汉诺塔问题

有三个编号分别为1,2,3的柱子；初始编号为1上的柱子上有n个盘子，下面的盘子大上面的盘子小，现在需要将这n个盘子移动到3号柱子上并输出移动步骤。
在移动的过程中

- 每次只能移动一个盘子
- 必须保持大盘子在下面，小盘子在上面。

算法描述 1

定义函数H(from, to, num)表示将num个盘子从from柱子移动到to柱子。则

```
H(from, to, num)
{
    if(num == 1)
    {
        output(from + "->" + to)
    }
    else
    {
        another = 三个柱子中除了from， to的另外一个柱子

        //将n-1个盘子从from移动到another
        H(from, another, num - 1)
        //将from上剩下的一个盘子移动到to上
        H(from, to, 1)
        //将another上的n-1个盘子移动到to上
        H(another, to, num-1)
    }

}
```

算法描述 2

使用javascript语言来实现这个算法，只需要做简单的修改以符合其语法规则即可。

```
function H(from, to, num)
{
    if(num == 1)
    {
        console.log(from + "->" + to);
    }
    else
    {
```

```
        //三个柱子中除了from， to的另外一个柱子
        var another = 6 - (from + to);

        //将n-1个盘子从from移动到another
        H(from, another, num - 1);
        //将from上剩下的一个盘子移动到to上
        H(from, to, 1);
        //将another上的n-1个盘子移动到to上
        H(another, to, num-1);
    }

}
```

复杂度分析

根据上面的算法描述可以看出：对于n个盘子的移动需要涉及2次n-1个盘子的移动和1次一个盘子的移动。设T(n)为移动n个盘子需要的移动的次数，则

- T(n) = 1 (n=1)
- T(n) = 2T(n-1) + 1 (n>1)

$T(n) = 2T(n-1) + 1$
= $2(2T(n-2) + 1) + 1 = 2^2T(n-2) + 2 + 1$
= $2^2\left(2 * T(n-3) + 1\right) + 2 + 1$ = $2^3T(n-3) + 2^2 + 2^1 + 2^0$
= ....
= $2^iT(n-i) + 2^0 + 2^1 + 2^2 + \ldots + 2^{i-2}$
=...
= $2^{n-1}T(1) + 2^0 + 2^1 + 2^2 + \ldots + 2^{n-2}$
= $2^0 + 2^1 + 2^2 + \ldots + 2^{n-2} + 2^{n-1}$

根据等比数列公式可得：

$T(n) = \frac{1*(1-2^n)}{1-2}$
= $2^n - 1$