# 二进制状态压缩

用2进制来储存状态，多出现在图有关的题目中。图的点数一般较少不超过20

## 哈密顿回路和旅行商问题：

旅行商问题(TravelingSalesmanProblem，简称TSP)是一个经典的组合优化问题。经典的TSP可以描述为：一个商品推销员要去若干个城市推销商品，该推销员从一个城市出发，需要经过所有城市后，回到出发地。应如何选择行进路线，以使总的行程最短。

从[图论](https://baike.baidu.com/item/%E5%9B%BE%E8%AE%BA)的角度来看，该问题实质是在一个带权完全[无向图](https://baike.baidu.com/item/%E6%97%A0%E5%90%91%E5%9B%BE)中，找一个权值最小的[Hamilton](https://baike.baidu.com/item/Hamilton)回路。由于该问题的可行解是所有顶点的[全排列](https://baike.baidu.com/item/%E5%85%A8%E6%8E%92%E5%88%97)，随着顶点数的增加，会产生组合爆炸，它是一个[NP完全问题](https://baike.baidu.com/item/NP%E5%AE%8C%E5%85%A8%E9%97%AE%E9%A2%98)。

NP难的问题意思是不存在较快算法的问题，较快算法指多项式时间的算法。（2^n或者n！都不是多项式复杂度算法，n或n^2叫多项式复杂度）

对于n<=15;是可以凭借常规算法求解的

用动态规划或者搜索求解。

搜索时间复杂度是n!，空间复杂度是n

动态规划采用了以空间换时间的方式,时间复杂度是n\*2^n,空间复杂度也是n\*2^n;

怎么存图的状态，哪点走了哪点没走，对每个点编号，用二进制表示；

比如0000 0000 0000 001，表示15个点第一个点走了其余没走，

1000 0000 0000 001，表示15个点第一个点和最后一个点走了其余没走，

仅仅这样还不足以做dp，同一个二进制数，对于以不同点为结尾应该算作不同状态。比如同样1000 0000 0000 111,可能是以第一个1结尾，可能是以第二个1结尾可能是以第三个1结尾，可能是以最后一个1结尾

dp[i][j]代表i的状态下，j为当前结尾的状态

怎么转移，首先由于用了数i存储，把十进制数i变成2进制就能知道当前状态哪点走了哪点没走。

K是j能到达的所有点，对于一个状态i和要到的点k,k没走过：

就去走，且那个状态的点应该是原本和当下最小的。

对于所有dp初始值应该是INF

dp[i+2^k][k]=min(dp[i+2^k][k],dis(j,k))

**但还有一个特别需要注意第地方:**

旅行商问题中要求每个点走一次返回起点再，怎么处理，可能想到，再开一个dp记录起点，这样不行，你要以不同起点取最小的，以每个点为起点的dp是不同的，正解是进行n次dp，每次重新初始化dp数组，分别以1到n-1为起点，最后算出的min(dp[up][i]+dis[i][b]),是二进制数上限的是进制数值,b是这次判定的起点。i从1到n-1。这仅仅是以一个起点的，算n次去一次总的最小值，得到最后结果

图覆盖问题(hdu6321)：

Dp[i][j]代表i下，j状态下