## 压缩状态动态规划

## 问题

一般情况下动态规划算法都能取得很好的效率，但是有一种情况会大大降低DP的效率，就是**当状态太多**的时候， 因为状态太多直接影响了状态转移的计算量。那我们有什么优化方法呢？这里给介绍通过状态压缩的手段来解决状态过多的情况。

所谓状态压缩：就是将不必要的状态进行舍弃，将结果相同的状态进行合并的方法。

## 经典问题

[青蛙过河](http://www.rqnoj.cn/problem/17)是一道非常经典的状态压缩的题目，题目的大意是：有一条小河长L米（小于10^9米），上面有一些石头（小于100个），青蛙非常讨厌跳到石头上，起初青蛙站在0点，青蛙每次能跳[S, T]的整数米距离，问青蛙至少要踩多少次石头才能跳过小河？

**分析**：如果按照最原始的方法就是用dp[i]保存青蛙跳到第i米时，需要踩到的最少石头数。

dp[i] = min{dp[j} +bool (i是否时石头)   ( i - T <= j <= i - S )

可是这样的话状态就回有10^9个。于是在这里我们要采用状态压缩的方法，进一步分析状态转移可以看到，如果有一个长段距离没有石头，那么若干米以后最优解会占领所有的状态。由下图可见2S - T < S，所以最优解点不可达区域越来越来越小，直到占领整个后续区域。

所以，在计算状态时最多只要计算石头后的100米区域状态，整个状态数被大量压缩。

## 常见题目

另一类状态压缩的题目是在二维平面上进行的，通常有平铺问题，这类题目本质上不能称为压缩状态**，**因为并没有减少状态的数量，只是将状态表示法进行了减化。

代表题目：POJ 3254 Corn Fields， POJ 1185 炮兵阵地，POJ 2411 Mondriaan's Dream

还有一类题目也经常用状态压缩的方法，这里的压缩还是有意义的，因为多种路径会被压缩成一种状态， 例如走1->2->3和1->3->2的状态都是111。不过我对于这种题目的解法长使用记忆化搜索，时间上为稍逊一筹。

代表题目：POJ 2411 Mondriaan's Dream，POJ 3229 The Best Travel Design，POJ 2688 Cleaning Robot，POJ 1669 Best Sequence，POJ 3311 Hie with the Pie，ZOJ 3471 Most Powerful，CodeForces 11D