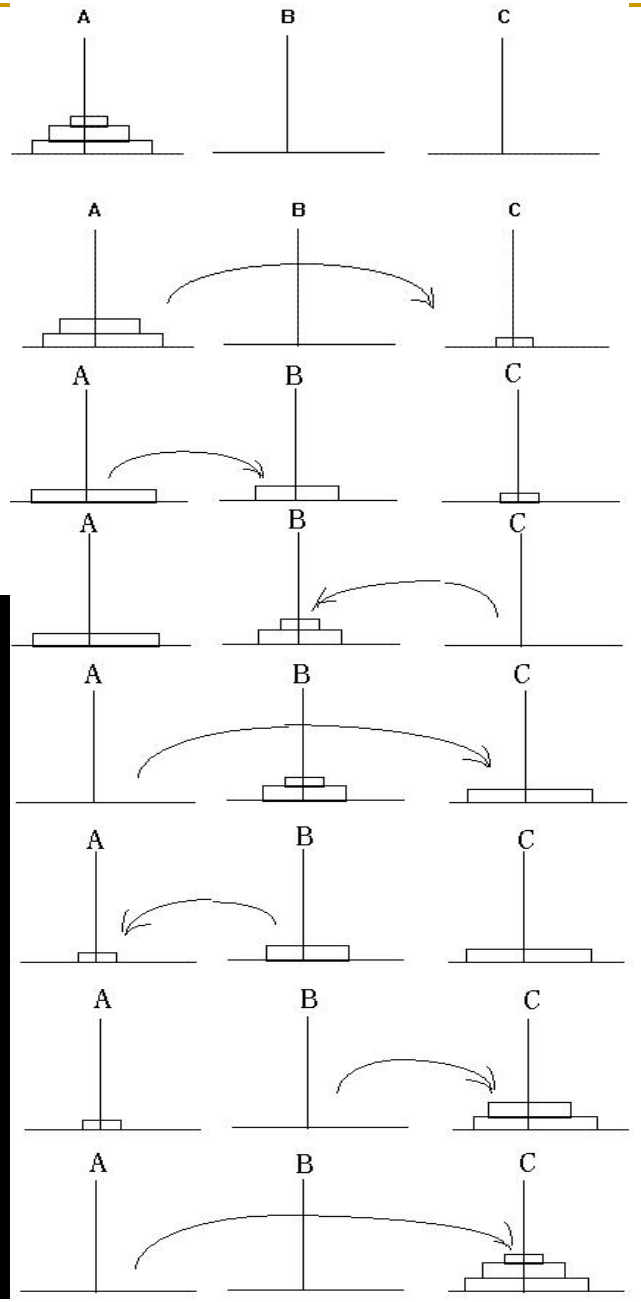
- 法国数学家爱德华·卢卡斯曾编写过一个印度的古老传说：在世界中心贝拿勒斯（在印度北部）的圣庙里，一块黄铜板上插着三根宝石针。印度教的主神梵天在创造世界的时候，在其中一根针上从下到上地穿好了由大到小的64片金片，这就是所谓的汉诺塔。不论白天黑夜，总有一个僧侣在按照下面的法则移动这些金片：一次只移动一片，不管在哪根针上，小片必须在大片上面。僧侣们预言，当所有的金片都从梵天穿好的那根针上移到另外一根针上时，世界就将在一声霹雳中消灭，而梵塔、庙宇和众生也都将同归于尽。



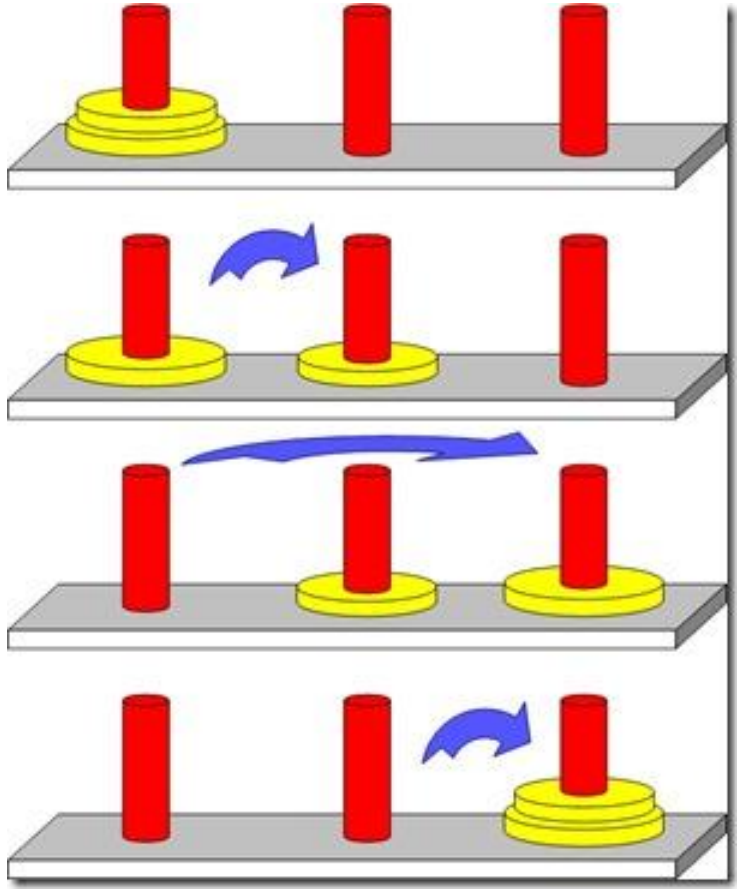
hanoi塔问题

- 输入n
- 输出移动次数最少的移动方案，格式见样例。
- 样例输入：3
- 样例输出：

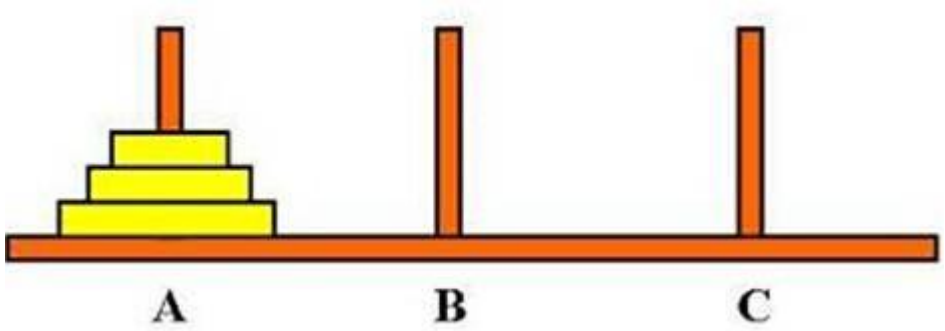
```
Step 1: Move 1 from A to C
Step 2: Move 2 from A to B
Step 3: Move 1 from C to B
Step 4: Move 3 from A to C
Step 5: Move 1 from B to A
Step 6: Move 2 from B to C
Step 7: Move 1 from A to C
```



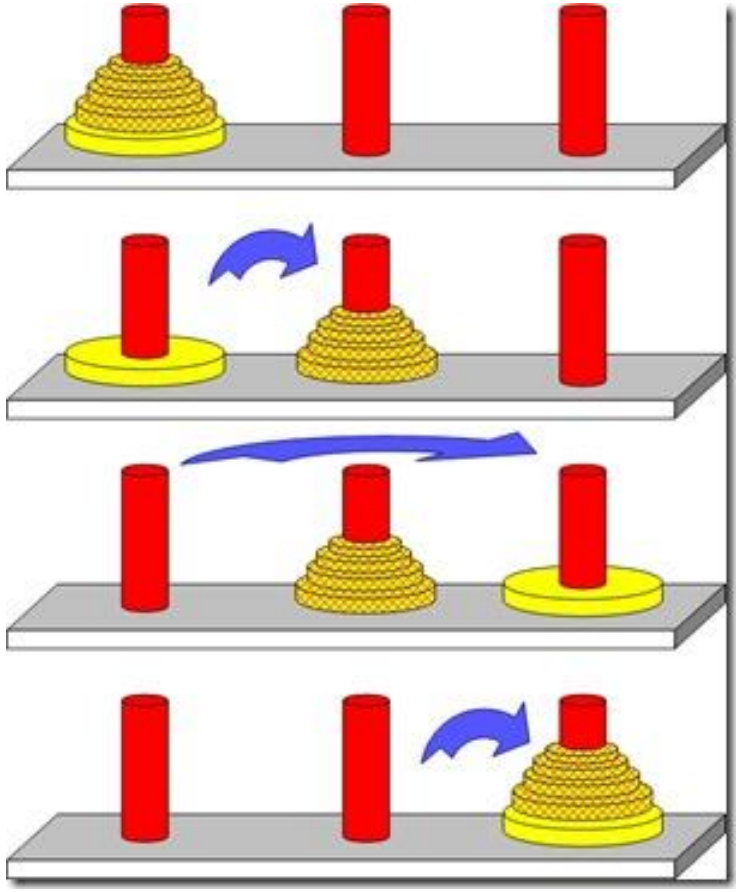
2个盘子的汉诺塔



3个盘子的汉诺塔



n个盘子的汉诺塔



算法分析

- 分治与递归

参考代码：

```
#include<iostream>
using namespace std;
int cnt;
void hanoi(int n,char a,char b,char c)
{
    if(n==0)return;
    hanoi(n-1,a,c,b);
    cout<<"Step "<<++cnt<<": Move "<<n<<" from "<<a<<" to "<<c<<endl;
    hanoi(n-1,b,a,c);
}
int main()
{
    int n;
    cin>>n;
    hanoi(n,'A','B','C');
    return 0;
}
```

例：求最大公约数

- 输入a,b求最大公约数。(0<=a,b<=2*10^9)
- 样例输入：
- 98 63
- 样例输出：
- 7

- 有哪些方法？

例：求最大公约数

- 穷举法
- 分解质因数法
- 辗转相除法（欧几里得算法）
- 更相减损法
- ……

辗转相除法、更相减损法

辗转相除法参考代码：

```
#include<iostream>
using namespace std;
int gcd(int m,int n)
{
        //return n ? gcd(n, m%n) : m;
        if(n==0)return m;
        return gcd(n,m%n);
}
int main()
{
        int a,b;
        cin>>a>>b;
        cout<<gcd(a,b)<<endl;
}
```

练习

- 完成更相减损法求最大公约数的程序。

思考与总结

- 什么是递归?
- 何时用递归?
- 如何用递归?
- 递归优缺点?

再举两个例子

- 感受下

- 递归的威力！
- 递归的精彩~

- 关于递归的效率缺点，我们暂不考虑~

例

- n个不同的球，任取m个，有多少种不同取法？
- 样例输入：
- 3 2
- 样例输出：
- 3

分析

- 1、数学方法
- 组合数

分析

- 2、递归
- 递归求解，数学公式验算

参考代码：

```
long long f(int n,int m)
{
    if(n<m)return 0;
    if(n==m)return 1;
    if(m==0)return 1;
    return f(n-1,m-1)+f(n-1,m);
}
int main()
{
    int n,m;
    cin>>n>>m;
    cout<<f(n,m)<<endl;
    return 0;
}
```

例 最长公共子序列的长度

- 例如两个字符串：
- “ababbcda”
- “abaacd”
- 最长公共子序列的长度为：
- 5

分析

- 分情况讨论：
- (1)
- (2)
- (3)

参考代码：

```
string s1,s2;
int len1,len2;
int lcs(int n, int m){
    if(n==len1||m==len2)return 0;
    if(s1[n] == s2[m])return lcs(n+1,m+1) + 1;
    return max(lcs(n+1,m),lcs(n,m+1));
}
int main(){
    cin>>s1>>s2;
    len1=s1.length(),len2=s2.length();
    cout<<lcs(0,0);
    return 0;
}
```

暴力递归

- 效率低下
- 大量的重复子问题
- 更好的方法——记忆化递归/动态规划 课下自学

其他应用举例

例：十进制转二进制

- 输入一个十进制整数N（0<=N<=10^9），输出其二进制形式。
- 样例输入：13
- 样例输出：1101

分析

- 1、直接模拟求解过程

- 2、递归

```
void convert(int n)
{
    if(n<=1) cout<<n;//递归出口，当n等于0或1时，直接输出
    else
    {
        convert(n/2);
        cout<<n%2;//输出放在递归调用之后，因为输出是反向的
    }
}
int main()
{
    int n;
    cin>>n;
    convert(n);
    cout<<endl;
    return 0;
}
```

例：判断回文串

- 输入一个字符串，判断是否为回文串，是则输出YES，否则输出NO。
- 样例输入：good
- 样例输出：NO
- 样例输入：12321
- 样例输出：YES

分析

- 1、设首尾指针

- “good”
- “12321”

```
cin>>s;
int len=s.length();
int l=0,r=len-1;
bool ok=true;
while(l<r)
{
    if(s[l]!=s[r])
    {
        ok=false;
        break;
    }
    l++,r--;
}
if(ok)cout<<"YES"<<endl;
else cout<<"NO"<<endl;
```

- 2、利用递归
- 先首尾判断

- 若首尾不同，则肯定不是回文

- 若首尾相同，则去掉首尾后，
- 判断剩余中间的新串是否回文串

```
string s;
bool f(string s)
{
    int n=s.length();
    if(n<=1)return true;
    if(s[0]!=s[n-1])return false;
    s.erase(0,1);
    s.erase(s.length()-1,1);
    return f(s);
}
int main()
{
    cin>>s;
    if(f(s))cout<<"YES"<<endl;
    else cout<<"NO"<<endl;
    return 0;
}
```

例 快速求幂

- 输入a、n、p，求a^n%p。（a,n,p<=10^9）
- 样例输入：2 3 3
- 样例输出：2

分析：以求2^19为例，暂不取模

- 1、朴素（暴力）

- 2^19
- =2*2*2……*2
- 18次乘法

分析：以求2^19为例，暂不取模

- 2、基于分治
- 2^19=2^18*2
- 2^18=2^9*2^9
- 2^9=2^8*2
- 2^8=2^4*2^4
- 2^4=2^2*2^2
- 2^2=2*2
- 6次乘法

参考代码：

```
int a,n,p;
int f(int a,int n,int p)
{
    if(n==0)return 1%p;
    int s=f(a,n/2,p);
    s=(1LL*s*s)%p;
    if(n%2)s=1LL*s*a%p;
    return s;
}
int main()
{
    cin>>a>>n>>p;
    cout<<f(a,n,p)<<endl;
    return 0;
}
```

```
int a,n,p;
int f(int n)
{
    if(n==0)return 1%p;
    int s=f(n/2);
    s=(1LL*s*s)%p;
    if(n%2)s=1LL*s*a%p;
    return s;
}
int main()
{
    cin>>a>>n>>p;
    cout<<f(n)<<endl;
    return 0;
}
```

分析：以求2^19为例，暂不取模

- 3、基于二进制拆分的快速幂

- 这是目前最常用的求快速幂的方法，课下自学。

例：装错信封

- 有n封信和n个信封，如果所有的信都装错了信封，共有多少种不同情况？
- 样例输入：
- 3
- 样例输出：
- 2

例：打靶

- 一个射击运动员打靶，靶一共有10环，连开10枪打中90环的可能打法有多少种？
- 比如，以下是两种不同的打法：
- 9 9 9 9 9 9 9 9 9 9
- 9 9 9 9 9 9 9 9 8 10

递归

- 递归是重要算法，是很多算法的基础。
- 历年NOIP中，递归应用几乎无处不在。

例 NOIP2001-普及组复赛-第1题-数的计算

- 问题描述
- 我们要求找出具有下列性质数的个数(包含输入的自然数n):
- 先输入一个自然数n(n<=1000),然后对此自然数按照如下方法进行处理:
- 1. 不作任何处理;
- 2. 在它的左边加上一个自然数,但该自然数不能超过原数的一半;
- 3. 加上数后,继续按此规则进行处理（每一次加的数不超过上一次加的数的一半）,直到不能再加自然数为止.

- 样例1:
- 输入：6
- 输出：6
- 样例1解释：满足条件的数为 6个 (此部分不必输出)
- 16
- 26
- 126
- 36
- 136

- 样例2：
- 输入：20
- 输出：60
- 样例2解释：
- 20
- 1020，920，820，……，220，120
- 51020，4920，4820，……　1220
- 251020　，24920，24820，……
- 1251020，124920，124820，……
- 共计60个。

例 NOIP1998-普及组复赛-第1题-2的幂次方表示

- 描述
- 任何一个正整数都可以用2的幂次方表示。例如：
- 137=2^7+2^3+2^0
- 同时约定方次用括号来表示，即a^b可表示为a(b)。由此可知，137可表示为：
- 2(7)+2(3)+2(0)
- 进一步：7=2^2+2+2^0（2^1用2表示）
- 3=2+2^0
- 所以最后137可表示为：
- 2(2(2)+2+2(0))+2(2+2(0))+2(0)
- 又如：
- 1315=2^10+2^8+2^5+2+1
- 所以1315最后可表示为：
- 2(2(2+2(0))+2)+2(2(2+2(0)))+2(2(2)+2(0))+2+2(0)

- 输入
- 一个正整数n（n≤20000）。
- 输出
- 一行，符合约定的n的0，2表示（在表示中不能有空格）。
- 样例输入
- 137
- 样例输出
- 2(2(2)+2+2(0))+2(2+2(0))+2(0)

结束语

- 递归有两个过程：
- 1、把大问题一步一步往小问题分解；
- 2、从最小的问题开始，根据递推关系一步一步得到新的解决方案，最后得到所得解。
- 所以，递归的本质其实还是递推关系，没有这个递推关系，就无法产生递归定义的合理性，合理性就是要求能够调用函数的本身，函数可以永远适用，这里面的思想就是使用一种相同的方法来描绘整个世界。只不过，递归还想强调一种分治的思想，这种思想在解决复杂问题中有非常重要的应用。

The end.

Thanks.