1、动态规划问题的**一般形式是求最值化**

1.1:核心问题：

穷举，但此处的穷举不是暴力穷举是基于备忘录(此类问题存在“重叠子问题”的性质)

1.2:列出正确的状态转移方程:（具有最优子结构性质）

对于状态转移方程的思考主要是从状态(原问题和子问题中的状态)，选择以及dp数组的定义.

```
框架:
 //初始化状态
 dp[0][0][...]=basecase;
 for 状态1 to 状态1的所有取值:
 for 状态2 to 状态2的所有取值:
 ....
 dp[状态1][状态2][...]=求最值(选择1，选择2....
递归的算法为自顶向下而迭代下的dp为自底向上的算法

```

计算递归问题的时间复杂度方法:子问题个数*每个子问题求解的时间

对于dp数组过大的情况下可以考虑状态压缩(State Compress)即我们如果发现每次状态转移只需要DP table的一部分时，那么可以尝试用状态压缩DP table的大小

在计算的过程中还要定义一个最大值和最小值表示无穷大和无穷小

例题1

```
凑硬币:
int coinChange(vector<int>& coins,int amount){
 vector<int> dp(amount+1,amount+1);
 dp[0]=0;
 for(int i=0;i<dp.size();i++){
 for(int coin:coins){
 if(i-coin<0) continue;
 dp[i]=min(dp[i],dp[i-coin]+1);
 }
  }
  return dp[amount]==amount+1?"INF":dp[amount];
  }
```

计算机解决问题其实没有什么特殊技巧，它唯一解决方法就是穷举只是分为聪明穷举和愚蠢的穷举之说而已