NOI 2007 总结与解题

余林韵

福州第一中学

高三九班

yydjtc1990@gmail.com

2007 年 8 月 23 日

目 录

Chapter 1

Survey

1.1 Day 1

Problem	Contest Status	Current Status	Solution
network	100	Accept	DP & Find Shortest Path
cash	100	Accept	DP & AVL
surround	40		Network Flow & Match

1.2 Day 2

Problem	Contest Status	Current Status	Solution
necklace	50	Accept	Segment Tree
count	50	Accept	DP & Matrix
catch	57	76	Construct

Chapter 2

Day 1

2.1 比赛情况

第一天的比赛非常紧张，因为这是我最后一次参加 NOI 了。刚看到第一题的时候觉得有些难以置信，因为迅速看出了题目的解法，怀疑想法出了问题，有漏洞。但是听到了周边噼噼啪啪的敲键盘声，便放下心来认真打代码去了做完后开始认真想第二题，由于刚开始没有注意到后面的 Hint，所以走入误区，我猜出来了那个 Hint，便认为是贪心，开始写程序，后来写着写着就发现出现问题了，还好错误代码并不很长，并没有花去很长时间。我及时“悬崖勒马，回头是岸”。想出 60 分的 DP 方法。然后我就去做第三题了，因为第三题是提交答案题，得分应该还是比第二题剩下的 40 分容易。提交答案题我遵循了“看数据，找规律”的方法，针对小型的特殊数据找了各种构造方法，结果得了 40 分还是令人满意的，但是还是有所失误。做到此时，比赛还有 2 个多小时。我决定重返第二题。由于第二题 DP 的状态转移方程那么华丽，因此我觉得必然存在某种单调性来对方程进行优化。果然，最后我找到了单调性，但是令人惊讶的是，此题的单调性需要用平衡树维护，虽然我写过平衡树，也了解单调性优化，但是还从未把两者结合在一起写过，由于时间充裕，最后我还是努力写了下来，虽然觉得味道有一些怪。但是，我成功了。5 个小时，紧张的缘故使我没有吃任何的点心，仅仅是喝了几口水。最后的成绩让我瞬间放松了下来。240，第一名，为我之后的 day2 奠定了相当好的基础。

2.2 社交网络

2.2.1 原题：

问题描述：

在社交网络 (**social network**) 的研究中，我们常常使用图论概念去解释一些社会现象。

不妨看这样的一个问题。在一个社交圈子里有 n 个人，人与人之间有不同程度的关系。我们将这个关系网络对应到一个 n 个结点的无向图上，两个不同的人若互相认识，则在他们对应的结点之间连接一条无向边，并附上一个整数权值 c，c 越小，表示两个人之间的关系越密切。

我们可以用对应结点之间的最短路长度来衡量两个人 s 和 t 直接的关系密切程度，注意到对最短路径上的其他结点为 s 和 t 之间的联系有一定的重要程度。那么我们通过统计通过一个结点 v 的最短路径的数目来衡量该结点在社交网络中的重要程度，是有一定道理的。

考虑到两个结点 A 和 B 之间可能会有多条最短路径。我们修改重要程度的定义录下：

定义1. 令 $C_{s,t}$ 表示从 s 到 t 的不同的最短路的数目；则定义

$$I(v) = \sum_{s \neq v, t \neq v} \frac{C_{s,t}(v)}{C_{s,t}}$$

为结点 v 在社交网络中的重要程度。

为了使 $I(v)$ 和 $C_{s,t}(v)$ 有意义，我们规定需要处理的社交网络都是连通的无向图，即任意两个结点之间都有一条有限长度的最短路径。

现在给出这样一幅描述社交网络的加权无向图，请你求出每一个结点的重要程度。

输入数据：

输入文件中第一行有两个整数，n 和 m，表示社交网络中结点和无向边的数目。在无向图中，我们将所有节点从 1 到 n 进行编号。

接下来 m 行，每行用三个整数 a,b,c 描述一条连接结点 a 和 b，权值为 c 的无向边。注意任意两个结点之间最多有一条无向边相连，无向图中也不会出现自环（即不存在一条无向边的两个端点是相同的结点）。

输出数据：

输出文件包括 n 行，每行一个实数，精确到小数点后 3 位。第 i 行的实数表示结点 i 在社交网络中的重要程度。

样例输入：

```
4 4
1 2 1
2 3 1
3 4 1
4 1 1
```

样例输出：

```
1.000
1.000
1.000
1.000
```

样例说明：

社交网络如图 2.1 所示。对于 1 号结点而言，只有 2 号到 4 号结点和 4 号到 2 号结点的最短路经过 1 号结点，而 2 号结点和 4 号结点之间的最短路又有 2 条。因而根据定义，1 号结点的重要程度计算为 $\frac{1}{2}+\frac{1}{2}=1$。由于图的对称性，其他三个结点的重要程度也都是 1。

数据规模和约定：

50% 的数据中：$n \leqslant 10$，$m \leqslant 45$。

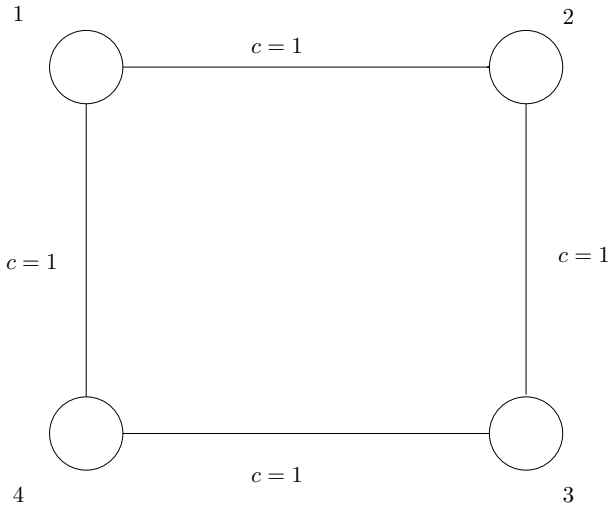

Figure 2.1: Explanation

100% 的数据中：$n \leqslant 100$，$m \leqslant 4500$。

所有数据中保证给出的无向图连通，且任意两个结点之间的最短路径数目不超过 10^{10}。

2.2.2 问题分析：

本题是这次比赛中最简单的一道题目，可以把它理解为送分题吧。具体考察的知识点有：最短路算法，加法原理以及乘法原理。

记 s 至 t 的最短路径长度为 $D_{s,t}$，s 至 t 的最短路径条数为 $F_{s,t}$。易知：

命题1.

$$C_{s,t}(v) = F_{s,v} \times F_{v,t}, (D_{s,v} + D_{v,t} = D_{s,t})$$
$$C_{s,t}(v) = 0, (D_{s,v} + D_{v,t} > D_{s,t})$$

故只需做一遍 Dijkstra，并在做的过程中记录下 $F_{s,t}$ 即可。

维护 $F_{s,t}$ 的方法：[1]

$$\begin{cases} F_{s,k} = F_{s,k} + F_{s,x} & \text{if } D_{s,k} = D_{s,x} + E_{x,k} \\ D_{s,k} = D_{s,x} + E_{x,k}, F_{s,k} = F_{s,x} & \text{otherwise} \end{cases}$$

[1] 下面的方程中 s 为起点，x 为当前确定 dist 的结点, k 为待扩展结点

时间复杂度： $O(N^3)$
空间复杂度： $O(N^3)$
期望得分： 100 分 □

2.3 货币兑换

2.3.1 原题：

问题描述：

小 Y 最近在一家金券交易所工作。该金券交易所只发行交易两种金券：A 纪念券（以下简称 A 券）和 B 纪念券（以下简称 B 券）。每个持有金券的顾客都有一个自己的帐户。金券的数目可以是一个实数。每天随着市场的起伏波动，两种金券都有自己当时的价值，即每一单位金券当天可以兑换的人民币数目。我们记录第 K 天中 A 券和 B 券的价值分别为 A_K 和 B_K（元/单位金券）。

为了方便顾客，金券交易所提供了一种非常方便的交易方式：比例交易法。比例交易法分为两个方面：

a) 卖出金券：顾客提供一个 $[0, 100]$ 内的实数 OP 作为卖出比例，其意义为：将 $OP\%$ 的 A 券和 $OP\%$ 的 B 券以当时的价值兑换为人民币；

b) 买入金券：顾客支付 IP 元人民币，交易所将会兑换给用户总价值为 IP 的金券，并且，满足提供给顾客的 A 券和 B 券的比例在第 K 天恰好为 $Rate_K$；

例如，假定接下来 3 天内的 A_K、B_K、$Rate_K$ 的变化分别为：

时间	A_K	B_K	$Rate_K$
第一天	1	1	1
第二天	1	2	2
第三天	2	2	3

假定在第一天时，用户手中有 100 元人民币但是没有任何金券。

用户可以执行以下的操作：

时间	用户操作	人民币（元）	A 券的数量	B 券的数量
开户	无	100	0	0
第一天	买入 100 元	0	50	50
第二天	卖出 100%	150	0	0
第二天	买入 150 元	0	75	37.5
第三天	卖出 100%	225	0	0

注意到，同一天内可以进行多次操作。

小 Y 是一个很有经济头脑的员工，通过较长时间的运作和行情测算，他已经知道了未来 N 天内的 A 券和 B 券的价值以及 Rate 。他还希望能够计算出来，如果开始时拥有 S 元钱，那么 N 天后最多能够获得多少元钱。

输入数据：

第一行两个正整数 N 、S ，分别表示小 Y 能预知的天数以及初始时拥有的钱数。

接下来 N 行，第 K 行三个实数 A_K、B_K、$Rate_K$，意义如题目中所述。

输出数据：

只有一个实数 MaxProfit ，表示第 N 天的操作结束时能够获得的最大的金钱数目。答案保留 3 位小数。

样例输入：

```
3 100
1 1 1
1 2 2
2 2 3
```

样例输出：

```
225.000
```

评分方法：

本题没有部分分，你的程序的输出只有和标准答案相差不超过 0.001 时，才能获得该测试点的满分，否则不得分。

数据规模和约定：

测试数据设计使得精度误差不会超过 10^{-7}。
对于 40% 的测试数据，满足 $N \leqslant 10$；
对于 60% 的测试数据，满足 $N \leqslant 1000$；
对于 100% 的测试数据，满足 $N \leqslant 100000$；
对于 100% 的测试数据，满足：

$0 < A_K \leqslant 10$；
$0 < B_K \leqslant 10$；
$0 < Rate_K \leqslant 100$；
$MaxProfit \leqslant 10^9$；

提示：

输入文件可能很大，请采用快速的读入方式。
必然存在一种最优的买卖方案满足：

- 每次买进操作使用完所有的人民币；
- 每次卖出操作卖出所有的金券。

2.3.2 问题分析：

解法一：

由于有 Hint 的存在，因此很容易想到一个 $O(N^2)$ 的算法。记 FB_i 为在第 i 天最多能够拥有多少 B 类券，FA_i 为在第 i 天最多能够拥有多少 A 类券。显然有

$$FA_i = FB_i \times Rate_i$$

那么，对于第 i 天最后一次购买，它在第 j 天 $\{i \leqslant j\}$ 的价值就是：

$$Worth_{i,j} = FB_i \times B_j + FA_i \times A_j$$

于是：

$$FB_j = \max_{i<j}\{\frac{Worth_{i,j}}{A_j \times Rate_j + B_j}\},$$
$$FA_j = FB_j \times Rate_j$$
$$Ans = \max_{1\leqslant i\leqslant n}\{Worth_{i,n}\}$$

时间复杂度： $O(N^2)$
空间复杂度： $O(N)$
期望得分： 60 分

解法二：

在解法一的基础上，通过优化方程，我们想到了解法二。
重新考察方程：

$$FB_j = \max_{i<j}\{\frac{Worth_{i,j}}{A_j \times Rate_j + B_j}\},$$
$$FA_j = FB_j \times Rate_j$$

我们可以把最后一次购入金券的时间看作一种决策。考虑不同的两种决策，在此时哪种决策能够获得最大的价值哪种决策显然此时更优。假设此时 A 券的价值为 P_A，B 券的价值为 P_B，两个不同的决策是：最后一次在第 x 天和第 y 天买入金券。那么，决策 x 不如决策 y 优等价于：

$$FA_x \times P_A + PB_x \times P_B \leqslant FA_y \times P_A + PB_y \times P_B$$
$$\Leftrightarrow (FA_x - FA_y) \times P_A \leqslant (FB_y - FB_x) \times P_B$$

$\because P_A > 0$

$$\therefore \text{决策}x\text{不如决策}y\text{优} \Leftrightarrow \begin{cases} \frac{FA_x - FA_y}{FB_y - FB_x} \leqslant \frac{P_B}{P_A} & \text{if } FB_y - FB_x > 0 \\ \frac{FA_x - FA_y}{FB_y - FB_x} \geqslant \frac{P_B}{P_A} & \text{if } FB_y - FB_x < 0 \\ FA_x \leqslant FA_y & \text{if } FB_y = FB_x \end{cases}$$

由此，我们可以构造决策序列，令：

$$\begin{cases} G(x,y) = \frac{FA_x - FA_y}{FB_y - FB_x} & \text{if } FB_y - FB_x < 0 \\ G(y,x) = \frac{FA_x - FA_y}{FB_y - FB_x} & \text{if } FB_y - FB_x > 0 \end{cases}$$

假设此时的决策序列为 i_1,i_2,...,i_k，则它们满足: $-\infty < G(i_1,i_2) < G(i_2,i_3) < \ldots < G(i_{k-1},i_k) < \infty$。

那么在选取有关 x 的最优决策时，记 $H(x) = \frac{B_x}{A_x}$，找到最大的 l 满足 $G(i_{l-1},i_l) \leqslant H(x) \leqslant G(i_l,i_{l+1})$，则决策 l 对于 x 决策最优。

由此可以计算出 $FA_i, FB_i\{i \in [1,n]\}$。由于 $H(x)$ 以及决策 y 均非单调，因此需要使用平衡树来维护决策序列与做出决策。考场上我使用的是 Treap 。

时间复杂度： $O(Nlog_2N)$

空间复杂度： $O(Nlog_2N)$

期望得分： 100 分 □

2.4 调兵遣将

2.4.1 原题：

问题描述：

我军截获的情报显示，敌军正在集结兵力试图向我军重要的军械研究所发起进攻。由于我军正处于多线作战的状态，无法抽调大批兵力前去支援，指挥部决定通过有效的战前部署来提高胜率，减少伤亡和损失。

该军械研究所的平面图可以看作是一个 $N \times M$ 的矩阵，每个 1×1 的格子都表示一个区域，每个区域只与它上下左右的四个区域相邻。每个区域的用途可分为以下 3 种之一：

1. 该区域被用于军事研究（用字母’O’表示）；
2. 该区域内驻扎有一个机械化中队（用’#’表示）；
3. 该区域是空地（用’.’表示）。

由于空间有限，任一个 1×1 的格子内都无法驻扎两队以上的机械化中队（包括两队），否则会大大降低战斗时的机动性。

遗憾的是，由于战前估计不足，我军的防御部署显得十分分散，这很容易让敌军所擅长的偷袭战术得逞。为了确保万无一失，我军决定利用为数不多的防御部队以最少的移动步骤将所有重要研究区域都包围起来。所谓的“包围”即从该矩阵边界侵入的敌军找不到任意一条路，使得他们不遭受任何机械化中队的反抗就能到达某研究区域。

由于军队内部的传令权限的限制，每个单位时间指挥部只能向所有中队中的一个中队下达指令（朝上/下/左/右移动 1 格）。由于时间紧迫，指挥部希望能够尽快完成部署，这个任务就交给你来完成。

注意：为了确保不在研究区域发生战斗，最终部署计划不允许某个研究区域驻扎有军队。

输入数据：

该题为提交答案型试题，所有输入数据 surround1.in ～ surround10.in 在考试开始前已被存入各位选手的试题目录下。

对于每个数据：

第一行包括 2 个整数 N,M，接下来 N 行，每行包括 M 个字符（’.’,’O’ 或’#’）。

输出数据：

针对给定的 10 个输入文件 surround1.in ～ surround10.in，你需要分别提交你的输出文件 surround1.out ～ sourround10.out 。

每个输出文件的第一行，包括你的答案所花费的时间 T

接下来 T 行，按顺序输出每条命令，每行包括 4 个整数 x1,y1,x2,y2，表示将位于 (x1,y1) 的部队移向 (x2,y2)。

样例输入：

```
5 5
..##.
#...#
#OOO#
#..O#
.###.
```

样例输出：

```
1
2 1 2 2
```

评分方法：

如果选手的输出方案不合法（方案执行过程中出现军队重叠，军队移出矩形边界，最终方案有军队和研究所在同一区域，军队没有包围研究所等），

则得零分，否则设选手输出的方案耗时为 ans，则得分按如下计算：

$$\begin{cases} 10 & ans \leqslant A_i \\ 1 + \left\lfloor \left(\frac{ans - B_i}{A_i - B_i} \right)^2 \times 9 \right\rfloor & A_i < ans \leqslant B_i \\ 1 & B_i < ans \end{cases}$$

对于每个数据，都有两个评分参数 A_i 与 B_i，其中保证 $A_i < B_i$。

你如何测试自己的输出：

我们提供 surround_check 这个工具来测试你的输出文件是否是可接受的。使用这个工具的方法是：

`surround_check <输入文件名> <输出文件名>`

在你调用这个程序后，surround_check 将根据你给出的输入和输出文件给出测试的结果，其中包括：

- 非法退出：未知错误；
- overlap：出现军队重叠，或最终方案有军队和研究所在同一区域；
- outside：军队移出矩形边界；
- move error：移动一个不存在的军队，或移动距离不只一格；
- not surround：军队没有包围所有研究所；
- time not match：输出文件中移动步数与输出答案不符；
- yes：输出可接受，将进行下一步的评分。

2.4.2 问题分析：

对于提交答案型试题，首先要做的就是分析各个数据，找出其中隐含的特殊性，并有针对性的编写程序来获得较优解。

第 1, 2 个测试数据为手工测试点。考场上时间不够 1 没做出来。

第 $3,4,5,8$ 为较特殊的数据。4 中 O 十分分散，并且有规律。我用的最短路做。$3,5,8$ 都是模板型数据，即：处理一种小数据然后复制若干遍。但是对于数据 8 要比较小心，它的小数据比较阴，稍有不慎就会失误，我就只拿了 $1,2$ 分。

剩余的 $6,7,9,10$ 为随机大数据。考场我没有做，但是据说将外围（边沿）全部覆盖就是一种好方法。

标准的算法是网络流 + 最佳匹配。首先运用最小割求出最少需要放置 # 的位置，其次计算出哪些 # 是可移动的。最后运用最佳匹配，将可移动的 # 作为 X 部，需放置的地方作为 Y 部，两者间距离作为权值，求出答案。更改结点间在网络流中的容量，多次运行，取最优解。 □

Chapter 3

Day 2

3.1 比赛情况

第二天的比赛我并没有发挥的很好，可能主要是自己比较保守的缘故吧，因为第一天的分数让我认为稳定，不出错最重要，因此三题都拿分了，但是都不高。三道题目是按顺序一题一题切的。第一题，保守拿到 50 分，与期望的 60 有差距，不知道是什么原因；第二题，是我考场上花时间最长的一题，由于我看出 $k = 2$ 与 Fibonacci 数列有关，因此我就一直致力于找出 $k = 3$ 的数列通项，这是考场上的一个败笔吧；其实如果我能跳出思维误区，应该也是能很快想到标准方法的，因为我的想法已经和解法比较接近了，都是矩阵连乘；第三题我的想法是随机化，我认为 DP 是错误的，没有优势，并且事实上也是这样的，但是数据没有卡 DP，亏死了 ...，不过话说回头，我自己时间没有掌控好，随机化的次数太少了，也很失败。157 分，差强人意。过去的就让它过去吧，总结经验，争取下次不要再犯同样的错误。

3.2 项链工厂

3.2.1 原题：

问题描述：

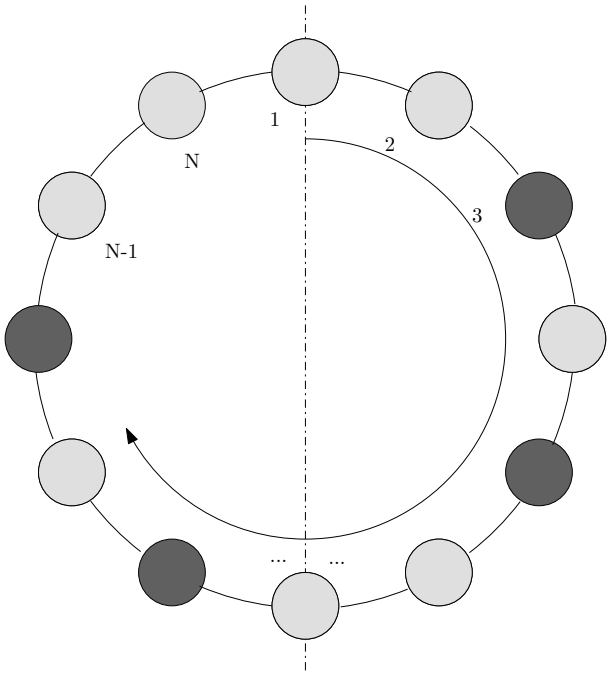

Figure 3.1: 一条项链

T 公司是一家专门生产彩色珠子项链的公司，其生产的项链设计新颖、款式多样、价格适中，广受青年人的喜爱。最近 T 公司打算推出一款项链自助生产系统，使用该系统顾客可以自行设计心目中的美丽项链。

该项链自助生产系统包括硬件系统与软件系统，软件系统与用户进行交互并控制硬件系统，硬件系统接受软件系统的命令生产指定的项链。该系统的硬件系统已经完成，而软件系统尚未开发，T 公司的人找到了正在参加全国信息学竞赛的你，你能帮助 T 公司编写一个软件模拟系统吗？

一条项链包含 N 个珠子，每个珠子的颜色是 1, 2, . . . ,c 中的一种。项链被固定在一个平板上，平板的某个位置被标记位置 1，按顺时针方向其他位置被记为 2, 3, . . . ,N 。

你将要编写的软件系统应支持如下命令：

命令	参数限制	内容
R k	$0 < k < N$	意为 Rotate k 。将项链在平板上顺时针旋转 k 个位置, 即原

		来处于位置 1 的珠子将转至位置 $k+1$，处于位置 2 的珠子将转至位置 $k+2$，依次类推。
F		意为 Flip 。将平板沿着给定的对称轴翻转，原来处于位置 1 的珠子不动，位置 2 上的珠子与位置 N 上的珠子互换，位置 3 上的珠子与位置 $N-1$ 上的珠子互换，依次类推。
S i j	$1 \leqslant i, j \leqslant N$	意为 Swap i, j 。将位置 i 上的珠子与位置 j 上的珠子互换。
P i j x	$1 \leqslant i, j \leqslant N, x \leqslant c$	意为 Paint i, j, x 。将位置 i 沿顺时针方向到位置 j 的一段染为颜色 x。
C		意为 Count 。查询当前的项链由多少个“部分”组成，我们称项链中颜色相同的一段为一个“部分”。
CS i j	$1 \leqslant i, j \leqslant N$	意为 CountSegment i, j 。查询从位置 i 沿顺时针方向到位置 j 的一段中有多少个部分组成。

输入数据：

输入文件第一行包含两个整数 N, c ，分别表示项链包含的珠子数目以及颜色数目。第二行包含 N 个整数，$x_1, x_2, \ldots, x_n$，表示从位置 1 到位置 N 的珠子的颜色，$1 \leqslant x_i \leqslant c$。第三行包含一个整数 Q ，表示命令数目。接下来的 Q 行每行一条命令，如上文所述。

输出数据：

对于每一个 C 和 CS 命令，应输出一个整数代表相应的答案。

样例输入：

```
5 3
1 2 3 2 1
4
C
R 2
P 5 5 2
CS 4 1
```

样例输出：

```
4
1
```

评分方法：

本题没有部分分，你的程序的输出只有和标准答案完全一致才能获得满分，否则不得分。

数据规模和约定：

对于 60% 的数据，$N \leqslant 1000$，$Q \leqslant 1000$；
对于 100% 的数据，$N \leqslant 500000$，$Q \leqslant 500000$，$c \leqslant 1000$。

3.2.2 问题分析：

考场上苦于无法解决翻转和旋转操作，没有得到优秀算法。

其实，只要发现：**不论无论怎样翻来倒去，珠子的相对位置总是不变的**，这题便可以迎刃而解。

建立一棵线段树，对于每个线段维护如下值：这条线段的颜色（如果整段同色的话），线段最左端的颜色，线段最右端的颜色，该条线段有多少部分组成。用 head 记录最初位于一号位置的珠子，再用 IsRetroflex 来判断是否需要翻转，接下来使用正常的线段树操作即可。注意：对于有需要 i , j 的操作而言, i , j 要替换成现在位于 i , j 的珠子的起始位置再进行操作：

- Flip: 直接将 IsRetroflex->not IsRetroflex;
- Rotate K: 如果 IsRetroflex 为真，则 head->head+k;
  otherwise,head->head-k;
- Change x into origin: 如果 IsRetroflex 为真，则 origin->head-x+1;
  otherwise,origin->x+head - 1;

时间复杂度： $O(Nlog_2N)$
空间复杂度： $O(Nlog_2N)$
期望得分： 100 分 □

3.3 生成树计数

3.3.1 原题：

问题描述：

最近，小栋在无向连通图的生成树个数计算方面有了惊人的进展，他发现：

- n 个结点的环的生成树个数为 n 。
- n 个结点的完全图的生成树个数为 n^{n-2}。

这两个发现让小栋欣喜若狂，由此更加坚定了他继续计算生成树个数的想法，他要计算出各种各样图的生成树数目。

一天，小栋和同学聚会，大家围坐在一张大圆桌周围。小栋看了看，马上想到了生成树问题。如果把每个同学看成一个结点，邻座（结点间距离为 1）的同学间连一条边，就变成了一个环。可是，小栋对环的计数已经十分娴熟且不再感兴趣。于是，小栋又把图变了一下：不仅把邻座的同学之间连一条边，还把相隔一个座位（结点间距离为 2）的同学之间也连一条边，将结点间有边直接相连的这两种情况统称为有边相连，如图 3.2 所示。

小栋以前没有计算过这类图的生成树个数，但是，他想起了老师讲过的计算任意图的生成树个数的一种通用方法：构造一个 $n \times n$ 的矩阵 $A = \{a_ij\}$

$$\text{其中} a_{ij} = \begin{cases} d_i & i = j \\ -1 & i \text{ 与 } j \text{ 有边相连} \\ 0 & otherwise \end{cases}$$

其中 d_i 表示结点 i 的度数。

与图 3.2 相应的 A 矩阵如下所示。为了计算图 3.2 所对应的生成数的个数，只要去掉矩阵 A 的最后一行和最后一列，得到一个 $(n-1) \times (n-1)$

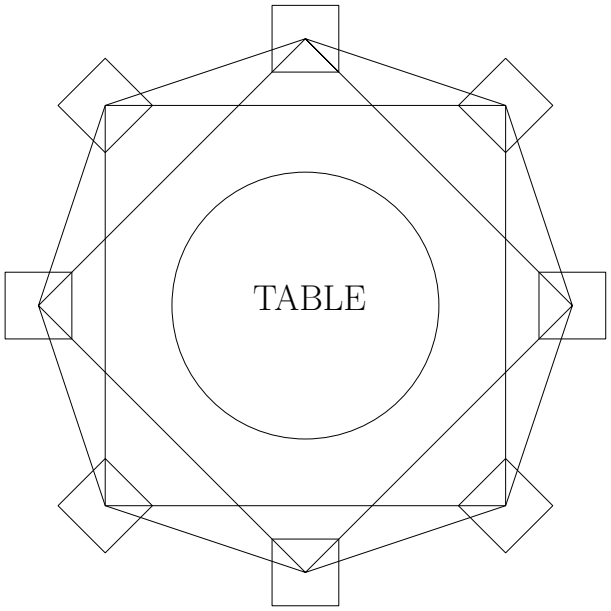

Figure 3.2:

的矩阵 B，计算出矩阵 B 的行列式的值便可得到图 3.2 的生成树的个数。

$$\mathbf{A} = \begin{bmatrix} 4 & -1 & -1 & 0 & 0 & 0 & -1 & -1 \\ -1 & 4 & -1 & -1 & 0 & 0 & 0 & -1 \\ -1 & -1 & 4 & -1 & -1 & 0 & 0 & 0 \\ 0 & -1 & -1 & 4 & -1 & -1 & 0 & 0 \\ 0 & 0 & -1 & -1 & 4 & -1 & -1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 4 & -1 & -1 \\ -1 & 0 & 0 & 0 & -1 & -1 & 4 & -1 \\ -1 & -1 & 0 & 0 & 0 & -1 & -1 & 4 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 4 & -1 & -1 & 0 & 0 & 0 & -1 \\ -1 & 4 & -1 & -1 & 0 & 0 & 0 \\ -1 & -1 & 4 & -1 & -1 & 0 & 0 \\ 0 & -1 & -1 & 4 & -1 & -1 & 0 \\ 0 & 0 & -1 & -1 & 4 & -1 & -1 \\ 0 & 0 & 0 & -1 & -1 & 4 & -1 \\ -1 & 0 & 0 & 0 & -1 & -1 & 4 \end{bmatrix}$$

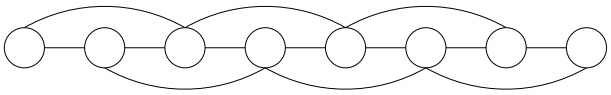

Figure 3.3:

所以生成树的个数为 $|B| = 3528$。小栋发现利用通用方法，因计算过于复杂而很难算出来，而且用其他方法也难以找到更简便的公式进行计算。于是，他将图做了简化，从一个地方将圆桌断开，这样所有的同学形成了一条链，连接距离为 1 和距离为 2 的点。例如八个点的情形如图 3.3：

这样生成树的总数就减少了很多。小栋不停的思考，一直到聚会结束，终于找到了一种快捷的方法计算出这个图的生成树个数。可是，如果把距离为 3 的点也连起来，小栋就不知道如何快捷计算了。现在，请你帮助小栋计算这类图的生成树的数目。

输入数据：

输入文件中包含两个整数 k,n ，由一个空格分隔。k 表示要将所有距离不超过 k（含 k ）的结点连接起来，n 表示有 n 个结点。

输出数据：

输出文件输出一个整数，表示生成树的个数。由于答案可能比较大，所以你只要输出答案除 65521 的余数即可。

样例输入：

```
3 5
```

样例输出：

```
75
```

样例说明：

样例对应的图如下：

数据规模和约定：

对于所有的数据 $2 \leqslant k \leqslant n$

数据编号	k 范围	n 范围	数据编号	k 范围	n 范围
1	$k = 2$	$n \leqslant 10$	6	$k \leqslant 5$	$n \leqslant 100$
2	$k = 3$	$n = 5$	7	$k \leqslant 3$	$n \leqslant 2000$
3	$k = 4$	$n \leqslant 10$	8	$k \leqslant 5$	$n \leqslant 10000$
4	$k = 5$	$n = 10$	9	$k \leqslant 3$	$n \leqslant 1015$
5	$k \leqslant 3$	$n \leqslant 100$	10	$k \leqslant 5$	$n \leqslant 1015$

提示：

行列式的一种计算方法，记 $\alpha(P)$ 表示 P 中逆序对的个数，令 B 的行列式

$$|B| = \sum_{P=p_1p_2\ldots p_n \text{是 1 到 } n \text{ 的排列}} (-1)^{\alpha(P)} \prod_{j=1}^{n} b_{i,p_i}$$

如，$B = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix}$，则计算如下：

P	$\alpha(P)$	b_{1,p_1}	b_{2,p_2}	b_{3,p_3}	$(-1)^{\alpha(P)} \prod_{j=1}^{n} b_{i,p_i}$
123	0	1	5	0	0
132	1	1	6	8	-48
213	1	2	4	0	0
231	2	2	6	7	84
312	2	3	4	8	96
321	3	3	5	7	-105

所以 B 的行列式为 $0 - 48 + 0 + 84 + 96 - 105 = 27$。

3.3.2 问题分析：

胡伟栋出题时" 邪恶" 得把人往行列式这条歪路上引导，但是我不会行列式的 $O(N^3)$ 的行列式算法，按他给的方法只能得到 40 分，因此我没有采用。

考场上对于 40% 的数据，我采用的是搜索。我一开始做这道题目的思路是通过小数据看出大数据的规律，但是除了发现 $N = 2$ 是 Fibonacci 数列以外，我一无所获。看胡伟栋给出的数据范围，有两个 $N \leqslant 10^15$，很明显的知道标准算法是 log_2N 级别的，于是我想到了矩阵连乘。但是做矩阵连成一样需要找到递推关系才能够继续。我在这上面失误了：我认为通过 $N-1, N-2, \ldots, N-K$ 的生成树个数能够推出 N 的生成树个数。

事实上，我们考虑的是 N 的情况下，最后 K 个节点的连通性情况，用最小表示法来表示它们，例如 $N = 3$ 的时候有如下 5 种情况[1]：$111, 112, 121, 122, 123$，对于 $N = 4$ 有 15 种状态，对于 $N = 5$ 则有 52 种。

这样通过 $N-1$ 的情况我们可以推出 N 的情况：考虑最后一个结点往前连边的情况，可以找出一一对应的关系，构造出基本矩阵。这个预处理我采用的是递归，因为递归比较易于实现。

接下来计算 $N = k$ 时不同连通性情况下的情况数，作为基底。一样采用递归的方法。

最后进行矩阵连乘。将基底乘以基本矩阵 $N-K$ 次即可。我们的答案是情况 N 下连通性为 $11\ldots1$ 的情况数。

时间复杂度： $O(log_2N \times 52^3)$ [2]

空间复杂度： $O(52^2)$

期望得分： 100 分 □

[1]数字相同表示他们处于同一连通块

[2]52 是 $K = 5$ 时连通性情况数，下同

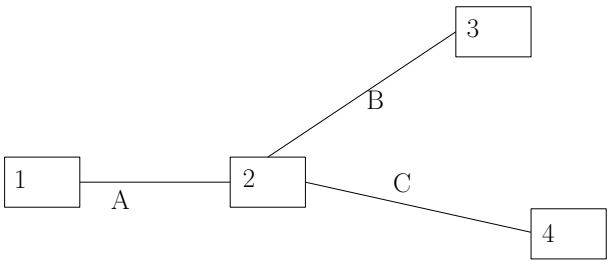

Figure 3.4: Magic Land 的一个可能格局

3.4 追捕盗贼

3.4.1 原题：

问题描述：

魔法国度 Magic Land 里最近出现了一个大盗 Frank，他在 Magic Land 四处作案，专门窃取政府机关的机密文件（因而有人怀疑 Frank 是敌国派来的间谍）。为了捉住 Frank， Magic Land 的安全局重拳出击!

Magic Land 由 N 个城市组成，并且这 N 个城市又由恰好 $N-1$ 条公路彼此连接起来，使得任意两个城市间都可以通过若干条公路互达。从数据结构的角度我们也可以说，这 N 个城市和 $N-1$ 条公路形成了一棵树。

例如，图 3.4：就是 Magic Land 的一个可能格局（4 个城市用数字编号，3 条公路用字母编号）：

大盗 Frank 能够在公路上以任意速度移动。

比方说，对于上图给出的格局，在 0.00001 秒钟内（或者任意短的一段时间内），Frank 就可以从城市 1 经过城市 2 到达城市 4，中间经过了两条公路。

想要生擒 Frank 困难重重，所以安全局派出了经验丰富的警探，这些警探具有非凡的追捕才能：

1. 只要有警探和 Frank 同处一个城市，那么就能够立刻察觉到 Frank，并且将其逮捕。

2. 虽然 Frank 可以在公路上以任意快的速度移动，但是如果有警探和

Frank 在同一条公路上相遇，那么警探也可以立刻察觉到 Frank 并将其逮捕。

安全局完全不知道 Frank 躲在哪个城市，或者正在哪条公路上移动，所以需要制定一个周密的抓捕计划，计划由若干步骤组成。在每一步中，可以做如下几件事中的一个：

1. 在某个城市空降一位警探。警探可以直接从指挥部空降到 Magic Land 的任意一个城市里。此操作记为“L x”，表示在编号为 x 的城市里空降一位警探。耗时 1 秒。

2. 把留在某个城市里的一位警探直接召回指挥部。以备在以后的步骤中再度空降到某个城市里。此操作记为“B x”。表示把编号为 x 的城市里的一位警探召回指挥部。耗时 1 秒。

3. 让待在城市 x 的一位警探沿着公路移动到城市 y，此操作记为“M x y”。耗时 1 秒。当然，前提是城市 x 和城市 y 之间有公路。如果在警探移动的过程中，大盗 Frank 也在同一条公路上，那么警探就抓捕到了 Frank 。

现在，由你来制定一套追捕计划，也就是给出若干个步骤，需要保证：无论大盗 Frank 一开始躲在哪儿，也无论 Frank 在整个过程中如何狡猾地移动（ Frank 大盗可能会窃取到追捕行动的计划书，所以他一定会想尽办法逃避），他一定会被缉拿归案。希望参与的警探越少越好，因为经验丰富的警探毕竟不多。

例如对于前面所给的那个图示格局，一个可行的计划如下：

1. L 2 在城市 2 空降一位警探。注意这一步完成之后，城市 2 里不会有 Frank，否则他将被捉住。

2. L 2 再在城市 2 空降一位警探。

3. M 2 1 让城市 2 的一位警探移动到城市 1。注意城市 2 里还留有另一位警探。这一步完成之后，城市 1 里不会有 Frank，公路 A 上也不会有 Frank 。也就是说，假如 Frank 还没有被逮捕，那么他只能是在城市 3 或城市 4 里，或者公路 B 或公路 C 上。

4. B 1 召回城市 1 的一位警探。注意虽然召回了这位警探，但是由于我们始终留了一位警探在城市 2 把守，所以 Frank 仍然不可能跑到城市 1 或者是公路 A 上。

5. L 3 在城市 3 空降一位警探。注意这一步可以空降在此之前被召回的那位警探。这一步完成之后，城市 3 里不会有 Frank，否则他会被捉住。

6. M 3 2 让城市 3 里的一位警探移动到城市 2。这一步完成之后，如果 Frank 还没有被捉住，那他只能是在公路 C 上或者城市 4 里。注意这一步之后，城市 2 里有两位警探。

7. M 2 4 让城市 2 里的一位警探移动到城市 4。这一步完成之后，Frank 一定会被捉住，除非他根本就没来 Magic Land 。

这个计划总共需要 2 位警探的参与。可以证明：如果自始至终只有 1 名或者更少的警探参与，则 Frank 就会逍遥法外。

你的任务很简单：对于一个输入的 Magic Land 的格局，计算 S，也就是为了追捕 Frank 至少需要投入多少位警探，并且给出相应的追捕计划步骤。

输入数据：

输入文件给出了 Magic Land 的格局。第一行一个整数 N，代表有 N 个城市，城市的编号是 1 N。接下来 $N-1$ 行，每行有两个用空格分开的整数 x_i，y_i，代表城市 x_i，y_i 之间有公路相连。保证 $1 \leqslant x_i, y_i \leqslant N$

输出数据：

向输出文件输出你所给出的追捕计划。

第一行请输出一个整数 S，代表追捕计划需要多少位警探。

第二行请输出一个整数 T，代表追捕计划总共有多少步。

接下来请输出 T 行，依次描述了追捕计划的每一步。每行必须是以下三种形式之一：

1 “L x”，其中 L 是大写字母，接着是一个空格，再接着是整数 x，代表在城市 x 空降一位警探。你必须保证 $1 \leqslant x \leqslant N$。

2 “B x”，其中 B 是大写字母，接着是一个空格，再接着是整数 x，代表召回城市 x 的一位警探。你必须保证 $1 \leqslant x \leqslant N$，且你的计划执行到这一步之前，城市 x 里面确实至少有一位警探。

3 “M x y”，其中 M 是大写字母，接着是一个空格，再接着是整数 x，再跟一个空格，最后一个是整数 y。代表让城市 x 的一位警探沿着公路移动到城市 y。你必须保证 $1 \leqslant x, y \leqslant N$，且你的计划执行到这一步之前，城市 x 里面确实至少有一位警探，且城市 x, y 之前确实有公路。

必须保证输出的 S 确实等于追捕计划中所需要的警探数目。

样例输入：

```
4
1 2
3 2
2 4
```

样例输出：

```
2
7
L 2
L 2
M 2 1
B 1
L 3
M 3 2
M 2 4
```

评分方法：

对于任何一个测试点：

如果输出的追捕计划不合法，或者整个追捕计划的步骤数 T 超过了 20000，或者追捕计划结束之后，不能保证捉住 Frank，则不能得分。

否则，用你输出的 S 和我们已知的标准答案 S^* 相比较：

1. 若 $S < S^*$，则得到 120% 的分。
2. 若 $S = S^*$，则得到 100% 的分。
3. 若 $S^* < S \leqslant S^* + 2$，则得到 60% 的分。
4. 若 $S^* + 2 < S \leqslant S^* + 4$，则得到 40% 的分。
5. 若 $S^* + 4 < S \leqslant S^* + 8$，则得到 20% 的分。
6. 若 $S > S^* + 8$，则得到 10% 的分。

数据规模和约定：

输入保证描述了一棵连通的 N 结点树，$1 \leqslant N \leqslant 1000$。

3.4.2 问题分析：

标准算法是定义了一个叫做“主链”的东西，然后沿着主链走。遇到分支递归做。

这种方法需要很高深的数学知识，了解一些定理。考场上几乎不可能做出来（事实上考场中也没有 100 分）。因此这题贪心便可以大显身手。

考场上很多人采用了明显错误 DP 的方法，取得了不错的效果。

我采用的方法很简单。就是选一个点作为起点，然后做，取最优。做的过程没有找主链，就是随便做下去。但是我考场上犯了个错误，由于我做的过程复杂度是 $O(N)$，我原本可以把每个点都当起点做一遍的。可是我却鬼使神差的只做了 10 次。导致最后只有 57 分。事后我将我的程序稍加修改重新测了一遍就成了 76 分，应该还算是一个令人满意的成绩。 □

参 考 文 献

[YM01] 杨沐。调兵谴将 (surround) 解题报告，2007。

[HWD01] 胡伟栋。生成树计数解题报告，2007。