4/30/2023

int cnt = 0;

void f(int l, int r) {

if(l > r)return ;

if(l == r)

{

cnt ++ ;

return ;

}

cnt ++ ;

int mid = (l + r) / 2;

f(l, mid);

f(mid + 1, r);

return ;

}

int main() {

f(1,1000);

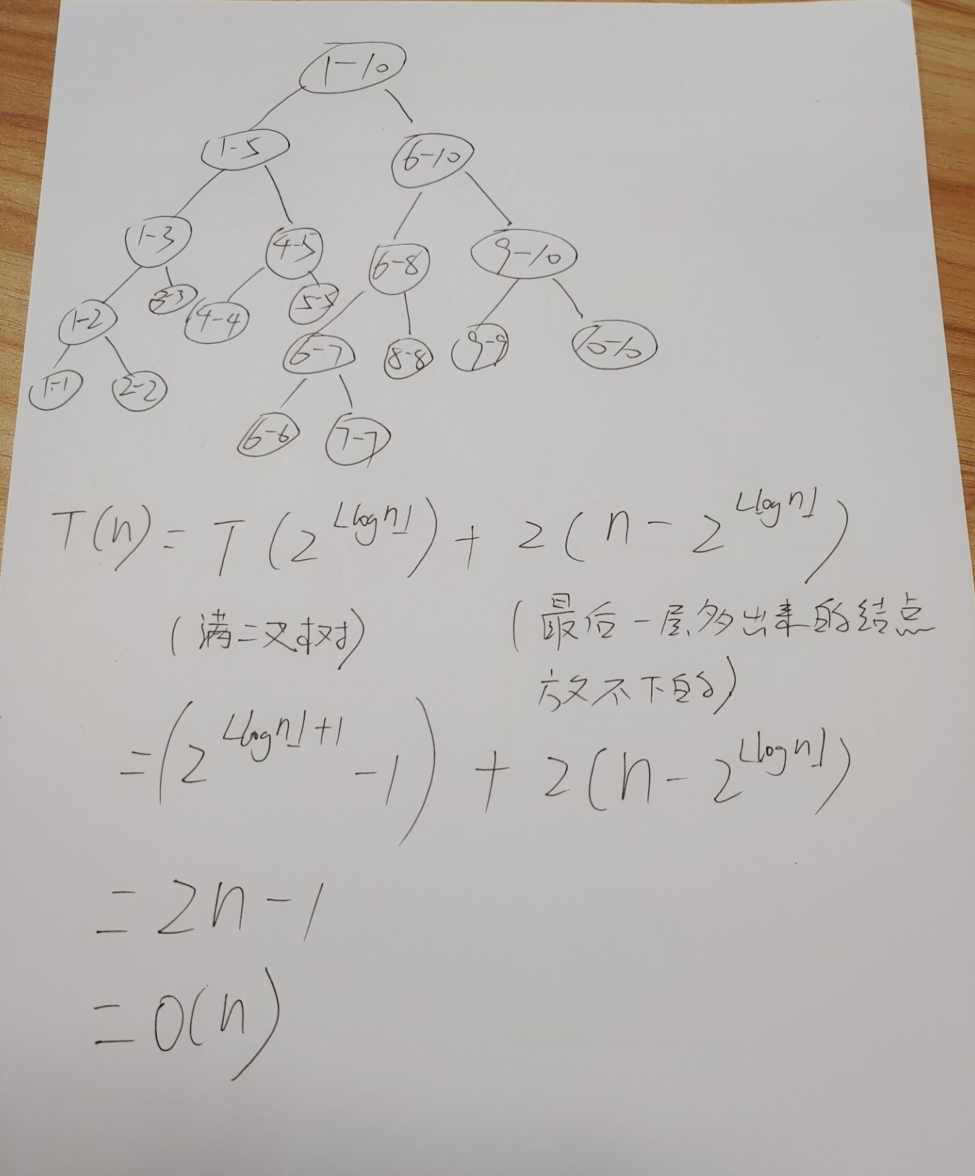
printf("%d",cnt);

}

易得，题目就是要求(l<=r)的递归函数的执行次数。

下面给出n=10的递归树的图示，以及递归函数执行次数cnt=T(n)=2n-1的证明过程

结论成立，故cnt=T(1000)=1999



6个圆盘的汉诺塔，总的移动次数是63

f(n)=f(n-1)+1+f(n-1)=2f(n-1)+1

最后f(n)=2^n-1

int f(int n)

{

if(n < 1) return 0;

if(n <= 3) return 1;

return f(n - 1) + f(n - 2) + f(n - 3);

}

这个和爬楼梯的算法问题差不多，需要推导出递推公式一个个计算（比较麻烦，有大佬有更好的方法的话欢迎补充）

理解题目：题目可以类比成爬楼梯的算法问题，一个人迈一步可以跨过1或2或3个台阶，现在一个人从20阶台阶高度下来，问一共有多少种走法可以走到台阶数为1到3之间的台阶。

初始化参数：

首先显然，若人在台阶1到3之间（包括1和3）时已经符合要求，可以直接返回1，所以，f（1）到 f（3）= 1；

当 n = 4时，走一步到3，走两步到2，走三步到1，三种走法都满足范围条件，所以 f（4） = 3；

当 n = 5时，走一步到4，根据 ii 知道有3种方法，走两步到3，走三步到2，所以有5种方法，即 f（5） = 5；

当 n = 6，7时，依照上述方法推导得 f（6） = 9 ，f（7）= 17（用于递推公式的验证，也可以运行代码求得）

推导递推公式

根据上述规律（走了一步后到达前一个点，可以直接使用该点的f（n）值，走两步后……）

所以当n >= 4时，f（n）= f（n-1）+f（n-2）+f（n-3）

所以当n >= 5时，f（n-1）= f（n-2）+f（n-3）+f（n-4）

两式相减得递推公式：f（n） = 2f（n-1）-f（n-4）

验证：

根据式子求得 ：

f（5）= 2f（4）-f（1）=2X3-1 =5

f（6）= 2f（5）-f（2）=2X5-1 =9

f（7）= 2\*f（6）-f（3）=2X9-1 =17

（后续也是符合规律的，想验证的同学可以运行代码求得正确的数进行比较）

求解

然后……就开始痛苦的计算了，一个个进行递推

f（8） = 2Xf（7）- f（4） = 2X17 - 3 = 31

……（中间就省略了）

f（20） = 2f（19）- f（16）=2X25281 - 4063=46499

long long gcd(long long x, long long y) {

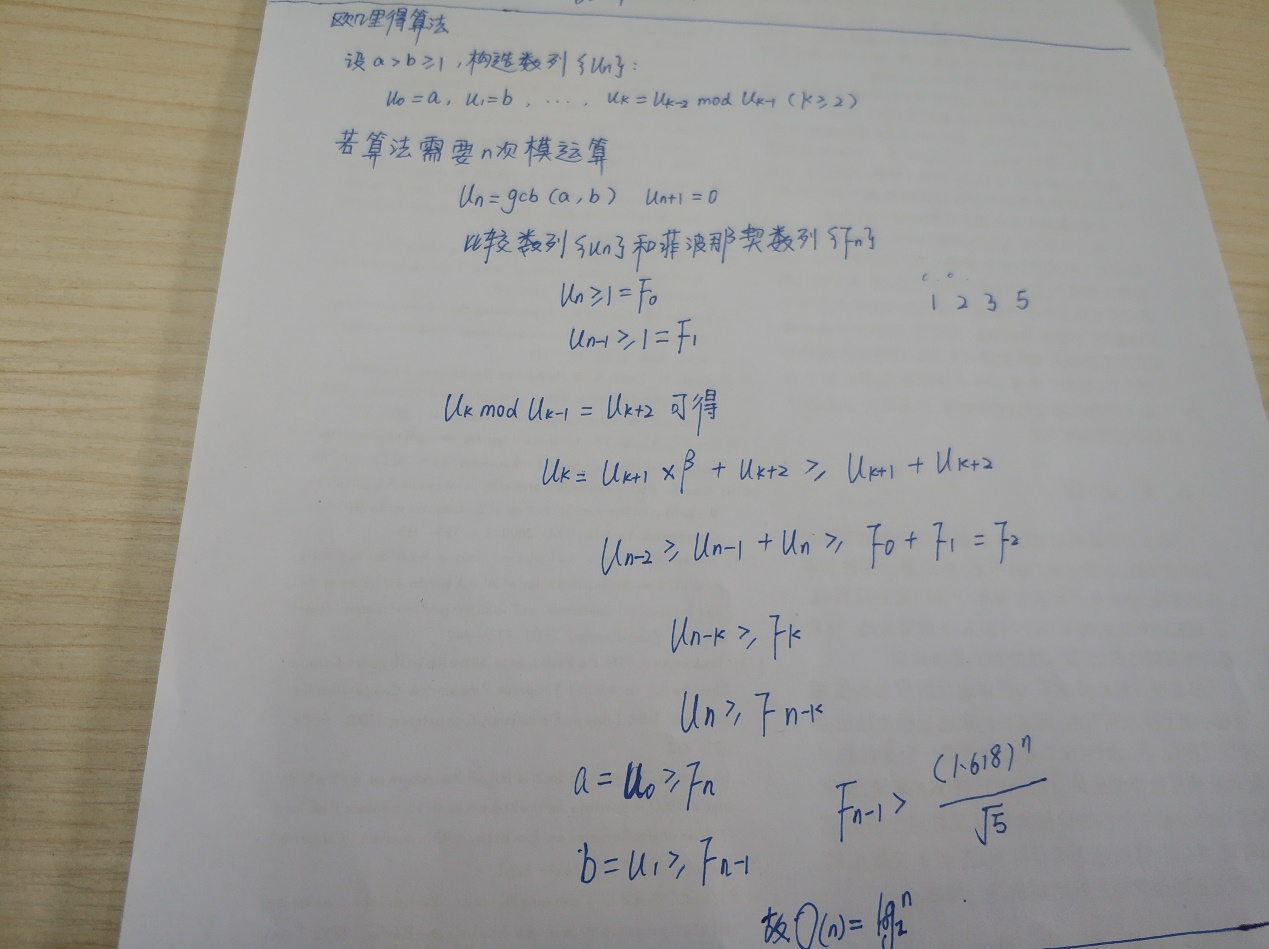
if (y == 0)

return x;

else

return gcd(y, x % y);

}该程序的时间复杂度为 O(logn)



一个凸N边形，可以用N-3条互不相交的对角线将凸N边形分成N-2个三角形，这称为凸N边形的一种三角剖分。例如N＝5时，共有以下5种三角剖分：

当N＝8时，总共有（）种三角剖分

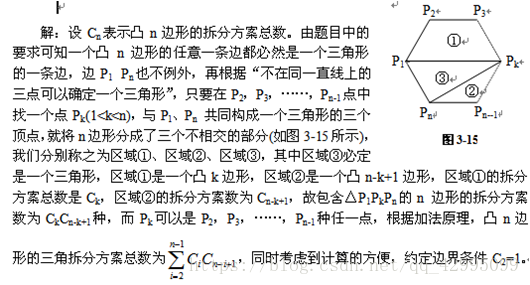
多边形三角剖分公式

D(n+1)\Dn =(4n-6)\n (Dn表示凸n边形的三角剖分数）

D8=(D8\D7)\*(D7\D6)\*(D6\D5)=(22\7)\*(3)\*(14\5)=132

卡特兰数又称卡塔兰数，英文名Catalan number，是组合数学中一个常出现在各种计数问题中出现的数列。以比利时的数学家欧仁·查理·卡塔兰 (1814–1894)的名字来命名。

最初，给卡塔兰数建立的数学模型是：一个凸n边形，通过不相交于n边形内部的对角线，把n边形拆分成若干三角形，不同的拆分数目用hn表示，hn即为Catalan数。例如五边形有如下五种拆分方案(如图)，故h5=5。求对于一个任意的凸n边形相应的hn



#include<cstdio>

long long a[42]={0,0,1,1};

int main()

{

int n;

scanf("%d",&n);

if(n==2) //如果是2边形，就输出0，单独处理一下

{

printf("0\n");

return 0; //输出后就可以结束了

}

for(int i=4;i<=n;i++) //用两重循环，实现递推式，外层循环是枚举边数

{

for(int j=2;j<i;j++)

{

a[i]+=a[j]\*a[i-j+1]; //递推式

}

}

printf("%lld\n",a[n]); //输出a[n]，就是正解

return 0;//结束程序

}

————————————————

版权声明：本文为CSDN博主「密码锁」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：<https://blog.csdn.net/qq_42995099/article/details/82193505>