## 《彩色圆环(circle)》命题报告

华东师范大学第二附属中学 吴佳俊

### 【命题思路】

《彩色圆环(circle)》的最原始版本是我与同学交流时得到的试题A。

**试题A：一个环上有N个点，每个点染为M种颜色之一，求相邻两点均不同色的方案数。（N≤1018, M≤109）**

这个问题的求解分两步进行。

首先拆环为链，对于链可以通过递推求解。用f[i, j]表示考虑了前i个点，最后一个点的颜色为j的方案数，故有；

进一步，对于环，仅当首尾相接时与链有所不同，故考虑增设一维表示第一个点的状态。即用f[i, j, k]表示考虑了前i个点，第一个点的颜色为j，最后一个点的颜色为k的方案数，故有。若令，每个状态的计算时间复杂度可降为O(1)，算法的时间复杂度为O(NM2)。

这样的时间复杂度不能在时限内出解。我们注意到，算法关注的只是第一个点和当前考察的最后一个点的颜色异同，而不是具体的颜色。所以，我们可以极大程度地优化状态设计。

用f[i]表示考虑了前i个点，第一个点和最后一个点颜色相同的方案数；g[i]表示考虑了前i个点，第一个点和最后一个点颜色不同的方案数。容易得到f[i] = g[i-1]; g[i] = f[i-1]\*(M-1) + g[i-1]\*(M-2)。显然，经这一优化，每个状态的计算复杂度为O(1)，算法的时间复杂度为O(N)，空间复杂度为O(1)。

试题A还可以进一步使用矩阵乘法将时间复杂度优化为O(logN)，由于和后续试题关联不大，此处不再详述。

试题A是一个并不太困难的问题，当时我尝试对这个问题提出推广，并得到试题B。

**试题B：一个环上有N个点，每个点染为M种颜色之一，求同色连续段长度均不超过K的方案数。（N≤1018, M≤109, K≤105）**

容易发现，试题B与试题A相比，需要多关注的即最初和当前最后同色连续段的长度。故我们对试题A的原始算法进行改进。用f[i, f\_col, f\_len, l\_col, l\_len]表示考虑了前i个点，第一个连续段的颜色为f\_col，长度为f\_len，当前最后一个连续段的颜色为l\_col，长度为l\_len的方案数。同样可以运用试题A的优化使递推的状态总数降为O(N3)，即f[i, f\_len, l\_len]表示考虑了前i个点，第一个连续段的长度为f\_len，当前最后一个连续段的长度为l\_len，且第一个连续段和最后一个连续段颜色相同的方案数；g[i, f\_len, l\_len]表示考虑了前i个点，第一个连续段的长度为f\_len，当前最后一个连续段的长度为l\_len，且第一个连续段和最后一个连续段颜色不同的方案数。从而有递推公式：

如果我们进一步使用部分和优化，即令f\_sum[i, j, k] = ，g\_sum[i, j, k] = ，则可使每个状态的计算复杂度降为O(1)。从而问题的时间复杂度为O(N3)。

事实上试题B还可以进一步优化，优化的方法将在下面试题C的解答中详述，优化后算法的时间复杂度为O(N+K)，空间复杂度为O(1)。本题同样可以使用矩阵乘法优化，优化后算法的时间复杂度为O(KlogN)，空间复杂度为O(1)。两种版本可视情况选用。

试题B的描述与试题A过于接近，对于选手而言，得到递推模型并进一步优化的难度并不太大。考虑到题目要与NOI竞赛试题难度相仿，经过与同学的交流，我对试题进行了进一步改编，得到了试题C，经过包装的试题显得与原来版本的试题截然不同。

**试题C：一个环上有N个点，每个点随机染为M种颜色之一。求环上同色连续段长度之积的期望值。（N≤200, M≤109）**

但事实上，经过仔细剖析可以发现，我们同样可以用递推算法解决问题。用f[i, f\_col, f\_len, l\_col, l\_len]表示考虑了前i个点，第一个连续段的颜色为f\_col，长度为f\_len，当前最后一个连续段的颜色为l\_col，长度为l\_len的同色连续段长度之积的期望值。同样地，运用颜色的等价性进行优化。

**用f[i, f\_len, l\_len]表示考虑了前i个点，第一个连续段的长度为f\_len，当前最后一个连续段的长度为l\_len，且第一个连续段与最后一个连续段颜色相同时所求的期望值(暂不乘以第一个和最后一个连续段的长度)；用g[i, f\_len, l\_len]表示考虑了前i个点，第一个连续段的长度为f\_len，当前最后一个连续段的长度为l\_len，且第一个连续段与最后一个连续段颜色相异时所求的期望值(暂不乘以第一个和最后一个连续段的长度)。从而有递推公式**

**类似地，使用部分和优化时，可令，。**

**上述计算完毕后，只需再计算 即为所求值。**

**递推的边界状态值得注意。边界状态包括：**

**综上得，递推状态总数为O(N3)，每个状态计算耗时为O(1)，算法的时间复杂度为O(N3)。使用滚动数组，算法的空间复杂度为O(N2)。**

**试题C的算法可进一步优化。首先需要进行除法剥离，然后通过观察状态的等价性再进一步优化状态设计和数据计算。优化的关键在于注意到f[i+j+k][j][k]在j, k不同的情况下是相等的，从而可以将状态总数优化到O(N)，每个状态的计算复杂度可通过维护四个部分和降为O(1)。具体的优化办法，状态设计及递推公式详见附录。故算法的总时间复杂度为O(N)，空间复杂度可通过滚动数组降为O(1)。**

试题C即最终的试题《彩色圆环(circle)》。试题C尽管存在O(N)的解法，但从思维复杂度和编程复杂度（避免高精度）两方面考虑，为使试题难度不过于高，只给出了较小的数据规模。事实上，如果需要加强试题，可将试题C稍作修改得到试题D。

**试题D：一个环上有N个点，每个点染为M种颜色之一。求所有环状态中环上同色连续段长度之积的期望值，结果用最简分数形式表示。（N≤2000, M≤104）**

容易发现，这一题的状态设计、递推公式和优化方法都与试题C相同。本题与试题C的关键差异在于结果的表示形式。这一改动使得程序中需要实现高精度计算，增加了编程复杂度。具体实现时，考虑到分母的特殊形式(MN)，可以通过多次高精度数除以单精度数较为简洁地实现。本题是试题C的加强版，只有思维难度和编程难度均很高的时间复杂度为O(NL)，空间复杂度为O(L)的最优算法能够得到满分。

### 【考查要点】

**本题考察的算法是递推及其优化。但值得注意的是，本题的算法探究过程是逐步挖掘问题本质的过程。本题实际考察的是选手抓住事物内在联系，摒除冗余信息，探索问题本源的能力。这一点才是更为重要的。**

### 【数据设计】

**考虑到选手的能力差异，本题的数据分为三类：**

**第一类数据规模很小，使得朴素的搜索算法能够得分，这一类数据占总数据的20%；**

**第二类数据规模中等，使得朴素的递推算法能够得分，这一类数据占总数据的30%；**

**第三类数据规模较大，使得优化了状态设计和递推公式计算的递推算法能够得分，这一类数据占总数据的50%。**

**在10个测试数据中，有两个边界情况数据，分别属于第一类和第三类。换言之，如果对这些边界情况进行特殊处理，朴素的搜索程序可以获得不超过30分，朴素的递推算法能够获得不超过60分。**

### 【期望得分】

**本题属于NOI竞赛中的中等难度题。考虑到参与NOI的选手的实力一方面不断增强，一方面又参差不齐，预计有超过30%的选手获得至少20分，超过5%的选手获得至少50分，少量选手获得满分。平均分预计在10分左右。**

### 【感谢】

我曾将试题B提供给北京大学的曹钦翔同学，他完成了该题的简要解题报告，收录在IOI2009国家集训队第二次作业中。该份报告对于本次试题创作和整理有很大的帮助。**华师大二附中的邓永行同学在试题的创作和整理过程中同样给予我很大的帮助，在此一并致谢。**

### 【附录】

1. **《彩色圆环(circle)》原题**

## **彩色圆环(circle)**

命题人：华东师范大学第二附属中学 吴佳俊

时间限制：1秒 空间限制：64M

## 【关键字】

递推优化

### 【问题描述】

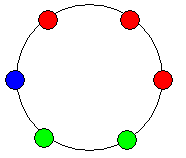
小A喜欢收集宝物。一天他得到了一个圆环，圆环上有*N*颗彩色宝石，闪闪发光。小A很爱惜这个圆环，天天把它带在身边。

一天，小A突然发现圆环上宝石的颜色是会变化的。他十分惊讶，仔细观察这个圆环后发现，圆环上宝石的颜色每天变化一次，而且每颗宝石的颜色都**等概率地**为特定的*M*种颜色之一。小A发现了这个秘密后，对圆环更是爱不释手，时时刻刻都在研究。

又经过了一段时间，小A发现因为圆环上宝石的颜色不断变化，圆环有时会显得比其他时候更美丽。为了方便比较，小A这样定义圆环的“**美观程度**”：

1. 设圆环上**相同颜色**的宝石构成的**连续段**长度分别为*a*1, *a*2, ..., *an*；
2. 定义圆环的“**美观程度**” *R* = = *a*1 \* *a*2 \* ... \* *an*。

以图一给出的圆环为例，有*a*1 = 3, *a*2 = 2, *a*3 = 1，故*R* = 6。



图一

现在小A想知道，在上述前提下，圆环的“**美观程度**”的**期望值***E*(*R*)是多少。因为如果知道了*E*(*R*)，他就可以判断每天变化出的新圆环是否比一般情况更美丽。

**说明**：“**美观程度**”的**期望值**即为对每种可能的圆环状态的“**美观程度**”与其**出现概率**的**乘积**进行**求和**所得的值。

### 【输入文件】

输入文件circle.in仅有一行，该行给出依次两个正整数*N*, *M*，分别表示宝石的**个数**和宝石在变化时可能变成的**颜色种类数**。

### 【输出文件】

输出文件circle.out应仅有一行，该行给出一个实数*E*(*R*)，表示圆环的“**美观程度**”的**期望值**。

### 【输入样例一】

3 2

### 【输出样例一】

2.25

### 【输入样例二】

200 1

### 【输出样例二】

200

### 【数据规模和约定】

20%的数据满足1 ≤ *N*, *M* ≤ 8；

50%的数据满足1 ≤ *N*, *M* ≤ 25；

100%的数据满足1 ≤ *N* ≤ 200, 1 ≤ *M* ≤ 109。

### 【评分标准】

对每个测试点，若你给出的*E*(*R*)与标准程序给出的*E*(*R*)’的**相对误差**不超过10-7，则该测试点得满分；否则该测试点得零分。

**说明**：**相对误差**　 .

1. **《彩色圆环(circle)》解题报告**

## 《彩色圆环(circle)》解题报告

华东师范大学第二附属中学 吴佳俊

### 【时空限制】

每个测试点时间限制1秒，空间限制64M。

### 【问题描述】

一个环上有N个点，每个点随机染为M种颜色之一。求环上同色连续段长度之积的期望值。

### 【数据规模和约定】

20%的数据满足1 ≤ N, M ≤ 8；

50%的数据满足1 ≤ N, M ≤ 25；

100%的数据满足1 ≤ N ≤ 200, 1 ≤ M ≤ 109。

### 【算法描述】

本题最朴素的方法是搜索。此算法的时间复杂度为O(MNN)，最多可优化至O(MN)，可以通过20%的测试数据。

对于环的处理，有一种常用的方法，即断环为链。如果将本题的环按链的形式考虑，则容易发现，递推是解决问题的好方法。

**用f[i, col, len]表示考虑了前i个点，当前最后一个连续段的颜色为col，长度为len时所求的期望值(暂不乘以最后一个连续段的长度)。递推公式即**

上述公式还可进一步使用部分和优化，即令。上述公式的时间复杂度为O(N2M+NM2)。

再考虑环的情况，环与链的差异在于头尾的连接可能会使所求值变化。为此，容易想到只需在递推中，把头尾的状态都表示出即可。为此，可增加两维，即

**用f[i, f\_col, f\_len, l\_col, l\_len]表示考虑了前i个点，第一个连续段的颜色为f\_col，长度为f\_len，当前最后一个连续段颜色为l\_col，长度为l\_len时所求的期望值(暂不乘以第一个和最后一个连续段的长度)。递推公式和链的递推公式大致相同，同样可以使用部分和优化。**

注意到这样的递推设计使得仅状态总数就高达O(N3M2)，算法的总时间复杂度达到O(N3M3)。由于常数很小，在使用了部分和优化的情况下可以通过50%的测试数据。

为了进一步优化算法，尝试研究状态的表示。注意到

**在f\_col确定的情况下，任何与f\_col不等的l\_col的值都是等价的。进一步，我们关注的只是f\_col与l\_col的异同，而并非f\_col或l\_col的具体取值。**

这是本题的核心结论。根据这个结论，状态设计就可大幅简化。

**用f[i, f\_len, l\_len]表示考虑了前i个点，第一个连续段的长度为f\_len，当前最后一个连续段的长度为l\_len，且第一个连续段与最后一个连续段颜色相同时所求的期望值(暂不乘以第一个和最后一个连续段的长度)；用g[i, f\_len, l\_len]表示考虑了前i个点，第一个连续段的长度为f\_len，当前最后一个连续段的长度为l\_len，且第一个连续段与最后一个连续段颜色相异时所求的期望值(暂不乘以第一个和最后一个连续段的长度)。从而有递推公式**

**类似地，使用部分和优化时，可令，**

**上述计算完毕后，只需再计算 即为所求值。**

**递推的边界状态值得注意。边界状态包括：**

**综上得，递推状态总数为O(N3)，每个状态计算耗时为O(1)，算法的时间复杂度为O(N3)。使用滚动数组，算法的空间复杂度为O(N2)。此算法可以得到满分。**

**考虑到试题的思维难度和编写难度，本题的数据规模使得O(N3)的算法可以获得满分。但事实上，本题可以进一步加强，算法也可以进一步优化。**

**首先，我们注意到可能的方案总数为MN。因此，计算期望值时的分母是确定的，我们可以不必考虑除法，只计算分子。进一步观察新递推式，我们可以发现，由此注意到l\_len这一维的冗余。**

**尝试削去冗余，用f[i, f\_len]表示考虑了前i个点，第一个连续段长度为f\_len，第一个连续段与第i个点之后的第一个连续段同色时所求的积的和；g[i, f\_len]表示考虑了前i个点，第一个连续段长度为f\_len，第一个连续段与第i个点之后的第一个连续段异色时所求的积的和。从而有**

如果直接完成计算，每次需要O(N)时间，但容易注意到上式可用部分和优化。即令

**则成功使得上式的时间复杂度降为O(N2)，使用滚动数组，空间复杂度降为O(N)。**

**事实上，算法还可以进一步优化。使用类似的方法，注意到，由此发现f\_len一维的冗余，不妨进一步改进，放弃f\_len的表示。故有**

**用f[i]表示考虑了第一个连续段之后的i个点，第一个连续段与第i个点之后的第一个连续段同色时所求的积的和；g[i]表示考虑了第一个连续段之后的i个点，第一个连续段与第i个点之后的第一个连续段异色时所求的积的和。从而有**

**可计算与O(N2)算法类似的部分和，从而递推的时间复杂度降为O(N)。如果我们能够将计算最终结果的时间复杂度也降为O(N)或更低，则算法的总时间复杂度降为O(N)，空间复杂度可用滚动数组降为O(1)。**

**事实上这是可行的，我们只需在计算每个f[i]/g[i]时，考虑环的其他部分仅构成一个首尾相连的连续段的情况并计算即可。具体地，即计算**

### 【其他算法】

目前已知的其他算法与上述算法本质相同，故不再赘述。

### 【实现情况】

circle.cpp实现了算法描述中时间复杂度为O(N3)的递推算法。