计数问题选讲

陈立杰

清华大学交叉信息学院

February 9, 2015

intro.

Table of Contents

大家应该都知道容斥原理吧。
好了，我们已经知道这个模型是怎么回事了，让我们看几道题目来学习一下吧！

Biconnected

给你一个$n(n \leq 10)$个点的无向简单图，问有多少个边的子集$E' \subset E$, 使得只保留E'中的边的话，整个图是双联通的。
也就是问你有多少个关于边的子图（点集合还是这n个点）双连通。
2014 ACM/ICPC Asia Regional Anshan Online, by me.

做法 1

直接处理这个问题并不容易，不妨考虑如何计算使得整个图不双联通的边集。注意到一个不双联通的图，必然能够唯一地分成几个双联通块，并且这些块之间组成了一棵森林。

做法 1 cont.

那么我们不妨先枚举这个图如何被划分成了几个联通块，然后计算这些联通块之间连成森林的方法数即可。每个联通块连成双联通图的方案数就是一个子问题了，集合dp计算即可。
后者是一个经典问题，我们先枚举森林中哪些点组成了树，然后使用生成树数量的计数公式即可。
复杂度比较高，所以只能处理$n \leq 10$。

做法 2 abs.

使用和下面一个题目类似的方法可以做到$n \leq 15$。

给你一个$n(n \leq 15)$个点的有向简单图，问有多少个边的子集$E' \subset E$，使得只保留E'中的边的话，整个图还是强连通的。
2014 Tsinghua Training Camp, by me.

做法1

还是非常难直接处理这个问题，和上一题类似，我们来考虑如何非强连通的子图数量。相信聪明的同学立马就发现了，非强联通的图当然就是一些强连通分量组成的DAG啦 。
那么我们立刻有了简单粗暴的做法1，枚举分成强联通分量的划分方案，对于每一个划分计算连成DAG的方案数。

那么我们现在需要考虑这个问题，给你n个点的图，计算有多少子图是DAG。
这个问题也是一个非常经典的问题，我们可以采用集合dp，令$D[S]$表示S这个集合有多少子图是DAG，我们知道一个DAG就会有一些没有出度的汇点，那么我们枚举S中汇的集合T，不能保证$S \setminus T$中就没有汇，可能出现重复计算。所以可以使用容斥原理来计算。列出式子就是：

$$D[S] = \sum_{T \subseteq S \quad |T| \geq 1} (-1)^{|T|-1} ways[T, S-T] \cdot D[S-T].$$

其中$ways[S, T]$表示从$S-T$到T单向连边的方案数。

做法1 cont.

但是注意到这里的时间复杂度非常大，只能够处理到$n \le 8$的情况，在当时的比赛中也就只能获得50分了。

做法2

要优化时间复杂度，我们来仔细的观察一下这个计算。
注意到，实际上我们没有必要预先枚举集合划分，我们可以直接枚举T，表示汇的强连通分量构成的集合是哪些，然后dp出T内部拆成奇数个强连通分量的方案数减去拆成偶数个强连通分量的方案数。那么实际上我们就一下子把那么多容斥一起计算了。
在通过一些技巧预处理$ways$这个数组，那么复杂度就只有$O(3^n)$了。

Biconnected 做法2

那么联系到biconnected这个题目，我们实际上也可以通过枚举叶子的集合，来容斥的计算生成树的个数。
使用和上一题一模一样的方法，也可以在$O(3^n)$的时间内解决这个问题了。

Endless Spin

有一行N个白色的球，每次你会随机选择一个区间$[l, r]$，将这些区间里的球都染黑。问期望多少次以后，所有的球都被染黑？

2013 Multi-University Training Contest 3, by me.

做法1

首先我们注意到，期望次数等于:

$$\sum_{L=0}^{\infty} P[L].$$

其中$P[L]$表示我们进行了L次操作后，还没有全部染黑的概率。这个式子是非常直观的。

做法1

那么接下来我们考虑如何计算$P[L]$，注意到在进行了L次操作后还没有全部染黑，意味着存在一些球，L次操作以后还是白色的。
那么我们不妨考虑使用容斥原理，枚举这些到最后也是白色的球有哪些。然后进行计算。
不妨假设这些白色球分别是$v_1, v_2, \ldots, v_k$，
那么只能在区间$[1, v_1 - 1], [v_1 + 1, v_2 - 1], \ldots, [v_k + 1, n]$的子区间中选择。
如果这样的子区间一共有A个，那么每次选到他们的概率就是:

$$p = \frac{A}{\binom{n+1}{2}}.$$

做法1 cont.

L轮都选到他们的概率就是p^L。注意到既然使用了容斥原理，那么如果有奇数个白球，$P[L]$就加上p^L，否则就减去p^L。
显然$\sum_{i=0}^{\infty} p^L = \frac{1}{1-p}$。所以直接处理$\frac{1}{1-p}$即可。
那么考虑dp，来计算上面的值，注意到我们只需要记录上一个目前的白点数量的奇偶性，目前可以选择的区间数，然后枚举下一个白点的位置即可。

做法2

先介绍一个容斥原理的简单变形。
考虑我们有随机变量$X_1, X_2, \ldots, X_n$，我们想要计算$\mathbb{E}[\max(X_1, X_2, \ldots, X_n)]$。
我们有如下的式子：

$$\mathbb{E}[\max_{i=1}^{n}(X_i)] = \sum_{S\subset[n],\ |S|\geq 1} (-1)^{|S|-1}\mathbb{E}[\min_{i\in S} X_i].$$

注意到这个式子其实是显然的，考虑X_i都是离散的简单情况。不妨直接将X_i表示成集合$1, 2, \ldots, X_i$，那么max就是它们的和的大小，min就是它们的交的大小。这个式子立马就变成了原来的并，交形式的容斥原理。由期望的线性性立刻就能得到这个式子。
对于X_i不是离散的情况，反正大家也用不着>_>，呵呵哒。

做法2

那么我们令X_i表示第i个球在哪一轮被染黑。那么我们想要计算的其实就是$\mathbb{E}[\max_{i=1}^{n}(X_i)]$。
这个并不好计算，通过之前的那个式子，我们能将问题转化成计算$(-1)^{|S|-1}\mathbb{E}[\min_{i\in S} X_i]$，进一步的分析能够得到和做法1一模一样的方法。
所以说这里其实只是思路不同啦，做法最后还是一样的<_<。

Table of Contents

简单地说就是通过一个外层的dp来计算使得另一个dp方程(子dp)最终结果为特定值的输入数。
(我坦白这名字是我瞎YY的)。
那么，这个问题要怎么解决呢，我们不妨一位一位确定子dp的输入，不妨考虑已经枚举了前i位了，那么注意到，由于我们只对dp方程的最终结果感兴趣，我们并不需要记录这前i位都是什么，只需要记录对这前i位进行转移以后，dp方程关于每个状态的值就可以了（这个的意思是，外层dp的状态是所有子dp的状态的值）。

看一个简单的例子，假设我们有一个DP A。
A读入一个序列$a_1, a_2, \ldots, a_n$.
返回关于这个系列的一个结果。
现在我们想要知道有多少种a的序列可以返回这种结果。

这种要怎么做？
考虑第一个DP，A，如果我们a_1, a_2这么依次枚举。
那么当我们处理到$a_1, a_2, \ldots, a_i$的时候，有意义的只有A目前每个状态的值。
也就是说，我们可以在状态里面记录“A的所有状态的值”。
好了，我们已经知道这个模型是怎么回事了，让我们看几道题目来学习一下吧！

给你一个只由AGCT组成的字符串$S(|S| \le 15)$，对于每个$0 \le i \le |S|$，问有多少个只由AGCT组成的长度为$m(1 \le m \le 1000)$的字符串T，使得$LCS(S, T) = i$?
LCS就是最长公共子序列。
2014 Multi-University Training Contest 4, by me.

做法

首先让我们来考虑，给你两个串S和T，我们怎么计算他们的LCS?
显然我们可以列出方程$D[i,j]$，表示T的前i个和S的前j个的LCS。S已经给定了，我们把这看成一个关于T的dp，那么立刻就能发现这和刚才说的模型是一模一样的。
那么，我们一位一位枚举T，用一个大状态记录每一个$D[i,j]$的值是什么，当然这样的复杂度会很大，但是注意到对一个特定的串，$D[i,j]$和$D[i,j-1]$的差只能是0或者1，那么实际上状态最多也只有2^{15}种。

Square

给你一个$n \cdot n(n \leq 8)$的棋盘，上面有一些格子必须是黑色，其它可以染黑或者染白，对于一个棋盘，定义它的优美度为它上面最大的连续白色子正方形的边长，对于每个$0 \leq i \leq n$，问有多少种染色方案使得棋盘的优美度为i?

2014 Asia AnShan Regional Contest,by me.

做法1

首先让我们考虑，如果给你一个$n \cdot n$的棋盘，怎么计算它上面最大连续白色子正方形的边长？
相信大家立马可以列出如下的dp式子：

$$D[i,j] = \begin{cases} max(D[i-1,j], D[i,j-1], D[i-1,j-1]) + 1 & (i,j) \text{ is white} \\ 0 & (i,j) \text{ is black} \end{cases}$$

那么我们考虑一个一个枚举棋盘上格子的颜色，枚举到格子(i,j)的时候，其实我们只需要记录之前白色正方形的最大边长，和它之前$n+1$个格子上的dp值就行了。
注意到如果$D[i,j] > 0$，那么$D[i,j-1] \geq D[i,j] - 1$，这意味着这些D值的可能状态数量不会太多，经程序实际验证最多只有几万，所以可以无压力跑出这个问题呢。

做法2

可是如果你说我比较naive，想不到上面这个看起来很厉害的做法，怎么办呢。

首先将问题转化成对于每个i，有多少种方案使得棋盘上存在$i*i$的空白正方形。

考虑一个比较暴力的想法，还是按刚才的思路进行dp，对每个点记录往上有几个连续的白格子。

这显然是不行的，因为状态数量最多是$n^n = 8^8 = 2^{24}$。但是我们注意到，对于i比较大的情况，实际上一共只有$(n-i+1)*(n-i+1)$种可能的白色正方形，我们不妨直接枚举这些正方形的出现情况进行容斥，那么对于$i \geq 5$的情况, $8-i \leq 4$，只有2^{16}的复杂度。

然后我们只需要考虑$i \leq 4$的情况，注意到这个时候计算暴力进行计算，状态数也只有$8^5 = 2^{15}$了，完全可以承受。

做法3

如果你见得多了，立马就能想到其实上面两个想法是可以结合到一起的。
我们还是可以先枚举$i \geq n-3$的情况，然后使用做法1的思路进行dp，那么做法1中的状态数量也大大减少，能够跑到$n \leq 12$的情况。

做法4

当然以上的做法还不是最优的。

还是考虑对于每个k，有多少种方案使得棋盘上存在$k*k$的空白正方形。考虑做法1中的$D[i,j]$数组。

实际上对于每一行，我们并没有必要记录$D[i,1],\ldots,D[i,k-1]$的具体值是多少，因为以这些点为右下角，不可能存在$k*k$的空白正方形，我们只需要记录这些点是否都是白色的就可以了。

那么实际上状态的数量就只有$(k+1)^{n-k+1}$。而这个函数的最大值在$n\leq 8$时只有几千。

我们有一个$n * m (n, m \leq 30)$的网格，每个格子上可能有障碍物，也可能没有。我们要对所有非障碍物的格子进行黑白染色，并且染色完成之后，从左上角到右下角任何一条只往下或往右的路径（不能经过障碍物）都恰好只经过了一个黑色格子。问染色的方案数。

2013 TCO Algorithm Semifinal 2 - Level Three, by rng_58.

做法

这个题目的正解是一个关于n的多项式的dp，但是使用我们刚才的思路也可以得到另外一个能够通过的做法。
首先注意到对于从左上角无法到达，或者无法到达右下角的点，我们可以随意染色，直接计算在答案里就可以，所以我们可以假定每个非障碍点都可从左上角到达，也可以到达右下角。
考虑给定了一个图的染色，如何判断是否满足条件？考虑使用DP，对于点(i,j)，用$D[i,j]$表示所有从左上角到(i,j)的路径都经过了几个黑点。注意到如果从左上角到(i,j)存在两条经过黑点数不同的路径，那么肯定不满足条件。同时$D[i,j]$的合法值只有0和1。

做法 cont.

注意到这样这个问题就变得和上个问题差不多了，我们也可以一个一个枚举(i,j)的颜色，并且记录从(i,j)往前m个格子的D值。
一眼看起来似乎有2^{30}个状态，但是注意到如果$D[i][j-1]=1$, 并且(i,j)不是障碍物的话，$D[i][j]$只能取1，这就意味着状态数量其实是非常有限的。最多只有2^{16}左右。可以通过数据。

Jigsaw Puzzle

给你一个$n * m(1 \leq n \leq 500, 1 \leq m \leq 6)$的棋盘，你可以在上面删去一些格子，但是你要保证删去了这些格子以后，剩下的格子我们可以使用$1 * 2$的多米诺骨牌完全覆盖。问删除格子的方案数。

2013-2014 ACM-ICPC, NEERC, Moscow Subregional Contest.

做法

还是一样的思路，先考虑给定了一个删完之后的棋盘以后，我们怎么判断是否可以用$1*2$的多米诺骨牌完全覆盖。
为了方便不妨考虑使用按行dp的思路，那么状态就是当前行有哪些空位置需要被覆盖，转移就是枚举如何在当前行放置$1*2$的多米诺骨牌。
考虑这个内部dp，它有$2^6=64$个状态，每个状态有2个可能值True或者False，那么外层的dp要完全记录它的话就有2^{64}种状态，这显然是不切实际的。

做法 cont.

那么让我们来考虑仔细地分析一下，首先注意到，一个状态里如果有相邻两个空格子（左边那个旁边是障碍物或者边界），那么容易看出横着放肯定比竖着放优，所以在转移的时候肯定优先考虑横着放。按这个方式，可以把任何状态里相邻的两个空格子填上而不丢失可行解，因此我们只需要考虑没有相邻两个空格子的状态就可以了。
而这样的状态只有21种，当然2^{21}还是不小。再仔细地观察可以注意到，一个棋盘对应的所有状态，他们中空位置的奇偶性肯定是相同的，那么状态的数量就只有$2^{10}+2^{11}$个了。问题也就解决了。

StringPath

给你一个$n*m(1 \le n, m \le 8)$的矩阵，再给你两个长度为$n+m-1$的小写字母组成的字符串A, B，问有多少种给矩阵在每个格子里填小写字母的方式，使得从矩阵的左上角到右下角（只能往右和往下），存在两条路径经过的字符分别组成A和B。

Single Round Match 591 Round 1 - Division I, Level Three, by gojira_tc.

做法

还是一样的思路，先考虑给了一个矩阵之后，如何判断这个矩阵是否满足条件？还是可以使用一个DP，容易看出我们可以对每个位置(i, j)，分别记录是否存在一个左上角到它的路是A或者B的前缀。然后我们一个一个枚举矩阵位置上的字符，枚举到(i, j)时，注意到我们只需要记录它之前的m个位置的DP值即可。复杂度就是$2^{2m} = 2^{16}$，可以接受。

Table of Contents

对于计数问题，除了容斥，正难则反这样比较间接的方法以外，还有一些更加直接的方法，最常见的就是将整个计数不重不漏地按照一些方法分成一些子部分(对计数对象的分类)，那么每个子部分的结果加起来当然就是答案啦。这里的关键当然就是分类的方法啦。
好了，我们已经知道这个模型是怎么回事了，让我们看几道题目来学习一下吧！

The only survival

$n(n \leq 12)$个点的无向完全图，每条边的边权在1到$L(L \leq 10^9)$之间，问有多少个这样的图使得点1到点n的最短路是$k(k \leq 12)$？

2014 Multi-University Training Contest 4, by me.

做法

这个问题一眼看过去，非常不可做，容斥啊，dp啊之类的方法想一想似乎也无法使用，怎么办呢？
不妨考虑枚举一点东西做一下分类，很容易想到我们可以预先枚举每个点的距离标号d_i。
有了距离标号以后，那么考虑点i和j之间的边，首先如果$d_i = d_j$，那么这个边的边权可以是任何值，不然假设$d_i < d_j$，那么边权不能小于$d_j - d_i$，否则就矛盾了。

同时对于点i，必须得存在一个$d_j < d_i$的点，使得他们之间的边权恰好是$d_i - d_j$。那么我们将所有点都按照标号排序，然后一个一个进行dp，计算出满足条件的边权的方案数量就可以啦。
接下来注意到，显然1的标号是0, n的标号是k，然后对于标号大于k的点实际上我们并不关心他们具体的标号，直接用$k+1$表示就可以。
当然对于2到$n-1$这些点，我们也不需要知道它们各自具体的标号，只需要知道每种标号各有几个就可以喽。那么我们就枚举一下每种标号各有几个，顺便计算一下边权的方案数，问题就解决了。

扑克牌

我们有$N(N \le 30)$张纸牌，每张牌都有数字和颜色两个属性。考虑把这N张牌排成一行，使得相邻的两个要么颜色相同要么数字相同。牌的数字在0到9之间，颜色有红黄蓝三种（用012来表示）。问方案数。不会有4张牌的数字和颜色都一样。

2013 Tsinghua Training Camp, by me.

做法

这个问题看起来非常难，也没有什么思路，不妨让我们深入的分析一下。

考虑一个合法的N张纸牌的放法，由于相邻的要么是同颜色，要么是同数字，不妨将相邻的分为两类，一类是数字相同但颜色不同，一类是颜色相同。

不妨令前者为一类边，后者为二类边。

考虑所有一类边连成的连续块，注意到实际上我们只关心他们中第一个的颜色A和最后一个的颜色 B，那么实际上这就意味着我们可以把它们缩成一条从$A \to B$的边。

对于每个数字的牌，我们可以枚举如何将这些牌通过一类边连接成一些颜色之间的边。由于同一颜色的牌数量不多，暴搜就可以。

做法 cont.

然后我们可以通过DP，依次考虑每张牌，记录各种边的数量(一共有9种边)是多少的情况下，一类边的连接方案数。
有了这样的一个分类以后，剩下的问题就变成了在一个3个点的DAG上，有多少种走法可以走完所有的边了，再使用一个dp计算即可。

题外话

关于这个题目还有一个比较有趣的话题，这个题目在出题的时候怎么对拍？也就是怎么使用暴力验证数据的正确性？

$O(n^2 \cdot 2^n)$的求哈密顿路径数量的方法，需要的空间也是$O(n \cdot 2^n)$的，虽然时间上多跑会儿没什么问题，但是空间并不是一般的电脑能够开下的。

实际上有一个指数时间，多项式时间的计算哈密顿路径数量的方法。

把问题转化成求1到n, 经过每个点至少一次，且长度为$n-1$的路径的数量。

容斥枚举没有经过的集合，再使用dp计算无限制情况下的路径数量即可。

Substring Pairs

求满足以下条件的字符串对S, T的个数，字符集的大小为A。

- $|S| = n, \quad |T| = m$
- T是S的子串。

其中$n \leq 200, m \leq 50, A \leq 1000$。
Makoto Soejima Contest 1, Japanese Grand Prix, by rng_58

做法

现在，让我们假设T已经确定了。
如何计算满足条件的S的个数？
不妨让我们考虑暴力容斥，枚举S中T的出现位置是哪些，然后用出现奇数次的减去出现偶数次的，这是一个非常经典的方法，可以使用一个简单的dp来实现。

做法 cont.

那么我们来看，如果要使用这个dp计算，我们需要知道关于T的什么信息？容易发现在这里我们只需要判断两个相邻的T的出现位置是否会导致矛盾即可，也就是说，我们只需要知道T的前缀和后缀的相等关系即可。

那么立马我们就把枚举量从A^m减少到了2^{m-1}，我们只需要枚举对于T，第i个前缀是否和第i个后缀相等即可。

我们可以用一个$m-1$位的mask来表示这个枚举的信息。

当然2^{m-1}还是不能接受的，通过暴搜，我们可以发现实际上合法的mask只有两千多种。
从直观上理解，我们可以发现实际上这里的约束是很强的，所以可能性的数量不大。
那么，问题就变成了两步，第一步是求出每种mask对应的T的数量，第二步是计算答案。
怎么求每种mask对应的T的数量呢，我们不妨考虑放缩条件，首先很容易求出，满足mask里那些前缀和后缀都相等的T的个数，只需要将相等的位合起来连起来，计算free的变量的个数就行了。
当然我们要求的是恰好 mask里那些前缀和后缀都相等的T的个数，于是我们就前去所有合法的mask中，现在这个mask的超集的答案的和就可以了，这个问题就解决了。

Table of Contents

Part4. 期望的线性性

大家都知道，对于一些随机变量，他们和的期望等于期望的和。
这个东西看起来非常非常的简单，但是可以很巧妙的用来解决各种各样的问题。

考虑一个$n(n \leq 10)$个点的无向简单图，每条边的边权是一个$[0,1]$之间的随机数，并且各个边边权都是独立的。问这个图的MST的期望大小是多少？

2015 ACM Shanghai Training Camp, by me.

做法

首先注意到，如果所有边的边权都不相同，那么最小生成树是唯一的。然后注意到，所有边边权都不相同的概率为1。所以我们可以假定最小生成树为1。

然后这个问题乍一看不太好做。但是我们注意到可以使用期望的线性性。考虑一条边e，如果它在MST里，它对MST的贡献就是它的边权，否则它对MST的贡献就是0。那么MST的大小就是所有边的贡献和。

所以我们可以反而过来计算每条边的贡献的期望。

做法

考虑一条边$e : a - b$，如果它的边权是x，注意到它出现在MST里当且仅当不存在从a到b的全部由$< x$的边组成的路径。这等价于，如果每条边出现的概率是x，在不考虑边e的情况下a和b不连通。
注意到，如果我们假定每条边出现的概率是x，然后再使用经典方法计算原图每个子图是联通图的概率，可以发现这些概率都是关于x的多项式，并且多项式的次数小于等于总边数。
所以边权是x的话，通过枚举a所在的连通分量是什么，我们也可以算出a和b不连通的概率多项式$P(x)$。那么期望贡献自然就是$\int_0^1 xP(x)dx$.
加起来就能得到MST的期望值了。

有一个$n*m(n,m \le 20)$的矩阵，有$r(r \le n \cdot m)$个萌萌的小兔子要走进来。它们一个一个走进来，每个兔子进来的时候都等概率随机选择一个矩阵中的空格子然后站在里面，当然有兔子了就不是空格子了。
等它们都选好格子以后，我们对第i个兔子，定义$f(i)$为另外一个离它欧几里得距离最近的兔子的编号，如果有距离一样的，那么优先选择所在行编号最小的，如果还有一样的，优先选择所在列编号最小的。
然后我们把i和$f(i)$连在一起，变成一个图，问这个图中联通块期望的数量是多少？
Single Round Match 636 Round 1 - Division I, Level Two, lyrically.

做法

首先我们来分析这个图的，可以把边看成$i \to f(i)$的有向边。那么注意到，由于这个图有n条边，那么他的每个连通分量必然是一个以一个环为根的内向树。

于是我们就能发现，联通分量的数量，等于环的数量。

于是我们只需要考虑环的数量的期望。

做法 cont.

让我们来进一步的分析，注意到一个环意味着环上所有点x和$f(x)$的距离都是相等的。

不妨考虑环上路径$a_{i-1} \to a_i \to a_{i+1}$，容易看出$a_{i+1}$的行列字典序要比$a_{i-1}$小，同样的道理$a_{i+3}$的行列字典序也要比$a_{i+1}$小，以此类推，可以发现如果这个环的长度大于2，那么就矛盾了。

因此只可能存在长度为2的环。

那么接下来我们不妨枚举两个位置(i_1, j_1)和(i_2, j_2)，然后来计算这两个点构成环的概率是多少。加起来就能得到环的数量的期望。

BitToggler

给你一个长度为$n(n \leq 20)$的01串，以及一个指针，初始时指针在第i_0个字符上。
每回合随机一个$1 \ldots n$中的数j，如果指针之前在i上，就花费$|i-j|$的时间把指针从i移动到j上，并且把01串的第j位取反。
不停这样随机，直到01串变成全0或者全1为止，问到终止前期望花费的时间是多少？
Single Round Match 641 Round 1 - Division I, Level Three, by Kriii

做法

实际上我们可以注意到这里的状态是没有办法压的，也就是说对于一个01串，如果我们想要从它开始到终止需要的期望时间，那么必须要记录这个串本身是什么。

做法 cont.

怎么办呢，注意到一共有n^2种代价，分别是指针从i移动到j。
不妨令随机变量$E_{i,j}$表示还是原来那个随机的过程，但是现在我们只有在指针从i移动到j的时候才计算代价$|i-j|$，其余的移动产生的代价一概忽略一概忽略。
令随机变量E为原过程产生的代价。我们原来就是要计算$\mathbb{E}[E]$。
很显然我们有$E=\sum_{i,j}E_{i,j}$。
根据期望的线性性，我们自然有：

$$\mathbb{E}[E]=\sum_{i,j}\mathbb{E}[E_{i,j}].$$

做法 cont.

那么我们只要分别计算出$\mathbb{E}[E_{i,j}]$就可以了。
怎么算呢？注意到由于我们只关心$i \to j$这个移动产生的代价，可以发现，其它$n-2$个位置都是等价的。
所以我们其实只需要记录i,j上的数，和其它位置上的0和1的个数就可以了。
这样我们就可以列一个方程求解，算出$\mathbb{E}[E_{i,j}]$，然后再把这些值加起来，就能得到$\mathbb{E}$了。问题也就解决了。

Problem links

1. Biconnected:
   http://acm.hdu.edu.cn/showproblem.php?pid=4997
2. sconnect:
   http://www.lydsy.com/JudgeOnline/problem.php?id=3812
3. Endless Spin:
   http://acm.hdu.edu.cn/showproblem.php?pid=4624
4. Hero meet devil:
   http://acm.hdu.edu.cn/showproblem.php?pid=4899
5. Square: http://acm.hdu.edu.cn/showproblem.php?pid=5079
6. OneBlack: http://community.topcoder.com/stat?c=problem_statement&pm=12844
7. Jigsaw Puzzle: http://codeforces.com/gym/100257

Problem links cont.

1. StringPath: http://community.topcoder.com/stat?c=problem_statement&pm=12620
2. The only survival: http://acm.hdu.edu.cn/showproblem.php?pid=4903
3. poker: http://oj.tsinsen.com/A1509
4. Substring Pairs: http://karelia.snarknews.info/index.cgi?data=macros/run&menu=index&head=index&round=02&sbname=2015w&class=2015w (can't submit solution)
5. MST: (unpublicized)
6. ClosestRabbit: http://community.topcoder.com/stat?c=problem_statement&pm=13366
7. BitToggler: http://community.topcoder.com/stat?c=problem_statement&pm=13328&rd=16084

Extra links

1. Kirchhoff's theorem:
   http://en.wikipedia.org/wiki/Kirchhoff%27s_theorem
   (Couting the number of spanning trees rooted at r)
2. Maximum-minimums identity:
   http://en.wikipedia.org/wiki/Maximum-minimums_identity
3. Expected value:
   http://en.wikipedia.org/wiki/Expected_value
4. Couting Hamiltonian Path in polynomial space
   http://dl.acm.org/citation.cfm?id=2308017