**状态压缩**

点序列d1,d2,...,dk

然后接下来遍历其它点，此时只有dk对接下来的操作有影响。

使用状态压缩，f[i][j]，i记录值（已出现的点），j记录当前点。每次操作，利用k个点序列推出k+1个点的序列，C(n,k)\*k。而且，通过当前点找到新的点，需要花费代价（乘上一个系数）。

以n=20为例，则C(20,1)\*1+C(20,2)\*2+...+C(20,20)\*20 = 10485760

更节省内存的方法

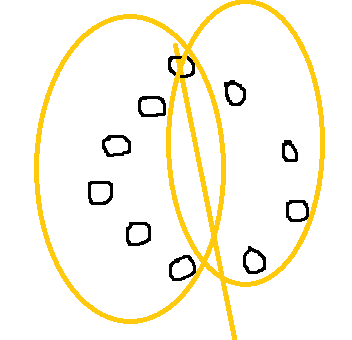
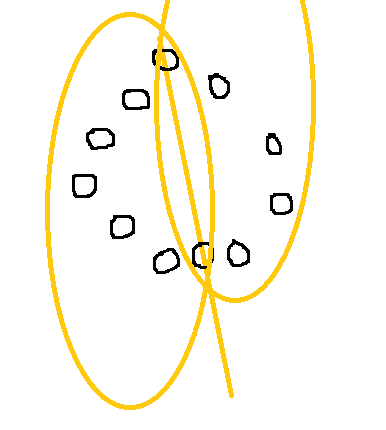
0-2^n-1构造出来，每个数位数和记录所有的位。

分成n部分。

使用循环队列，记录含有k个1的数和含有k+1个1的数。

**折半搜索**

对循环分成两半，分别求，再并在一起。

 或 

分解求：

第一种方法：

W[(n-1)/2]

W[n/2]

第二种方法：

W[n/2]

W[(n+1)/2]

合并：

第一种方法：

n\*n/2\*C(n,(n-1)/2)

（理想情况的最大值，非为实际最大值）

第二种方法：

n\*C(n,n/2)

n为偶数时，感觉使用第二种方法较好写一点。

状态压缩+折半搜索

分成两半

状态压缩

前面一部分和后面一部分 (quite the same)

C(n,k)=C(n,n-k)

n为奇数

C(n,0)+C(n,2)+...+C(n,n/2) = C(n,n/2+1)+C(n,n/2+2)+...+C(n,n)

n为偶数时，……

折半搜索没有使效率提高。

状态压缩

C(n,1)+C(n,2)+...+C(n,n)=2^n