

BAIDU

百度之星程序设计大赛十周年算法合集

(2005~2014)

百度校园品牌部

2014/7/29

百度之星程序设计大赛十周年算法合集	1
百度之星 2005	6
初赛第一题：连续正整数	6
初赛第二题：重叠区间大小	7
初赛第三题：字符串替换	7
初赛第四题：低频词过滤	8
决赛：八方块移动游戏	9
百度之星 2006	10
初赛第一题：百度语言翻译机	10
初赛第二题：饭团的烦恼	11
初赛第三题：变态的比赛规则	12
初赛第四题：蛴蛴计分	13
初赛第五题：座位调整	14
初赛第六题：剪刀石头布	14
复赛第一题：另类杀人游戏：	15
复赛第二题：空中飞猴：	16
复赛第三题：星球大战：	16
复赛第四题：彩球游戏：	17
复赛第五题：追捕：	18
决赛题目：俄罗斯方块	19
百度之星 2007	21
初赛第一题：水果开会时段	21
初赛第二题：大话西游与数字游戏	22
初赛第三题：繁忙的会议室预定问题	23
初赛第四题：SQL 中的 SELECT 语句	25
初赛第五题：百度时间	26
初赛第六题：实习生小胖的百度网页过滤器.....	27
初赛第七题：Wii 游戏开始啦！	29
初赛第八题：百度的高级搜索方法	30
复赛第一题：简单印象	31
复赛第二题：好心的出租车司机	33
复赛第三题：Robots.txt 协议	33
复赛第四题：紧急修复	35
决赛	38
百度之星 2008	39
初赛一第一题：广告排名区间	39
初赛一第二题：LZW 网页判重	40

初赛一第三题：钉子与木板	41
初赛二第一题：圆内五角星	42
初赛二第二题：传输方案规划	43
初赛二第三题：公平数	44
初赛二第四题：成语纠错	45
复赛第一题黑白树	46
复赛第二题：寻找所有的同构象	47
复赛第三题：验证码识别	50
决赛	52
百度之星 2009	54
初赛第一题：火柴游戏	54
初赛第二题：电子商务平台商品推荐问题	57
初赛第三题：葫芦娃	60
初赛第四题：争车位	63
初赛第五题：Sorry，打错了	66
初赛第六题：树形控件	68
初赛第七题：交点覆盖	72
初赛第八题：我的地盘	74
复赛第一题：高频 Query 的识别	75
复赛第二题：图形检索	78
复赛第三题：网页的相似度计算	80
复赛第四题：拼车	84
决赛第一题：Spider	88
决赛第二题：限制性图灵测试，搜索引擎的非法用户检测	90
百度之星 2010	92
复赛第一场第一题：A+B 问题	92
复赛第一场第二题：i-Doctor	94
复赛第一场第三题：url 规范化	96
复赛第一场第四题：并行修复	99
复赛第一场第五题：猜猜你在哪儿	101
复赛第二场第一题：购物搜索调研	102
复赛第二场第二题：内存碎片	104
复赛第二场第三题：蜗牛	106
复赛第二场第四题：午餐聚会	107
复赛第二场第五题：玉树驰援	110
决赛：植物大战僵尸	112
百度之星 2011	120

初赛第一场第一题 图标排列	120
初赛第一场第二题：篮球场	122
初赛第一场第三题：度度熊大战僵尸	123
初赛第二场第一题 圆环	124
初赛第二场第二题：园艺布置	125
初赛第二场第三题：数据还原	126
复赛第一题 跳跃	127
复赛第二题：股票代码	129
复赛第三题：数据校验	130
决赛第一题 德州扑克游戏	132
百度之星 2012	135
初赛第一场第一题:	135
初赛第一场第二题：小小度刷礼品	138
初赛第一场第三题：集合的交并	139
第一题 度度熊就是要刷排名第一	140
第二题：度度熊的礼物	141
第三题：网页聚类	142
第四题 小王子的表演	143
复赛第一题、蛛网	144
复赛第二题、改变数字	147
复赛第三题、消灭病毒	148
复赛第四题	150
复赛第五题	151
百度之星 2013	152
区域赛第一场第一题	152
区域赛第一场第二题	153
区域赛第二场第一题	154
区域赛第二场第二题	154
区域赛第三场第一题	156
区域赛第三场第二题	157
区域赛第三场第三题	158
月赛第一场第一题	158
月赛第一场第二题	160
月赛第一场第三题	160
月赛第二场第一题	162
月赛第二场第二题	162
月赛第三场第一题	163

月赛第三场第二题	164
决赛	164
百度之星 2014	170
资格赛 1: DiskSchedule.....	170
资格赛 2: Energy Conversion	172
资格赛 3: Labyrinth.....	174
资格赛 4: Xor Sum	175
初赛 1: BestFinancing.....	176
初赛 2: Chess.....	178
初赛 3: CycleCocycle	180
初赛 4: Grids.....	181
初赛 5: Information.....	183
初赛 6: JZP Set	186
初赛 7: party.....	187
初赛 8: Scenic Popularity	188
复赛 1: FindNumbers.....	190
复赛 2: Race	192
复赛 3: The Game of Coins	193
复赛 4: The Patterns	201
复赛 5: Query on the tree	203
决赛	206

百度之星 2005

初赛第一题：连续正整数

1.1. 题目：

题目描述：

一个正整数有可能可以被表示为 $n(n \geq 2)$ 个连续正整数之和，如：

$$15=1+2+3+4+5$$

$$15=4+5+6$$

$$15=7+8$$

请编写程序，根据输入的任何一个正整数，找出符合这种要求的所有连续正整数序列。

输入数据：一个正整数，以命令行参数的形式提供给程序。

输出数据：

在标准输出上打印出符合题目描述的全部正整数序列，每行一个序列，每个序列都从该序列的最小正整数开始、以从小到大的顺序打印。如果结果有多个序列，按各序列的最小正整数的大小从小到大打印各序列。此外，序列不允许重复，序列内的整数用一个空格分隔。如果没有符合要求的序列，输出“NONE”。

例如，对于 15，其输出结果是：

1 2 3 4 5

4 5 6

7 8

对于 16，其输出结果是：

NONE

评分标准：

程序输出结果是否正确。

1.2. 关键思路：

枚举最终得到的长度 $length$ ，考虑当 $length$ 为奇数时，应满足 $N \% length == 0$ ；当 $length$ 为偶数时，应满足 $N \% length != 0$ 且 $N * 2 \% length == 0$ 。于是可以得到所有的方案，并注意左端点应当大于 0。

初赛第二题：重叠区间大小

2.1. 题目：

题目描述：

请编写程序，找出下面“输入数据及格式”中所描述的输入数据文件中最大重叠区间的大小。对一个正整数 n ，如果 n 在数据文件中某行的两个正整数（假设为 A 和 B ）之间，即 $A <= n <= B$ 或 $A >= n >= B$ ，则 n 属于该行；如果 n 同时属于行 i 和 j ，则 i 和 j 有重叠区间；重叠区间的大小是同时属于行 i 和 j 的整数个数。例如，行 $(10\ 20)$ 和 $(12\ 25)$ 的重叠区间为 $[12\ 20]$ ，其大小为 9；行 (2010) 和 $(12\ 18)$ 的重叠区间为 $[10\ 12]$ ，其大小为 3；行 $(20\ 10)$ 和 $(20\ 30)$ 的重叠区间大小为 1。

输入数据：

程序读入已被命名为 `input.txt` 的输入数据文本文件，该文件的行数在 1 到 1,000,000 之间，每行有用一个空格分隔的 2 个正整数，这 2 个正整数的大小次序随机，每个数都在 1 和 $2^{32}-1$ 之间。（为便于调试，您可下载测试 `input.txt` 文件，实际运行时我们会使用不同内容的输入文件。）

输出数据：

在标准输出上打印出输入数据文件中最大重叠区间的大小，如果所有行都没有重叠区间，则输出 0。

评分标准：

程序输出结果必须正确，内存使用必须不超过 256MB，程序的执行时间越快越好。

2.2. 关键思路：

将所有区间按左端点排序，从左到右扫并维护当前最靠右的端点。贪心可知，维护最靠右的区间必然是最优的，因此扫描到一个新的区间后直接和当前的最右端点进行比较并更新即可。

初赛第三题：字符串替换

3.1. 题目：

题目描述：

请编写程序，根据指定的对应关系，把一个文本中的字符串替换成另外的字符串。

输入数据：

程序读入已被命名为 `text.txt` 和 `dict.txt` 的两个输入数据文本文件，`text.txt` 为一个包含大量字符串（含中文）的文本，

以 `whitespace` 为分隔符；`dict.txt` 为表示字符串（ s_1 ）与字符串（ s_2 ）的对应关系的另一个文本（含中文），

大约在 1 万行左右，每行两个字符串（即 s1 和 s2），用一个 /t 或空格分隔。dict.txt 中各行的 s1 没有排序，并有可能有重复，这时以最后出现的那次 s1 所对应的 s2 为准。text.txt 和 dict.txt 中的每个字符串都可能包含除 whitespace 之外的任何字符。text.txt 中的字符串必须和 dict.txt 中的某 s1 完全匹配才能被替换。（为便于调试，您可下载测试 text.txt 和 dict.txt 文件，实际运行时我们会使用不同内容的输入文件。）

输出数据：

在标准输出上打印 text.txt 被 dict.txt 替换后的整个文本。

评分标准：

程序输出结果必须正确，内存使用越少越好，程序的执行时间越快越好。

3.2. 关键思路：

考虑到数据规模，最直接的做法是直接用 map 进行映射即可。如果还需要更高的效率，可以考虑将字符串进行 Hash，以及可以把 map 换成 HashMap。

初赛第四题：低频词过滤

4.1. 题目：

题目描述：

请编写程序，从包含大量单词的文本中删除出现次数最少的单词。如果有多个单词都出现最少的次数，则将这些单词都删除。

输入数据：

程序读入已被命名为 corpus.txt 的一个大数据量的文本文件，该文件包含英文单词和中文单词，词与词之间以一个或多个 whitespace 分隔。（为便于调试，您可下载测试 corpus.txt 文件，实际运行时我们会使用不同内容的输入文件。）

输出数据：

在标准输出上打印删除了 corpus.txt 中出现次数最少的单词之后的文本（词与词保持原来的顺序，仍以空格分隔）。

评分标准：

程序输出结果必须正确，内存使用越少越好，程序的执行时间越快越好。

4.2. 关键思路：

与上一题类似，用 Map 或者 HashMap 维护所有字符串的出现次数（也可以对所有字符串做 Hash），在输出时扫描所有串并过滤出现次数最少的串即可。

决赛：八方块移动游戏

5.1. 题目：

题目描述：

八方块移动游戏要求从一个含 8 个数字（用 1-8 表示）的方块以及一个空格方块（用 0 表示）的 3x3 矩阵的起始状态开始，不断移动该空格方块以使其和 相邻的方块互换，直至达到所定义的目标状态。空格方块在中间位置时有上、下、左、右 4 个方向可移动，在四个角落上有 2 个方向可移动，在其他位置上有 3 个方向可移动。例如，假设一个 3x3 矩阵的初始状态为：

```
8 0 3
2 1 4
7 6 5
```

目标状态为：

```
1 2 3
8 0 4
7 6 5
```

则一个合法的移动路径为：

```
8 0 3      8 1 3      8 1 3      0 1 3      1 0 3      1 2 3
2 1 4 => 2 0 4 => 0 2 4 => 8 2 4 => 8 2 4 => 8 0 4
7 6 5      7 6 5      7 6 5      7 6 5      7 6 5      7 6 5
```

另外，在所有可能的从初始状态到目标状态的移动路径中，步数最少的路径被称为最短路径；在上面的例子中，最短路径为 5。如果不存在从初试状态到目标状态的任何路径，则称该组状态无解。

请设计有效的（细节请见评分规则）算法找到从八方块的某初试状态到某目标状态的所有可能路径中的最短路径，并用 C/C++ 实现。

输入数据：

程序需读入已被命名为 start.txt 的初始状态和已被命名为 goal.txt 的目标状态，这两个文件都由 9 个数字组成（0 表示空格，1-8 表示 8 个数字方块），每行 3 个数字，数字之间用空格隔开。

输出数据：

如果输入数据有解，输出一个表示最短路径的非负的整数；如果输入数据无解，输出-1。

自测用例：

如果输入为：start.txt 和 goal.txt，则产生的输出应为：

5

又例，如果用

```
7 8 4
3 5 6
1 0 2
```

替换 start.txt 中的内容，则产生的输出应为：

21

评分规则：

1) 我们将首先使用和自测用例不同的 10 个 start.txt 以及相同的 goal.txt，每个测试用例的运行时间在一台 Intel Xeon 2.80GHz 4CPU/6G 内存的 Linux 机器上应不超过 10 秒

(内存使用不限制), 否则该用例不得分;

2) 每个选手的总分 (精确到小数点后 6 位) = 10 秒钟内能产生正确结果的测试用例数量 $\times 10 + (1 / \text{产生这些正确结果的测试用例的平均运行毫秒})$;

3) 如果按此评分统计仍不能得出总决赛将决出的一、二、三等奖共计九名获奖者, 我们将先设 $N=2$, 然后重复下述过程直至产生最高的 9 位得分: 用随机生成的另外 10 个有解的 start.txt 再做测试, 并对这 $10 \times N$ 个测试用例用 2) 中公式重新计算总分, $N++$ 。

5.2. 关键思路:

由于总的状态数最多只有 $9! = 362880$, 因此就单组数据而言肯定能够通过 bfs 很快跑出结果。几个可用的优化: 状态可以使用 hash 判重, 不枚举无用状态, 每次 $O(1)$ 生成新状态, $O(1)$ 计算新 Hash, A* 优化决策顺序等等。每一个微小的优化都能带来算法性能上的提升。

百度之星 2006

初赛第一题: 百度语言翻译机

1.1. 题目:

题目描述:

百度的工程师们是非常注重效率的, 在长期的开发与测试过程中, 他们逐渐创造了一套独特的缩略语。他们在平时的交谈、会议, 甚至在各种技术文档中都会大量运用。

为了让新员工可以更快地适应百度的文化, 更好地阅读公司的技术文档, 人力资源部决定开发一套专用的翻译系统, 把相关文档中的缩略语和专有名词翻译成日常语言。

输入要求:

输入数据包含三部分:

1. 第一行包含一个整数 $N (N \leq 10000)$, 表示总共有多少个缩略语的词条;
2. 紧接着有 N 行的输入, 每行包含两个字符串, 以空格隔开。第一个字符串为缩略语 (仅包含大写英文字符, 长度不超过 10 字节), 第二个字符串为日常语言 (不包含空格, 长度不超过 255 字节);
3. 从第 $N+2$ 开始到输入结束为包含缩略语的相关文档 (总长度不超过 1000000 个字节)。

注意事项:

1. 输入数据是中英文混合的, 中文采用 GBK 编码。

GBK: 是又一个汉字编码标准, 全称《汉字内码扩展规范》。采用双字节表示, 总体编码范围

为 8140-FEFE，首字节在 81-FE 之间，尾字节在 40-FE 之间，排除 xx7F。总计 23940 个码位，共收入 21886 个汉字和图形符号，其中汉字（包括部首和构件）21003 个，图形符号 883 个。

2. 为保证答案的唯一性，缩略语的转换采用正向最大匹配（从左到右为正方向）原则。请注意样例中 PMD 的翻译。

1.2. 关键思路：

用 map 记录下所有的关键字，由于缩略语长度不超过 10，则可以对于每一位向后枚举，找到最长可匹配的串，从而得到答案。

初赛第二题：饭团的烦恼

2.1. 题目：

题目描述：

“午餐饭团”是百度内部参与人数最多的民间组织。同一个部门的、同一所大学的、同一年出生的、使用同一种型号电脑的员工们总是以各种理由组织各种长期的、临时的饭团。参加饭团，不仅可以以优惠的价格尝到更加丰富的菜式，还可以在吃饭的时候和同事们增进感情。

但是，随着百度的员工越来越多，各个饭团的管理变得繁杂起来。特别是为了照顾员工们越来越挑剔的胃，饭团的点菜负责人的压力也越来越大。现在，这个任务就交给“百度之星”了，因为，你将要为所有的百度饭团设计一个自动点菜的算法。

饭团点菜的需求如下：

1. 经济是我们要考虑的一个因素，既要充分利用百度员工的午餐补助，又不能铺张浪费。因此，我们希望最后的人均费用越接近 12 元越好。
2. 菜式丰富是我们要考虑的另一个因素。为简单起见，我们将各种菜肴的属性归结为荤菜，素菜，辛辣，清淡，并且每个菜只能点一次。
3. 请谨记，百度饭团在各大餐馆享受 8 折优惠。

输入要求：

1. 输入数据第一行包含三个整数 N, M, K ($0 < M < NK < BR />2$ 。紧接着 N 行，每行的格式如下：
2. 菜名（长度不超过 20 个字符） 价格（原价，整数） 是否荤菜（1 表示是，0 表示否） 是否辛辣（1 表示是，0 表示否）；
3. 第 $N+2$ 行是 $a\ b\ c\ d$ 四个整数，分别表示需要点的荤菜，素菜，辛辣，清淡菜的数目。

输出要求：

对于每组测试数据，输出数据包含 $M+1$ 行，前 M 行每行包含一个菜名（按菜名在原菜单的顺序排序）。第 $M+1$ 行是人均消费，结果保留两位小数。

2.2. 关键思路:

由于 n, m 很小, 可以枚举出所有菜选与不选的情况, 在结果中取最优值。

初赛第三题: 变态的比赛规则

3.1. 题目:

题目描述:

为了促进各部门员工的交流, 百度举办了一场全公司范围内的“拳皇”(百度内部最流行的格斗游戏)友谊赛, 负责组织这场比赛的是百度的超级“拳皇”迷 W.Z. W.Z 不想用传统的淘汰赛或者循环赛的方式, 而是自己制定了一个比赛规则。由于一些员工(比如同部门或者相邻部门员工)平时接触的机会比较多, 为了促进不同部门之间的交流, W.Z 希望员工自由分组。不同组之间的每两个人都会进行一场友谊赛而同一组内的人之间不会打任何比赛。

比如 4 个人, 编号为 1~4, 如果分为两个组并且 1, 2 一个组, 3, 4 一个组, 那么一共需要打四场比赛: 1 vs 3, 1 vs 4, 2 vs 3, 2 vs 4。而如果是 1, 2, 3 一组, 4 单独一组, 那么一共需要打三场比赛: 1 vs 4, 2 vs 4, 3 vs 4。

很快 W.Z 意识到, 这样的比赛规则可能会让比赛的场数非常多。W.Z 想知道如果有 N 个人, 通过上面这种比赛规则, 总比赛场数有可能为 K 场吗? 比如 3 个人, 如果只分到一组则不需要比赛, 如果分到两组则需要 2 场比赛, 如果分为三组则需要 3 场比赛。但是无论怎么分都不可能恰需要 1 场比赛。相信作为编程高手的你一定知道该怎么回答这个问题了吧? 那么现在请你帮助 W.Z 吧。

输入要求:

每行为一组数据, 包含两个数字 N, K ($0 < k \leq 10^5, N \leq 10^5$)。

输出要求:

对输入的 N, K 如果 N 个员工通过一定的分组方式可以使比赛场数恰好为 K , 则输出“YES”, 否则输出“NO” (请全部使用大写字母), 每组数据占一行。例:

3.2. 关键思路:

由题目可知, 若一组的人数为 a , 则本组人要参与的比赛总数为 $a*(N-a)$, 对所有组进行求和就得到了所有比赛场次数, 但考虑到 a vs b 等价于 b vs a , 需要将结果除以 2。将公式合并后得到 $N*N - \sum a_i*a_i = 2*K$, 问题变成是否存在 $\sum a_i*a_i = N*N - 2*K$ 。利用搜索剪枝, 加 $vis[i][j]$ 表示剩余 i 的情况下是否存在和为 j 的情况, 递推求解。

初赛第四题：蝈蝈计分

4.1. 题目：

题目描述：

蝈蝈小朋友刚刚学会了 0~9 这十个数字, 也跟爸爸妈妈来参加百度每周进行的羽毛球活动。但是他还没有球拍高, 于是大人们叫他记录分数。聪明的蝈蝈发现只要记录连续得分的情况就可以了, 比如用 “3 2 4” 可以表示一方在这一局中连得三分后, 输了两分, 接着又连得到四分。可是, 后来大人们发现蝈蝈只会用 0~9 这十个数字, 所以当比赛选手得分超过 9 的时候, 他会用一个 X 来表示 10 完成记分。但问题是, 当记录为 “X 3 5” 的时候, 蝈蝈自己也记不起来是一方连续得到十三分后, 再输五分; 还是先赢十分输三分再赢五分。

因为百度内部就要开始进行羽毛球联赛了, 要先摸清大家的实力才好分组比赛呢~于是, 大人们想知道以前每局的比分是怎样的, 以及谁获得了胜利。要是遇到了根据比赛记录无法确认比赛过程的情况, 也要输出相应的提示哦。

需要进一步说明的是, 比赛是五局三胜的, 每局先获得二十一分的为胜, 但是胜方必须领先对手两分或以上, 否则必须继续比赛直到一方超出对手两分为止, 比分多的一方获胜。任何一方先获胜三局后就获得最终胜利, 比赛也相应的结束。而且蝈蝈保证是完整的无多余信息的记录了比赛。

输入要求：

文件中第一行只有一个整数 M, 表示蝈蝈记录了多少场比赛的分数;

在接下来的 2M 行里, 每场比赛用两行记录, 第一行是一个整数 N ($N \leq 1000$) 表示当前这个记录中有多少个字符, 第二行就是具体的 N 个字符表示记录的分数 (相邻字符用空格隔开)。

输出要求：

对应每一个分数记录, 输出相应的每局分数, 每局分数都使用两个整数表示, 表示两个选手的得分, 中间用 “:” 分隔开; 每组分数记录间使用一个空行分隔开。如果相应的比赛结果无法预测, 以 “UNKNOWN” 一个单词独占一行表示 (请全部使用大写字母)。

4.2. 关键思路：

题目保证给定数据存在合法方案, 由于存在比分合法, 局数比分合法等限制, 可以进行搜索求解, 最终得到答案。

初赛第五题：座位调整

5.1. 题目：

题目描述：

百度办公区里到处摆放着各种各样的零食。百度人力资源部的调研发现，员工如果可以在自己喜欢的美食旁边工作，效率会大大提高。因此，百度决定进行一次员工座位的大调整。

调整的方法如下：

1. 首先将办公区按照各种零食的摆放分成 N 个不同的区域（例如：可乐区，饼干区，牛奶区等等）；
2. 每个员工对不同的零食区域有不同的喜好程度（喜好程度是 $1 \sim 100$ 的整数，喜好程度越大表示该员工越希望被调整到相应的零食区域）；
3. 由于每个零食区域可以容纳的员工数量有限，人力资源部希望找到一个最优的调整方案使得总的喜好程度最大。

输入要求：

文件第一行包含两个整数 $N, M (N \geq 1, M \leq 300)$ 。分别表示 N 个区域和 M 个员工；

第二行是 N 个整数构成的数列 a ，其中 $a[i]$ 表示第 i 个区域可以容纳的员工数 ($1 \leq a[i] \leq M, a[1] + a[2] + \dots + a[N] = M$)；

紧接着是一个 $M \times N$ 的矩阵 P ， $P(i, j)$ 表示第 i 个员工对第 j 个区域的喜好程度。

输出要求：

对于每个测试数据，输出可以达到的最大的喜好程度。

5.2. 关键思路：

题目显然为一个最大费用最大流问题，可以把每个区域看成一个点，每个人也看成一个点。

连边的时候先在 每个区域 和 每个人之间连一条费用为喜好程度，容量限制为 1 的边。

源点 到 每个区域 之间连一条费用为 0 容量限制为区域最大人数的边。

每个人 到 汇点 之间连一条费用为 0 容量限制为 1 的边。

费用流可以反复用 spfa 求出最长的路径并调整流量。

初赛第六题：剪刀石头布

6.1. 题目：

题目描述：

N 个小孩正在和你玩一种剪刀石头布游戏（剪刀赢布，布赢石头，石头赢剪刀）。 N 个小孩中有一个是裁判，其余小孩分成三组（不排除某些组没有任何成员的可能性），但是你不知道谁是裁判，也不知道小孩们的分组情况。然后，小孩们开始玩剪刀石头布游戏，一共玩 M 次，每次任意选择两个小孩进行一轮，你会被告知结果，即两个小孩的胜负情况，然而你不会得

知小孩具体出的是剪刀、石头还是布。已知各组的小孩分别只会出一种手势（因而同一组的两个小孩总会是和局），而裁判则每次都会随便选择出一种手势，因此没有人会知道裁判到底会出什么。请你在 M 次剪刀石头布游戏结束后，猜猜谁是裁判。如果你能猜出谁是裁判，请说明最早在第几次游戏结束后你就能够确定谁是裁判。

输入要求：

输入文件包含多组测试数据，每组测试数据第一行为两个整数 N 和 M ($1 \leq N \leq 500$, $0 \leq M \leq 2000$)，"="、">" 和 "<"，分别表示和局、第一个小孩胜和第二个小孩胜三种情况。

输出要求：

1. 每组测试数据输出一行，若能猜出谁是裁判，则输出裁判的编号，并输出在第几次游戏结束后就能够确定谁是裁判，小孩的编号和游戏次数以一个空格隔开；
2. 如果无法确定谁是裁判，输出-2；如果发现剪刀石头布游戏的胜负情况不合理（即无论谁是裁判都会出现矛盾），则输出-1。

6.2. 关键思路：

二分寻找第一次可以判定裁判出现的位置，把前面的边全部加入图中，再枚举每一个人为裁判，此时其他人的输赢关系已确定，应有唯一解，对于每一个点深搜染色，若遇到冲突说明会有矛盾发生，若有多个人，当他是裁判时可以推出唯一解，则裁判不能确定，最终求得第一次裁判出现的位置。

复赛第一题：另类杀人游戏：

1.1. 题目：

周末的晚上，百度的员工们总喜欢聚集在公司的会议室玩杀人游戏。从 1 警 1 匪到 n 警 n 匪，他们尝试了几乎所有流行的杀人游戏规则。终于有一天，连最热衷杀人游戏，“杀人”不眨眼的 Austin 也开始对无休止的辩论感到厌烦。于是，他决定改变他的一贯作风，他开始变成了一个“杀人不眨眼”的杀手。

如何做到杀人不眨眼呢？Austin 早已构思好他的杀人计划：N 个人（包括 Austin）坐成一圈玩杀人游戏，按顺时针编号 1，2，3，4..... Austin 从 1 号开始顺时针开始数到第 m 号就杀掉第一个人。被杀掉的人要退出游戏。

3. 如果第 m 个人恰好是 Austin 自己，他就杀掉他顺时针方向的下一个人。

4. Austin 从被杀的人的下一个顺时针数 m 个人，把第 m 个杀掉。

5. 重复 2-4，直至杀掉所有人。

Austin 把这个杀人计谋告诉了法官小 k，他便可以闭起眼睛杀人啦。作为一个正直善良的法官，小 k 当然不能让残忍的 Austin 得逞，于是，她偷偷把 Austin 的杀人计划告诉了作为警察的你，聪明的百度之星。现在，你的任务是活到最后，与 Austin 单挑。

输入：

第一个行包含一个整数 T，表示有 T 组测试数据。

对于每组测试数据：三个整数 N，M，T，($3 \leq N \leq 10000$, $1 \leq M$, $T \leq 10000$) 分别表示参与游戏的人数，Austin 每隔 M 个人会杀掉一人，Austin 初始位置的标号。

输出：

每个测数数据输出一个整数。

你需要选择的初始位置的序号，以确保最后剩下的两个人是你与 Austin。

1.2. 关键思路:

一共有 n 个人参加游戏，每一次一个人退出，则总共退出 $n-2$ 次，用 vector 维护一个参与游戏的人的链表，每一次直接寻址删除一个节点，操作都在 $O(1)$ 完成，总复杂度为 $O(n)$ 。

复赛第二题：空中飞猴：

2.1. 题目：

马戏团里新来了一只很特别的小猴子皮皮 —— 不仅长得漂亮，还很聪明。自从它来到马戏团之后，“空中飞猴”成了马戏团里保留节目，慕名观看的人络绎不绝。“空中飞猴”表演开始时，空中架着两根长长的钢丝。皮皮在其中一根上，它的目标是到达另一根钢丝上。皮皮必须在爬行一定距离后纵身一跃，直接跳到另一根钢丝的某个位置。由于皮皮的速度非常快，它的运动轨迹可以近似的看成一条直线段。为了不让自己太危险，皮皮希望自己的跳跃距离尽量短，而为了不让观众等得太不耐烦，它在钢丝上的爬行距离不能超过 d 。在爬行距离不超过 d 的情况下，皮皮的跳跃距离最短是多少？

输入格式：

输入文件包含多组测试数据。每组测试数据包含 16 个实数 $x_1, y_1, z_1, x_2, y_2, z_2, x_3, y_3, z_3, x_4, y_4, z_4, x_p, y_p, z_p, d$ ，表示两根钢丝分别为线段 $(x_1, y_1, z_1)-(x_2, y_2, z_2)$ 和 $(x_3, y_3, z_3)-(x_4, y_4, z_4)$ ，皮皮的坐标为 (x_p, y_p, z_p) ，最大爬行距离为 d 。皮皮保证在第一条钢丝上，保证每条钢丝长度大于零。但两条钢丝有可能相交甚至重叠。

输出格式：

每组测试数据输出一行，仅包含一个非负实数，四舍五入保留三位小数，即最短跳跃距离。

2.2. 关键思路：

本题的关键在于求空间两线段间最短距离，源代码采用三分法，在两条线段上做二维三分求的距离的极小值，并得到线段上距离最近的点对，通过点对坐标判断是否在离 P 点 d 内，是则所求距离为答案，否则离点对最近的边界点为最小距离点。

复赛第三题：星球大战：

3.1. 题目：

公元 4999 年，人类科学高度发达，绝大部分人都已经移居至浩瀚的宇宙，在上千颗可居住星球上留下了人类的印记。然而，此时人类却分裂成了两个联盟：正义联盟和邪恶联盟。两

个联盟之间仇恨难解，时有战争。

现在，正义联盟计划要破坏邪恶联盟的贸易网络，从而影响邪恶联盟的经济状况，为下一次战争作好准备。邪恶联盟由数百颗星球组成，贸易通过星球间的运输航道来完成。一条运输航道是双向的且仅连接两个星球，但两个星球之间可以有多条航道，也可能没有。两个星球之间只要有运输航道直接或间接的相连，它们就可以进行贸易。正义联盟计划破坏邪恶联盟中的一些运输航道，使得邪恶联盟的星球分成两部分，任一部分的星球都不能与另一部分的星球进行贸易。但是为了节省破坏行动所需的开支，正义联盟希望破坏尽量少的运输航道来达成目标。请问正义联盟最少需要破坏多少条运输航道呢？

输入格式：

输入文件包含多组测试数据。每组测试数据第一行为两个整数 N 和 M ($2 \leq N \leq 500$, $0 \leq M \leq N(N-1)/2$)， N 为邪恶联盟中星球的数量。接下来 M 行，每行三个整数 A 、 B 和 C ($0 \leq A, B < N$, $A \neq B$, $C > 0$)，表示星球 A 和星球 B 之间有 C 条运输航道。运输航道的总数量不超过 10^8 。

输出格式：

每组测试数据输出一行，包含一个整数，表示需要破坏的运输航道的数量。

如果输入的贸易网络本来就是不连通的，则输出 0。

3.2. 关键思路：

把两点间的边数看成权重，转化为求带权无向图的最小割，应用 Stoer-Wagner 算法即可。

复赛第四题：彩球游戏：

4.1. 题目：

X 博士是一个研究儿童智力开发方法的科学家，他为幼儿教育领域做出了许多贡献。最近，X 博士正在研究一种适合儿童的游戏，用以辅助发展儿童的观察力、注意力和思维能力。经过连日的构思，X 博士终于设计出了一种游戏：彩球游戏。

彩球游戏是一种单人参与的游戏，游戏首先给出一串由许多不同颜色的小球组成的小球序列，以及一个整数参数 M ($M \geq 2$)。一段连续的具有相同颜色的小球序列称为连续同色序列。小孩，即游戏参与者，每次可以向任意一段连续同色序列插入一个同色小球，使该序列的长度加一。当一段连续同色序列在插入一个同色小球后其长度达到 M 时，该序列就会爆炸消失，然后原序列两边的其余小球会重新连成一串，如果两段相同颜色的连续同色序列在此时连接在一起，它们就会合并形成一段新的连续同色序列。如果新形成的连续同色序列长度达到 M ，这段序列也会爆炸消失，然后重复上述过程，直到没有新的长度达到 M 的连续同色序列出现为止。游戏的目标很简单，就是插入尽量少的小球，使得所有小球都爆炸消失掉。通过长时间的游戏和不断提高游戏水平，这个游戏可以很好地开发儿童的观察力、注意力和思维能力。但是 X 博士仍然面临着一个困难的问题，他还需要设计出一个游戏演示 AI 程序，可以以最优的方式（即插入的小球数量最小）进行游戏，用于游戏教学，或者在游戏中对小孩给出提示。X 博士并不擅长此类程序，因而他无法完成这个任务，你可以帮助他吗？

输入格式：

输入文件包含多组测试数据。每组测试数据第一行为整数 M ($2 \leq M \leq 20$)，第二行为一

条非空的字符串，由大写字母组成且长度不超过 200，表示初始的一串小球，不同的字母表示不同的小球颜色。初始时可能会存在一些长度达到 M 的连续同色序列，但这些序列不会马上爆炸消失。

输出格式：

每组测试数据输出一行，表示至少需要插入多少次小球才能使所有小球爆炸消失掉。

4.2. 关键思路：

我们先把原来的珠子串表示为颜色相同的 n 块，用二元组 (颜色, 个数) 表示。比如说 ABBABBA 就表示为 (A, 1) (B, 2) (A, 1) (B, 2) (A, 1)。

令 $dp[i][j]$ 表示从第 i 块到第 j 块需要多少小球才可以消去，最后的答案就是 $dp[0][n-1]$ ；

$dp[i][i]$ 一开始存储的是第 i 块的小球个数（也即长度），显然当连续长度 $L \geq m$ 的时候 $dp[i][i]=1$ ； $L < m$ 的时候， $dp[i][i]=m-dp[i][i]$ 。

对于状态 $dp[L][R]$ 我们可以进行如下转移：

- 把第 R 段的珠子强行全消掉，转移到 $dp[L][R-1]$ ；
- $dp[R][R] < M$ 时，我们找一个 $L \leq X < R$ ，且 X 段的 color 和 R 段相同，消掉 $[X+1, R-1]$ 之间所有的珠子，转移到 $dp[L][X]$ ；

我们不妨定义 $cost(x, y)$ 表示已经有 x 个同色珠子段，后面又来了 y 个同色珠子，暴力消掉这些珠子需要多少次操作。那么显然有：

$cost(x, y) = 0$ 当 $y > 0$ 且 $x + y \geq M$

$cost(x, y) = 1$ 当 $y = 0$ 且 $x \geq M$

$cost(x, y) = M - x - y$ 当 $x + y < M$

于是最后总的转移方程有两种：

$dp[L][R] = dp[L][R-1] + dp[R][R]$

$dp[L][R] = dp[L][X] + dp[X+1][R-1] + dp[R][R] \quad \text{if } (dp[R][R] < M)$

复赛第五题：追捕：

5.1. 题目：

四个小孩正在花园里玩追捕游戏。一个小孩扮演逃亡者，其余三个小孩做追捕者。花园是一块由 N 行 M 列方格组成的草地，花园周围有木栏包围着，不能走出，花园里面还有一些障碍物不能够通过。游戏可以进行许多回合，每个回合分成两轮，第一轮追捕者可以进行追捕行动，第二轮逃亡者可以根据前一轮追捕者的行动开展逃亡旅程。在第一轮里，三个追捕者必须在三人中选择一个人向某个相邻的方格走一步，只有在三个人都没有可以走的相邻方格时，他们才允许选择停留在原地。在第二轮里，逃亡者也必须选择某个相邻的方格走一步，如果逃亡者没有任何可走的方格，那么逃亡者就被捕了。四个小孩都不允许走到有障碍物或其他人的方格上，也不能走出花园，因而，四个小孩总是会位于不同的方格上面。

这些小孩都是非常聪明的，三个追捕者也是团结一致的。追捕者如果有可以捉到逃亡者的方法，那么他们就一定不会错过。逃亡者如果有不被捕获的方法，那么他也不会犯错。除此之

外，追捕者会希望尽快地捉到逃亡者，而逃亡者即使在会被捕获的情况下也会尽可能地拖延时间。给定花园的障碍物的分布图和四个小孩的初始位置，你知道追捕者有方法捉到逃亡者吗？如果有，他们要经过多少轮后才能捉到逃亡者呢？

输入格式：

输入文件包含多组测试数据。每组测试数据第一行为两个整数 N 和 M ($1 \leq N \leq 10$, $1 \leq M \leq 10$)，为花园方格阵列的行数和列数。接下来 N 行，每行 M 个字符，可以为“.”、“#”、“0”和“X”，分别表示空地、障碍物、追捕者和逃亡者。追捕者总是会有三个，而且四个小孩一开始也都会在空地上面。

输出格式：

每组测试数据输出一行，若追捕者能够捉到逃亡者，则输出他们要经过多少轮后才能成功。轮数的计算包括追捕者和逃亡者进行行动的两轮，逃亡者被捕获的那一轮不算，因而结果总是一个奇数。具体输出格式请参考输出样例。

题目说明：水平所限，此题难以在常规时间内找到合适的解法，抱歉无法提供源代码。

决赛题目：俄罗斯方块

6.1. 题目

俄罗斯游戏中共有七种方块，每种方块都由四个方格组成，如下图所示，七种方块分别编号为 1~7。

游戏中，每次落下一个方块，落到一个宽度为 10 格的槽中。方块的下部一旦碰到槽的底部，或槽中已有的方块，就不能再移动。方块落下不动后，如果有某些行因落下的方块而填满，这些行将被消去。方块下落前，你可以控制方块的左右移动和旋转，以将其放在合适的位置。你对方块的所有移动和旋转操作在下落前（槽外）就计算完毕，然后直接下落到底，下落过程中不能再做操作。如果方块刚刚落下后顶部高度大于 17 行，游戏结束——即使此时有些行可以消除。

交互方式

你的程序应当包含 `tetris_lib.h`，并连接相应的库文件。库中的两个重要函数是：

```
void StartGame(int* t1, int* t2);  
int Step(int r, int l, int* next);
```

你的程序应该首先调用 `StartGame`，其中 $t1$ 和 $t2$ 表示前两个方块的编号（ $t2$ 对应于传统游戏中的“下一个方块”）。接下来，你的程序每次可以使用 `Step` 函数下落一个方块，返回消去的行数。 r 表示旋转方式（ $r=0, 1, 2, 3$ 分别表示顺时针旋转 0 度、90 度、180 度、270 度）， l 表示方块在旋转后的最左边一格的列编号（从左到右依次为 1, 2, ..., 10），而 $next$ 表示方块落下后新的下一个方块编号（0 代表没有下一个方块，下一次 `Step` 调用后库将自动终止你的程序）。你的程序不应自行终止。

关于自测的提示

调用 `StartGame` 函数时，库将从标准输入中读入若干行，每行包括一个整数，表示方块的编号。你可以利用这一点对你的程序进行测试。程序运行结束后，测试库将把结束原因和得分显示在标准输出中。

库中还有两个函数可以用于自测：

```
void SetLog(const char* filename);
```

```
void Snapshot();
```

如果需要测试库记录程序的行为，请在调用 StartGame 之前调用 SetLog 函数。

评分规则

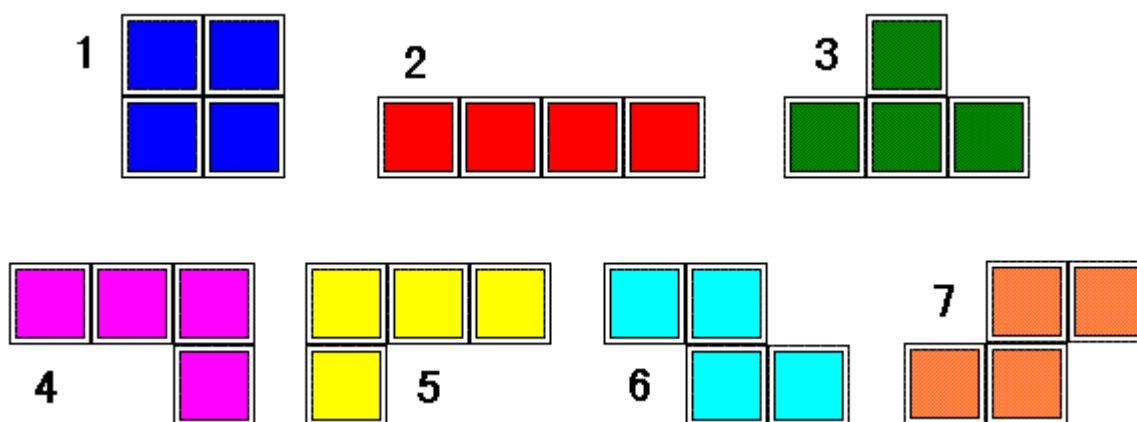
在与测试库交互的过程中，出现以下条件之一，则库将终止程序：

- 1、调用参数非法
- 2、方块刚刚落下后，其顶部高度大于 17 行
- 3、所有方块均已落下
- 4、本数据的运行总时间超过 1 秒钟

程序终止后，假设一次消去 1 行、2 行、3 行、4 行的次数分别为 a, b, c, d，则该数据原始得分为 $2b+6c+10d$ 。换句话说，消去单独的 1 行不得分。

对于每个数据，得分排名前八的程序分别得到 10, 7, 6, 5, 4, 3, 2, 1 分。如果原始得分相同，则消去行数多的排名在前；如果原始得分和消去行数都相同，则下落方块总数多的排名在前。如果三者都相同，则得分相同。消去行数为 0 的程序不得分，即使它排在前八。

最终成绩按照 50 个数据的总分从大到小排序，如果总分相同则按所有数据的原始得分之和排序；如果仍有相同，则按所有数据的消去的总行数排序；如果仍相同，则名次相同。



题目说明： 由于本题已找不到现场决赛时提供的交互接口和其他 API，因此无法提供源代码。

百度之星 2007

初赛第一题：水果开会时段

1.1. 题目：

题目描述

每个百度工程师团队都有一笔还算丰裕的食品经费，足够每天购置多种水果。水果往往下午送达公司前台。前台的姐姐们只要看到同时出现五种或以上的水果，就称之为“水果开会”。从搜索引擎切词的语法角度，只要两种水果的名字中有一个字相同就属于同样的类别。例如“小雪梨”和“大雪梨”是同一种水果，而“核桃”和“水蜜桃”也被认为是同一种水果。尤其要指出的是，如果有三种水果 x , y , z 同时在前台出现，且 x 和 y 是同一种水果， y 和 z 也是同一种水果的时候， x 和 z 在此时也被认为是同一种水果（即使 x 和 z 并不包含相同的字）。现在前台的姐姐们想知道，今天是否有“水果开会”——五种或更多的水果同时在前台出现。

输入格式

输入的第一行只有一个整数 n ，表示购置水果的组数。接下来的 n 行表示水果的到达时间、取走时间（时间用 1200 到 1900 之间的正整数表示，保证取走时间大于到达时间）。剩下的字符串以空格分割每一种水果。如“1400 1600 雪梨 水蜜桃”，表示下午两点到四点（包含两点和四点这两个时间点），雪梨和水蜜桃会在前台等待开会。每种水果名称由不超过十个汉字组成。

输出格式

输出仅一行，包含一个字符串 Yes 或 No，分别表示今天水果开会与否。

输入样例 1 例

3

1200 1400 雪梨 柠檬

1300 1400 西瓜 苹果

1400 1800 花生 水蜜桃

输出样例 1 例

Yes

输入样例 2 例

3

1200 1400 雪梨 柠檬

1400 1500 哦 大梨 呀

1500 1800 咦 大梨

输出样例 2 例

No

样例解释

在样例 1 中，时刻 1400 有六种水果在前台；在样例 2 中，由于雪梨和大梨在任何时刻都是同一种水果，最多只有四种水果同时在前台。

评分规则

程序将运行在一台 Linux 机器上（内存使用不作严格限制），在每一测试用例上运行不能超过 1 秒，否则该用例不得分；

要求程序能按照输入样例的格式读取标准输入数据，按照输出样例的格式将运行结果输出到标准输出上。如果不能正确读入数据和输出数据，该题将不得分；

该题共有 10 个测试数据集，每组数据均满足 $n \leq 10$ ，每个时段最多有 10 个水果，一共不超过 50 个水果；

该题目 20 分。

1.2. 关键思路：

枚举每个时间点，在当前时间点搜索所有的水果，并将同类的水果进行扩展。搜索的次数即为水果的种类数。

初赛第二题：大话西游与数字游戏

2.1. 题目：

题目描述

“叉烧鸡翅膀，我呀最爱吃！……”，百度 spider 组的“黑龙潭之行”在烤着鸡翅，唱着星爷的经典时达到高潮。大家在篝火旁围成一圈，开始玩“数 7”加强版游戏，规则如下：

规则 1：遇 7 的倍数或含 7 的数时 pass。

规则 2：遇有包含相同数字的数时 pass。注意相同数字不必相邻。例如 121。

数错的惩罚很残酷——吞食烤全羊。为避免惩罚，百度工程师们需要你——史上最强程序员的帮助。百度工程师想知道：

req1 x：符合规则 1 的第 x 个数是什么？

req2 y：符合规则 2 的第 y 个数是什么？

req12 z：同时符合规则 1、2 的第 z 个数是什么？

query n：数 n 是规则 1 中的第几个数，是规则 2 中的第几个数？

输入格式

输入的每一行为一个查询，由一个查询词和一个无符号整型数组成。共有四种查询，查询词分别为 req1、req2、req12、query（区分大小写）。

输出格式

前三种查询输出一个无符号整型的解。对于“query n”的查询，若 n 是规则中的数则输出相应的解，否则输出 -1。

输入样例 例

req1 10

req2 10

req12 10

query 14

输出样例 例

11

10

12

-1 13

评分规则

程序将运行在一台 Linux 机器上（内存使用不作严格限制），在每一测试用例上运行不能超过 1 秒，否则该用例不得分；

要求程序能按照输入样例的格式读取标准输入数据，按照输出样例的格式将运行结果输出到标准输出上。如果不能正确读入数据和输出数据，该题将不得分；

该题目共有 10 个测试数据集，其中数据 1~5 主要考查正确性，满足 $x, y, z, n \leq 1000$ ；输入 6~10 主要考查时间效率，满足 $x \leq 10,000,000$ ， $y \leq 1,000,000$ ， $z \leq 240,000$ ， $n \leq 20,000,000$ 。数据 1 和 6 只包含 req1，数据 2 和 7 只包含 req2，数据 3 和 8 只包含 req12，数据 4 和 7 只包含 query，数据 5 和 10 包含全部四种查询。每组数据都恰好包含 100 个查询。

该题目 20 分。

2.2. 关键思路：

数位 DP。分别设 $dp0[len][first][mod]$ ， $dp1[len][first][mask]$ ， $dp2[len][first][mod][mask]$ 表示三种情况下，长度为 len ，当前是否为首位，模 7 的值为 mod ，已用过的数的状态为 $mask$ 的数的个数，枚举当前位进行转移即可。

初赛第三题：繁忙的会议室预定问题

3.1. 题目：

百度由最开始的 7 人团队迅速发展为几千人的大团队，而工程师们经常需要在一起进行“头脑风暴”，这样会议室就成了紧缺资源。为了有效利用资源，大家决定制定规则，自动安排会议室的使用。

为了公平起见，应按照申请时间从早到晚依次考虑，先到先得，且申请一旦被接受就不能取消。在处理每条请求时，只要当前请求可以和前面已接受的所有请求同时满足时就必须被接受（如有必要，可以调整给已接受申请安排的会议室和开会时间）。注意同一时间开的不同会议必须在不同的会议室，而同一个人不能同时参加两个会议。

输入格式

输入第一行为会议室总数 n 和请求总数 m ；第二行是 n 个整数，表示会议室能够容量的人数。以下 m 行每行是一个请求，按请求时间先后顺序排列（即应优先满足在输入中更早出现的请求）。

每个请求中第一个是整数，表示会议需要的时间长度（单位：小时）；之后为与会人名单。

人名由不超过四个汉字组成，用半角逗号分隔（每人名字固定且唯一，有重名的也在登记时区分开）。名单后的数字表示可以安排会议的时间，也以半角逗号分隔，如 10, 11, 14, 15 表示第 10, 11, 14, 15 个小时可以开会（会议时间为 9 到 19 之间的正整数）。

输出格式

输出 m 行，依次表示每个请求是否被接受。1 表示接受，0 表示不接受。

输入样例：

例
2 4
20 2
3 张三, 李四, 王五 10, 11, 12, 14, 15
1 张三 12
4 王六, 王七, 王八, 王九, 王十 9, 10, 11, 12, 13, 14, 15
2 张三 14, 15

输出样例：

例
1
0
0
1

样例解释

请求 1 可以满足，因此接受；在请求 1 接受的前提下请求 2 和请求 3 都无法满足，因此不接受。请求 1 和请求 4 可以同时满足（都在会议室 1，前者用时间 10~12，后者用时间 14~15）。需要特别注意的是：如果没有请求 1，后三个请求可以同时满足。但是规则是先到先得，请求 1 只要可以满足就必须接受。

评分规则

程序将运行在一台 Linux 机器上（内存使用不作严格限制），在每一测试用例上运行不能超过 2 秒，否则该用例不得分；

要求程序能按照输入样例的格式读取标准数据，按照输出样例的格式将运行结果输出到标准输出上。如果不能正确读入数据和输出数据，该题将不得分；

该题共有 15 个测试数据集，均满足 $n \leq 10$, $m \leq 10$ 。每个会议最多有 10 人参加；

该题目 30 分。

3.2. 关键思路：

搜索。主要的剪枝包括：和当前的最优解比较，如果当前位比最优解差则放弃该解；如果当前搜到一个解满足后面所有位都是 1 则结束；将所有人 and 会议室的状态都用位运算进行加速判断，避免冗余的循环；对当前层的所有状态设计估价函数（例如考虑当前时间点被后面的多少人覆盖），并调整搜索顺序，尽可能快地靠近最优解，使得最优化剪枝能得到发挥。由于前面的状态对后面的会议安排有影响，因此可以考虑迭代加深搜索，但这尽管能在一些情况下避免陷入死胡同，但也可能会放慢趋近最优解的速度，因此需要合理使用。

初赛第四题：SQL 中的 SELECT 语句

4.1. 题目

SQL 中的 SELECT 语句用于从数据库中查询记录。某个工程项目数据库中有一个所有数据均为字符串的表，需要查询一些满足条件的记录数。本题考虑 SELECT 语句的简化形式，相关语句格式如下：

1. 计数语句，查询满足条件的记录条数。有两种格式：

格式 1: SELECT COUNT(*) WHERE <条件>

格式 2: SELECT COUNT(*)

2. 子集选择语句，选择满足条件的记录并组成一个集合。有两种格式

格式 1: SELECT * WHERE <条件>

格式 2: SELECT *

上述两种语句中的 FROM 子句具有相同的格式：

格式 1: FROM 格式 2: FROM (子集选择语句)

其中 TABLENAME 为该工程中惟一的表名，子集选择语句即上述用 SELECT * 开头的语句。

条件的格式为一条或多条=用关键字 and 连接（不区分大小写），其中 FIELD 为字段名，VALUE 为数据值，它们均为由大小写字母和数字组成的长度不超过 10 的非空字符串。该条件表示所有特定的字段必须等于给定值。

给定表中的所有记录和若干条计数语句，输出所有语句的结果。

输入格式

输入第一行为三个整数 c, n, q, 分别表示数据库中表的列数、记录数和查询次数；第二行为表名(即 TABLENAME)；第三行为表中的 c 个字段名(FIELD)，之间用一个或多个空格隔开，字段名各不相同；接下来 n 行，每行表示一个记录，有 c 个数据值(VALUE)，之间用空格隔开；接下去有 q 行，每行一条 SELECT 记录数语句，该语句长度（包括空格）不超过 1000。输入数据保证每条语句满足题目中给出的计数语句的定义，并且 FROM 子句的格式 1 中出现的表名和输入的表名一致。

输出格式

输出 q 行，每行一个整数，表示相应语句输出的结果（即满足条件的记录数）。

输入样例 例

4 5 6

Book

BookName Price PublishDate Author

NBAsports 10 2004 dearboy

SQL 20 2002 absorbed

IntrotoAlgorithm 59 2002 Thomas

MultipeView 60 2002 RichardHautley

NBAsports 10 2004 dearboy

SELECTCOUNT(*) FROM Book WHERE BookName=NBA sports and Author=dearboy

SELECTCOUNT(*) FROM Book WHERE Price=20

SELECTCOUNT(*) FROM Book WHERE Author=lala

SELECTCOUNT(*) FROM (SELECT * FROM Book WHERE BookName=NBA sports)

SELECTCOUNT(*) FROM (SELECT * FROM Book WHERE BookName=NBA sports) WHERE Price=20

SELECTCOUNT(*) FROM Book

输出样例 例

```
2
1
0
2
0
5
```

评分规则

程序将运行在一台 Linux 机器上（内存使用不作严格限制），在每一测试用例上运行不能超过 2 秒，否则该用例不得分；

要求程序能按照输入样例的格式读取标准数据，按照输出样例的格式将运行结果输出到标准输出上。如果不能正确读入数据和输出数据，该题将不得分；

该题共有 10 个测试数据集，数据 1 的表与样例相同，并包含 15 条 SELECT 语句。数据 2, 3, 4, 5 的表分别有 1, 2, 5, 7 列，数据 6~10 的表均有 8 列。数据 2~5 的表均有恰好 1000 条记录，并包含 100 个 SELECT 语句。数据 6~10 的表不超过 3000 条记录，并包含不超过 20000 条 SELECT 语句。本题的后 5 组数据着重考查程序的时间效率；
该题目 30 分。

4.2. 关键思路：

由于查询都是 and 的关系，因此只需要从原语句中剖析出所有的等号并对所有的查询结果取交即可。考虑到数据范围，查询时的优化是至关重要的。一些可供参考的优化包括：对每列数据进行索引，查找时优先查询更高质量的索引（参考代码中统计了该列中不同的条目数和最大条目数进行加权计算），保存出现过的查询等等。实现的细节中，尽可能地对字符串和其他数据进行 Hash 以加速运算。

初赛第五题：百度时间

5.1. 题目

Baidu 的服务器上使用的不是北京时间，而是 Baidu 时间。Baidu 时间的时分秒与北京时间相同，但是日期与北京时间不同，是用一个正整数表示从 2000 年 1 月 1 日开始经过了几天。现在就请大家设计一个程序将北京时间转换为百度时间。在本题中，闰年的年份是 400 的倍数，或者是 4 的倍数但不是 100 的倍数。比如 2000 和 8888 均为闰年，但 6100 不是。

输入格式

输入数据的每一行为一个待转化的北京时间（不含空格和 TAB），正确的格式包括两种：

一种为：YYYY-MM-DD，（YYYY 表示四位数年份，MM 为两位月份，DD 为两位日期）；

另一种为：MM/DD/YYYY，（YYYY 表示四位数年份，MM 为两位月份，DD 为两位日期）；

输出格式

每个数据输出一行。如果可以成功转换，输出一个正整数，否则输出 Error。

输入样例 例

2000-01-01
AStar2007
05/26/2007
输出样例 例
0

Error

2702

评分规则

程序将运行在一台 Linux 机器上（内存使用不作严格限制），在每一测试用例上运行不能超过 1 秒，否则该用例不得分；

要求程序能按照输入样例的格式读取标准输入数据，按照输出样例的格式将运行结果输出到标准输出上。如果不能正确读入数据和输出数据，该题将不得分；

该题共有 5 个测试数据集，数据 1 和数据 2 中的所有日期均能成功转换。所有数据中，每行不超过 20 个字符，每组数据最多包含 100 行；

该题目 20 分。

5.2. 关键思路：

使用有限状态自动机来判断输入是否合法，将两种情况分别加入到自动机内即可，除此之外还需判断年月日是否合法。在输入合法的前提下，再分别统计从 2000 到相应年份中每一年的天数，当前年经过的月份的天数，以及当前月份经过的天数。全部天数相加可得到答案。

初赛第六题：实习生小胖的百度网页过滤器

6.1. 题目：

百度网页采集器(Baiduspider)每天从互联网收录数亿网页，互联网的网页质量参差不齐。百度的工程师们每天都在改进方法来判断一个网页质量的好坏，使质量差的网页出现在检索结果中较后的位置。现在实习生小胖想到一个很简单的方法来判断一个网页内容的好坏，方法如下：

1. 利用数据挖掘技术在互联网语料库中挖掘出一批有特点的词汇，分为好词和坏词两种，好词标上正的权重，坏词标上负的权重；
2. 通过好词和坏词词典对每个网页计算网页总权重：从第一个字开始匹配，找到一个好词则加上相应的权重，找到一个坏词则减去相应的权重，下一次匹配将从找到的词末尾的下一个位置开始。
3. 坏词采用正向最短匹配：从当前匹配位置开始的若干连续汉字，如果形成多个坏词，则只计算最短的那个坏词的权重，下一次匹配将从这个最短坏词末尾的下一个位置开始。
4. 好词采取正向最长匹配：从当前匹配位置开始的若干连续汉字，如果形成多个“有效”好词，则只计算最长“有效”好词的权重，下一次匹配从这个最长“有效”好词末尾的下一个位置开始。
5. “无效”好词的定义：好词的一部分本身是一个坏词；或者好词的一部分与后续相邻的若干字组成一个坏词。

现在小胖已经做好了第 1 步的工作，有一个好词和坏词的列表（词典），但是由于没有对中

文文本处理的程序经验，他想请未来的百度之星们帮他完成这个程序。

输入格式

输入第一行为一个字符串（网页正文）。从第二行开始为词典，格式为“词 空格 词的权重”。权重为一个带符号 32 位整数。如果权重为正，则为好词，反之则为坏词；不存在重复的词，不存在权重为 0 的词。

作为“网页”的字符串中同时包含中文和 ASCII 字符，每个汉字占 2 个字节。并非“网页”中的所有字都在词典中。

输出格式

输出仅一行，为网页总权重（答案保证不超过带符号 32 位整数的范围）。

样例输入 例

小胖之喷火龙骑士!!

小胖 6

喷火 -1

喷火龙 -1

火龙 -1

龙 4

龙骑 3

龙骑士 2

骑士 -2

士 3

样例输出 例

7

样例解释

从“网页”中找到的好词为“小胖”和“龙”，坏词为“喷火”和“骑士”。特别要说明一下“龙”被识别为好词的原因——“喷火”和“喷火龙”均为坏词，按正向最短匹配得到“喷火”，接着往下匹配到好词“龙”、“龙骑”和“龙骑士”，但是由于“骑士”是坏词，所以“龙骑”、“龙骑士”无效而“龙”是最长的有效好词。注意题目描述中的匹配规则，好词的“有效”和“无效”只考虑该好词的一部分与后续字是否能够组成坏词，而不考虑和前面的字是否能够组成坏词——样例中的“龙”虽然可以与前面的字组成坏词“喷火龙”和“火龙”，但由于这两个词都是未能匹配成功的坏词，因此对好词“龙”的词性没有影响，可以累积“龙”的权重。

注意事项

输入数据的中文采用 GBK 编码。GBK：是又一个汉字编码标准，全称《汉字内码扩展规范》。采用双字节表示，总体编码范围为 8140-FEFE，首字节在 81-FE 之间，尾字节在 40-FE 之间，排除 xx7F。总计 23940 个码位，共收入 21886 个汉字和图形符号，其中汉字（包括部首和构件）21003 个，图形符号 883 个。评分规则 程序将运行在一台 Linux 机器上（内存使用不作严格限制），在每一测试用例上运行不能超过 1 秒，否则该用例不得分； 要求程序能按照输入样例的格式读取标准输入数据，按照输出样例的格式将运行结果输出到标准输出上。如果不能正确读入数据和输出数据，该题将不得分； 该题共有 10 个测试数据集，前 7 组数据的大小不超过 1K 字节，数据 8 和数据 9 不超过 600K 字节，数据 10 的网页正文不超过 1M 字节。所有数据的词典不超过 50,000 项，且词典中的词保证由 1 到 5 个汉字组成。词典不包含重复的单词； 该题目 20 分。

6.2. 关键思路:

将文本中所有坏词找出，再在由坏词分割的文本中进行好词匹配。匹配时由于字典中单词的大小不超过 5，因此可以对文本中长度不超过 5 的所有子串和字典进行 Hash 即可。GBK 读入时可以将字符以 int 的形式储存。

初赛第七题：Wii 游戏开始啦！

7.1. 题目:

为了在紧张的上班时间内让员工们轻松些，百度休息室里放置着按摩椅、CD、高尔夫套装和 Wii 游戏机等休闲用品。其中最受欢迎的当然是游戏机。

Wii 游戏机有两个手柄，每个手柄使用两节电池（这两个电池可以是不同的品牌），其中至少一块电池没电时该手柄没电。

工程师们在玩游戏时，总是用最简单的方式更换电池：有手柄没电时把所有没电的电池拿走，一一换上新电池即可（有电的电池总是继续使用）。当有手柄没电且没有新电池可用时才停止玩 Wii。

告诉你每个品牌电池的使用时间以及该品牌电池的个数，请计算工程师们玩游戏时间的最小值和最大值。

输入格式

输入第一行为一个正整数 n ，表示电池的种数。接下来 n 行，每行两个整数 L 和 F ，表示使用时间为 L 的电池有 F 个（电池不必成对出现，即 F 可以是奇数）。

输出格式

输出仅一行，包含两个整数，分别表示工程师们的最短游戏时间和最长游戏时间（短的时间在前）。两个整数之间以空格隔开。

输入样例 例

```
3
3 2
5 2
8 2
```

输出样例 例

```
5 8
```

样例解释

有三对电池，使用时间分别为 3 小时、5 小时和 8 小时。

方案 1：一开始给手柄 1 使用一对 3 小时的电池，给手柄 2 使用一对 5 小时的电池，则 3 小时后手柄 1 没电，换上一对 8 小时的。再过 2 小时后，手柄 2 没电。此时已经没有电池可用了。总时间为 5 小时。

方案 2：一开始给手柄 1 使用一对 8 小时的电池，给手柄 2 使用一对 5 小时的电池，则 5 小时后手柄 2 没电，换上一对 3 小时的。再过 3 小时后，手柄 1 和手柄 2 同时没电。此时已经没有电池可用了。总时间为 8 小时。

评分规则

程序将运行在一台 Linux 机器上（内存使用不作严格限制），在每一测试用例上运行不能超过 2 秒，否则该用例不得分；

要求程序能按照输入样例的格式读取标准输入数据，按照输出样例的格式将运行结果输出到标准输出上。如果不能正确读入数据和输出数据，该题将不得分；
该题共有 15 个测试数据集，均满足 $n \leq 10$, $L \leq 200$, F 的总和不超过 30；
该题目 30 分。

7.2. 关键思路:

在分析问题和数据范围之后可以发现，本题应当是一个用搜索进行解决的题目，尽管直观上有很多看似正确的贪心策略。搜索时可以考虑以四堆电池来划分阶段，并保证从左到右每堆大小递增，枚举电池时也采用递增策略进行枚举。一些有效的剪枝包括当前的最优化剪枝，答案不超过总和的 $1/4$ ，当前剩余的电池必须能填充完剩余的堆，等等。

初赛第八题：百度的高级搜索方法

8.1. 题目:

你尝试过在百度上使用 `site inurl` 语法查询吗？如果还没有的话可以试一下:)

如输入 `site:www.baidu.com inurl:news`

则会搜出所有在 `www.baidu.com` 站点上的包含“news”子串的 url。

现在有一个 `inurl` 查询列表和一个 `url` 列表，你能找出所有至少被查询过一次的 url 吗？

输入格式

输入第一行是一个整数 n ，表示一共有 n 个查询。以下 n 行每行一个查询。查询的 `site` 部分和 `inurl` 部分中间恰好用一个空格分割，且每行不包含其他空格。下一行是一个整数 m ，表示 `url` 列表中一共有 m 个 url。以下 m 行每行一个 url。

输出格式

每个 url 输出一行。如果至少符合一条查询，输出 1，否则输出 0。

输入样例 例

```
3
site:www.baidu.com inurl:/more
site:zhidao.baidu.com inurl:/browse/
site:www.sina.com.cn inurl:www20041223am
7
http://www.baidu.com/more/
http://www.baidu.com/guding/more.html
http://www.baidu.com/events/20060105/photomore.html
http://hi.baidu.com/browse/
http://hi.baidu.com/baidu/
http://www.sina.com.cn/head/www20021123am.shtml
http://www.sina.com.cn/head/www20041223am.shtml
```

输出样例 例

```
1
1
0
```


0
0
0
1

评分规则

程序将运行在一台 Linux 机器上（内存使用不作严格限制），在每一测试用例上运行不能超过 2 秒，否则该用例不得分；

要求程序能按照输入样例的格式读取标准输入数据，按照输出样例的格式将运行结果输出到标准输出上。如果不能正确读入数据和输出数据，该题将不得分；

该题共有 6 个测试数据集，数据 1, 2, 3, 4, 5, 6 的大小分别约为 4K, 750K, 1.5M, 6.5M, 12M, 18M。所有查询和 url 均合法，url 均以 http:// 开头。url 和查询中可能包含中文。输入文件的每行不超过 256 个字节；

该题目 30 分。

8.2. 关键思路：

对每一个 site 维护一个 AC 自动机，将他所拥有的所有 inurl 字段都插入到自动机中。对于每一个给定的 url，先分析它的 site 并找到对应的自动机，然后在自动机内进行整个 url 的匹配，如果匹配到某个 inurl 字段的终点则答案为 1，否则为 0。当然在细节上还需注意是否存在对应的自动机，以及实现时字符集大小的问题等等，可以使用 HashMap 进行速度上的优化。

复赛第一题：简单印象

1.1. 题目：

题目描述

简单、可依赖是百度的性格，百度之星们又有怎样的性格呢？

有 n 名选手入围了百度之星程序设计大赛的复赛阶段。为了让选手相互之间有更多的了解和更好的交流，组委会工作人员邀请每位选手选择一些不同的词语来介绍自己的性格。每名选手需要为自己选定的每一个词语指定一个绝对值不超过 100 的整数权值 q ，表示这个词语在多大程度上描述了自己的性格。例如“美丽 90”表示该选手认为自己非常美丽，而“冷淡 -60”表示比较热情，而“外向 -5”表示稍微有一点点偏内向。你不需要考虑词语本身的语义，比如“外向 -5”不代表“内向 5”。

组委会发现不少选手的性格都有相似之处，所以希望找到一组词语尽量准确的描述所有的选手。对于选出的词语集合 S ，定义选手 i 被描述的精确程度 $P(i)$ 为 S 中所有词语在选手 i 眼中的权值之和（选手 i 没有提到的词权值视为 0），组委会希望让最小的 $P(i)$ 尽量大，因为这样才能让选出的词语能够描述所有选手，而不仅仅是部分。

另外，所选词语的个数也是有讲究的。词语太少会略显单薄，而词语太多又不方便宣传，所以组委会设定了词语个数的最小值 \min 和最大值 \max 。

输入数据

输入数据第一行为三个整数 n , \min 和 \max ，表示选手的个数、词语的最小值和最大值，

接下来的 n 行，每行包括若干“词语 - 权值”对。词语保证由不超过 10 个汉字组成（用 GBK 编码，不含非汉字），权值为绝对值不超过 100 的整数。

输出数据

仅一行，输出各个词语，用空格隔开。次数总数必须不小于 \min 且不大于 \max ，每个词语都必须至少被一个选手提到过，否则视为非法输出。

输入样例

```
3 2 4
美丽 90 冷淡 -60 外向 -5 乐观 20
美丽 10 冷淡 20
外向 20 乐观 -5
```

输出样例

```
美丽 冷淡 外向
```

样例解释

如果把选手按照输入中出现的顺序编号为 $1 \sim 3$ ，则 $P(1)=90-60-5=25$ ， $P(2)=10+20=30$ ， $P(3)=20$ ，所有 P 中的最小值为 20。

评分方法

本题包含 30 个测试点，每个测试点 10 分，共 300 分。

测试点 $1 \sim 10$ 满足 $n \leq 100$ ， $1 \leq \min \leq \max \leq 5$ ，涉及到的词语不超过 500 个。

测试点 $11 \sim 20$ 满足 $n \leq 200$ ， $10 \leq \min \leq \max \leq 15$ ，涉及到的词语不超过 2000 个。

测试点 $21 \sim 30$ 满足 $n \leq 400$ ， $25 \leq \min \leq \max \leq 30$ ，涉及到的词语不超过 10000 个。

每个测试点独立评分。对于每个测试点，非法输出的得分为 0，合法输出的得分大于 0。

设合法输出的程序数为 M ，比程序 i 的方案严格更优的程序数为 $Y(i)$ ，则该测试点程序 i 的分值为 $10(1-Y(i)/M)$ 。换句话说，输出该测试点最优解的程序将获得 10 分，而最差解唯一的情况，输出最差解（但合法）的选手将得到 $10/M$ 分。注意：每个测试点的得分不必为整数。

1.2. 关键思路：

考虑用估价函数对每个单词计算出相应的价值，然后通过贪心得到较优解之后进行随机调整。估价函数可以有多种设计方法，例如可以考虑单词对所有人的贡献之和，以及它对人的最坏贡献，并统计出有最坏贡献的人数，通过加权的方式进行结合。随机调整时包括对集合大小的随机以及集合元素的随机，可以考虑对加权排序后的单词进行非均匀的随机选取，例如可以令每次选取的单词 id 为 $s^k * M$ ，其中 s 为 $[0, 1]$ 间的随机数， M 是单词数量， k 是常数，这样在随机时能趋近于估价函数更高的单词。在所有单词中随机出若干个单词之后进行暴力扫描，得到其中最优的单词并进行插入。删除时同理，直接扫描当前维护的单词并找到贡献最坏的单词将其删除即可。参考代码中还加入了卡时来避免超时的情况，通过这些手段就已经可以在时限内取得非常不错的解了。

复赛第二题：好心的出租车司机

2.1. 题目：

题目描述

北京的一位出租车司机向你抱怨：城市发展太快，公路越来越多，他已经疲于计算行驶路线，于是求助你开发一个自动导航的工具。

出租车只能在公路上行驶。所有的公路都是笔直、双向的，相交的公路视为连通（可以在交叉点处从一条公路开到另一公路上）。由于道路施工，整个城市的公路系统可能并不完全通畅。如果乘客的目的地不在公路边，则乘客下车后要步行前往，步行路线不受公路限制。这位好心的司机还特别提出，乘客步行距离越短越好；其次，出租车行驶里程越短越好。方便起见，用笛卡尔坐标系来描述城市地图，所有坐标都在第一象限 $[0, 1000]$ 的范围内。公路宽度忽略不计。

输入格式

第一行是一个数字 k ，代表公路条数。以下 k 行每行用 4 个实数描述一条公路（用空格隔开），前两个表示公路起点，后两个表示公路终点。下一行包含 4 个实数（用空格隔开），前两个表示乘客上车点，后两个表示乘客目的地坐标。不相交公路间的最短距离至少为 10^{-4} 。

输出格式

仅一行，为出租车行驶的里程数，保留一位小数。

输出格式

```
2
2.0 2.0 10.0 10.0
10.0 2.0 2.0 10.0
3.0 3.0 9.0 4.0
```

输出样例

7.8

评分方法

本题有 20 组数据，满足 $k \leq 100$ ，公路的交点数不超过 10000。

2.2. 关键思路：

将所有相邻的事件点连边，事件点包括起点，线段端点，线段交点和终点对最近线段的垂足。随后从起点开始做一次最短路即可。最后取答案时需要取步行距离最短的垂足。

复赛第三题：Robots.txt 协议

3.1. 题目：

题目描述

搜索引擎是靠 Web Robot（又称 Spider）来收集互联网上浩如烟海的网页的。Spider

就像一个旅行家一般，不知疲倦地奔波于万维网的空间，将遇到的页面收集下来供搜索引擎索引。对于一个网站的管理员来说，如果希望搜索引擎只收录自己指定的内容，或者指定某些不希望搜索引擎访问的内容，该如何去做呢？他需要的就是 Robots Exclusion Protocol 协议，这里简单的称它做 Robots.txt 协议。

Robots.txt 是一个放置在网站根目录下的纯文本文件。举例来说，当 Spider 访问一个网站

（比如 `http://www.example.com`）时，首先会检查该网站中是否存在 `http://www.example.com/robots.txt` 这个文件，如果

Spider 找到这个文件，它就会根据这个文件的内容，来确定它访问权限的范围。

`www.robotstxt.org` 是 robots.txt 协议的 Home Page，在这个站点上你可以找到很多 robots.txt 协议相关的资料。Robots.txt 协议在

`http://www.robotstxt.org/wc/norobots.html` 上有比较详尽的描述。

你的任务就是编写 Spider 中的一个逻辑单元，这个单元的作用就是来判断一个网站的一些 URL 是否被禁止抓取。对方网站的站点在这里假设是 `www.example.com`，这个 Spider 的 User-agent 当然是 Baiduspider。注意，这个逻辑单元除了本身的判断逻辑要求与 robots.txt 协议一致外，还要注意容错的问题。互联网上纷繁芜杂，会出现很多意想不到的错误。

如何能够对一个对错参半的 robots.txt 进行解析，把其中正确的挑拣出来、把错误的部分忽略掉，也是一个不小的挑战哦。都会遇到什么错误？在开始爬行互联网之前，谁都不知道。

输入格式

第一行是一个正整数 m ，表示 robots.txt 文件的行数，后面跟 m 行，是 robots.txt 的全文。下一行包含一个正整数 n ，表示 URL 的行数，后面跟 n 行 URL，这个就是你要判断的 URL 的列表。

输出格式

每条 URL 输出一行，每行两列，第一列是一个数字，如果这条 URL 被禁止，则输出 0，否则输出 1。第二列是这条 URL 本身。

输入样例

```
2
User-agent: *
Disallow: /tmp/
2
http://www.example.com/index.html
http://www.example.com/tmp/somepage.html
```

输出样例

```
1 http://www.example.com/index.html
0 http://www.example.com/tmp/somepage.html
```

评分方法

本题包含 20 组数据，均满足 $0 \leq n, m \leq 100$ 。

3.2. 关键思路：

由于本题已无法找到当年比赛时提供的 robot.txt 版本（版本更新不一致），因此没能提供参考代码。但关键思路仍然是通过自动机对所有的格式串进行匹配，并注意特判字符*的匹配情况。

复赛第四题：紧急修复

4.1. 题目

背景

2050 年的一天，某市 k 家公司的计算机系统突然同时瘫痪。市政府很快找到了问题的所在，并开始研发一套自动修复整个城市的系统。自动修复系统启用后整个城市的计算机系统将恢复正常，但由于研发时间较长，在此之前各公司仍将蒙受不小的损失。为此，市政府决定派出 n 支维修能力相同的维修队到各公司进行手工修复，力图在自动修复系统启用前整个城市的总损失达到最小。

参数

城市被划分成 $R \times C$ 的网格，各行从上到下编号为 $1 \sim R$ ，各列从左到右编号为 $1 \sim C$ 。每个格子为空地、障碍物和建筑物之一（公司总是位于某个建筑物格子中）。维修队每次只能往上（行编号减 1）、下（行编号加 1）、左（列编号减 1）、右（列编号加 1）四个方向移动，第 i 个维修队每小时可以移动 $s(i)$ 格。维修队在任何时候都不能位于障碍物中，但建筑物可以从它上下左右的相邻空地进入，也可以从这些格子出去。注意：不能直接从一个建筑物格移动到另一个建筑物格，而必须先移动到相邻的空地，在空地上移动，然后再进入另一个建筑物。每个格子的实际尺寸很大，因此可以有多个维修队同时在同一个格子中。第 i 家公司的位置为 $(r(i), c(i))$ ，瘫痪程度为 $B(i)$ ，每小时的经济损失为 $P(i)$ 元。瘫痪程度的单位是“队·小时”，即一支维修队每小时可以把一个公司的瘫痪程度降低 1，而如果 m 支维修队同时为一个公司修复，每小时可以把该公司的瘫痪程度降低 m 。注意，在完全修复之前，每个小时的经济损失不变。

修复计划

政府的修复计划应包含每个小时各维修队所执行的命令。这些命令包括：

1. REST

该命令不进行任何操作。

2. MOVE <seq>

按照 seq 进行移动。其中 seq 为一个长度不超过 s 的非空字符串（如果不需要移动，请使用 REST 命令）。如果 seq 的长度超过 s ，则从第 $s+1$ 个字符开始的剩余部分将被删除；如果该命令不能完全成功的执行，则从第一个非法移动开始的所有后续移动将被忽略。

3. REPAIR

对当前公司进行维修。如果当前公司已经修复或者当前位置不是公司，该命令将被忽略。该命令后面加的所有参数将被忽略。

需要注意的是，每个小时只能执行一条指令，例如不能在 MOVE 之后立刻 REPAIR，必须等待下一个小时。

输入格式

输入的第一行为三个正整数 R, C, T 即网格的行数、列数和自动修复系统的启用时间。以下 R 行每行 C 个字符，表示该城市的地图。点表示空地，# 表示障碍物，0 表示建筑物。下一行包含一个正整数 k ，表示公司的数目。以下 k 行每行四个整数 r, c, B, P ，其中 (r, c) 表示该公司坐标（ $1 \leq r \leq R, 1 \leq c \leq C$ ）， B 为该公司的初始瘫痪程度， P 为每小时的经济损失。 (r, c) 处保证为一个建筑物格。 B 和 P 保证大于 0。

下一行包含两个正整数 n, s ，表示维修队的个数和每小时最多移动的格子数。以下 n 行每行包含两个整数 r, c ，表示该维修队的初始位置。维修队按照输入文件中的顺序编号为

$1 \sim n$ 。

输出格式

输出每 n 行为一组，描述一个小时各维修队的发出的命令。共 T 组，一共 nT 行。所有格式非法的指令将被忽略（等效于 REST）。尽管如此，如果程序输出的命令中没有 REPAIR，或者所有的 REPAIR 都没有成功执行，则输出将视为非法。

输入样例

```
4 7 5
...#0##
#. ....#
0...##0
#. ....
2
1 5 3 5
3 7 4 6
3
4 7 5
1 1 5
3 1 5
```

输出样例

```
MOVE U
MOVE RRRD
MOVE RDRURD
REPAIR
MOVE DRRU
MOVE DRRRU
REPAIR
REPAIR
REPAIR
REPAIR
REPAIR
MOVE D
REST
REST
REPAIR
```

模拟器

每小时的模拟过程如下：

第一步：对于每个尚未修复的公司 i ，将 $P(i)$ 累加进总损失。

第二步：按照序列从小到大的顺序依次执行各维修队的指令。

为了更清晰的说明规则并帮助选手设计和测试程序，命题组开发了一个简单的模拟器。该模拟器根据输入数据和程序输出进行模拟，给出详细模拟过程，以及最后的结果。最终的测试将严格按照该脚本的输出进行评判。

模拟器用 python 编写，没有安装 python 的选手可以在 <http://www.python.org> 下载最新版本。

评分方法

本题包含 30 个测试点，每个测试点 10 分，共 300 分。

测试点 1~10 满足 $R, C \leq 10, n \leq 5, k \leq 10, B \leq 15, T \leq 30$

测试点 11~20 满足 $R, C \leq 30, n \leq 10, k \leq 20, B \leq 50, T \leq 500$

测试点 21~30 满足 $R, C \leq 100, n, k \leq 100, B \leq 1000, T \leq 10000$

每个测试点独立评分。对于每个测试点，非法输出的得分为 0，合法输出的得分大于 0。

设合法输出的程序数为 M ，比程序 i 的方案严格更优的程序数为 $Y(i)$ ，则该测试点程序 i 的分值为 $10(1-Y(i)/M)$ 。换句话说，输出该测试点最优解的程序将获得 10 分，而最差解唯一的情况，输出最差解（但合法）的选手将得到 $10/M$ 分。注意：每个测试点的得分不必为整数。

4.2. 关键思路：

本题是一个开放性题目，但由于本题没有确定的最优解法以及已经找不到比赛时提供的模拟器，因此没能提供源代码。

题目整体的思路如下：在进行模拟时可以考虑对每个状态进行估价，并将估价的结果采纳入决策当中。一个状态的估价涉及到的元素有：当前机器人和待修理的建筑物的距离，建筑物的损失速度和修理程度等，在统计各元素之后即可得到每个机器人在每个状态下最适合修理的建筑物，并直接模拟即可。注意如果一旦让一个机器人停下的话，之后就可以直接将这个机器人删去（由贪心易证），因此可以进一步缩小规模。由于算法复杂度不大，可以采取非均匀随机策略（如令随机到的 $id = r^2 * N$ ），其中 r 为 $[0, 1]$ 的随机数， N 为决策数，这样可以随机到估价更好的决策。

决赛

5.1. 题目:

比赛目标:

完成基于 CounterStrike1.5 的 podbot AI, 在 fy_iceworld_plus, de_dust2, de_inferno 三张地图上与对手 AI 展开较量。Podbot 的基本代码框架由平台给出, 请选手自行参考研究。有待完成的代码位于 bot_T.cpp bot_T.h bot_CT.cpp bot_CT.h 中, 各个函数功能与接口请选手参考代码中的注释说明。

比赛方式:

1. 提交代码: 请将 bot_T.cpp bot_T.h bot_CT.cpp bot_CT.h 压缩成一个 rar 压缩文件, 通过比赛网站提交:

第一二组使用 <http://192.168.0.3/>

第三四组使用 <http://192.168.0.4/>

第五六组使用 <http://192.168.0.5/>

2. 赛制:

第一阶段: 从比赛开始至之后 4 小时整。提交第一阶段 bot 代码用于 iceworld 测试, 测试结果将决定第二阶段分组。

第二阶段: 使用 dust2 和 inferno 两张地图, 赛程是由小组循环决出前八。前八由淘汰赛决定胜负, 判定最终名次。

操作说明编译

进入目录 C:/astar2007/MSYS

双击运行 msys.bat

在 msys 界面中输入 `cd /c/astar2007/PODBOT`

执行命令 `make`, 如果 `make` 成功, 则在 .obj.win32 子目录中有 podbot_mm.dll 生成

调试 1. 在代码中使用 UTIL_ServerPrint 可以向 console 打印信息在游戏中使用

2. 可以记录 log 文件以供分析检查之用

运行 1. 将 podbot_mm.dll 拷贝到 C:Program Files 反恐精英中文站 CS1.5 中文硬盘版 CS1.5 中文硬盘版 cstrikeaddonspodbot 覆盖原有文件即安装成功。

2. 双击桌面上的 cstrike 快捷方式, 启动游戏, 单击进入游戏, 建立一个 11 人局域网游戏。进入游戏后按 6 进入观察模式

3. 控制 bot

(1) 按 = 7 杀掉所有自动 bot

(2) 按 = 5 添加 bot, 选择 godlike 且 aggressive 的 bot

(3) 观察游戏结果 按 ~ 可以观察 Server 调试信息输出

规则说明: 0. 裁判组拥有最终裁定权

1. 胜负判定原则

(1) 地图为炸弹图时, 如 de_dust2 和 de_inferno 时, 以炸弹是否成功爆炸为胜负条件。当地图为 fy_iceworld_plus 时以杀伤数为胜负条件。

(2) 当由于赛程问题导致平局出现时, 由裁判组判定

2. 违例判定原则

(1) AI 程序不正常导致程序无法运行时, 经裁判组认定后判负

(2) 以各种形式 hack 对手数据, 经裁判组认定后判负

5.2. 参考思路:

由于本题没有提供当年比赛时现场提供的 AI 接口，因此无法提供源代码。

AI 设计的算法思路如下：设计 AI 时需要参照接口文档严格进行调用，否则很难拿到 AI 的分数。AI 的设计可以有多种风格：狙击型，冲锋型，埋伏型，等等。具体的 AI 设计时，仍然需要对当前的状态进行估价，并可以结合模拟退火或随机调整来得到比较理想的决策，决策方法可以与复赛中的题目进行参照。当然对于专业的 CS 游戏的 AI，需要考量的因素非常多，在短短的比赛时间中也很难面面俱到，但只要基本的功能得到实现，并结合一些简单的策略即可。

百度之星 2008

初赛一第一题：广告排名区间

1.1. 题目描述:

shifen 广告消费预估系统可以估计出一段时间内一个特定的广告在检索结果中排在各个位置的几率。比如系统对某广告的输出如下：

$p_1 = 0.03, p_2 = 0.08, p_3 = 0.04 \dots$

这说明该广告展现在第 1 位的概率是 3%，展现在第 2 位的概率是 8%，展现在第 3 位的概率是 4%……

问题是：如何给出一个排名估计区间 $[i, j]$ ，使得广告出现在该区间中的概率大于或等于一个预设值 p ，同时这个区间所包含的元素尽可能的少。也可用数学语言来描述：给定数 p 和数列 p_1, p_2, \dots, p_n ，求 i 和 j ($1 \leq i \leq j \leq n$)，在满足 $p_i + p_{i+1} + \dots + p_j \geq p$ 的前提下让 $j-i$ 最小。

一般来说， p_i 只需保留 6 位小数就足够了。这样，若令 $a_i = 10^6 p_i$ ， $a = 10^6 p$ ，则 a 和所有的 a_i 均为 $[0, 10^6]$ 之间的整数。这样就避免了对实数的处理。

输入数据：

第一行包含一个整数 n ($1 \leq n \leq 100,000$)。

以下 n 行每行包含一个 $[0, 10^6]$ 内的整数，依次为 a_1, a_2, \dots, a_n 。这 n 个整数之和保证不超过 10^6 。

最后一行包含一个 $[0, 10^6]$ 内的整数 a 。保证所有 a_i 之和不小于 a 。

输出数据：

输出仅一行，包含一个整数，即 $j - i$ 的最小值。

评分标准：

程序输出结果是否正确。

1.2. 关键思路:

问题为: $\min(j-i: \sum_{i \leq k \leq j} p[k] \geq a)$

记 $\text{sum}[i] = \sum_{1 \leq k \leq i} p[k]$;

$\Rightarrow \min(j-i: \text{sum}[j] - \text{sum}[i-1] \geq a)$

由于 $p[i] > 0$, 则 $\text{sum}[i]$ 严格单调增

可以枚举 i , 二分出最小的使得 $\text{sum}[j] - \text{sum}[i] \geq a$ 成立的 j

这时候区间 $[i+1, j]$ 即为在当前 i 条件下满足 $(j)-(i+1)$ 最小的符合条件的解
更新答案即可

初赛一第二题: LZW 网页判重

2.1. 题目描述:

有一种简单的网页判重的方法, 通过求两个网页内容的最长公共子序列 (LCS) 长度来判定两个网页的相似程度。

如:

(网页 A) 老师: 请用“果然”造句。

(网页 B) 学生: 先吃水果, 然后喝汽水……

它们的最长公共子序列为“果然”, 长度为 2。注意这里的“子序列”并不要求连续。

类似的, 下面两个网页:

(网页 A) 老师: 请用“果然”造句。

(网页 B) 学生: 先吃水果, 然后喝汽水, 果然拉肚子……

最长公共子序列还是“果然”, 长度为 2。但不难看出, 由于“果然”两个字在网页 B 中也曾连续出现, 第二组网页比第一组更加“相似”。为了区分开这两种情况的区分度, 我们改用一种称为 LZW 的理论。为了严格的叙述相似度的计算方法, 我们首先定义“文本单元”。假定网页用一个不包含空白字符 (空格、回车换行、水平制表符) 的字符串来表示。它只包含纯文本, 没有标签。在计算相似度之前, 你应该首先对该字符串进行处理, 划分成一个个“文本单元”。每个文本单元可以是一个中文字、英文单词 (由一个或多个连续的半角英文字母和数字组成, 正规表达式为 $[a-zA-Z0-9]^+$)、或者一个标点符号。

根据上述定义, 同一个标点符号的全角和半角应该被作为不同的文本单元, 尽管他们看起来可能很相近; 每个单独全角英文和全角数字都应该被看成一个单独的文本单元, 而连续的半角英文字母和数字应被看成一个整体。总之, 全角的字符可以与中文字同等对待。

这样, 网页被看成文本单元序列。例如, 网页“内容? 1 2 345 6 ??web2.00#”切分出的文本单元序列为 (为了显示方便, 用下划线分隔各文本单元):

内_容_?_1_2_345_6_?_?_web2_.00_#

而网页“why 内容相似??1234567890,web#00”的切分结果为:

why_内_容_相_似_?_?_1234567890_,_web_#_00

黑体部分给出了两个网页的一个公共子序列。注意“内容”、“??”分别在两个网页中都是连续出现的文本单元。为了奖励这种情况, LZW 规定一段由连续 k 个文本单元组成的字符串权值为 k^2 。在刚才的例子中, “内容”、“??”的权值均为 4。但“00”是一个数字串, 应当被看成一个单独的文本单元。所以权值仅为 1。

根据上述规则，公共子序列“内容 ?? 00”的权值为 $22+22+1=9$ 。在所有可能的子序列中，这个权值是最大的。

给定两个网页，求他们的 LZW 相似度，即所有可能的公共子序列中的最大权值。

注意

1) 输入的网页内容以 GBK 编码

2) 除了大小写英文字母和数字之外的其他半角字符均视为标点符号。

输入数据：

包含两行，分别是网页 A 和 B 对应的字符串（不包含空白字符）。每行至少包含 5 个字节，最多包含 200 个字节

输出数据：

输出仅一行，包含一个整数，为两个网页的 LZW 相似度。

评分标准：

程序输出结果是否正确

2.2. 关键思路：

首先将数据按照题意分割

题目有点像最大公共子序列但由于连续的计算为 k^2

所以需要增加一维表示之前连续的次数，为 $dp[i][j][k]$ 表示串 1 在 i 处，串 2 在 j 处，且之前连续 k 次

转移为

$dp[i][j][0] = \max(dp[i-1][j][1 \sim len], dp[i][j-1][1 \sim len])$

$if(s[i-1] == t[j-1])$

$dp[i][j][k(k>0)] = dp[i-1][j-1][k-1] + (k*k - (k-1)*(k-1));$

答案即为 $\max(dp[len_s][len_t][0 \sim len])$

初赛一第三题：钉子与木板

3.1. 题目描述：

墙上有 n 个钉子，编号为 $1, 2, \dots, n$ 。其中钉子 i 的横坐标为 i ，纵坐标初始为 x_i 。可以进行两种操作：

0 k v ：移动操作。竖直移动钉子 k ，坐标变为 (k, v) 。

1 s t v ：测试操作。若在高度为 v 处放一块横坐标范围是 $[s, t]$ 的水平木板，它将下落到什么高度？换句话说，求出钉子 $s, s+1, s+2, \dots, t$ 的纵坐标中，不超过 v 的最大值。如果这些钉子的高度全部大于 v ，则木板将落到地上，高度为 0。

注意，在测试操作时，水平木板只是用来测试的“临时木板”，将在测试后立即被拿走，不会影响到后续测试工作。

输入数据：

第一行包含两个整数 n, m ，即钉子的个数和操作的个数 ($1 \leq n, m \leq 10^5$)。以下 n 行每行一个不超过 10^9 的非负整数，即 x_i 。以下 m 行为各操作 ($1 \leq k \leq n, 1 \leq s < t \leq n, 1 \leq v \leq 10^9$)

输出数据:

按照输入的顺序, 对于每个测试操作输出一个整数, 即该测试水平木板的最后高度。

评分标准:

程序输出结果是否正确

3.2. 关键思路:

在线段树的每一个节点中维护一个 multiset, 记录对应区间内所有钉子的高度。

Multiset 可以在 $O(\log n)$ 的时间复杂度内查询小于等于某个数的最大值。

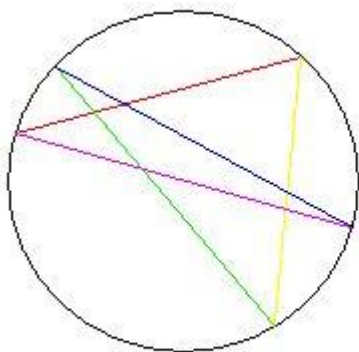
修改钉子高度的时候, 沿着根节点到叶节点, 从 multiset 中删掉原来的钉子的高度, 加入新的钉子的高度, 时间复杂度为 $O(\log n \log n)$

查询区间内小于等于 limit 的最大值, 在线段树中查询的时候只要查询对应节点的 multiset 中小于等于 limit 的最大值即可, 时间复杂度为 $O(\log n \log n)$ 。

总共 m 个操作, 总的时间复杂度为 $O(m \log n \log n)$, 空间复杂度为 $O(n \log n)$ 。

初赛二第一题: 圆内五角星

4.1. 题目描述:



如图, 一个半径为 1 的圆周上有 5 个点。按角度制给出 5 个点的极角 A_i ($0 \leq A_i < 360$, $i=1..5$)。按下图的方法连成一个五角星, 计算圆被切割成的 11 个部分面积的方差。

具体地说, 假定 11 个区域的面积分别为 S_1, S_2, \dots, S_{11} , 那么面积的均值计算方法为:
$$M = (S_1 + S_2 + \dots + S_{11}) / 11$$

面积的方差计算方法为:

$$D = ((S_1 - M)^2 + (S_2 - M)^2 + \dots + (S_{11} - M)^2) / 11$$

输入数据:

输入仅一行, 包含 5 个 $[0, 359]$ 内的互不相等的整数。

输出数据:

输出仅一行, 包含一个实数, 即各部分面积的方差。输出保留小数点后 4 位。

评分标准:

程序输出结果是否正确

4.2. 关键思路:

本题题目意思直接,按部就班的求出每一块的面积即可

朴素算法为首先求出中间相交得到的五个点,然后通过求出扇形面积来求弓形面积,然后可以求出 10 个三角形面积,5 个弓形面积剩下的五边形面积也较好求解标程为了稳妥起见,采取了检验过正确性的代码。

$\text{area}(r, \text{info})$ 返回的是以 $(0, 0)$ 为圆心, r 为半径的圆与多边形 info 的面积交对每个圆上的点构造切线,相邻两点的切线的交点,这样弓形面积可以转化为多边形与圆的面积

初赛二第二题: 传输方案规划

5.1. 题目描述:

面对艰巨复杂的技术挑战,百度所崇尚的系统设计哲学是“简单可依赖”,而百度的工程师们正在互联网世界中实践着这种理念。这里正好有一个挑战,让作为百度之星的你小试牛刀:

在处理数以百亿计的网络信息的过程中,有一个很常见的问题:

怎么样将一个集群上的信息以最低的成本传输到另外一个集群上?

数据源集群 A 有 n 台服务器,编号为 $1, 2, \dots, n$, i 号服务器上待传输的数据量为 A_i , 单位是 GB。

目的地集群 B 有 m 台服务器,编号为 $1, 2, \dots, m$, j 号服务器上的空闲容量为 B_j , 单位为 GB。

A 集群的 i 号服务器上的每 GB 数据对于 B 的集群的 j 号服务器收益为 $V_{i,j}$, 从 A 集群的 i 号服务器向 B 集群的 j 号服务器传输 1GB 数据的开销为 $C_{i,j}$ 。

你的任务是在保证 A 中的所有数据传输完毕的前提下,性价比 V/C 尽量高。其中 V 为所有数据在 B 集群上的价值之和, C 为总开销。换句话说,若 A 集群的 i 号服务器向 B 集群的 j 号服务器发送了 $T_{i,j}$ 个 GB 的数据 ($T_{i,j}$ 不一定是整数), 则性价比定义为:

输入数据:

第 1 行两个整数 n, m ($1 \leq n, m \leq 50$), 即集群 A 和 B 各自的服务器台数。

第 2 行包含 n 个不超过 100 的正整数 A_1, A_2, \dots, A_n , 即集群 A 中每台服务器的待传输数据量 (单位: GB)。

第 3 行包含 m 个不超过 100 的正整数 B_1, B_2, \dots, B_m , 即集群 B 中每台服务器所能接受的最大数据量 (单位: GB)。

第 4 ~ $n+3$ 行每行包含 m 个不超过 100 的非负整数 $V_{i,j}$, 表示集群 A 的 i 号服务器中每 GB 数据对于集群 B 中的 j 号服务器的价值。

第 $n+4$ ~ $2n+3$ 行每行包含 m 个不超过 100 的正整数 $C_{i,j}$, 表示集群 A 的 i 号服务器中每 GB 数据传输到集群 B 中的 j 号服务器所需要的开销。

输出数据:

仅一行, 为最大性价比。输出保留三位小数 (四舍五入)。如果 A 的数据无法传输完毕, 输

出-1。

评分标准：

程序输出结果是否正确

5.2. 关键思路：

为网络流模型。

题目要求必须要全部数据传输完毕，即要求满足满流量。

而性价比： V/C 为典型的分数规划问题

令： $V = v_1 + v_2 \dots + v_n$

$C = c_1 + c_2 \dots + c_n$

$\Rightarrow V / C = X$ (取最大)

$\Rightarrow V = C * X$

$\Rightarrow v_1 + v_2 \dots + v_n = X * (c_1 + c_2 \dots + c_n)$

$\Rightarrow (v_1 - X * c_1) + (v_2 - X * c_2) \dots + (v_n - X * c_n) = 0$

即二分 X ，跑满足条件的最大费用最大流，边费用为 $(v_n - X * c_n)$

当最大费用为 0 时候， X 取最大最大值即可

初赛二第三题：公平数

6.1. 题目描述：

如果一个整数的十六进制表示（不含前导 0）中，前一半数字之和等于后一半数字之和，我们称它为公平数。

注意，如果该数的十六进制表示中包含奇数个数字，则正中间的数字既不属于前一半，又不属于后一半。

例如在十六进制下 $1D=7+7$ ，因此 $1DE77$ 是公平数。数字 E 并不参与计算。

再例如，

所有单个数字的十六进制数（即 $0 \sim F$ ）均为公平数，但 $F0$ 不是（不能把 $F0$ 补充前导 0 写成 $0F0$ ，进而认为它是公平数）。

给出十六进制数 K ， X ， Y 和十六进制数字集合 S ，求区间 $[X, Y]$ 之内，有多少个公平数满足：

十六进制表达式（不包含前导 0）中每个数字均在集合 S 中

并且为 K 的倍数

输入数据：

输入第一行为数字集 S ，包含 $0 \sim 9$ 以及大写字母 $A \sim F$ 。

每个数字或字母最多出现一次。

第二行包含 3 个十六进制正整数 K ， X ， Y ，均不超过 10 个数字（包含 $0 \sim 9$ 以及大写字母 $A \sim F$ ，不包含前导 0）。

输出数据：

仅一行，包含一个整数，即满足条件的公平数个数。

评分标准：

程序输出结果是否正确

6.2. 关键思路:

数字集最多 16 个 ($0^9 + A^F$)

而 K, X, Y 不超过 10 个数字, 也就是说 Y 最大为 10 个 F, 为 $16^{10} = 2^{40} = 1024^4 = 10^{12}$

当 K 较大的时候: $K > 10^5$, 直接枚举, 复杂度为 $\leq 10^7$

当 K 较小的时候: 为数位 dp, 枚举长度 len, dp[len][sum][modk] 表示长度为 len, sum 为前 len/2 个数求和减去后 len/2 个数求和, modk 为取模值

复杂度为 $\text{len} * \text{sum} * \text{modk} = 10 * (5 * 16) * 10^5 = 10^8$, 实际采用记忆化搜索复杂度达不到 10^8

初赛二第四题: 成语纠错

7.1. 题目:

题目描述:

成语是中华民族的文化瑰宝, 作为历史的缩影、智慧的结晶、汉语言的精华, 闪烁着睿智的光芒。

你的任务是给一个错误的四字成语进行纠错, 找到它的正确写法。具体来说, 你只允许修改四个汉字中的其中一个, 使得修改后的成语在给定的成语列表中出现。原先的错误成语保证不在成语列表中出现。

有时, 这样的“纠错”结果并不惟一。例如“一糯千金”可以改为“一字千金”也可以改成“一诺千金”。但由于“糯”和“诺”是同音字, “一糯千金”实为“一诺千金”的可能性比较大。

因此, 我们还将提供一个汉字分类表, 要求修改前后的两个字必须属于同一个分类。在这样的限制下, 我们保证成语纠错的结果惟一。

注意

1、汉字均采用 GBK 编码(详见 2008 百度之星第一场初赛题目 02)

2、每个汉字分类至少包含两个汉字, 同一个汉字可能出现在多个类别中, 同一类别的汉字各不相同。

3、成语列表中的成语都是真实存在的四字成语, 未在分类表中出现的汉字不允许修改。

输入数据:

输入第一行包含两个整数 n, m ($1 \leq n \leq 200$, $1 \leq m \leq 20000$)。n 表示汉字类别的个数, m 表示成语的个数。

以下 n 行每行用一个无空白分隔符(空格、TAB)的汉字串表示一个分类中的所有汉字。注意, 该汉字串最多可能包含 200 个汉字。

以下 m 行为成语列表, 每行一个成语, 恰好四个汉字。

最后一行为待纠错的成语, 恰好四个汉字, 且不在成语列表中出现。

输出数据:

仅一行, 为一个四字成语。在“修改必须在同一分类中进行”的限制下, 输入数据保证纠错结果惟一。

评分标准：

程序输出结果是否正确。

7.2. 关键思路：

$$n = 200, m = 20000$$

根据目标串，一一对比成语列表，发现只有一个汉字不同的成语，取出这个两个字，看是否在同一个集合

首先需要预处理出来某个字在哪几个集合内，最多 200 个集合，用数组装着即可

故复杂度为 $O(4 * m * n)$

复赛第一题黑白树

1.1. 题目描述：

在图论中，包含 n 个结点（结点编号为 $1 \sim n$ ）， $n-1$ 条边的无向连通图被称为树。在树中，任意一对结点间的简单路径总是惟一的。你拥有一棵白色的树——所有节点都是白色的。接下来，你需要处理 c 条指令：

1. 修改指令 ($0v$)：改变一个给定结点的颜色（白变黑，黑变白）；

2. 查询指令 ($1v$)：询问从结点 1 到一个给定结点所经过的第一个黑色结点编号（假设沿着简单路径走）。

注意，在查询指令中，必须从结点 1 而不是结点 v 出发。如果结点 1 本身就是黑色，查询指令应该返回 1。

输入数据：

第一行包含两个正整数 n, c ，即结点数和指令条数。以下 $n-1$ 行，每行两个正整数 (u_i, v_i) ($1 \leq u_i$ 示结点 u_i 到 v_i 之间有一条无向边。以下 c 行，每行两个整数 (c, v) 。当 $c=0$ 时表示修改指令，其中 v 代表被修改的结点编号； $c=1$ 时表示查询指令。你的程序需要输出结点 1 到结点 v 之间路径的第一个黑色结点编号。在第一条指令执行前，所有结点都是白色的。

输出数据：

对于每个查询操作 ($c=1$ 的操作)，输出一行，包含一个整数，即第一个黑色结点的编号。如果不存在黑色结点，输出 -1。

评分标准：

程序输出结果是否正确。

1.2. 关键思路：

先以 1 为根节点 DFS，计算每个节点的入栈时间 $in[i]$ 和出栈时间 $out[i]$ 。把每个节点按照入栈时间的顺序进行排序，建线段树。线段树中节点存的是该区间内，颜色为黑色的节点的出栈时间最大值。

对于两个操作：

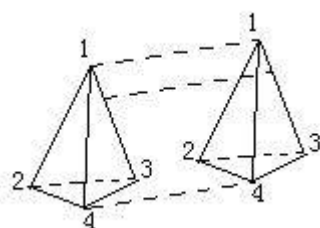
1. 修改节点 v 颜色。修改节点颜色后，需要更新线段树中的最大值。时间复杂度 $O(\log n)$ 。
 2. 查询从节点 1 到节点 v 的简单路径上第一个黑色节点，就等于在线段树中查询 $[1, \text{in}[i]]$ 这个区间内的最大的值所对应的节点。如果最大值不为 0，最大值对应节点的出栈时间大于等于节点 v 的出栈时间，那么该点就是答案。时间复杂度 $O(\log n)$ 。
- 总的时间复杂度为 $O(n \log n + c \log n)$ 。

复赛第二题：寻找所有的同构象

2.1. 题目描述：

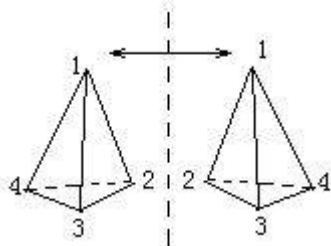
三维空间中，为一个物体定义如下三种基本的变换：

平移：物体沿着一条直线 α 进行一段位移 δ 。对于物体中的任意一点，其起始位置和终止位置连成的线

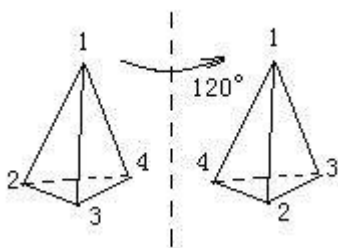


段平行于 α ，长度为 δ 。

反射：物体相对于一个平面 β 作镜像。对于物体中的任意一点，其起始位置和终止位置连成的线段垂直于 β ，并且到 β 的距离相等。



旋转：物体绕着一条轴线 γ 转过一定角度 φ 。对于物体中的任意一点，由其起始位置和终止位置对 γ 作垂线，垂足重合，垂线段长度相等，并且两条垂线所成的角度为 φ 。



物体 A 经过上述三种基本变换或基本变换的有限次组合，变成物体 A' ，若 A' 与 A 完全重合（对于空间中的任意一点，如果它属于 A' ，则一定属于 A ，反之亦然），则称 A' 为 A 的一个同构象。

需要注意的是，上面所说的物体和它的同构象并不一定是相同的物体，可以理解为物体上的每个点都是“有记号”的，虽然二者在形状、体积、位置上完全重合，但其中只要有一个点经过变换以后位置发生了改变，就是不同的物体。例如，线段 u 关于它的中垂面做反射得到线段 v ，虽然 u 和 v 在三维空间中重合，但 u 上的点变换到 v 以后位置发生了改变，所以 u 和 v 是不同的物体。

从定义出发，容易证明，同构象满足下面的 3 条性质：

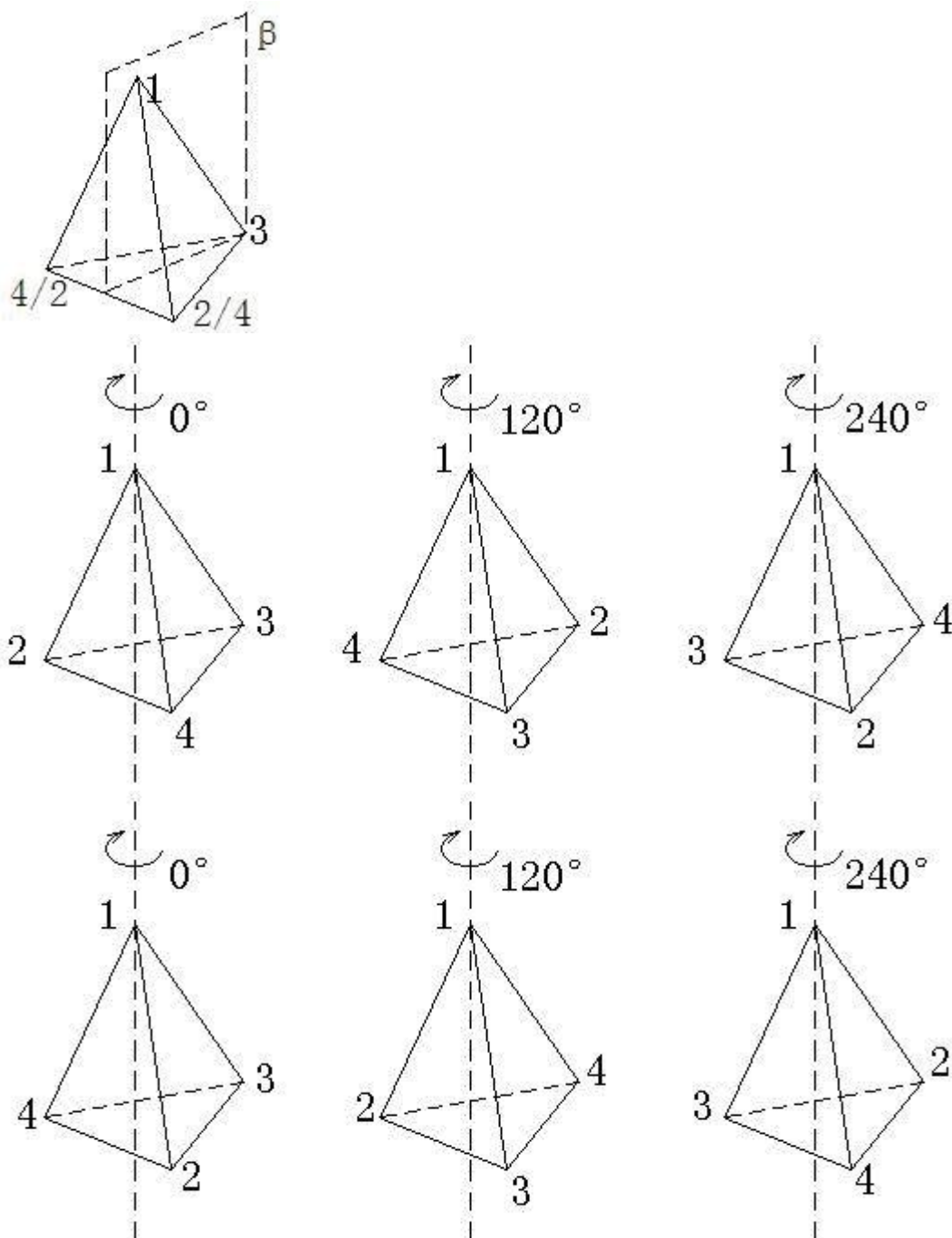
1. 自反性： A 是 A 自身的一个同构象。
 2. 对称性：若 B 是 A 的一个同构象，则 A 是 B 的一个同构象。
 3. 传递性：若 B 是 A 的一个同构象， C 是 B 的一个同构象，则 C 是 A 的一个同构象。
- 推论：若 B 是 A 的一个同构象， C 也是 A 的一个同构象，则 B 和 C 彼此互为同构象。

B 和 C 都是 A 的同构象，若有一个基本变换序列 S 把 B 变换到 C，且满足对于任意点 p 属于 B，S 把 p 变换到 p 自身，则称 B 和 C 为相同的同构象。例如，线段 u 绕着中垂线旋转 180 度得到一个同构象 v，还可以关于中垂面反射得到一个同构象 v'，v 和 v' 就是相同的同构象。

同构象的相同和不同是对本题非常重要的概念。事实上，由于 B 和 C 都是 A 的同构象，由上面的推论，B 和 C 一定互为同构象，因此 S 是一定存在的。那么 B 和 C 相同的唯一要求就是 S 把 B 变换到 B 自身，其实同构象“相同”的本质就是 B=C。

A 的全部不同的同构象组成的集合叫做 A 的同构象集。例如，上面所举的线段的例子中，u 的同构象集就是 {u, v}。

又例如，一个正三棱锥的同构象集共有 6 个元素，分别沿中轴旋转 0、120、240 度得到其中 3 个，再沿图中的 β 平面做反射以后，同样旋转又得到另外 3 个，这 6 个同构象集中的元素任意两者都不相同：



题目输入一个凸多面体的所有顶点坐标，要求输出这个凸多面体的同构象集所包含的元素个数。

输入的点至少有 4 个，每个输入作为一个顶点可以唯一确定一个多面体。不需要考虑非凸的情况，不需要考虑多点共线的情况，也不要求处理所有点退化到一个平面多边形的情况。但是有可能存在 4 个或以上的点共面。

多点共面的判断规则如下：四个点组成的四面体中，一个面积最大的表面的面积为 S1，三个面积较小的

表面的面积之和为 S_2 ，若，则可认为四点共面。若有更多的点共面，输入保证其中任意四个点都满足上述不等式。

如有需要，可使用三角形面积的 Heron 公式，这个公式是数值稳定的：

$$A = \frac{1}{4} \sqrt{(a + (b + c))(c - (a - b))(c + (a - b))(a + (b - c))}.$$

其中 a 、 b 、 c 分别为三角形三边长，且。

输入数据：

输入第一行为一个整数，表示凸多面体的顶点数 n ，此后每行为三个浮点数，精度为小数点以后 10 位有效数字，表示一个顶点的直角坐标 (x, y, z) ，顶点可能按照任意的顺序给出。

输入顶点数目； $.,$ 均小于 10000。

输出数据：

输出只有一行，为一个整数，表示同构象集元素的个数。

评分标准：

程序输出结果是否正确

2.2. 关键思路：

本题的目标是统计同构置换的个数

程序的大概思路如下

- 1 扫出 3 维凸包作为预处理
- 2 寻找该凸多面体的对称面, 存在则 $p=2$, 不存在则 $p=1$
- 3 寻找该凸多面体的旋转轴, 第 i 个轴可以旋转 a_i 种度数使得重合
- 4 根据 2, 3 结果计算答案 $ans=p*\prod a_i$

1. 0 比较好处理, 拥有成熟的时间复杂度 $O(n^2)$ 算法, 而且几乎不存在达到上界的数据

2. 0 最朴素的思想是 $O(n^2)$ 枚举两个点关于某平面 α (两点连线的垂直平分面) 是对称的, $O(n)$ 的处理处所有点在 α 上投影的坐标和距离然后统计是否能构成对应, 总复杂度是 $O(n^3 \lg n)$

3. 0 对于给定的直线和角度, 可以 $O(1)$ 的得到每个点旋转后的坐标, 用 set 维护可以 $O(n \lg n)$ 的确定能否以该度数旋转

3. 1 旋转轴可能是一个面的旋转轴, 面的个数是 $O(n)$ 级别, 对于一个面可以 $O(\text{点数})$ 的得到重心 W , 然后可以判断这个面上的点到重心距离是否相等, 如果相等, 选定面上一个点 A , 枚举面上另外一个点 B , 以 WA, WB 的夹角做为旋转角度根据 3. 0 进行判断. 这一部分总复杂度 $O(n^3 \lg n)$

3. 2 旋转轴还可能由顶点 $O(n)$ 个和棱中点 $O(n)$ 构成的点集中两个点的连线, 取两点, 可以枚举旋转轴, 但是旋转角度需要 $O(n)$ 处理出每个顶点关于这条直线的相对角度排序后类似 3. 1 枚举角度, 复杂度 $O(n^4 \lg n)$

3. ?

4. 0 上述步骤完成后很容易得到答案

本题数据较难构造

第三步并没有想到也没有讨论出较好的结果, 不能保证上述情况能够包含所有的旋转轴

无法实现正确的代码

测试数据

Input:

4

0.000000000000.000000000000.000000000000

1.000000000000.000000000000.000000000000

0.500000000000.86602540380.000000000000

0.500000000000.288675134620.000000000000

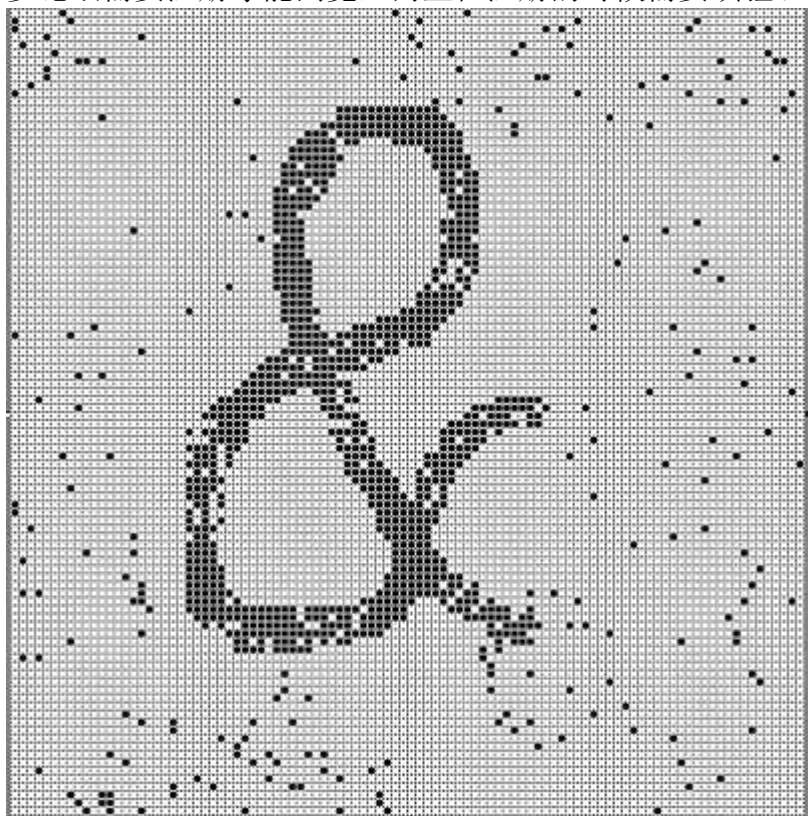
Output:

6

复赛第三题：验证码识别

3.1. 题目描述：

Baidu 的论坛抓取机器人小 A，最近的抓取表现越来越差了。工程师小明发现原来是很多论坛需要注册才能浏览，而且在注册的时候需要填验证码。如图所示



于是小明准备升级小 A 机器人，让它能够自动识别验证码。你能帮助小明设计识别验证码的程序吗？为了控制开发难度，这个版本只需要识别！@#¥%&*~+九个符号即可。我们已经为你将图片拆分成 100*100 个点的位图，每个位图只包含一个符号，如下图所示的符号&。

同时为了让这个过程更有趣一点，我们将程序设计成交互式的，即你的程序向测试程序提问，

通过测试程序的回答收集信息，当信息足够的时候输出解答即可。

例如：（注意“_”代表下划线，而不是空格）

1. 提问:你的程序向 stdout 输出字符串 Q_1_3\n，代表查询坐标（1，3）点的黑白信息；
2. 回答:测试程序向你的程序的 stdin 写入 P_0\n，代表（1，3）点的颜色为白，同理，如果你的程序读到 P_1\n，代表（1，3）点的颜色为黑；
3. 1 和 2 步骤不断循环，经过若干次交互，你的程序已经找到了答案，则可以输出结果：你的程序向 stdout 输出 R_&\n，测试程序就会记录识别结果和询问次数并退出测试。

测试程序

我们准备了一个帮助你测试的客户端程序，点击下载 Test.zip，需要的 jre 版本是 1.6.0_05，下载 jre。

注意

1. 识别单个图片，询问次数超过 2500 次则不得分，识别正确，且询问次数分数不高于 500 次得全分，高于 500 次后分数线性递减；
2. 识别单个图片，交互总时间超过 15s 则不得分；
3. 识别结果正确得 30%的分数，识别结果错误不得任何分数；
4. 所有测试图形都由同一个出题人书写，字体方向正放向上；
5. 尺度不小于 50 像素，即主体符号无噪声包围盒的长宽不同时小于 50 像素；
6. 图片随机噪声点比例不高于 15%，图片上的黑白点都有可能随机翻转，而在添加噪声的过程中我们已经保证主体符号的含义是确定的；
7. Test.zip 中的测试数据是无噪声的，与评测数据不同，请特别注意。

3.2. 关键思路：

100 个询问（10*10 采样）：确定验证码中心位置

100~200 个查询确定黑色格子大致分布：在已知的黑色像素点周围做 A*搜索（策略是找黑色像素点分布较密集的方向搜索）

100~200 个查询最终确定交叉数和交叉类型：在上一操作中找出可能的交叉点（交叉点是周围的黑白像素点均匀交叉分布），然后在可能的交叉点周围做精细搜索，确定交叉类型

最后跟 9 个图案做交叉点及交叉方向匹配

决赛

1.1. 题目：直升机大战 Air Strike Helicopters

Air Strike Helicopter 是一种虚拟的 3D 武装直升机，选手需要为这种直升机编写 AI 控制程序控制飞行机动，并使用火箭击落对手 AI 控制的直升机。



比赛赛制：赛制为挑战赛，选手可以在比赛平台选择任意对手的 AI 进行挑战，挑战成功则获得对手当前积分的一半，挑战失败则失去自己积分的一半，以最后总积分为排名决定最终名次。

接口：

选手需要实现如下控制接口

```
int heli_control( struct scene_info sinfo[], struct flight_info *finfo, struct cmd_info *cmd )
```

sinfo 为系统提供的比赛场景信息，定义为

```
struct scene_info
```

```
{
```

```
    int obj_num; //障碍物数量
```



```

    struct obj_info obj_arr[]; //障碍物信息数组
}

struct obj_info
{
    double x; //空间坐标 x
    double y; //空间坐标 y
    double z; //空间坐标 z
    double r; // 物体半径 r
    int type; // 物体类型，可能为障碍物，直升机，对手火箭，己方火箭
}

```

finfo 为己方直升机状态

```

struct flight_info
{
    double x; //空间坐标 x
    double y; //空间坐标 y
    double z; //空间坐标 z
    double a; // 火箭炮指向 a
    double b; // 火箭炮指向 b
    double c; // 火箭炮指向 c
}

```

cmd 为选手经过计算返回控制指令

```

struct cmd_info
{
    int thrust; //旋翼升力
    int rotate; //尾桨出力，用于改变航向 0 为居中，大于 0 为左转 小于 0 为右转
    double alpha; //旋翼与主轴的夹角
    double dir; // 旋翼偏转方向，0 为正前方，右手原则
    int shoot; // 射击指令 0 为不射击 大于 0 为射击
}

```

1.2. 解题思路:

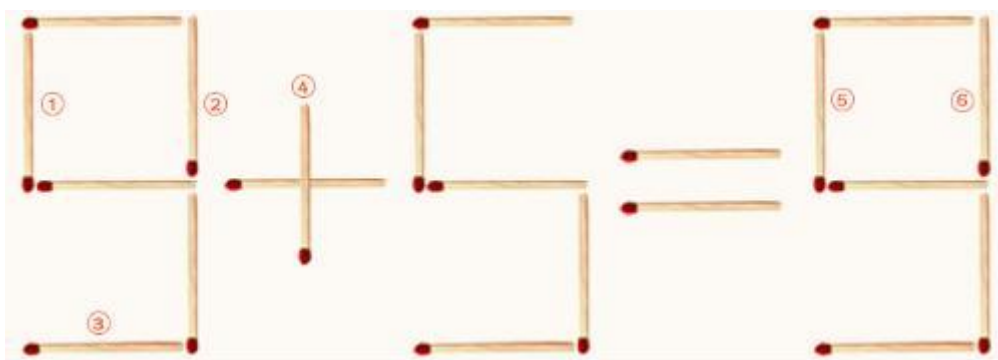
1. 因为整个比赛环境是仿真的 3D 物理世界，选手首先需要学习直升机的物理特性，建立利用用几个基本控制量控制直升机动作的方法
2. 在掌握了直升机的基本控制的基础上，需要建立瞄准射击的方法，仍然依赖对于直升机姿态的控制
3. 在飞行和射击动作建立基本控制函数的前提下，需要建立导航和策略模型。有两种策略可供参考，一种是偏向防守的固定策略，不论对手策略如何，均在比赛空间内选择比较有利的地形和位置防守，等待对手进入射击范围进行精确射击。另一种是动态搜索的策略，将比赛空间划分成很多小的区域。不断根据场上数据的变化更新局势评估，选择飞向最有利的区域位置，同时不断评估是否有射击机会。
4. 选择对手也很重要，比赛策略往往是相生相克的，选择比较容易的对手，容易获得积分但积分不一定高，选择难对手容易失去积分，但一旦成功收获巨大。

百度之星 2009

初赛第一题：火柴游戏

1.1. 题目描述

在百度，同事们之间喜欢交流游戏。其中，火柴游戏是一个比较经典的例子。游戏的规则很简单：恰好移动一根火柴，使等式成立。如下面的等式可以变成 $3+6=9$ （还有其他解）：移动哪一根火柴能使等式成立？



下面是所有火柴数字的样子

0 1 2 3 4 5 6 7 8 9

请你写一个程序，找出所有的规范解。所谓规范是指：

- * 只能改变数字，不能改变符号；
- * 数字和符号的组成方式必须严格的和图示的一样（减号由一根火柴组成）；
- * 新等式必须形如 $a+b=c$ 或 $a-b=c$ ，其中 a 、 b 、 c 都是不含前导 0 的非负整数。

当然，最重要的是：新的等式必须在数学上成立。

输入格式

输入仅一行，为一个格式为 $a+b=c$ 或 $a-b=c$ 的表达式，其中 a 、 b 、 c 均为不含前导 0 的非负整数。表达式长度不超过 100，且不含空白字符。因此，加号/减号紧跟在 a 的后面、 b 紧跟在加号/减号的后面、等号紧跟在 b 的后面、 c 紧跟在等号的后面。

输出格式

输出所有规范解，按字典序输出（请注意：输出顺序不对将不得分）。无解时，仅输出一行-1。

样例输入 1

9+5=9

样例输出 1

3+5=8

3+6=9

样例输入 2

1+1=2

样例输出 2

-1

测试数据

共 10 个测试点，基本参数如下表：

测试点编号	表达式的长度
1-2	1-10
3-4	10-25
5-6	26-50
7-10	51-100

24.

25.

26.

裁判问答:

Q: 第一题的表达式长度不是只有 5 吗? A: 可以多位整数

Q: 把某根移出来再移到原来的位置上算不算移动? A: 不算

1.2. 解题思路:

首先处理每一个数字在被拿走一根火柴之后, 可能与哪些数字变成新的数字。

枚举给出的运算数字中的每一位数字, 再枚举从其中拿走火柴后和哪一位数字进行变化, 若变化后等式两端值相等则记录。

初赛第二题: 电子商务平台商品推荐问题

2.1. 题目

百度网络交易平台(“百度有啊”)是建立在百度旗下独有的搜索技术、强大社区资源基础上的中文互联网领域最具规模的网上个人 C2C 交易平台。伴随着“百度有啊”的成长, “有啊”的顾客也蜂拥而至; 面对如此大量的用户, 如何把平台上数以千万计的商品按一定的规则推荐给他们以促成交易, 是“百度有啊”面临的重要问题。

在本题中, 假设有 M 个用户和 N 种产品, 每个用户的浏览历史可以用一个 N 维特征向量 X 描述: $X_i=1$ 当且仅当该用户曾经浏览过商品 i 。如果用户 A 和用户 B 曾浏览过(部分)相同的商品, 我们说用户 A 和用户 B 相似; 如果用户 A 和用户 B 相似, 或者用户 A 和一个“与用户 B 相似”的用户相似, 则需要把用户 A 和用户 B 划分到同一个用户群。该用户群中所有用户的特征向量的“按位或”便是整个用户群的特征向量——它表示至少被其中一个用户浏览过的商品集合。

每当一个新用户到来时, 可以计算出它和所有用户群之间的相似度。假定他的特征向量为 A , 用户群的特征向量为 B , 则:

$$\text{similarity}(\vec{A}, \vec{B}) = \cos(\vec{A}, \vec{B}) = \frac{\vec{A} \cdot \vec{B}}{\|\vec{A}\| \|\vec{B}\|}$$

其中计算特征向量模长时使用的是二范数, 即所有维度数值的平方和的算术平方根。

接下来, 我们应找出该用户最接近的用户群, 并从该用户群购买过的商品中选三个商品进行推荐。一般来说, 一个商品被购买的次数越多, 就越应该被优先推荐(若购买次数相同, 优先推荐 id 值小的商品), 但为了避免马太效应, 我们需要做一个特殊处理: 不推荐最畅销的商品(如果有多个商品的购买次数都是最多的, 则它们都不应该被推荐, 为简单计, 如果所有商品的购买次数都一样的话就都不推荐了)。另外, 推荐给用户的商品不能是他已经购买过的商品。

如果从最接近的用户群中无法推荐出三个商品，应从次接近的用户群购买的商品中以相同的规则补充，以此类推，直到选出三个推荐商品，或者无法选出更多商品。

测试数据保证任何一个用户不会跟两个不同用户群的相似度相同，因此商品推荐的结果总是惟一的。

注意：如果用户浏览记录跟某用户群的相似度为 0，则在任何情况下都不从该用户群购买的商品中推荐。

输入格式

第 1 行：M、N，分别是平台用户数和商品总数。 $0 < M, N \leq 100\,000$

第 2 行：K，表示接下来有 K 行记录。 $0 < K \leq 100\,000$

第 3~K+2 行，每一行是一次用户的浏览-购买记录，记录格式为：

uid i1, i2, ..., ip b1, b2, ..., bq

其中 uid 是不超过 M-1 的非负整数，代表用户 id。i 为本次浏览的商品 id 集合（无重复元素，元素顺序无意义），而 b 为该用户在此次浏览后购买的商品集合（无重复元素，元素顺序无意义）。i 中的每个元素均为不超过 N-1 的整数，b 是 i 的子集。 $q \leq p \leq 50$ 。注意：同一个用户 ID 可以对应多条记录

第 K+3 行：Q，表示接下来有 Q 次查询。 $0 < Q \leq 125$

第 K+4~K+Q+3 行：每一行是一次用户的浏览记录，格式为

uid i1, i2, ..., ip

含义类似于浏览-购买记录。注意：用户群划分方式完全取决于第 3 行开始的 K 条记录。这里的查询不会导致用户群划分方式的变化（在实际的系统中，用户群数据也是定期更新，而非实时修改）。

输出格式

对每个查询输出一行，为推荐的最多三个商品的 id（为不超过 N-1 的非负整数），按推荐度降序排列。如果没有任何可推荐的商品，输出 NONE；如果有商品可推荐，但不足三个，应全部输出。

样例输入

```
9 12
8
0 0, 1, 2 1
1 3, 4, 5 3, 4
2 1, 5 1
3 6, 7 6, 7
4 8, 9 8
5 8, 10 8, 10
0 11 11
```


8 6 6

3

6 0, 1, 2

1 0, 1, 2, 6

3 8, 9

样例输出

3 4 11

11 7

10

PS: 原样例有错, 于 22: 45 分修正样例

样例解释

首先对购买历史进行预处理形成用户群, 然后对每个用户群购买的商品按购买次数进行排序, 结果如下:

用户群 ID	用户 ID	浏览过的商品 ID 集合	购买的商品 ID
0	0, 1, 2	0, 1, 2, 3, 4, 5, 11	1 (2 次), 3, 4, 11
1	3, 8	6, 7	6 (2 次), 7
2	4, 5	8, 9, 10	8 (2 次), 10

接下来处理查询。

输入：0, 1, 2 输出：3, 4, 11

此浏览记录与用户群 0 最接近。根据规则推荐 3, 4, 11（1 是最畅销商品，不推荐；对于被购买数量相同的商品 3 和 4，优先推荐 id 值小的商品）

输入：0, 1, 2, 6 输出：7, 11

此浏览记录与用户群 1 最接近，根据规则推荐 7（6 是最畅销商品，不推荐）；由于不足三个商品，从次接近的用户群中推荐，推荐 11（1 是最畅销商品，3、4 是购买过的商品）；仍然不足三个商品，但是已无更多商品可以推荐。

输入：8, 9 输出：10

此浏览记录与用户群 2 最接近，根据规则推荐 10；不足三个商品，但是已无更多商品可推荐。

2.2. 解题思路

1、本题主要是一道码力题，没有太多的算法性。整场比赛只有三个人满分，题解的代码为 lunarmony 的满分代码

2、本题需要的算法知识主要是对于并查集的基本使用。最大的难点我想是在对输入输出很多细节的考虑上，详情请参考代码

3、另外一个小技巧是将相似度的计算转化为位操作，提高了速度。具体说来，一个 id，用它除 20 的商做数组下标，余数做数组的值。

初赛第三题：葫芦娃

3.1. 题目描述

蝎子精和蛇精为祸人间，葫芦七兄弟准备与之决一死战。不幸的是，七弟不慎被两只妖精抓住，困在了蛇蝎山的囚笼里，其余六兄弟必须尽快去营救。二娃使用千里眼，查看到七弟被囚的位置，但蛇蝎山地势复杂，机关遍布，如何才能又快又安全的把七弟救出来呢？于是，六兄弟请您来帮忙了。

在二娃的帮助下，大家先绘制了一张蛇蝎山的地图，并把能安全停留的地方以点标记。你很快就发现，这些安全点组成了一个六邻接图——每个点都与左上、左、左下、右下、右、右上六个点等距。于是，你以其中两条坐标轴：“左——右”和“左上——右下”，给各点设置坐标（见图 1）。只要把囚笼的六个邻接点都占了，然后六兄弟一起施法，就能把七弟营救出来。

图 1. 六邻接图及坐标

虽然已经有了地图，但怎样走才能最快的把七弟救出来呢？葫芦兄弟告诉你，他们有两种移动方式：

1、跑步。可在一单位时间移动一单位距离，即从一个点移动到某个邻接点（见图 2）。

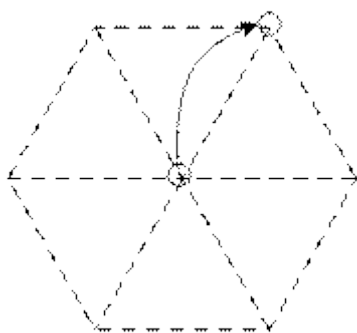


图 2. 跑步移动示意图

2、翻跟头。可在一单位时间沿着一个坐标轴方向移动多个单位距离，但其飞过的每个点上都必须有葫芦兄弟站在那里施法。例如，在点 $(0, 0)$ 和点 $(1, 1)$ 都有葫芦娃，那么位于点 $(2, 2)$ 的葫芦娃便可在兄弟的帮助下，沿着“左下——右上”坐标轴直接翻跟头到点 $(-1, -1)$ （见图 3）。

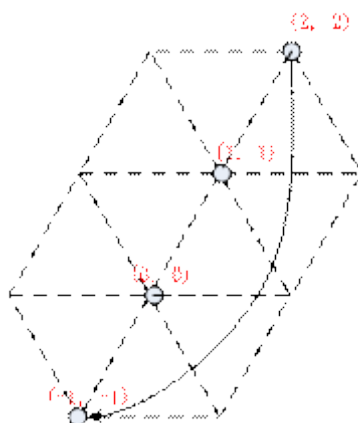


图 3. 翻跟斗移动示意图

另外，为了不引起妖精的注意，每一单位时间最多只有一个葫芦娃能移动，且每个点上只能站一个葫芦娃。由于葫芦兄弟心灵相通，被困的七弟也能为兄弟施法。

六个葫芦娃的出发位置为 $(0, 0)$ ， $(1, 0)$ ， $(2, 0)$ ， $(1, 1)$ ， $(2, 1)$ ， $(2, 2)$ 。如果按照最快的方案，六兄弟需要多长时间才能救出七弟呢？

输入格式

输入仅包含一行，包含两个整数 X ， Y ，表示囚禁七弟的位置。只要把此点的六个邻接点都占了，就能把七弟救出来。 (X, Y) 不会和上述六个出发点重合。 $0 \leq X, Y < 7$

输出格式

输出一行，包含一个整数，即营救的最短时间。

样例输入 1

3 1

样例输出 1

4

样例输入 2

3 4

样例输出 2

8

样例输入 3

0 5

样例输出 3

14

测试数据

共 30 组数据，输出结果近似在 $[0, 16]$ 内均匀分布。

3.2. 解题思路：

题目给出了 6 兄弟的初始位置和七弟的位置。可以想到每次将离七弟最远的葫芦娃移动到较近的位置是最优的情况。所以我们先将能移动的 6 个方向记录，bfs 求出每个位置到七弟的距离。然后进行搜索。

在搜索的过程中，我们每次将距离七弟最远的葫芦娃移动，并且利用之前求得的距离限制来保证葫芦娃不会移动到更远的位置。

在移动的过程中判断目前的状态是否已满足要求，是则更新答案 ans，若目前 dfs 的步数 \geq ans，那么可以直接返回。步数的初始限制定位题目中提到的 16。

即可求得答案。

初赛第四题：争车位

4.1. 题目描述

争车位是目前 SNS 网站上比较热门的游戏之一。假如小明有 N 个好友和 M 辆车。每个好友都有 K 个车位。这些车位可能停放了小明的车，也可能停放了其它人的车，还能没有停任何车辆（空车位）。在当前状态中，小明的 M 辆车全停放在他的好友的车位上。

任务一：考虑如下的移车规则：

1. 必须按一定顺序依次移动自己的车，而不能移动别人的车。
2. 每辆车只可以移动一次。
3. 车只能从原来的位置移到空车位处，不能移动到一个已停放了车的车位（无论这辆车是谁的）。移动后，原来的位置就成了空车位。
4. 只能跨好友移车。也就是说，一辆车不能从某好友的车位移到这个好友的另一车位。

请帮助小明把他的所有 M 辆车都移动一遍。

任务二：考虑如下的移车规则：

1. 每个车位都对应一个金额。当小明把一辆车最终停在某个车位时，将会得到该车位对应的金额。
2. 不一定需要把所有车都移一遍，但只有移动前后处于不同车位的车才能得到对应的金额。
3. 可以把自己的车开到附近的空地上（那里足以停下他所有的车）作为临时中转。因此车不必依次移动，每辆车也可以多次移动。但请注意：所有移动结束之后，每个车位最多只能停一辆车。
4. 和任务一相同的是：仍不许动其他人的车，且每辆车在移动之前和所有移动结束后所在的车位仍必须属于不同好友。特别地，不许最终把一辆车停到用于中转的那个空地上。

请帮助小明获得最大的总金额。

输入格式

第 1 行包含一个整数 T，即任务编号（T=1 或 2）。

第 2 行包含两个整数 N、K，分别表示小明的好友数和每个好友的车位数。以下 N 行每行有一个包含 K 个字符的字符串，描述每个好友当前车位的状态。其中 ‘#’ 表示该车位停放的是小明的车； ‘*’ 表示该车位停放的是其它人的车； ‘.’ 表示该车位是空车位。

如果 $T=1$ ，输入到此结束；否则接下来的 N 行每行包含 K 个不超过 100 的整数，分别表示每个车位对应的金额。

输入据至少有一辆小明的车。

输出格式

如果 $T=1$ ，输出任意可行的移车方案。假如有 M 个步骤，则第一行输出 M ，紧接着 M 行表示 M 步的具体内容。格式为： $(x1, y1) \rightarrow (x2, y2)$ ，表示把第 $x1$ 个好友第 $y1$ 个车位上的车移到第 $x2$ 个好友的第 $y2$ 个车位。如果无解，输出 -1。

输入 $T=2$ ，输出一个数字，表示最多的金额总数。

样例输入 1

```
1
3 4
###.
###
****
```

样例输出 1

```
3
(0, 0) → (1, 0)
(1, 1) → (0, 3)
(0, 1) → (1, 1)
```

35. 样例输入 2

```
36. 1
37. 3 4
```


38. ##*#

39. .#**

40. ****

41. 样例输出 2

42. -1

43. 样例输入 3

44. 2

45. 2 2

46. #*

47. #*

48. 2 1

49. 1 2

50. 样例输出 3

51. 3

对于规则一：

只需先将任意一个人的车移动到一个合法的空位处，然后将其他人的车移动到这个位置上，这样不断进行下去即可。如果某一时刻没有其他人的车可以放在这个空位而这个空位的拥有者还有其他的车子没有移动的话就尝试这个人的其他车子移到其它空位处。如果空位数量不足的话，那么将无法完成这个任务，输出无解-1。否则一定有解。

对于每一个人的车位，我们总是从左向右处理。这样只需将所有车位扫描一遍，复杂度 $O(NK)$ 。

对于规则二：

我们只需贪心地进行选取，将所有车位的金额从大到小排序，然后依次把可以放在这个位置的车放到这个位置来即可。

4.2. 关键思路:

首先询问数和一开始的初始值范围一样，要想到可以预处理后统一回答或者是离线。

如果设计状态 $f[i, j]$ 表示已经拨打到了字母树中的 i 节点，后面想拨打的号码为 j 的概率。这样的空间是开不下的

不妨用 $P[x][i]$ 表示在 x 的前 i 位数字已经拨了的情况下，

如果接下来准备要拨的数字是 x 的后 $6-i$ 位，其结果会拨到电话簿中任一电话的概率，用整数表示。这样就可以比较完美的解决此题。

初赛第五题：Sorry，打错了

5.1. 题目

龙先生是一位著名的记者，平时最喜欢报道一些鲜为人知的故事。最近，由于听说索马里海盗猖獗，他打算实地探访，做一个深入的调查。

龙先生联系了索马里当地的一些朋友，做了周密的计划——坐船从三亚出发，越过南海，趟过印度洋，最后到达索马里海域的亚丁湾。可就在船离海岸仅 10 公里时，突然一伙海盗突袭客船，所有人被劫持到了索马里城内。

在人质被运送到“海盗基地”的途中，龙先生凭着多年的经验，乘海盗不注意，跃下了卡车，在无数砰砰的枪声中，没命的向外跑去。跑了几分钟后，龙先生突然看到一个电话厅。他迅速向电话厅奔去，想打电话向住在索马里的朋友求助。然而，随着追赶脚步声的临近，龙先生估算留给自己打电话的时间最多 30 秒，决不容许拨错电话。

遗憾的是，越是这种危机的时候，越容易犯错。身为“智者”的您，请帮龙先生算一算：当他打电话给一位朋友的时候，恰好打给了另外一个人（不一定是他朋友）的概率是多少？已知索马里是一个有不超过 10 万人的小城，电话号码只有 6 位。

输入格式

第 1 行：索马里人数 n 。 $n \leq 100000$

接下来的 n 行：所有人的电话号码

接下来的 10 行：一个 10×10 错按表，表示按 a 键时按成 b 键的概率（第一行第一列表示按 0 时按成 0 的概率，第一行第二列表示按 0 时按成 1 的概率。所有的概率用整数 $0 \sim 10$ 表示，即实际概率要除以 10）

下一行：索马里城中你的朋友人数 m 。 $m \leq n$ 。

接下来的 m 行：每个朋友的电话号码。

输出格式

打给你的每一个朋友时候，打通其他人的电话的概率（每行一个概率，乘以 10^6 后用整数表示）

输入样例

```
5
267535
229127
693606
861879
902375
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
5
267535
229127
693606
861879
902375
```

输出样例

```
4
4
4
4
4
```

测试数据

共 10 个测试点，基本参数如下表：

测试点编号	n	m
1	10	5
2	50	25
3	300	100
4	2000	1000
5	8000	3000
6	30000	10000
7	50000	10000
8	100000	20000
9	100000	50000
10	100000	100000

初赛第六题：树形控件

6.1. 题目描述

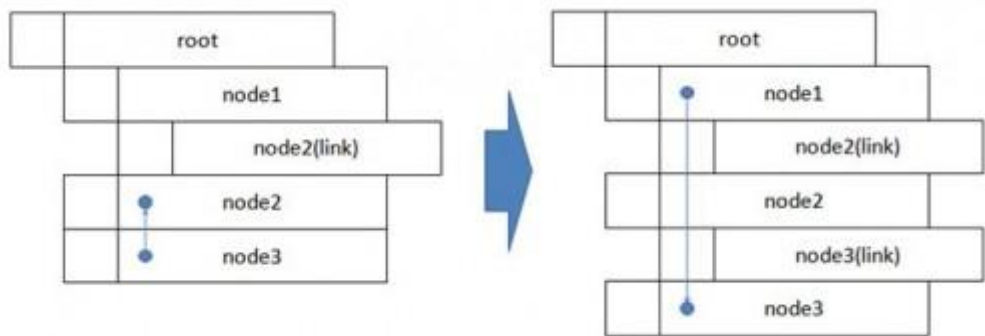
在本题中，你需要和一种通用的交互控件——树形控件打交道。树形控件中的结点分为两类：实体结点和链接结点，后者类似于 Linux 中的符号链接。

用户可以通过拖拽节点改变树的结构。用户在一个结点 A 上按下鼠标左键，移动到另外一个结点 B 上，松开鼠标，即可将结点 A 移动到结点 B 下面，作为 B 的最下方的儿子。如果改用鼠标右键拖动，表示结点 A 的位置不变，只是在结点 B 下创建一个到源结点 A 的链接结点。树中的所有实体节点始终是全部展开的，而链接结点只显示结点本身，不展开被链接的子树。

屏幕是一个矩形的区域，左上角是坐标原点 (0, 0)，x 轴正方向指向右，y 轴正方向指向下。树的每个结点的高度为 10 个像素，宽度为 50 个像素。子结点相对父结点要向右缩进 10 个像素。另外，每个结点前都有一个展开/折叠的控件，宽度和高度均为 10 像素。为了简单起见，这里所有的展开/折叠控件均处于失效状态，不响应与此相关的用户事件。

换句话说，要想把结点 A 拖拽到结点 B 下，点击和释放时，鼠标指针必须分别位于结点 A/结点 B 所属的 10*50 区域中（不能位于该矩形的四个边界上）。如果点击或释放时鼠标指针不在任何结点区域内，或者点击与释放时鼠标在同一个结点区域内，此操作应被认为是无效的（状态码为 1）。

即使控件能顺利检测到点击和释放时鼠标所处的结点 A 和结点 B，拖拽操作也可能是非法的（状态码为 2）。下图展示了两个连续的链接操作，其中第二个操作是非法的，因为这将导致树上形成一个环。其他非法操作包括：将任意结点拖拽到其后代结点（不管是移动还是链接）、尝试把任意结点拖拽到链接结点上。



如果操作完整，并且没有出现上述任何一种非法情况，则视为操作合法，状态码为 0。注意：可以将一个结点移动至其父结点下，尽管树结构不会发生任何变化。

请编写控件交互逻辑，输出每个用户操作的状态码。

输入格式

第 1 行是两个正整数 m, n，其中 m 为树的非叶结点个数，n 为操作次数。 1<= m, n <= 1000

以下 m 行是初始树的描述。每行的开头是一个非叶结点的名称，接下来是它的各个子结点名称。每个实体结点的名称都是唯一的，是一个由数字和小写字母组成的字符串（长度不超过 20）。如果一个结点是链接节点，其名称与源结点相同，其后用“(link)”标注（如果多次创建链接，最后也只有一个“(link)”而不是多个）。输入保证合法（没有环，链接结点没有子结点）。在任何时候，树上最多有 10000 个结点。

之后的 n 行描述交互动作，每行格式为：

Button x1 y1 x2 y2

例如，L 15 65 25 55 表示在 (15, 65) 单击左键并拖拽到 (25, 55)； R 15 65 25 55 表示在 (15, 65) 单击右键并拖拽到 (25, 55)；

输出格式

输出为 n 行，依次为每个输出的返回码。

样例输入

```
2 3
root node1 node2 node3
node1 node2(link)
L 25 25 25 15
R 25 45 25 35
R 25 15 25 55
```

样例输出

```
1
0
2
```

测试数据

共 20 个测试点，基本参数如下表：

测试点编号	m	n	备注
1	1	4	
2	1	2	
3	1	3	
4	3	6	
5	1	999	
6	2	4	
7	100	100	
8	500	10	
9	500	10	
10	500	100	
11	100	300	
12	150	400	
13	200	450	
14	250	500	
15	300	550	
16	400	600	
17	600	600	
18	800	800	
19	900	900	
20	1000	1000	

解出一个题目后请将解题报告按照原题面、解题源代码、测试数据用例和关键思路文字说明顺序编制成 Word 文档。每组题目完成后请从下一页开始一个新的题解文档。

解题源代码与测试数据用例见附件

6.2. 思路说明:

题目的背景是树形控件，不是很好理解题意，现阐述我的想法：

输入为一棵树，树的节点分为实体节点与链接节点(添加‘link’以示区别)两种，每种节点宽 10，长 50，节点头部有 10*0 大小的标识(无实际作用)。

然后父子节点之间有 10 长度的缩进。

现有两种操作 L x1 y1 x2 y2 与 R x1 y1 x2 y2，

L 操作为将覆盖(x1, y1) 的节点移至覆盖(x2, y2)的节点下方作为**最后一名**孩子。

R 操作为覆盖(x2, y2) 的节点添加指针指向覆盖(x1, y1)的节点。

操作结果有 3 种，分别为 0，1，2.

1 表示 x1 y1 或 x2 y2 不合法，即不再任何节点**内部**。

2 表示拖拽造成的结果不合法，分三种情况：

使树成环，将任意结点拖拽到其后代结点上（不管是移动还是链接）、尝试把任意结点拖拽到链接结点上。

0 即除了 1，2 外的情况。

输出为操作的结果。

典型的模拟题，码量应该算较大的。但算法性降低。

现在我们的主要任务就是

- 1，判断点的位置是否在节点区域内，即判定结果态 1
- 2，自然使用树形结构，构造树，代码中使用了 stl 的 list，简化部分操作。
3. 成环 节点 子节点 连接节点 这些均为树上的搜索，用 dfs 函数完成即可，即判断结果态 2
- 4，剩余情况即为结果态 0

题目重点考察了代码量，分类情况讨论多，选手在比赛时应沉着思考，冷静面对！

初赛第七题：交点覆盖

7.1. 题目描述

平面上有 N 条直线，至少有两直线不相互平行。

任务一：求一个周长最小的凸多边形，包围住所有直线的交点。

任务二：求一个周长最小的矩形，包围住所有直线的交点。

输入格式

第 1 行包含一个整数 T ，即任务编号 ($T=1$ 或 2)。

第 2 行为一个整数 N ，为直线的数目。接下来有 N 行，每行包含四个数，为一条直线上两点的坐标。

仅一行，为周长最小的矩形的周长，保留两位小数。

输出格式

如果 $T=1$ ，输出凸多边形的最小周长，保留两位小数；

如果 $T=2$ ，输出矩形的最小周长，保留两位小数。

三、测试用例

样例输入 1

```
1
4
0 0 1 0
0 0 0 1
1 1 1 0
1 1 0 1
```

样例输出 1

```
4.00
```

样例输入 2

```
2
3
0 2 3 0
3 0 3 2
3 1 0 2
```

样例输出 2

8.85

样例输入 3

2

6

2 0 0 9

6 5 3 8

9 0 7 1

9 4 0 3

6 2 2 6

0 5 2 8

样例输出 3

51.47

7.2. 解题思路

题意很简单，给 n 条直线，求包围这 n 条直线的所有交点的凸包。因为交点的数量级在 $O(n^2)$ ，而这题的 n 是 100000，所以朴素的写法是不能过大数据的。

高效算法的基本思路就是尽量先排除肯定不在凸包上的点，只考虑那些有可能在凸包上的点。首先我们考虑没有任何两条直线平行的情况。算法实际上很简单，先将直线按斜率排序，只要排成一圈即可，哪一条在最前面无所谓。设直接为 $\{L_1, L_2, \dots, L_n\}$ 然后，只需考虑所有相邻直线 L_i 与 L_{i+1} （包括 L_n 与 L_1 ）的交点即可，这样的交点共有 $O(n)$ 个，然后再用 $O(n \log n)$ 的算法求凸包。故总的复杂度为 $O(n \log n)$ 。

算法证明直观描述如下：显然，对于组成凸包的每条线段而言，所有的交点都在这条线段的同侧（或在这线段上）。分两种情况考虑凸包上的线段：

(1) 这线段是原有直线的一部分。因为所有交点都在其一侧，可以发现，这些直线在其另一侧是呈“放射状”的，故凸包线段两端的点，一定是该直线与最上和最下（也即斜率与该直线最接近的两条）直线的交点。

(2) 这线段不是原有直线的一部分。在这种情况下，线段的两端一定是某两条直线的交点，而这两条直线一定也是斜率相邻的。不然的话，若有一条直线的斜率在两者之间，则一定会有一个交点交到凸包这条线的另一侧。（这里感觉不好说清楚……）

最后再加入平行直线的情况。对于一组平行直线，容易发现只需考虑最“外侧”的两条即可。这样的话，原来的一条直线最多变成两条线，故原来的一个交点最多变成四个点，但这只是一个常数倍数，所以时间复杂度不变。

初赛第八题：我的地盘

8.1. 题目描述

百度公司的员工们在工作之余，经常以产品组为单位组织一些活动，包括吃大餐、春游秋游、公益活动、唱 KTV、看电影、体育比赛等。这些活动有一个专业的名字，叫做 team building，我们也亲切的称之为“bui”。



最近，地图产品组刚刚完成一个大项目，大家决定大 bui 一场。一阵七嘴八舌后，很多内容被提了出来，最终确定先打乒乓球，然后吃饭，最后 K 歌。问题是，谁也不知道有什么地方可以同时满足这三个需求。

不过没有什么问题可以难倒我们的工程师。很快，就有人写出了程序，为大家找到了合适的地点。地图覆盖之处，皆为我的地盘。你想挑战一下我们的工程师吗？想为我们找出更合适的地点吗？那就来吧。

输入格式

第 1 行是一个整数 k，表示某范围内所有的 POI (Point of Interest) 点数量，后续 k 行每行用 5 个字段描述一个 POI 点。它们的含义和格式如下表：内容数据格式数据范围

POI 编号 Int, 唯一 [0, 231-1]

POI 类型字符串 0-16 字节(不超过 15 种类型)

POI 级别 Int 0-5(越大表示越高级)

POI 经度 Double [0, 180], 6 位有效精度

POI 纬度 Double [0, 180], 6 位有效精度

需要注意的是这里的经纬度跟通常的经纬度范围是不一样的

随后是一个整数 n (0 < n ≤ 20)，表示共 n 组查询。以下 n 行，每行表示一组查询，格式为：

POI 类型 1 POI 类型 2 POI 类型 3 最低级别最高级别

分别表示三个 bui 地点各自的类型、最低级别和最高级别。

输出格式

对于每组查询” t1 t2 t3 min max”，输出三个 POI 编号 p1、p2、p3，满足：

- p1、p2、p3 的类型分别为 t1、t2 和 t3。
- p1、p2、p3 的级别不小于 min，不大于 max。
- p1、p2、p3 的两两欧几里得距离之和应尽量小。

输入数据保证至少存在一个解。

三、测试用例

样例输入

5

1 休闲娱乐 3 11.122843 12.431021

```
2 餐饮服务 2 13.384021 10.230425
3 旅游景点 3 12.234492 9.234268
4 休闲娱乐 5 20.242391 39.304233
5 教育机构 1 42.243292 67.232065
1
休闲娱乐餐饮服务旅游景点 0 5
```

样例输出

```
1 2 3
```

8.2. 解题思路

由于不要求最优解，因此不用全部遍历，使用贪心法寻找离当前已选择点集合最近的点。第一个 POI 选择所有可能的值，然后对每个可能的第一个 POI，选择离他最近的第二个 POI，选好第二个后，选择离前面两个距离之和最小的那个为第三个 POI，选好三个后计算总距离，如果小于前一个解则更新解，终结条件是找到所有可能的第一个 POI。

复赛第一题：高频 Query 的识别

内存限制:1MB

1.1. 题目描述

百度每天都会接受数亿的查询请求，如何在这么多的查询(Query)中找出高频的 Query 是一个不小的挑战。而你的任务则更加艰巨，你需要在极其有限的资源下来找出这些高频的 Query. (使用内存不得多于 1MB, 本题的规定覆盖其它地方的规定)

关于内存限制：我们评测程序计算内存使用量的方法是将选手程序实际使用的内存减去以下空程序实际使用的内存。

```
#include<stdio.h>
#include<iostream>
#include<string>
```

```
intmain() { }
```

注意，测试的机器是 64 位的. 在测试机上下面代码输出：88 4 2

```
#include<stdio.h>
intmain() {
printf( "%d %d %d\n", sizeof(long), sizeof(int*), sizeof(int), sizeof(short));
}
```

关于时间限制：下面是一个能得到正确输出的程序，你的程序使用的时间不能大于以下程序的运行时间（对相同的输入，g++ -O2 编译）。

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
intmain() {
constint L = 16+1;
charbuf[L];
charout[100][L];
while(scanf( "%s", buf)==1) {
intp = 0;
for(inti=0; i<200; i++)
p =rand()%100;
strcpy(out[p], buf);
}
for(inti=0; i<100; i++)
if(strlen(out[i])>0)
printf( "%s\n", out[i]);
}
```

输入格式

一行一个 Query，以文件结束符结尾。每个 Query 字节数 L（一个汉字两个字节）满足： $0 < L \leq 16$ 。输入大小不超过 1GB（包括换行符）。

输出格式

你认为最高频的 100 个 query。每行一个，不能有重复，不能多输出，但可以少输出（见样例）。

样例输入

美女
帅哥
美女
百度
美女
百度
百度

美女
美女
美女

样例输出
百度
美女

评分方法

如果你的程序运行超时或使用内存峰值超过限制, 那么你的得分为 0. 否则得分非 0. 你的得分是你输出的 query 的实际频次的总和 (在样例中为 3+6=9) 在所有有提交的选手中的排序而定.

具体来说, 设测试点分数为 S , 得分非 0 的程序数为 M , 比程序 i 的方案严格更优 (实际频次的总和更大) 的程序数为 $Y(i)$, 则该测试点程序 i 的分值为 $S(1-Y(i)/M)$. 换句话说, 输出该测试点最优解的程序将获得 S 分, 而最差解唯一的情况, 输出最差解 (但合法) 的选手将得到 S/M 分. 注意: 每个测试点的得分不必为整数.

提示, 请使用 C 语言的 `stdio` 函数而不要使用 `iostream`, 否则在 I/O 速度上会处于明显劣势.

1.2. 解题思路:

要求使用 1MB 以内的空间, 在只有一遍扫描的情况下, 尽可能找出出现次数在 TOP100 里面的 Query.

于是很显然, hash 是必须的了.

由于内存限制相当严格, 所以 hash 表只能开到很小, 比如, 103 (因为 103 是个素数).

在扫描的过程中, 为了限制内存的使用, 你需要按照某种规则不断地替换掉一些 entry.

在遍历完以后需要归并, 因为要求 TOP100, 所以每个 hash 值对应的 list 应该要有 100 个元素 (因为只要近似, 所以并不需要很严格).

然后你就发现, 这个东西怎么那么像计算机组成原理里面的多路组相联 Cache 阿....

那替换的规则几乎就不用想了, 找出出现次数最少的那个, 直接掐掉.

复赛第二题：图形检索

2.1. 题目描述

和人类一样，度度熊也喜欢上网搜美女的图片，不过和人类不同，他搜的是熊熊。他经常发现现有的搜索功能无法满足他的要求。看到喜欢的熊熊就喜欢狂搜那个熊熊的其他图片。

遗憾的是人类并没有给每个熊起个名字，他非常羡慕人类可以用“金泰熙生活照”这样精确的 Query 来进行图片搜索。有一天他终于受不了了，决定开发一个“度度熊”图片检索系统。目的就是从一张图片出发，检索和该图片相似的图片。

这可不是一件容易的事情，度度熊心里当然很清楚。因此他要先实现一个简化版的检索系统。具体的描述如下：

系统目标：给定任意一张包含特定图形的位图，从一系列候选集中，正确检索出与之相似的结果。

检索对象：黑白位图(即 0/1 bitmap), 尺寸统一为 180*180，其中每一个点 $P_{x,y}$ 保存像素点的颜色，1 表示黑色，0 表示白色。



输入格式

只有一个测试点，共 $X=150$ 幅图像，分为 $M=15$ 个类别，每个类别 $N=10$ 个样本。在人眼看来，每个类别中的样本两两相似，但任意两个不同类别的图像都不相似。

每幅图像(按照输入顺序依次编号为 $0..X-1$)包含 180 行，每行 180 个字符(为 0 或 1)，字符之间无空白。不同图像间用单个空行隔开。没有多余的输入(即保证只有 X 幅图像)

输出格式

共 X 行，其中第 i 行用 10 个整数描述图像 i 的相似图像的序号，按照相似度降序排列（即：越接近的图像越早出现）。

评分方法

对输入的每幅图像 i ，评测程序将给出一个原始得分 $R(i)$ ，然后计算出所有图像的总原始得分 $R = R(0) + R(1) + \dots + R(X-1)$ 。 R 越大，你的最终得分也越高，但具体分数还取决于其他选手的表现。

具体来说，设原始分数非 0 的程序数为 M ，比程序 i 严格更优（总原始得分 R 更大）的程序数为 $Y(i)$ ，则程序 i 的最终得分为本题总分的 $100(1 - Y(i)/M)\%$ 。换句话说，原始分数最大的程序将获得本题 100% 的分数，而最差解唯一的情况，输出最差解（但合法）的选手将得到 $100/M\%$ 的分数。注意：本题的最终得分不必为整数。

第 i 幅图像的原始得分 $R(i)$ 在很大程度上取决于检索出的正确图像个数 C 。具体来说，当 $C=0$ 时， $R(i)=0$ ，否则 $(C-1)^2 < R(i) \leq C^2$ 。当 C 相同时，设正确图像在输出序列中的编号分别为 T_0, T_1, \dots, T_{C-1} ，则 $T = T_0 + T_1 + \dots + T_{C-1}$ 越小越好（这里不提供具体公式）。

测试数据

本题只有一组输入。具体见“输入格式”的描述。

共 12 类别 * 6 样本 = 72 幅图像，以及相应的 0/1 矩阵的输入文件 `sample.in`，你可以使用该数据调试和测试你的程序。压缩包里有一个 `typerecord.txt` 文件。格式如下

0 113

表示 13.bmp 是第 0 个类别的第 1 个样本。

请注意样例数据和测试数据的图片个数是不同的。

2.2. 解题思路：

将图像抽象为几个特征值，比如 Trace 变换，图像哈希或者 Sift 特征向量等等，也可通过网络流来计算图像之间的差别完成聚类。

复赛第三题：网页的相似度计算

3.1. 题目描述

度度熊最近发现日本的互联网有很多恶意的作弊者用程序制造了大量的垃圾站点，分布在成千上万的主域上，这些站点初看起来感觉还行，但当度度熊发现几千个格式一模一样的站点，变化的只是其中的垃圾内容时，度度熊觉得实在作呕。例如下面这两个网站首页



他决定开发出一个工具，这个工具可以用于自动比较两个网页间格式的相似程度。

网页的 HTML 标签信息组成了一棵(有根的)DOM 树。TAG 名就是树的结点的标签。下面是一个例子：

```
<HTML>
<BODY>
<table>
<tr>
<td><img></img></td>
</tr>
<tr>
<td>
Welcome!
<br></br>
<a>ClickHere!</a>
</td>
```

```

<td><img></img></td>

</tr>

</table>

</BODY>

</HTML>

```

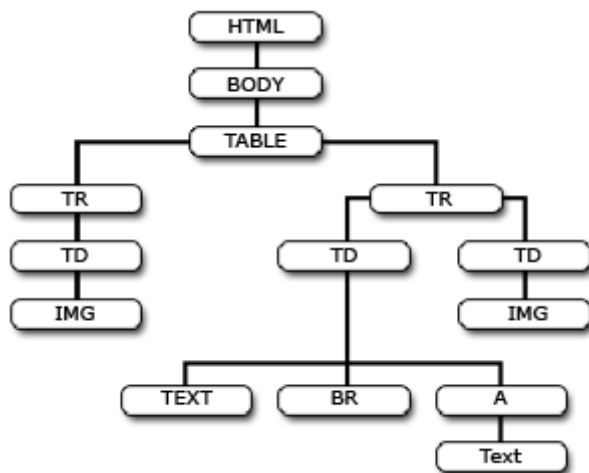


图 1

一棵”正则”的树是指结点数>1 的树.

两个 DOM 树相等当且仅当根的标签相等且各个儿子结点对应的子 DOM 树也都对应相等(各兄弟结点之间的顺序是重要的)

一个树的子树是指去掉某一结点与其父亲的边后留下来的以该结点为根的树.

如上图中以 TABLE 结点为根的子树是

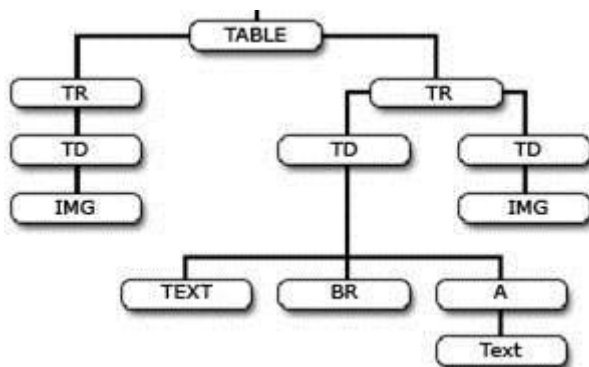


图 2

一个树的”根余子树”是指去掉一系列的子树, 但不去掉子树的根后余下的树.

例如图 1 去掉 2 个以 TR 为根的子树后的”根余子树”是：

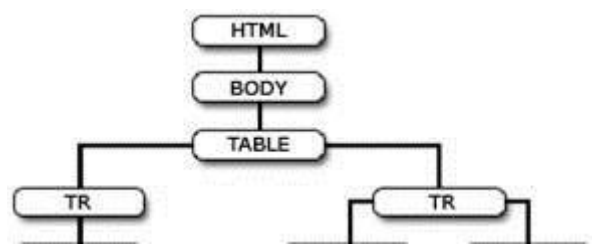


图 3

一个树的”广义子树”是指某个子树的”根余子树”，例如图 2 子树的一个”根余子树”如下：

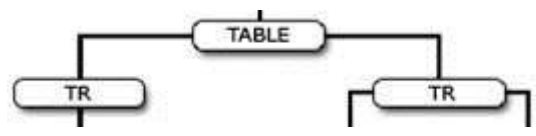


图 4

两个 DOM 树的相似度定义为这两个 DOM 树所拥有的共同的，”正则”的”广义子树”的数目。

你的任务是给出 N 个 HTML 网页，求出这 N 个网页的两两相似度 (共 $N*(N-1)/2$ 对). 并按相似度从大到小排序输出。

由于内容信息是不重要的. 故你可以把连续的文本当作一个单独的虚拟 PURE_TEXT 标签. 准确来说, 你必须将

```
<b>why why </b>
```

看作是

```
<a><PURE_TEXT></PURE_TEXT></a>.
```

但要注意. 连续的空白 (空格, TAB, 回车换行) 不是 TEXT, 空白只起到分隔符的作用. 如果网页中有 PURE_TEXT 标签, 则和上述虚拟的 PURE_TEXT 标签同等对待。

输入格式

第一行 N, 表示有 N 个 HTML. ($2 \leq N \leq 10$)

从第二行开始描述 N 个 HTML, 每个 HTML 的描述格式是

第一行 URL, 以 http:// 开头。每个 HTML 的 URL 均不相同。

以下为 HTML 源码, 直到一个以只有 4 个等号 (====) 的行表示结束。

为简单计, HTML 里的标签只有名字, 没有属性信息, 并且所有的标签都是成对出现的. 如
</br>.

标签名字不区分大小写 (且不一定是标准的 HTML 标签).
和
 是相同标签。

用正则表达式表示如下, 括号表示分组:

ID:=[a-zA-Z0-9]+

TEXT:=[a-zA-Z0-9<SPACE>]+

START_TAG:=<ID>

END_TAG:=</ID>

TREE:=EMPTY|TEXT|(START_TAGTREE END_TAG)

HTML:=<html>TREE</html>

其中<SPACE>指空白(空格, TAB, 回车换行等等, 具体的说, 是 c 语言中 isspace(c)为真的字符). 空白起到划分语法边界的作用, 因此可以出现在任意语法边界上. 如< / pre >是合法的. 但< / p re>则不是合法的. 输入的 HTML 保证合法, 我们不会刻意出数据来测试选手对语法的理解(数据都是由真实的网页数据而来).

但选手最好作一些简单的容错. 如果我们的数据不严格符合上面的正则表达式. 但是 95%的选手程序都会正确处理, 我们将认为数据是正常的. 比赛规则中保证输入一定为 LINUX 格式, 但此题不保证.

输入文件长度<=10MB. 总的标签数目(包括 PURE_TEXT)<=10000.

输出格式

共 $N*(N-1)/2$ 行.

每行格式为:

相似度<空格>对应的 URL1<空格>对应的 URL2

在同一行中, URL1 和 URL2 按字典序排序(即 URL1<URL2).

全部结果按相似度从大到小排序. 相似度相同的按 URL1 的字典序排序. 相似度保证小于 64 位有符号整数的最大值.

注意:我们的测试机器是 64 位, 故 long 可以保存得下相似度的结果. 请用 scanf(“%ld”, &x)或 cin>>x 来读取 long

样例输入

2

http://url_b/

<html>

<A>TEXT
T</br>TEXT

</html>

====

http://url_a/


```
<html>
<a>U</a><br>U</br>
</html>
====
```

样例输出

```
2http://url_a/ http://url_b/
```

测试数据

共 10 组测试数据，测试数据是真实的网页数据(经过必要的转换以满足上述的格式描述和规模)加部分手工数据。

3.2. 解题思路:

根据给出规则对两颗子树暴力枚举广义子树进行计算。

复赛第四题：拼车

4.1. 题目描述

虽然北京的公交系统很发达，但对上班一族来说，出租车仍是很常用的交通工具。度度熊就经常坐出租车上下班。有一天他发现一个现象：他的同事基本上都是每人单独坐一辆车，虽然有些人住在同一个方向，甚至同一个小区。他想，可以考虑让大家拼车，即节约出租车资源，又可以减少大家的支出。

度度熊作了以下假定：

出租车最多一辆可以坐 4 个人(为简单计，不考虑像度度熊这样因体积太大而坐不了 4 个人的情况)。

假设公路都是横平竖直的，可以用正方形框格来表示公路。相交的公路视为连通（可以在交叉点处从一条公路开到另一公路上）。车只可以在公路上走。

员工家都在公路的交叉点处。员工下班后，都从公司所在位置直接打的士回家。

每位员工只能乘坐一辆出租车到达目的地，不能中途下车，换乘其他出租车。

同往常一样，度度熊只喜欢出主意，却不喜欢写代码. 所以这个重任就只好落到你身上了:)

你的任务是写一个程序：求出一种员工打车方案，使得把所有员工送回家，在这种情况下，要使所有的士所走的路程尽量短。

为了方便起见，用笛卡尔坐标系来描述员工家所在区域。假定员工家所在区域的长和宽均为 9。

下面是公司所在位置为 (3, 3) 时，坐标描述图：

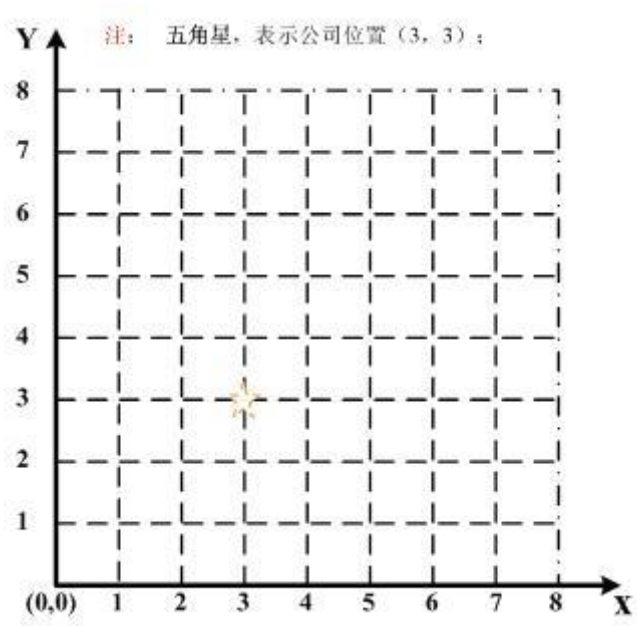


图 1. 公路示意图.

输入格式

输入包括：公司位置，员工总人数和每个员工家所在的位置。

具体格式见下：

第 1 行：有 1 个数 N ($0 < N < 100$)，表示当天需要打的回家的员工总人数。

第 2 至 $N+1$ 行，每行有 2 个整数，分别为 X_i ($0 \leq X_i \leq 8$) 和 Y_i ($0 \leq Y_i \leq 8$)，中间用空格隔开，表示员工 i 所要到达目的地的坐标。例如，第 2 行是员工 1 的家的坐标位置，第 3 行是员工 2 家的坐标位置，依次类推。不同员工的住址可能完全相同，员工的住址也可以和公司重合（可以在程序中直接忽略他）。

第 $N+2$ 行有 2 个数，分别为 M ($0 \leq M \leq 8$) 和 W ($0 \leq W \leq 8$)，中间用空格隔开，表示公司的坐标。

输出格式

输出包括：总的最短公里数，所需车的数目，以及每辆车所载的员工（用坐标表示）和每辆车所走的路线（用坐标表示）。

具体格式见下：

第 1 行：总的最短公里数。

第 2 行：所需车的总数目，假定是 C；

下面第 3 行至第 C+2 行，每一行输出包含三部分：

第一部分是该车所走路程；

第二部分是该车所载的员工，目的地坐标表示；

第三部分是该车所走的路线，用经过的交叉点坐标序列表示。

第一部分和第二部分之间用空格分割；

第二部分和第三部分之间用 “:” 分割。

样例输入

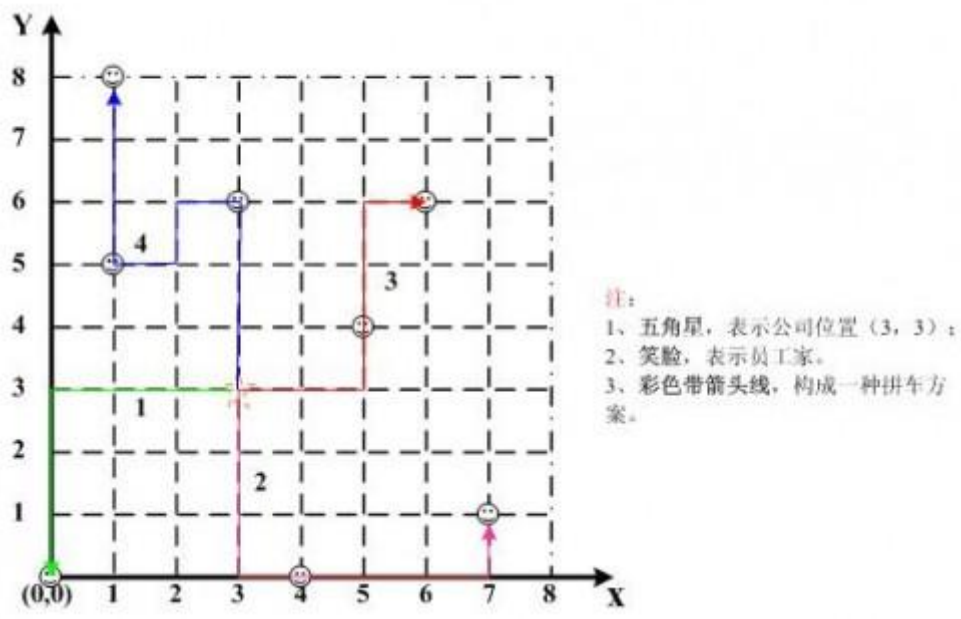
```
8
0 0
4 0
7 1
5 4
1 5
3 6
6 6
1 8
3 3
```

样例输出

```
29
4
6(0,0) : (3,3) (2,3) (1,3) (0,3) (0,2) (0,1) (0,0)
8(4,0) (7,1) : (3,3) (3,2) (3,1) (3,0) (4,0) (5,0) (6,0) (7,0) (7,1)
6(5,4) (6,6) : (3,3) (4,3) (5,3) (5,4) (5,5) (5,6) (6,6)
9(1,8) (1,5) (3,6) : (3,3) (3,4) (3,5) (3,6) (2,6) (2,5) (1,5) (1,6) (1,7) (1,8)
```

样例解释

输入的员工家、公司具体位置以及拼车方案见下图：



此例的一个打车方案为：

- 1 号线路程为 6 公里，载 1 人。
 - 2 号线路程为 8 公里，载 2 人。
 - 3 号线路程为 6 公里，载 2 人。
 - 4 号线路程为 9 公里，载 3 人。
- 总计最短路程为 29 公里。

评分方法

如果你的程序输出方案非法，那么你的得分为 0. 否则得分非 0. 你的得分是由你的方案得到的总的最短公里数在所有有提交且得分非 0 的选手中的排序而定.

具体来说，设测试点分数为 S ，得分非 0 的程序数为 M ，比程序 i 的方案严格更优(总的公里数更小)的程序数为 $Y(i)$ ，则该测试点程序 i 的分值为 $S(1-Y(i))/M$ 。换句话说，输出该测试点最优解的程序将获得 S 分，而最差解惟一的情况，输出最差解（但合法）的选手将得到 S/M 分。注意：每个测试点的得分不必为整数。

测试数据

共 25 组，员工人数近似在 $[0, 100]$ 中均匀分布，公司位置和每个员工家所在位置都是随机生成的。

4.2. 题目思路

尽管地图很小，但是 100 人次的数据仍然难以处理，本题目并没有一个确定性的最优解，但是可以构造一些可能的最优解。

考虑如下解法：

1. 对所有员工的家排序。
2. 对于排好序的序列，分配出租车和计算线路。

其中排序的步骤可以采取多种不同的排序方法，对于不同的测试样例，不同的排序算法的表现可能会不同。这里提供以下几种排序方式供选择：

1. 先按 x 坐标排序再按照 y 坐标排序；
2. 先按 y 坐标排序再按照 x 坐标排序；
3. 以公司坐标为原点对所有点进行极角排序；
4. 坐标轴倾斜 90 度排序；
5. 坐标轴倾斜 270 度排序。

排好序后处理排序后的序列，这里采取的策略是只有连续的一段人可以乘坐同一辆出租车。该子问题可以用动态规划来解决， $dp[i]$ 表示前 i 个人回家的最小距离，转移方程为 $dp[i] = \min(dp[I - j] + cost(I - j + 1, i), 1 \leq j \leq 4)$ 。其中 $cost(a, b)$ 表示第 a 个人到第 b 个人乘坐同一辆车回家的最小距离。

对于 $cost$ 子算法，可以枚举所有可能的排列，来选择把所有人都送到家的最小的花费。

复杂度上，排序算法时间复杂度为 $O(n \log n)$ ， dp 算法时间复杂度为 $O(n)$ ， $cost$ 函数时间复杂度近似 $O(1)$ ，整个问题时间复杂度为 $O(n \log n)$ 。

通过这种算法，可以求得一个近似的最优解。

决赛第一题：Spider

1.1. 题目

Spider 是百度用来抓取网页的程序的名称。面对信息几乎无穷无尽的互联网，如何快速，有效的抓取到优质的网页是一个艰巨的任务。

在本次任务中，我们以站点为单位，你的任务就是设计一个特殊的 Spider，这个 Spider 以站点为单位进行抓取，目标就是在有限的抓取次数下抓到尽量多的优质站点。

你的 Spider 所处的具体环境描述如下：

假设互联网上有 n 个站点，编号从 $0..n-1$ ，注意，编号是纯随机的。你有 m 次的抓取机会，一次抓取一个站点，抓取完成后，你将被告知该站点是否是高质量站点 (GOOD/BAD)，并且告诉你该站点指出去的所有站点的 ID，站点 A 指向 B 当且仅当 A 站点上某个页面里面包含有“指向 B 站点某个页面”的链接。

下面是一个样例程序：

```
#include "spider.h" // 所需函数的声明
// #include // 不要输出任何东西，否则得 0 分
int rand_long() {
    return abs((rand() < 16) + rand());
}
int main() {
    int n, m;
    spider_init(n, m); // n: 站点数, m: 你能抓取的最大次数
    vector crawled(n, false);
    while (1) {
        int k = rand_long() % n;
        if (crawled[k]) continue;
        vector link_out;
        // 所有站点 k 指出去的站点都在 link_out 中
        int ret = spider_crawl(k, link_out);
        // < 0 表示你的抓取行为非法 (如 id < 0) 或者已经到达最大抓取次数
        // = 0 表示站点 k 为 BAD, = 1 表示 GOOD
        if (ret < 0)
            break;
        /* 调试代码
        string t = spider_get_name_by_id(k); // 该函数在系统测试时不可用
        if (t != "")
            printf("%s\n", t.c_str());
        */
    }
    // 抓取完毕，系统计分
    spider_done();
}
```

运行时间限制为 3 分钟 (最终测试数据)，赛时评测的时限为 30 秒 (数据量较小)。

实际使用的测试数据约有千万级别的站点，样例数据是随机抽取出来的 300 万个站点的链接子图，并且同一个站点的 ID 号与实际测试数据是不一样的。

评分方式

设站点总数为 n ，则最多允许抓取 $m = n/8$ 次，假设当程序分别进行了 $m/8, m/4, m/2, m$ 次抓取后抓到的 GOOD 站点数分别为 g_1, g_2, g_3, g_4 ，则最终的得分为 $g_1 * 8 + g_2 * 4 + g_3 * 2 + g_4$ ，重复抓取的只计一次。选手可随时提交代码，并选择代码进行测试 (数据规模小于实际数据但大于提供给选手的样例数据)，但测试的次数限制为每个选手最多 8 次。在本机选手可任意使用样例数据进行测试。选手的最终得分依据最后一次提交在最终测试数据集上测试所得的分数。

注意：DON'T CHEAT。(例如对文件系统进行读写或 HACK，以及其它明显违反比赛公平性的行为)，我们会人工检查前几名选手的代码。另外，测试时使用的数据格式和处理方式和提供的样例是不一样的。实际测试时程序流程如下：

初始化，把需要的数据全部 load 进内存 (<10 秒)

每个函数调用使用的时间与返回的数据量成正比。并且全部是内存操作

时空限制

选手内存限制使用 500MB (不包括系统使用的内存)

背景知识

一般来说，互联网上的链接代表一种“推荐”的关系，如果 A 发出一条链接指向 B，则可以认为 A 在某种程度上推荐了 B。但是由于作弊者的存在，使得实际的情况远比单纯的推荐来得复杂，作弊者经常制造大量的所谓的 link farm，即一堆 BAD 的结点相互链接在一起。有一个直观的简单想法认为，一个 GOOD 的

结点一般指向 GOOD 的结点，而一个 BAD 的结点既有可能指向 GOOD 也有可能指向 BAD，被很多 GOOD 指向的结点很有可能是 VERY GOOD 的，然而，实际的情况仍然要比这种观点复杂得多。

1.2. 解题思路：

开放性题目。

所有的节点列出被 good 指向和 bad 指向的次数。还有完全没有被爬到的节点，根据这些信息构造参数进行选择。

决赛第二题：限制性图灵测试，搜索引擎的非法用户检测

2.1. 题目

用户使用百度时会发生以下行为：

查询一个 Query，记为 QI，其中 I 是 0...99 的整数

点击一条搜索结果，记为 CI，其中 I 是 1...10 的整数

点击”下一页”链接进行翻页，记为 N

点击”相关检索”中的结果，记为 R

例如用户查询了一个 Query “美女”，点了 1, 2, 3, 10 条结果，然后翻页，然后再点了 3, 4 条结果，然后点了相关检索 “帅哥”，然后

又点了 1, 2, 3 条结果，然后离开了。这个用户的行为可以表示如下：

Q1 C1 C2 C3 C10N C3 C4 R C1 C2 C3

由于各种各样的原因，并不是每个用户都是人在做这个动作序列的，例如有些人会写程序会在百度上做以下事情：检索 “sex”，翻页，翻页，点第 3 条结果，如此不断重复。这个怪异的”用户”的行为可以表示如下：

Q1 N N C3 Q1 N NC3 Q1 N N C3 Q1 N N C3 Q1 N N C3

当你面对着数亿的正常用户和不计其数的”机器人”时，你会怎么做呢？

如果你是程序员

给出搜索引擎一个用户行为序列，这个序列是在较短时间（如分钟级）内完成的，你的任务是写一个程序，判断其是否是正常的用户行为。如果是，输出 yes，否则输出 no

输入样例

Q1 C2 Q1 C2 Q1C2 Q1 C2 Q1 C2 Q1 C2

Q1 C1 Q2 C2 Q3C3

输出样例

no

yes

如果你是测试者

你的任务是给出一系列的测试数据（最多 20 组），每一行的格式如下：

answerbehavior_sequence # comment

其中 answer 是 yes 或 no，behavior_sequence 是用户行为序列片断，不超过 64 个元素，每个元素格式如题目开头所述，必须是 QI，CI，N，R 中的一个。comment 是诠释，可选。例如：

yes Q1 C1 Q2 C1Q3 C1 # a normal user

no Q1 C1 Q1 C1Q1 C1

注意 测试者给的 answer 只是参考答案，请诚实给出，我们会对所有的答案进行检查并进行必要修正，如果有明显非诚实的行为，将取消该测试者的所有测试数据。测试者提供的测试数据中，要求参考答案

为 yes 和 no 的数据数目一样多。请所有测试者在 30 分钟内提交 4 组测试数据，这 4 组数据将作为测试者提交的前 4 组测试数据。我们会尽快把所有提交打包发送给 15 名程序员

计分方法

如果程序对测试数据输出的答案和测试者提供的答案一样，令 15 名程序员中得到正确结果的个数为 C ，那么程序在此测试点得 $1/C$ 分。如果程序对测试数据输出的答案和测试者提供的答案不一样，测试者得 $1/R$ 分，其中 R 为该选手的本场比赛排名。总分为所有测试数据的得分和。

2.2. 解题思路：

开放性题目。

可通过给出的一些数据，通过机器学习等方法提取信息特征构造神经网络进行判断。

百度之星 2010

复赛第一场第一题：A+B 问题

1.1. 题目描述：

Suzumiya 最近开始无端刁难她的同学 ViVo，总是莫名其妙的问他一些简单而又离谱、没有实际意义的数学问题，比如三千加上五百等于多少。回答一次两次还可以，但不断这样的纠缠致使 ViVo 已经无法忍受了。所以 ViVo 决定制作一个语音装置来自动回答 Suzumiya 提出的无聊问题。

ViVo 知道你是个伟大的算法艺术家，所以希望该装置核心的数学计算模块你能够来帮忙。装置接收到的语音会自动为你转化为对应的中文字符串，经过你的模块处理完成后输出中文字符串，传递给装置来朗读出来。

为了给你带来方便，ViVo 已经提前收集好了 Suzumiya 可能会问到的问题，发现这些问题中大部分是数学加法，也还有一些不是加法的问题，但答案依然都可以用数字来表示。

输入数据：

输入的第一行是一个数字 n ，表示接下来有 n 个问题，每个问题占一行。

2

一加一等于几？

三千加上五百等于多少？

输出数据：

每行包含一个没有语病的中文表示最终的结果。

二

三千五百

时间限制：

5000ms

评分标准：

程序输出结果是否正确。

1.2. 关键思路

由题意可把题目分为三个子问题：

1) 把两个中文数字转化为阿拉伯数字 a, b

2) 求 $a+b$

3) 把 $a+b$ 转化为中文数字

故我们可以写两个函数，分别用作中文数字转化为阿拉伯数字、阿拉伯数字转化为中文数字。

由于题目并没有给出中文数字的读法规则，故我根据网上一些资料以及平时的一些习惯，大概定了如下一些规则：

1) 每个数位必须由一个字词来表示，中文数位有：个，十，百，千，万，十万，百万，千万，亿，十亿，百亿，千亿，万亿，十万亿，百万亿，千万亿，亿亿……

2) 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 分别用 零，一，二，三，四，五，六，七，八，九来表示。

3) 两个数位间无论有多少个 0，都用一个“零”来表示。为了方便，这里规定只在 4 个数位以内才加“零”。

4) 以“一十”开头的数字，省略“一”。

中文数字转化为阿拉伯数字：

枚举每个数位，若找到这个数位，则把这个数字分为数位前与数位后两部分，再分别转化为阿拉伯数字，得到 x , y 。再根据数位的大小 $unit$ ，返回结果 $x * unit + y$ 。注意“十”的情况

若没找到任何数位，则表示这个是数字，直接枚举 0 ~ 9 的中文数字判定即可。

阿拉伯数字转化为中文数字：

枚举每个数位大小，若不小于这个数位，则把这个数字分为数位前与数位后两部分，再分别转化为中文数字，得到 x , y 。再根据中文数位 $unit$ ，返回结果 $x + unit + y$ 。注意数位间“零”的情况。

若没有找到“个”以上的数位，则表示这个是一位数字，直接返回相应的中文数字即可。

复赛第一场第二题：i-Doctor

2.1. 题目描述：

百度计划开发一个在线的健康专家系统，帮助用户足不出户就能初步了解自己所患的疾病，并以此向用户推荐适合的医院就诊。经过对海量数据的分析，百度提取出了若干表征疾病性状的特征，并把每种疾病是否符合某项特征都进行了标记，最终得到如下数据表格：

疾病名称	A_0	A_1	...	A_{n-1}
D_0	Yes	Yes	...	Probably
D_1	No	Probably	...	Yes
...
D_{m-1}	Probably	No	...	Yes

其中， D_0, D_1, \dots, D_{m-1} 表示疾病名称， A_0, A_1, \dots, A_{n-1} 表示疾病的特征。 m, n 均为正整数且 $m < 4096$ ， $n < 128$ 。特征的取值为 Yes（符合该项特征），Probably（有可能符合该项特征）或 No（不符合该项特征）。

这个专家系统被命名为 i-Doctor，因为它的工作方式很人性化，就像医院的专家一样通过与病人的一问一答来得出诊断。每当开始一个诊断时，i-Doctor 首先提问：“你是否感觉到有 A 症状？”其中，A 为一疾病特征。用户依据自己的感觉回答。不幸的是，有时候病人对自己是否有 A 症状不能肯定，甚至会给出错误的回答。统计表明，病人的回答及置信度如下

用户回答	含义	置信度
Yes	是	0.95
Probably yes	很可能是	0.80
Probably no	不像是	0.80
No	否	0.95
Don't know	不知道	-

注意：每个病人在诊断之前患有一种（且仅一种）确定的疾病，且该疾病保证存在于上述疾病数据库中。

现在，请你编写一个程序来让 i-Doctor 开始工作。

交互：

你的程序应当读写标准输入输出，以便与测试库交互。交互方式如下：

首先，你的程序（doctor）应从标准输入读取疾病特征表。第一行是两个正整数 m 和 n ，表示疾病的种类数和特征的种类数。 m 和 n 之间以一个空格 隔开。接下来共有 m 行，其中每一行描述一种疾病，格式为：

疾病名称 特征值 0 特征值 1 ... 特征值 $n-1$

开头的字符串为疾病名称，长度不超过 7 字节；一个空格之后依次是各特征值，取值为英文字母 Y 或 N 或 P，分别表示 Yes、No 和 Probably。相邻特征值以一个空格隔开。

接下来，你的程序可以向病人（patient）提问，提问方式为在标准输出上打印一行，格式为：Do you feel A_i ? 其中 A_i 表示特征特 i 。此后，你的程序应当从标准输入读取病人的回答。病人每次的回答也为一行，内容为 Yes、Probably yes、Probably no、No 和 Don't know 之一。

如此一问一答，直到你的程序诊断出病人所患疾病，此时应在标准输出上打印一行：I think of D_j ! 其中 D_j 为此疾病名称

如果无法确诊，你的程序可以在标准输出上打印一行：Give up. 表示你放弃诊断该病人。

注意：很快你将看到，放弃诊断总比错误的诊断要好。

在确诊或放弃后，你的程序应自行终止。

程序举例：

下面是一个示例程序（省略了 include），它能正确的与测试库进行交互，尽管它的得分可能不高：

```
int main()
{
    int m, n;
    char name[10], line[256];
    char table[4096][128][2];
    int i, j, q;
    srand(time(0));
    scanf( "%d %d\n" , &m, &n);
    for(i = 0; i < m; i++)
    {
        2scanf( "%s" , name);
        for(j = 0; j < n; j++)
            scanf( "%s" , table[j]);
        fgetc(stdin); /* consume '\n' */
    }
    for(q = 0; q < 4; q++)
    {
        printf( "Do you feel A%d?\n" , rand() % n);
        fflush(stdout); /* Important */
        fgets(line, sizeof(line), stdin);
    }
    if(rand() % 3 == 0)
        printf( "I think of D%d!\n" , rand() % m);
    else
        printf( "Give up.\n" );
    fflush(stdout); /* Important */
    return 0;
}
```

注意，

你的程序应当像上面的程序一样，在每次输出后立即执行 `fflush(stdout)` 语句，使输出的内容立即被送入测试库（而不是留在输出缓冲区中）。另外，你的程序应能独立编译，不能依赖于任何其他外部文件（包括测试库）。

时间限制：

3000ms

2.2. 关键思路

题目大意是有 m 种病， n 种特征，每种病有一些特征。现在有一个病人，你可以问他有没有一些特征，他的回答有 5 种，分别是肯定（是否）和也许（也许是否）以及不知道，肯定和也许的置信概率分别为 0.95, 0.8，然后通过询问得出结论这个病人患了何种病。我的做法是问出他所有的特征回答，然后根据每种病的特征和回答进行概率乘法，然后找到最大

的概率值。

复赛第一场第三题：url 规范化

3.1. 题目

题目描述：

互联网上很多不同 url 都是指向同一页面的，比如：

<http://tieba.baidu.com/f?kw=%C9%CF%BA%A3%CA%C0%B2%A9%BB%E1>

http://tieba.baidu.com/f?kw=%C9%CF%BA%A3%CA%C0%B2%A9%BB%E1&fr=tb0_search

[http://tieba.baidu.com/f?kw=%C9% ... search&ie=utf-8](http://tieba.baidu.com/f?kw=%C9%...search&ie=utf-8)

都指向的是同一页面：<http://tieba.baidu.com/f?kw=%C9%CF%BA%A3%CA%C0%B2%A9%BB%E1>。

我们需要把不同形式的 url 规范化，用统一的 url 代替。目前已经定义了一批规范化规则，你的任务是把一些待处理的 url 按规则替换成新的 url。

每条规则都是一个字符串。为了方便理解，我们定义如下术语：

通配符：字符 ‘#’ 或者 ‘*’，以及紧跟其后的长度限定符（可选）。其中 ‘#’ 匹配数字，‘*’ 匹配任意字符。

长度限定符：紧跟通配符后，格式为 [min-max]，表示该通配符匹配的字符个数的最小值和最大值，其中 min 和 max 均可以省略（但不能同时省略），当 min=max 时可以简写为 [min]。‘#’ 的默认最小长度为 1，‘*’ 为 0。二者的默认最大长度均为 无穷大。例如，”#[3]“表示 3 个连续数字，’*[3-9]‘表示 3 到 9 个任意字符，’*[-8]‘表示 0-8 个任意字符，’#[7-]‘表示至少七个连续数字。注意，当通配符的后一个字符为左方括号 ‘[’ 时，它总是应当被看作长度限定符的开始标志。

尖括号：成对的小于符号 “<” 和大于符号 “>”。尖括号总是两两配对，不存在无法配对的孤立尖括号，且尖括号内部不会出现通配符，也不会嵌套另一对尖括号。

规则片段：在处理规则之前，我们把规则分成若干个连续的片段，每个片段是单个通配符（以及紧随其后的长度限定符，如果有的话）、一对尖括号（以及括号内的字符，如果有的话），或者若干个连续的普通字符（不含通配符和尖括号）。为了避免歧义，规则应当被划分成最少 的片段。不难证明，此时的划分方法是惟一的。

例如，规则 “abc*d#[4-7]eeef<gh>i” 包含 7 个规则片段：abc、*、d、#[4-7]、 eeef、<gh>、i。

用某一条规则处理 url 时，我们遵循以下步骤：

第一步：匹配。首先，忽略含有尖括号的规则片段，用其他片段来匹配 url，使得每个片段所匹配的字符串从左到右 连接之后和该 url 完全相等。注意，连接顺序必须按照规则片段的顺序从左到右进行。根据定义，由普通字符组成的片段只能匹配和它完全相同的字符串，而通配符可以匹配多样化的字符串，规则如前所述。例如，abc<x>def*[3-5]ghi 在忽略尖括号后得到四个片段：abc、def、* [3-5]、ghi，因此，abc<x>def*[3-5]ghi 能匹配到的 url 是那些以 abcdef 开头，ghi 结尾，中间有 3 到 5 个字符的字符串。

如果无法匹配，说明该规则不适用于此 url，处理结束；但有时会出现匹配方式不止一种的情况，还需要消除歧义。例如 *bc<d>*<xyz> 匹配 abcabca 的方法有两种：

规则片段	*	bc	<d>	*	<xyz>
方案一中匹配的url片段	a	bc	无（匹配时忽略尖括号所在片段）	abca	无（匹配时忽略尖括号所在片段）
方案二中匹配的url片段	abca	bc	无（匹配时忽略尖括号所在片段）	a	无（匹配时忽略尖括号所在片段）

在本题中，如果出现像这样匹配方法不惟一的情况，你应当选择让从左到右第一个通配符匹配字符数最少的方案。如果还有多种方案，再选择其中让第二个通配符匹配字符串最少的方案，以此类推。 在上面的例子中，应选择第一种方案。

第二步：替换。对于每对尖括号，把它左边相邻的 **url 片段** 替换为尖括号内的 文本，然后连接所有 url 片段（输入数据保证规则不以左尖括号开头，并且相邻两对尖括号之间至少包含一个字符）。

我们仍然借助上例说明问题：

也就是说，” abcabca” 被规则” *bc<d>*<xyz>” 处理后变成了” adxyz” 。

再举一个实际的例子：*/viewthread.php?tid=#<>*<> 表示将 url 文件名为 viewthread.php，第一个参数名为 tid 且参数值为数字时，把其它参数删除。 换句话说，这条规则将把 www.baidu.com/dir/viewthread.php?tid=123&arg=aa 替换为 www.baidu.com/dir/viewthread.php?tid=123。

输入格式：

第一行为一个正整数 M，说明共有 M 条规则。第二行为一个正整数 N，说明共有 N 条 url。接下来 M 行，每行代表一条规则，不超过 256 个字符。 再接下来 N 行，每行代表一条 url。所有 URL 都去掉了 http:// 头，且不含#*<>四种特殊字符，也不超过 256 个字符。M≤1000，N≤10000，输入的 url 和规则均不含汉字。50%的数据保证 N≤100，90%的数据保证 N≤1000。

输出格式：

输出共 N 行。对每一行输入的 url，按照输入顺序依次考虑所有规则，直到某个规则修改 url 时输出替换后的新 url（如果匹配成功，但修改前后 url 并没有发生变化，不算修改，应继续尝试剩下的规则），然后忽略剩下的规则。如果没有匹配的规则，则把输入的 url 按照原样重新输出。

输入数据：

```
5
7
blog.sina.com.cn/s/blog_*.html~type=<>*<>
you.video.sina.com.cn/*[0]falsepage/<>#<>.html<>
club.*<club>.sina.com.cn/*[1]*
www.sitea.com.cn:#[2-4]<80>/*
blog.sina.com.cn/s/circleinfo_<blog.sina.com.cn/>*<>.html<>
blog.sina.com.cn/s/blog_4ab97cd40100hwx9.html?t=j1~type=v5_one&label=rela_prevar
ticle
you.video.sina.com.cn/falsepage/123456789.html
club.edu.sina.com.cn/index.html
www.163.com/
blog.sina.com.cn/s/circleinfo_4859f1fa010004zx_1.html
www.sitea.com.cn:8080/a/index.asp
www.baidu.com/
```

输出数据：

```
blog.sina.com.cn/s/blog_4ab97cd40100hwx9.html?t=j1
you.video.sina.com.cn/
club.club.sina.com.cn/index.html
www.163.com/
blog.sina.com.cn/
www.sitea.com.cn:80/a/index.asp
```


3.2. 关键思路

这道题是一道典型的模拟类型的题目，需要按照题目要求的方法对规则进行解析，以及对 URL 进行操作。算法上没有太大难度，重点在数据结构的建立以及对题目描述的归纳实现。本题主要包含两个部分：规则的解析，和 URL 的处理。

在规则解析中，可以建立了三个保存规则的结构体分别表示规则、替换规则片段以及普通规则片段，其中规则包含两个列表，分别保存替换以及普通规则片段。在替换规则片段中，保存替换的字符串以及被替换普通片段的下标；在普通片段中则保存该片的类型（通配符/字符串），以及通配符的长度限定信息。通过扫描一次字符串，可以将读入的每条规则保存到上述的结构中，以便后面的匹配过程使用。

在 URL 处理过程中，利用上述结构中的规则数据，对每一条读入的 URL 进行匹配。匹配过程中将对应的几种规则片段进行分别考虑。当对应的规则片段为字符串时，则直接比较，若成功进入下一个片段，否则返回失败信息；当对应的规则片段为通配符时，则从小到大对通配符指代的字符串长度进行枚举，对每一种情况都进入下一片段继续匹配，当某种情况可以匹配到字符串末尾时，则返回成功。完成对应规则的匹配后，根据规则中附带的替换规则对匹配后的 URL 进行替换，最后输出结果。

这种方法在规则解析阶段中为线性时间复杂度，URL 处理过程中除了通配符会造成一些冗余外也为线性复杂度。当通配符较多时会对代码执行性能造成一定影响，但总体上还是十分高效的。空间上根据输入规则的不同进行动态分配，可以满足题目的要求。

复赛第一场第四题：并行修复

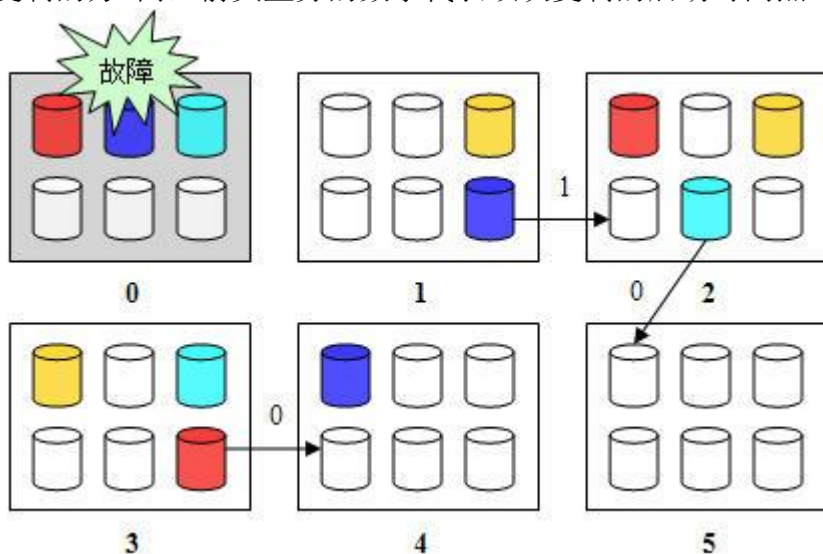
4.1. 题目

题目描述：

百度的网页按照某种标准均匀划分为 N 组，每组数据编号为 0 到 $N-1$ 的整数。一组网页可以全部存入一个数据库中。为了保证数据的可靠性，每一组网页按 X 份镜像存储，互为备份，且任两份镜像不会分布在同一台主机上。当有 F 台机器因故障而失效时，部分网页组对应的镜像数目可能会低于 X 份，此时需要复制镜像，使系统重新恢复到每组网页均有 X 份镜像的状态。

假设系统中一共有 M 台主机，主机编号从 0 到 $M-1$ 。每台主机的性能和网络带宽均相同。一台主机最多可以存储 T 个大小相同的数据库。每台主机任意时刻最多允许 C 个并发网络连接。一次复制操作占用源主机和目标主机各一个网络连接，源主机和目标主机不允许相同。若一对源主机和目标主机在同一时刻进行多次 ($B > 1$) 复制操作，则占用源主机和目标主机各 B 个网络连接。记一份镜像的一次复制用时为 1 单位。假定每次复制的启动时间点必须为非负整数，请设计一种修复算法，对于给定的某种镜像分布状态和失效的主机编号集合，输出一种恢复方案，使数据恢复的总用时尽可能短。输入集合保证系统能够从故障中恢复。

下图中给出了当 $N=4$, $X=3$, $M=6$, $T=6$, $C=1$ 时， 0 号主机故障时的一种恢复方案，箭头表示复制的方向，箭头盘旁的数字代表该次复制的启动时间点。



输入格式：

输入包含 $N+2$ 行。第一行包含五个正整数 N , X , M , T , C ，接下来的 N 行依次表示第 0 到 $N-1$ 组网页的镜像分布状态，每行有 X 列整数，为该组网页的镜像所在的主机编号。最后一行由 F 个整数组成，表示故障主机的编号（注意： F 本身不在输入中出现，你需要统计最后一行的整数个数，以获取 F 的值）。 $N \leq 40000$, $2 \leq X \leq 8$, $2 \leq M \leq 4000$, $T \leq 600$, $C \leq 4$, $F \leq 10$ 。

输出格式：

输出一个数据恢复方案。方案包含若干行，每行代表一次镜像复制操作，由四个整数 W , S , D , T 组成。 W 代表网页组编号， S 为源镜像所在主机编号， D 为目标镜像所在主机编号， T 为复制的启动时间点（起始时间为 0 ）。各复制操作应严格按照 T 从小到大排序。

输入数据:

4 3 6 6 1
0 2 3
0 1 4
0 2 3
1 2 3
0

输出数据:

0 3 4 0
2 2 5 0
1 1 2 1

该方案用了 2 单位时间 (耗时最多的主机编号为 2)。

时间限制:

5000ms

4. 2. 关键思路

Step 1: 注意到数据组数 N 虽然达到 40000, 但是损坏的机器只有 $F \leq 10$, 并且每个机子的数据容量 $T \leq 600$, 也就是说损坏的数据种类最多不超过 $F * T \leq 6000$, 其余的都是没有用的。

Step 2: 先考虑一个简化的问题, 即每种数据只损坏了一个, 我们现在要对每个有用的数据 (没有损坏的数据看作是没用的数据) 将它复制到其中一台没有损坏且不包含这个数据的机器上, 这个过程中必须保证不超过机器的数据容量。

假设我们替每个数据选定了一个待复制的机器, 并且确定了复制的源机器, 我们将源机器和目标机器连一条有向边, 那么问题就抽象为: 给一张带有重边的无向图 $\langle V, E \rangle$, 我们需要将它拆分成数量尽量少的子图 $\langle V, E_1 \rangle, \langle V, E_2 \rangle, \dots, \langle V, E_k \rangle$, 使得 $E_i \cap E_j = \emptyset$ ($i \neq j$), $E_1 \cup E_2 \cup \dots \cup E_k = E$, 且每个子图的度数不大于 C 。很明显, 原图的最大度数越小, 所分的期望子图个数就越少。

Step 3: 对于原本的问题, 假设我们得到了每组数据所需要复制到的机器集合, 并且我们得到了复制的每个目标机器和源机器, 如果我们忽略掉中间过程 ($A \rightarrow B \rightarrow C$, 我们假设 $A \rightarrow C$, $A \rightarrow B$ 也是合法的) 那么问题就和 Step 2 完全相同。但是我们无法忽略掉这个条件, 是否 step 2 的思路就不能用了? 我们注意到 $X \leq 8$, 说明每组数据最多复制 7 遍, 也就是说, 如果我们按照 step 2 的思路做的话, 简化问题的时间和原问题的时间相差不会超过 7, 所以我们的目标还是使得原图的最大度数最小。

Step4: 如果我们得到了数据 i 所要复制的机器集合 S 和为损坏的源机器集合 S' , 我们的目标是选择适当的边集 E' , 使得整张图的最大度数最小, S 中的任意点 x , 可以通过 E' 到达 S' 中至少一个点。我们使用贪心来解决这个问题: 将 S' 所有点标为蓝点, S 所有点标为白点, 每次我们从蓝点中取出一个度数最小的点, 并且取出任意一个白点, 然后将两个点连边, 更新度数并且将当前白点标为蓝点, 直到没有白点为止 (注意这里的度数代表的是原图的度数)。

Step5: 算法实现: 我们刚开始先选一个可行解 (可以先将每个数据需要复制的组数从大到小排序, 然后再将可复制的机器选出, 按容量从大到小排序取出前 Y 个 (Y 为当前数据需要复制的组数), 再将这些目标机器向任意一源机器连一条边), 然后进行随机调整。随机调整分为两种调整: 1、随机选一组数据, 删掉当前数据所有的边信息, 然后按照 step4 进行调整 (显然不同调整顺序会导致解的不同, 但是每次调整必然会导致最大度数的减少, 要不然

就回调整为调整前的状态即最大度数不变), 由于该调整会降低最大度数, 所以更新原图。2、随机选一组数据, 删掉当前数据所有的边信息, 然后随机选一个目标机器删除, 再选一个新的目标机器加入, 然后俺 step4 调整当前数据 (注意该调整可能会导致最大度数的增加), 最后比较调整完的最大度数和调整前的最大度数, 若变大则取消调整, 否则调整成功。

完。
由于本题解不唯一, 所以只给出一个参考解, 若有更好的思路欢迎补充。

复赛第一场第五题: 猜猜你在哪儿

5.1. 题目

题目描述:

有一个半径为 1 米的圆, 圆心位于 (0, 0) 点。你和圆处在同一平面上, 准备和这个圆一起玩游戏。这个圆每过一分钟会随机移动一次, 一共移动 t 次。圆有一个移动距离限制参数 s , 每次圆心从 (x_1, y_1) 移动到 (x_2, y_2) 时总是满足: $|x_1 - x_2| \leq s, |y_1 - y_2| \leq s$ 。

每次圆移动完成后, 你可以依次询问 k 个点是否在圆内, 然后任意走到一个新的位置, 结束这个回合。你的目标是尽量让自己位于圆内, 并且离圆心越近越好。每个回合结束后, 若你在圆的边界上或者圆外, 得 0 分; 若在圆内, 得分为 $(1 - \text{你到圆心的距离} / \text{圆的半径})$ 。所有 t 个回合结束后, 每个回合的平均得分就是你对于该测试点的原始得分。

交互:

你的程序应当读写标准输入输出, 以便与测试库交互。交互方式如下:

首先, 你的程序应从标准输入读入三个数 t, s, k , 分别表示总回合数、圆心移动的距离限制, 以及每回合你可以询问的点数。 $1 \leq t \leq 100, 0 < s < 1, 1 \leq k \leq 10$ 。

接下来, 你应当分 k 行给出各个询问点的坐标——每行两个数 x 和 y , 即询问坐标为 (x, y) 的点。每次询问一个点之后, 你的程序应当从标准输入读入一个数 a , 1 表示询问点在圆上或者圆的边界上, 0 表示在圆外。

再接下来, 你的程序应当往标准输出写两个数 x 和 y , 表示你要走到的新位置的坐标。

完成所有 t 个回合之后, 你的程序应当自行退出, 否则将按超时处理。

程序举例:

下面是一个示例程序 (省略了 include), 它能正确的与测试库进行交互, 尽管它的得分可能不高:

```
int main()
{
    int t, k;
    float s;
    scanf("%d%f%d", &t, &s, &k);
    for(int i = 0; i < t; i++)
    {
```

```

for(int j = 0; j < k; j++)
{
    int ans = 0;
    printf( "0.%d 0.%d\n", rand(), rand());
    fflush(stdout);
    scanf( "%d", &ans); // 1 表示仍然在圆里或圆的边界上，0 表示在圆外
}
printf( "0.%d 0.%d\n", rand(), rand());
}
}

```

注意，你的程序应当像上面的程序一样，在每次读取新的输入之前调用 `fflush(stdout)`，以确保这之前输出的内容立即被送入测试库（而不是留在输出缓冲区中）。另外，你的程序应能独立编译，不能依赖于任何其他外部文件（包括测试库）

时间限制：

1000ms

5.2. 关键思路

由于此题无正确解，且当 $k=1$ 时几乎不可做。根据此题采用相对评分，所以我们的目标是“比其他选手的得分要高”，而不是“最优”。由于时间紧迫，相对于其他题目来说，此题应投入较少时间。

普遍提交的“纯随机”做法在询问的 k 个点都不在圆内时，很容易就找不到圆了，以至于以后都无法找到圆，后面的轮数中得分均未 0。以下给出一种稍加修改的带阈值随机投点做法。相比于上述做法，在询问 k 个点均不在圆内时，由于我们知道圆心初始位置 $(0, 0)$ 以及圆心每次移动范围 s 为一个矩形，那么第 i 轮圆心在的位置便为 $i * s$ 的矩形内。由于 s 很小，所以在第 i 轮我们找不到圆时，下一轮我们采取在 $(i + 1) * s$ 的矩形内随机投点，试图找到圆大概的位置。而一旦找到圆的大概位置，我们再在新的圆心所在坐标差不超过 s 的矩形内寻找下一次移动的位置。

复赛第二场第一题：购物搜索调研

6.1. 题目

题目描述：

“有啊”是百度旗下的电子商务购物平台，每天都有上百万的用户在平台上搜索自己想要购买的商品。为了给用户更好的搜索体验，工程师们准备做一次搜索策略的调研工作。假设对于某一个特定的查询词来说，一共有 N 个相关商品，并且通过人工标注得到每个商品与查询词的相关度 $reli$ 。当用户输入这个查询词时，可以采取一种非常简单的响应策略，就是在 N 个相关商品中随机的找一段连续的商品子序列返回给用户。工程师认为，在每次返回结果中相关度的最小值代表了本次搜索结果的分数 $score$ 。

用户在搜索时，心中往往会对结果有个期望分数 $expect$ ，当搜索结果的分值 $score$ 高于用户的预期 $expect$ 时，该用户的满意度为 $score - expect$ ；否则用户的满意度为 0。假设在上述响应策略中，返回的子序列是完全随机的（即每段连续子序列被选中的概率完全相同），你的任务是计算每个用户满意度的数学期望。

输入格式：

输入第一行包含一个正整数 T ，表示测试数据的组数。每组测试数据的第一行为相关商品的数量 N ，第二行为 N 个整数，分别表示每个商品与查询词的相关度 $reli$ （用空格分隔）。第三行为一个正整数 M ，表示参与调研的用户数量，接下来的 M 行中的每一行都是一个整数 $expect$ ，表示该用户在搜索时预期的分数值。其中 $1 \leq T \leq 10$ ， $1 \leq N$ ， $M \leq 100,000$ ， $-109 \leq reli$ ， $expect \leq 109$ 。

输出格式：

对于每组测试数据，首先输出一行 “Case #:”，其中#表示测试数据编号（从 1 开始），注意空格与大小写。接下来的 M 行，按照输入顺序依次给出每个用户在这种响应策略下满意度的期望值，每个用户占一行。为了避免精度问题，期望值用最简分数 “A/B” 表示，其中 A 与 B 互质， B 是正整数。当结果是整数时，应只输出 A 。

样例输入：

```
3
3
1 3 5
1
2
4
5 -2 1 4
2
1
-1
3
2 4 10
1
0
```

样例输出：

```
Case 1:
5/6
Case 2:
7/10
3/2
Case 3:
4
```

样例说明：

在 Case 1 中，可能返回的子序列一共有 6 个 $[1]$ ， $[3]$ ， $[5]$ ， $[1\ 3]$ ， $[3\ 5]$ ， $[1\ 3\ 5]$ ，对应的分数（相关度最小值）为 1，3，5，1，3，1，用户满意度依次为 0，1，3，0，1，0。每个子序列被返回的概率相同，即为 $1/6$ 。根据数学期望的定义 可以得出该用户的满意度期望值是 $5/6$ 。

时间限制：

1000ms

6.2. 关键思路

题意比较明确，给一个长度为 n 的序列 a_i ，然后 M 次询问。每次询问是给出一个数 k ，然后在序列 A 里面随机取一段连续的序列，其中这个序列中的最小值为 x ，然后这个序列的得分是 $\max(x-k, 0)$ ，问得分的数学期望是多少。

总共可能产生的序列是 $n*(n+1)/2$ ，这个是要的数学期望的分母，关键是分子怎么求。我们计算将每一个 a_i 作为最小值的序列个数。首先可以通过单调栈处理出 a_i 左边第一个比 a_i 小的位置 li ，右边第一个比 a_i 小的位置 ri ，那么在任意序列左端点在 $[li+1, i]$ 之间，右端点在 $[i, ri-1]$ 之间的最小值都是 a_i ，个数是 $(i-li)*(ri-i)$ 。

但有一种特殊情况，就是 $[li+1, ri-1]$ 这段区间内不止一个数为 a_i ，也就是最小值不唯一。比如 3 2 4 2，在处理第一个 2 的时候会得到区间

$\{2\}, \{3, 2\}, \{2, 4\}, \{3, 2, 4\}, \{2, 4, 2\}, \{3, 2, 4, 2\}$ ，在处理第二个 2 的时候 $\{3, 2, 4, 2\}, \{2, 4, 2\}$ 这两个区间重复计算了。也就是说同一段区间里面有几个不同的最小值需要去重。

解决方法也很简单：假设当前处理区间为 $[l, r]$ ，最小值为 a_i ，位置分别为 $l < p_1 \leq p_2 \leq \dots \leq p_m < r$ ，处理 p_1 的时候依然是 $(p_1 - l)*(r - p_1)$ ，处理 p_2 的时候发现多出的部分为左端点在 $[p_1+1, p_2]$ ，右端点在 $[p_2, r-1]$ 的部分，因此去重后应当加 $(p_2 - p_1) * (r - p_2)$ ，后面都是同理了。这里需要处理一下每个数前一个和他相同的数的位置。

最后算答案的时候就是 $\sum (a_i - k) * g_i / (n*(n+1)/2)$ ($a_i \geq k$ ， g_i 为最小值为 a_i 的序列个数)，由于询问次数很多，考虑每次取的都是较大的几个 a_i ，排序后处理一下 a_i 的后缀和以及 $a_i * g_i$ 后缀和，然后根据查询的 k 二分位置给出答案就完成了。

复赛第二场第二题：内存碎片

7.1. 题目

题目描述：

在服务器上运行的模块，往往要连续运行几个月不重启。这期间，各种程序会不停地申请、释放内存，因而导致大量的内存碎片。例如，有一块大小为 4K 的内存，还有一块 5K 的内存，但是由于二者地址不连续，无法从中申请一块 6K 的连续内存。

解决这个问题方法之一是使用内存池，即把已释放的内存块链起来，而不直接还回操作系统，下一次申请这个大小的内存块的时候，直接从链表里获得内存即可。然而，这样做有个风险：用户可能在一段时间内只需要长度为 L 的内存，过一段时间后全部释放；接下来申请长度为 L' 的内存，造成长度为 L 的内存大量浪费。为了避免上述风险，一般的做法是：当程序申请长度为 L 的内存时，也可以给它分配一块长度为 L' ($L' > L$) 的更大的内存块，并且限制内存池中内存长度的种类。

给出 n 个内存块申请，你的任务是计算出在最好情况下，至少需要多少内存才能满足所有的需求。

注意：有的内存池允许将内存块串连起来，组成一个更大的内存块，但是串联需要使用指针，在管理小内存的时候比较浪费，所以本题规定内存块不许串连。

输入格式：

第一行为两个正整数 N 和 K ，其中 N 表示内存请求的数目， K 表示内存池中允许有 K 种不同的长度的内存块。以下 N 行每行包含两个正整数 L_i 和 C_i ，表示申请长度为 L_i 的内存块

C_i 次。 $N \leq 10000$, $K \leq 200$, $L_i < 1000000$, $C_i \leq 10000$ 。

输出格式:

输出仅一个整数，即在最好情况下，至少需要多少内存才能满足所有的需求。

样例输入:

3 2

10 1

11 1

20 1

样例输出:

42

样例解释:

长度为 11 的块两个，长度为 20 的一个。

时间限制:

1000ms

7.2. 关键思路

首先合并相同长度的内存碎片，然后根据题意很快就可以得到一个 DP 转移方程：

$F[N][K]$ 表示 前 N 个数分成 K 块的最小答案

$F[N][1] = \text{sum}(N) * \text{length}[N];$

$F[N][K] = \min(F[u][K-1] + (\text{sum}[N] - \text{sum}[u]) * \text{length}[N]);$

// $\text{length}[i]$ 表示从小到大排序之后，第 i 小的块的长度

// $\text{sum}[i]$ 表示排序后的块的数量的前缀和

但是这个 DP 是 $N*N*K$ 的，考虑优化。

将式子化简：

$F[N][K] = \min(F[u][K-1] - \text{sum}[u] * \text{length}[N]) + \text{sum}[N] * \text{length}[N];$

想一想 F 数组的意思，就会发现有着单调性，即 $F[u][K-1] - \text{sum}[u] * \text{length}[N]$ 这个东西是随着 u 的增加递减的。于是我们只需要直接将 $u=N-1$ 即可。

复赛第二场第三题：蜗牛

8.1. 题目

题目描述：

一只蜗牛某天早晨掉进了深为 L 尺的井中。蜗牛每天白天可以向上爬若干尺，晚上休息时会向下滑若干尺。蜗牛一旦到达井口或井底，便不再下滑。

假设蜗牛每天向上爬的尺数均为不超过 10 的正整数，而下滑的尺数为不超过 5 的正整数。

蜗牛在第 N 天白天里(含第 N 天白天结束时)爬出了井，你的任务是统计有多少种可能的爬升/下滑情况。对于两种爬升/下滑情况，当存在对应的白天上爬或者晚上下滑的尺数不同时，即视为不同的情况。

输入格式：

第一行：井深 L 。其中 L 为正整数，且 $L \leq 100$ ；

第二行：爬出的天数 N 。其中 N 为正整数，且 $N \leq 300$

输出格式：

输出一个正整数，为可能的爬升/下滑情况总数。如不可能在 N 天白天里(含第 N 天白天结束时)爬出深为 L 的井，则应输出 0。

样例：

输入：

27

3

输出：

6

解释：

输入指明井深为 27。蜗牛掉下去后，在第 3 天白天爬出了井。一共有 6 种可能的上升/下滑情况组合：

(9, -1) (10, -1) 10 8+9+10=27

(10, -1) (9, -1) 10 9+8+10=27

(10, -1) (10, -1) 9 9+9+ 9=27

(10, -1) (10, -1) 10 9+9+10>27 (第 3 天白天未结束时，爬出了井)

(10, -1) (10, -2) 10 9+8+10=27

(10, -2) (10, -1) 10 8+9+10=27

样例 2

输入：

5

4

输出：

5033

样例 3

输入：

42

12

输出：

3106744105061936231

时间限制:

1000ms

8.2. 关键思路

这道题大概是这次复赛中最水的题目，是一道简单的 dp。设状态 $f(i, j)$ 表示蜗牛用 i 天爬出深度为 j 的井的情况总数，注意到爬升和下滑的尺数分别只有 10 和 5，故而可以枚举爬升和下滑的尺数，从第 i 天的情况推出第 $i + 1$ 天的情况总数，在 $O(10 * 5 * N * L)$ 的复杂度下完成计算。

在枚举下滑的尺数时要注意，蜗牛下滑到井底后不会再下滑；此外最后一天一定是在白天爬出，所以只爬升不下滑。因为方案数比较大，这道题需要高精度运算。此外如果把 dp 写成记忆化搜索的形式，极端数据下(如 $L = 100, N = 300$)会爆栈，需要手动扩栈；在 G++ 和 VC++ 下扩栈方式不一致，这里不再多述。下面给出两种 dp 写法的标程。

复赛第二场第四题：午餐聚会

9.1. 题目

题目描述:

随着百度公司业务规模的不断扩大，员工人数数量激增。为适应互联网的快速变化，保持公司的灵活性，原来的一个部门被划分为好几个小组便于组织和管理。划分成几个小组后，基本保证了组内员工的协作性和灵活性，然而不同组之间的员工由于业务上无太多联系，致使组与组之间的员工相对生疏。为加强公司团队的整体性，促进不同组之间经验的分享，增进不同组之间员工的交流，公司决定在同一部门不同组之间开展“午餐聚会”活动。操作方法很简单：每次从每个组中各随机选出一名员工一起吃午饭。假设这个部门有 N 个组，那么每次参加午餐聚会的人数就为 N 。注意，如果这 N 个人的组合已经出现过一次，则需要重新随机选取。如果某次聚会后，部门中任意两个不同组的员工都曾一同参加过午餐聚会，表明认识、交流的目的已经达到，因此不再继续举行午餐聚会。由于每次参加聚会的人选是随机确定的，总的聚会次数可能很小，也可能很大。为了做预算，你需要计算出聚会次数的最小值和最大值，并给出最小值对应的方案。

输入格式:

第一行为一个正整数，即该部门的组数 N ($N \leq 6$)，接下来的 N 行，每行包含一个正整数 i ，代表该组的员工数 ($i \leq 10$)。

输出格式:

输出第一行包含两个数 \min 和 \max ，其中 \min 为最小聚会次数， \max 为最大聚会次数。以下 \min 行描述了聚会次数最小的方案，其中每行描述一次聚会，包含 n 个参加聚会的员工代号，从左到右依次代表第一组、第二组、第三组...的组员编号（每组的各组员编号为 0 到 $n-1$ ）。

样例输入:

```
4
2
```

2
2
2

样例输出:

5 13
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
1 1 1 1

时间限制:

5000ms

9.2. 关键思路

首先，本弱菜水平有限，无法给出完整解法，下面给出部分解法以及本人的一些想法。

题意：有 N ($N \leq 6$) 组人，每组 K_i ($K_i \leq 10$) 人。同组内的人互相认识，不同组的人只能通过聚会认识。每次聚会选 N 个人，每组 1 人。聚会之前这 N 个人不能都互相认识。聚会结束后这 N 个人便互相认识。当所有人互相认识后便不再举行聚会。现在要安排一些聚会，使得所有人互相认识，求聚会次数的最小值和最大值，并给出最小值时的方案。

关于最大值

首先给出最大值的答案：

$$\text{ans_max} = \max \left\{ \prod K_i - \prod_{i! = p \& \& i! = a} K_i + 1 \mid 1 \leq p < a \leq n \right\}$$

证明：

我们把第 i 组编号为 j 的人记做 i^j 。

假设最后一对互相认识的人是 (a^b, p^q) ，那么 (a^b, p^q) 能且只能在最后一次聚会出现。则，前面所有的聚会只要不出现 (a^b, p^q) ，即可任意组合。此时前面聚会次数的最大值等于 a^b 和 p^q 不同时出现的所有聚会方案数。那么

不出现 a^b 的聚会方案数： $(K_a - 1) \prod_{i! = a} K_i$

不出现 p^q 的聚会方案数： $(K_p - 1) \prod_{i! = p} K_i$

不出现 a^b 和 p^q 的聚会方案数： $(K_a - 1)(K_p - 1) \prod_{i! = p \& \& i! = a} K_i$

根据容斥原理： a^b 和 p^a 不同时出现的所有聚会方案数 =

$$\begin{aligned} & (K_a - 1) \prod_{i \neq a} K_i + (K_p - 1) \prod_{i \neq p} K_i - (K_a - 1)(K_p - 1) \prod_{i \neq p \& i \neq a} K_i \\ & = \prod K_i - \prod_{i \neq p \& i \neq a} K_i \end{aligned}$$

于是在 (a^b, p^a) 最后出现的前提下，有

$$\text{Ans_max} = \prod K_i - \prod_{i \neq p \& i \neq a} K_i + 1$$

因为 $\prod K_i$ 是定值，要使 Ans_max 最大，只需使 $\prod_{i \neq p \& i \neq a} K_i$ 最小，即 $K_a * K_p$ 最大即可。

关于最小值求解

复杂度简要分析

已知聚会的组合有 $\prod K_i$ 种可能，对于极限数据有 $S = \prod K_i = 10^6$ ，而 ans_min 的值（亦即搜索的深度 d ）有可能达到几十甚至更大，那么复杂度粗略估计就是 A_5^d ，所以必须要有强力的剪枝和优化。在这里我把数据分为两类处理。

- $N \leq 3$ 的情况，此时可直接构造解。

$N = 1$ 时， $\text{ans_min} = \text{ans_max} = 0$ ；

$N = 2$ 时， $\text{ans_min} = K_1 * K_2$ ；

$N = 3$ 时， $\text{ans_min} = \max \{K_i * K_j \mid 1 \leq i < j \leq 3\}$ ；

这里要讲一下关于 $N = 3$ 时最小聚会次数方案的构造。

不妨令 K_3 和 K_2 分别是最大值和次大值。我们先构造出第二组和第三组的所有组合 $S = \{ (2^i, 3^j) \mid 0 \leq i < K_2, 0 \leq j < K_3 \}$ ，然后把第一组加进来。

对于 1^0 ，我们从先前的组合 S 里面选出 K_3 个组合，使得这 K_3 个组合覆盖 $0 \cdots K_2 - 1$ ， $0 \cdots K_3 - 1$ ；然后对于 1^1 ，我们再从剩下的组合里面选出 K_3 个组合，使得这 K_3 个组合覆盖 $0 \cdots K_2 - 1$ ， $0 \cdots K_3 - 1$ ， \cdots 一直进行下去。由于 $K_1 * K_3 \leq K_2 * K_3$ ，所以 S 中可能还有剩余，此时把 1^0 和剩下的组合重新组合成三人组即可。

由于采用构造法，因此对于 $N \leq 3$ 的情况的解可以秒出。

- $N \geq 4$ 的情况

这种情况下，本代码做的很不好，没有想出较好的解法。

下面谈谈关于这种情况的一点想法

一次聚会的状态表示

因为 $K_i \leq 10$, 所以我们对每组人员按照 $0..K_i-1$ 编号, 那么每人的编号都是个位数, 因此可以用一个 N 位的十进制数表示一次聚会的组合。例如有 $N = 3, K_1 = 2, K_2=3, K_3=4$, 那么第一组选 0 号, 第二组 1 号, 第三组 2 号的一次聚会可以表示为 012。

一个极其暴力的做法

设总人数为 sum , 则对每个人从 $0-sum$ 开始编号。建立 $sum*sum$ 关系的矩阵, 相互认识用 1 表示, 不相互认识用 0 表示。先获取所有的聚会组合存放到 `state[]` 里面。搜索到每一层时枚举所有没用过的组合, 并更新关系矩阵, 接着继续递归搜索。若当前递归深度大于已知的最小值则剪枝; 当所有人都相互认识停止搜索, 更新最小解。

该方法纯暴力, 处理的数据规模很有限。

复赛第二场第五题：玉树驰援

10.1. 题目

题目描述：

4 小时以前：青海玉树发生强烈地震，大量房屋倒塌，人员伤亡惨重。灾区急需救援。

2 小时以前：政府迅速成立救灾指挥中心。全国各地情系玉树，纷纷筹集救援物资。

1 小时以前：指挥中心将若干与玉树有直接或间接道路交通的地区作为集散点。救援物资分批就近到达集散点后，再由汽车运进灾区。一批物资有一个唯一的编号，并由若干个集装箱组成。集散点已有不同数量的汽车在等候，指挥中心可以根据需要进行运输调度。汽车装卸物资的时间可忽略不计。一辆汽车一次最多只能运 1 个集装箱。

30 分钟以前：部分道路由于损坏和堵塞，汽车通行需要较长的时间；部分道路情况更加严重，已经不具备通车条件。指挥中心决定修复和疏通道路，使道路恢复通车，或缩短通行时间。施工需要一定的时间，而且要封闭道路，即在此过程中不能有汽车在该道路上行驶。足够数量的施工队已经待命，可以随时对多条道路进行施工。

现在（2010/4/14 11:49）：你被紧急征召到指挥中心。你的任务是调度物资运输和道路施工，在最短的时间内将所有物资运送到玉树灾区。

输入格式：

输入由三部分信息组成，依次为交通信息、汽车初始分布信息和物资就绪信息。相邻部分之间用一个空行（`\n`）隔开。

第一部分：玉树周边的交通信息。每行描述一条道路，格式如下：

地点 A 地点 B 该道路施工所需小时数 (t_0) 施工前行驶所需小时数 (t_1) 施工后行驶所需小时数 (t_2)

值间以一个空格隔开。道路总数不超过 256 条，任意两地间最多只有一条双向道路直达。地

点名为英文字母和数字组成的字符串，长度不超过 7 字节。玉树的地点名为” Yushu”。 t0, t1, t2 为整数。如果 t1 = -1, 表示该条道路在施工完成前无法通车；其它情况下, t0, t1, t2 均大于 0 且不大于 24。

一个空行之后，是第二部分：汽车初始分布信息。每行描述某个集散点初始时有多少汽车在等候，格式如下：

地点 数量

值间以一个空格隔开。集散点总数不超过 64，单个集散点汽车的初始数量不超过 1024。

一个空行之后，是第三部分：物资就绪信息。每行描述一批物资在何时何地就绪，格式如下：

yyyy/mm/dd HH:MM arrive 地点 物资编号 集装箱数量

值间用一个空格隔开。其中，yyyy/mm/dd HH:MM 的时间格式含义为：年/月/日 时:分。物资编号为英文字母和数字组成的字符串，长度不超过 7 字节。一批物资的集装箱数量不超过 2048。

最后一部分以文件结束符结尾。

输入保证所有物资可以在有限时间内运到玉树。

输出格式：

你的程序应当按时间顺序在标准输出上打印你下达执行的指令。每条指令一行（不超过 255 字节，以’ \n’ 结 尾），每行各值间用一个空格隔开。指令格式具体如下：

道路施工指令

yyyy/mm/dd HH:MM fix 地点 A 地点 B

该指令表示在指定时间开始对地点 A 和地点 B 的直达道路上进行施工。一条道路不得重复施工。

货物运输指令

yyyy/mm/dd HH:MM trans 地点 A 地点 B 物资编号:本指令运输的集装箱数量

该指令表示在指定时间开始将某种指定数量的物资从地点 A 运输到地点 B。地点 A 和地点 B 必须有直达道路。集装箱数量应为正整数。

空车移动指令

yyyy/mm/dd HH:MM move 地点 A 地点 B 本指令调动的汽车数量

该指令表示在指定时间开始将指定数量汽车从地点 A 调度到地点 B。地点 A 和地点 B 必须有直达道路。汽车数量应为正整数。

注意：所有指令涉及到的时间不得早于 2010/4/14 11:49（但可以相等），所有指令一旦下达就不能取消或中断。

样例输入：

Yushu A1 7 14 7

Yushu A2 2 -1 5

A2 A3 4 9 6

Yushu 3

A1 40

A2 5

A3 15

2010/4/14 11:59 arrive A2 G1 10

样例输出：

2010/4/14 11:49 fix Yushu A2

2010/4/14 11:49 move A3 A2 5

2010/4/14 13:59 trans A2 Yushu G1:5

2010/4/14 20:49 trans A2 Yushu G1:5

时间限制：

120000ms

10.2. 关键思路

本题并未要求给出最优解，而是依据各选手提交程序运行结果来综合给分，故在此给出一种可行解。

道路总数不超过 256 条，则点的总数也不超过 256 条，首先以玉树为源点做一遍 SPFA 求出每点到的最少时间，边的权值取 t_1 ，通过此时间来估计所有物资送达所需的最长时间，估计函数可以为：对每批物资有 $f(i) = C_i \cdot \text{dis}[c_src] / \max(1, \text{tot_car} / \text{tot_sup}) + \text{time}[i]$ ， C_i 为该物资的集装箱数， $\text{dis}[c_src]$ 为其降落点的到玉树的距离， tot_car 为汽车总数， tot_sup 为物资总数， $\text{time}[i]$ 为其到达时间，记录 max_f_pre 为 $\max\{f(i); 1 \leq i \leq \text{tot_sup}\}$ ；再以修复后的边权做一遍，记录 max_f_aft ，若 $\text{max_f_aft} \leq \text{max_f_pre} + \max\{t_0\}$ 则一开始就直接修复所有道路，否则不修复。

判断完是否修复后，开始对车辆运输情况决策。按物资到达的时间排序，每次让所有空余车辆来运输该物资，对每一辆要执行运输的车做 SPFA 来判断他的路线，执行完毕后对每辆车标记其空闲的时刻及地点，每次记录各车辆的运行情况保存，运输完所有物资后按时间排序并输出。

决赛：植物大战僵尸

11.1. 题目：

题目描述



这是植物大战僵尸里面的一款小游戏《我是僵尸》endless 模式，根据植物布局，我们合理的安排僵尸攻击植物，直到吃光每行最左边的大脑。每派出一个僵尸，将会花费你的太阳（购买僵尸的货币），我们目标就是用最少的太阳通过给定的一组关卡。

题目简化和细节说明

植物的脸都朝右，僵尸脸都朝左，僵尸是从最右面攻过来的；

僵尸不是慢悠悠走的，而是一次走一格或者两格；

僵尸前面有植物，必须吃完植物后才能前进，除非植物被梯子僵尸架了个梯子；

僵尸跟植物（大脑）在同个格子的时候，僵尸才可以吃植物（大脑）；

大脑的血量为 1，僵尸咬一口就吃掉。吃完大脑后，僵尸消失；

地图上布满了植物（没有空位），僵尸从地图最右边开始攻击；

一个僵尸只会攻击同行的植物，而且永远在这一行上；

没有特殊说明，植物的攻击都对同一行上的僵尸有效；

有些植物会攻击旁边行的僵尸，如磁铁蘑菇（请参考磁铁蘑菇的说明）；

有些植物会受到旁边行的僵尸影响，比如胆小蘑菇（请参考胆小蘑菇的说明）；

铁桶僵尸，橄榄球僵尸和梯子僵尸在被植物攻击的时候由铁桶，铁帽子，梯子先承受攻击，最后才轮到僵尸本体承受攻击。如果周围有磁铁蘑菇，这三样都会被磁铁蘑菇吸走，被吸走后就是本体直接被攻击。吸走后的铁器将消失；

植物选中一格攻击，发现该格有很多僵尸，而自己只能攻击一个，没有特殊说明，选择先放入的僵尸。如果这些僵尸是同一时刻放入的，选择先出现在输入文件的僵尸。

模拟器细节

我们模拟器分，植物，僵尸，子弹来设计。植物攻击僵尸有几种方式

1、直接攻击僵尸，比如窝瓜，地雷，食人花，磁铁蘑菇等，这种靠自身能力攻击僵尸，在攻击的瞬间就会对僵尸造成伤害。

2、选择发射子弹，发射子弹的行为是不会让僵尸立即扣血，而是等到判断子弹是否能够打击到僵尸时，才会扣僵尸的血。

我们的模拟器将地图变成一个 $\text{Array}[\text{Row}][\text{Col}]$ 的矩阵，僵尸的行走速度是 1/s 或者 2/s 只会从 $\text{Array}[\text{Row}]$ 走到 $\text{Array}[\text{Row}][i-1]$ 或者 $\text{Array}[\text{Row}][i-2]$ 。速度为 2 的僵尸，如果前面有植物挡住，只会前进一格；同理子弹的也是一样。

其它模拟细节

- 1、这个游戏每 1 秒检测一次，生成该帧的状态图，然后开始判断；
- 2、每帧的状态生成之后，判断伤害顺序为：
 - 1、检查每个僵尸吃植物（大脑）；
 - 2、检查每个植物是否要攻击僵尸（直接攻击，或者发射炮弹）；
 - 3、检查每枚炮弹是否打击到僵尸；
 - 3、僵尸吃植物，植物死亡时，将立即从植物列表删除；同理炮弹打到僵尸，或者僵尸死亡，也都会立即从对应的链表中删除。极端例子：当植物和僵尸都只剩一滴血的时候，由于僵尸先咬植物，所以植物死了，僵尸不被攻击；
 - 4、植物发射炮弹的时候，炮弹的起始位置为植物所在位置，下一秒开始移动。

植物数据

植物的伤害，攻击性弹药碰到僵尸后都造成 1 的伤害。

僵尸数据

僵尸吃植物每秒 2 格血。

输入输出

现在假设我们每个游戏场景都是在 ROW*COL 的方格中满植物，僵尸只能放置在最右面一列的方格中，并发起进攻。地图上放满植物每个方格中的可以同时存在多个僵尸。

输入：

输入由标准输入给出

第一行为 1 个整数 M ($0 < M < 100000000$)，第一个表示下面这些关卡你总共只能用 M 的太阳；

第二行也是 1 个整数 N ($0 < N < 100000$)，表示下面有 N 个 case；

第三行开始为 N 个 case 的具体信息，每个 case 第一行为 2 个整数 Row, Col 其中 Row 表示多少行，Col 表示多少列 ($1 \leq \text{Row} \leq 10000$, $1 \leq \text{Col} \leq 10000$) 接下来一共跟着 Row 行，并用代号标识每行的植物类型，并且用空格隔开。

输出：

结果需要输出到标准输出

对于每个关卡，第一行为一个整数 K，表示本关使用的僵尸数，接下来 k 行表示每个僵尸的布局，每行有 3 单词来表示一个僵尸的信息，以空格隔开，分别表示 僵尸代号 (ID)，行 (r)，时间 (time)。它们表示在时间 time 的时候把一个僵尸放到第 r 行的最右边上。对于放置时间相同的僵尸，植物攻击顺序和输出中的顺序一致的。

时间是整数，从 0 开始计算。我们要求输出的行按僵尸先后放置的时间，顺序一致。

如果你无法解出本关，又不想浪费太阳，请输出 1 行 0；

如果你输出了僵尸，那么即使这个僵尸是过关之后放置的，都要计算它的花费。

在桌面上有一个模拟器，填上你的输入输出文件它可以帮忙计算出你总的花费。

评判标准

我们评判程序会运行多个输入，然后我们会统计你的解决方案购买僵尸的总开销。

1、通过关卡最多的获胜；

2、关卡数一样，根据开销由小到大进行排序，决定谁获胜。

如果花费购买僵尸的开销相同，那么会以总的游戏的运行时间为标准，游戏运行总时间少的程序获胜，每关游戏运行时间指顺利过关的时间而不是程序执行时间。

注意事项

1、如果僵尸放置的位置不在地图上，那么这个僵尸会被认为无效的，同时相应的花费依然会被扣除；

2、如果僵尸的代号错误（注意大小写和连字符），那么这个僵尸会被认为是非法的，不进行任何操作，同时会按照最贵的橄榄球僵尸扣除 175；

3、如果程序出现异常情况或者程序本身没有完成全部的关卡，那么排名的时候会以通过的

关卡数为准;

4、如果在某一关植物已经全部被消灭,但是还有僵尸被放置到地图上,这些僵尸的花费依然会被计算;

5、如果花费的太阳数超过数据的给定值,模拟器在放置僵尸花费超过现在太阳数的时候结束游戏,以此时的结果作为最后结果。

【Sample Input】

```
100000
3
5 1
Potato-Mine
Potato-Mine
Potato-Mine
Potato-Mine
Potato-Mine
5 2
Potato-Mine Potato-Mine
Potato-Mine Potato-Mine
Potato-Mine Potato-Mine
Potato-Mine Potato-Mine
Potato-Mine Potato-Mine
5 2
Peashooter Wall-Nut
Peashooter Wall-Nut
Peashooter Wall-Nut
Peashooter Wall-Nut
Peashooter Wall-Nut
```

【Sample Output】

```
5
Pole 0 0
Pole 1 0
Pole 2 0
Pole 3 0
Pole 4 0
10
Imp 0 0
Imp 1 0
Imp 2 0
Imp 3 0
Imp 4 0
Pole 0 1
Pole 1 1
Pole 2 1
Pole 3 1
Pole 4 1
5
Pole 0 0
```

Pole 1 0

Pole 2 0

Pole 3 0

Pole 4 0

输入输出说明

第一关：每列都是土豆雷，我们都用撑杆跳的过去。如果你派小僵尸过去，那么僵尸会全砸死，虽然植物也全被消灭了，但是你还是没过关，还需要派小僵尸，花费更多。

11.2. 关键思路

此题作为百度之星 2010 的决赛题目，要求选手在 8 至 10 个小时内完成“僵尸大战植物”小游戏的模拟。这个游戏模式在《植物大战僵尸》中异常风靡，从一个守备者转向一个攻击者，就更需要玩家了解各种僵尸以及植物的特性，寻找突破口通过关卡。

这道题本身和是该游戏模式的简化：

- 1、没有了向日葵，阳光是一开始就给够了。
- 2、植物众多的攻击模式都有了简化
- 3、没有减速
- 4、僵尸模式比较单一。

我们把整个游戏分为两个部分：

- 1、模拟器
- 2、计算器

（一）模拟器：

模拟器应该是这道题目的重头戏，如何在短时间内就能写出一份能实现要求的代码也是对每个人的考验。

不难看出，整张地图只有三个大类：

- 1、植物
- 2、僵尸
- 3、子弹

对于植物来说，我们可以设计一个大类 `PlantObject`，共有属性包括：坐标、类别、血量、是否被架过梯子。然后对于每种细分的植物，都设计一个类，继承 `PlantObject`。比如对于食人花和磁力菇，需要额外记录冷却时间对于豌豆射手、小喷菇以及胆小菇，需要记录它的子弹是否还在。这样我们就设计好了植物的部分。

同理对于僵尸，我们也需要设计一个大类 `ZombieObject`，共有属性包括：坐标，移动速度，本体血量。对于细分的僵尸种类，在记录其额外信息，比如防具的血量，是否撑杆跳过，是否放过梯子等等。僵尸部分也如此设计好。

子弹部分也只需记录这个子弹的坐标，以及是从哪个植物发出的即可。

接下来的就是模拟器的设计了，我们来再看一下题目中给出的模拟器实现细节：

- 1、这个游戏每 1 秒检测一次，生成该帧的状态图，然后开始判断；
- 2、每帧的状态生成之后，判断伤害顺序为：

- 1、检查每个僵尸吃植物（大脑）；
- 2、检查每个植物是否要攻击僵尸（直接攻击，或者发射炮弹）；
- 3、检查每枚炮弹是否打击到僵尸；

4、僵尸吃植物，植物死亡时，将立即从植物列表删除；同理炮弹打到僵尸，或者僵尸死亡，也都会立即从对应的链表中删除。极端例子：当植物和僵尸都只剩一滴血的时

候，由于僵尸先咬植物，所以植物死了，僵尸不被攻击；

5、植物发射炮弹的时候，炮弹的起始位置为植物所在位置，下一秒开始移动。好像只需要按照题意进行模拟，但是还有很多细节要处理：

模拟僵尸的动作：

先考虑移动，移动到能移动到的最近的植物处。

如果移动到的地方存在植物，就要分好情况了：

如果僵尸是有杆的撑杆僵尸？

如果植物上面有梯子？

模拟植物的动作：

如果是普通的攻击型植物，先判断他能否攻击：

胆小菇胆小了？上次的子弹还没有攻击到？本行没有僵尸？

如果是食人花，即时消灭面前的一个僵尸，进入 CD。

如果是土豆地雷，即时消灭本格的所有僵尸（地雷血量无限不怕咬）

如果是窝瓜，看他左中右三格是否有僵尸，优先消灭最左边的。

如果是 CD 好的磁力菇，就要按照磁力菇的规则进行判定它要吸哪一件防具。

模拟子弹的动作：

子弹前进到能移动到的最近的僵尸/梯子处。

攻击僵尸/梯子，子弹消失，对应的植物可以下一次攻击。

综上，模拟器设计起来是不难的，但非常考验你的编程功底，以及需要一些 OOP 的技巧来简化你的工程量。但实现模拟器并不是这道题目的重点。

（二）计算器

重头戏来了，你要怎么样分配僵尸以求最小的阳光消耗来吃掉每一行的脑子？

比较明显的结论：

只有地雷和窝瓜具有面杀伤的能力

每一行之间是近似于独立的（磁力菇与胆小菇除外）

有了这两条理论基础，我们可以很好的设计一些直观的想法。

相同速度的僵尸尽可能在相近的时刻出发。


路障僵尸和橄榄球僵尸是绝对的主力

等等……

所以需要对每一行的火力做一个大致的估价，再大致的对这一行作一个战力的分配，再用模拟器进行模拟看最后的结果如何。这里就不方便进行详细的展开了。

最后提一件有趣的事，据说当年这道题的得分第四名的选手，提交的计算代码只有不到 80 行，他采用了最直观最暴力的策略，每行平均大量撒路障僵尸，再穿插些其他僵尸，取得了出乎意料的效果。

僵尸	特性	速度	血	花费	代号
 小鬼僵尸	移动速度快	2 格/s	3	50太阳	Imp
 路障僵尸	很普通	1 格/s	29	75太阳	Conehead
 铁桶僵尸	优先攻击铁桶，铁桶脱落后攻击本体 铁桶受到一定次数攻击脱落或者被磁铁蘑菇吸走	1 格/s	铁桶 能够承受56次攻击 去掉铁桶后能承受10次攻击 总计66	125太阳	Bucket
 橄榄球僵尸	优先攻击铁帽，铁帽脱落后攻击本体 铁帽受到一定次数攻击脱落或者被磁铁蘑菇吸走	2 格/s	铁帽子能够承受70次攻击 去掉铁帽子之后僵尸承受12次攻击 总计82	175太阳	Football
 梯子僵尸	碰到第一个植物，将梯子放在上面，下一帧计算才会爬过去。 植物被架了梯子之后，会被僵尸直接越过 无论梯子在僵尸手里，还是在植物身上，都会被磁铁蘑菇吸走。 只有僵尸手上没有梯子，才会攻击到僵尸本体 梯子受到一定次数攻击后会脱落，消失。 植物被架了梯子，只有植物死了或者被磁铁蘑菇吸走，梯子才会消失。	有梯子时2格/s， 没有梯子时1格/s	梯子能够承受24次攻击 僵尸能够承受24次攻击 总计 48	150太阳	Ladder
 撑杆僵尸	前进途中会翻过遇到的第一植物落到植物后面的那格上。检测到可以翻越，立即翻越不需要等到下一帧。 翻过后将会失去跳杆,行走速度变慢 注意它可以跳过窝瓜，当最终可能会被窝瓜砸死	有杆时2格/s 没杆时1格/s	24	75太阳	Pole

植物	特性	血	代号
 <p>土豆地雷</p>	<ul style="list-style-type: none"> 能够给予僵尸致命一击,僵尸碰到土豆雷后,本格僵尸全死。 土豆地雷爆炸后消失 	无穷	Potato-Mine
 <p>食人花</p>	<ul style="list-style-type: none"> 食人花会吞掉他前面1格或本格的僵尸,优先吃本格的 在咀嚼僵尸的时候,食人花不能动弹,咀嚼僵尸需要花费40秒的时间 咀嚼的时候,食人花被杀死,肚子里的僵尸不会复活 	12	Chomper
 <p>豌豆射手</p>	<ul style="list-style-type: none"> 能够发射豌豆把僵尸干掉,一发豌豆会消耗僵尸1格血,不能穿透僵尸。 只有发射的豌豆消失之后,才能发射下一枚 发射速度,每秒3格,打中僵尸后消失,超出地图范围也消失 豌豆只有发现自己前面有僵尸才会开火(同行) 	12	Peashooter
 <p>坚果</p>	<ul style="list-style-type: none"> 坚果具备坚硬的外壳,可以用来挡住其它僵尸。 没有攻击能力 	140	Wall-Nut
 <p>喷射蘑菇</p>	<ul style="list-style-type: none"> 僵尸(同行,蘑菇前面)靠近喷射蘑菇3格以内,喷射蘑菇才会开火 喷雾只有攻击到僵尸或者超出地图范围才消失 发送速度每秒3格 	12	Puff
 <p>窝瓜</p>	<ul style="list-style-type: none"> 靠近窝瓜的僵尸会被压扁,消灭该格上的所有僵尸。 窝瓜的可以攻击本格和邻近1格的僵尸,而且可以向后攻击(对于撑杆僵尸会出现向后攻击的情况) 窝瓜只会选择一个格子的僵尸攻击。 窝瓜选择最靠大脑最近的僵尸攻击 	无穷	Squash
 <p>胆小蘑菇</p>	<ul style="list-style-type: none"> 胆小蘑菇在O的位置,当敌人在x的位置的时候,停止攻击 .xxx. xxOxx .xxx. 其他跟豌豆射手一样 	12	Scharedy
 <p>磁体蘑菇</p>	<ul style="list-style-type: none"> 能够移除僵尸身上的金属制品,具体能影响的僵尸,参考僵尸的介绍。 能够除去梯子僵尸的梯子,无论梯子在僵尸身上,还是在植物身上,还是在自己身上。 攻击范围如下:O是磁铁蘑菇所在位置,x是僵尸(植物)所在位置 .xxx. xxOxx .xxx. 一次只能吸收一个僵尸的铁器装备 如果有很多僵尸(植物身上的梯子)进入攻击领域,优先选择中间,上面,下面这顺序,每列选择最靠近大脑的格子 同个格子先对付植物身上的梯子,然后才按前面策略选择僵尸 吸收铁器之后,过6秒才能重新吸收 	12	Magnet

百度之星 2011

初赛第一场第一题 图标排列

1.1. 题目描述:

百度应用平台上有很多有趣的应用，每个应用都由一个开发者开发，每个开发者可能开发一个或多个应用。百度的工程师们想把应用尽可能好的推荐给用户。

研究发现，同一个开发者开发的程序的图标有很大的相似性。如果把同一个开发者开发的应用放在一起，用户很快就会厌倦相似的图标，如果把这些图标穿插摆放效果就会好很多。

现在工程师想给用户推荐来自 m 个开发者的 n 个应用，在推荐的时候这些应用的图标将排成整齐的一行展示给用户，相邻两个图标之间的距离正好是 1，工程师们想让这些图标尽可能的穿插摆放。为了衡量穿插摆放的效果，给每个图标定义一个“分离度”，分离度的值是指当前图标和它左边最近的来自同一个开发者的图标之间的距离。如果一个图标左边没有来自同一个开发者的图标，则分离度为 0。所有图标穿插摆放效果的值定义为所有图标的分离度之和。

已知每个开发者开发的应用个数，请帮助百度的工程师找到图标穿插摆放效果的最大值。

输入描述

输入的第一行包含两个整数 n 和 m ，用一个空格分隔，分别表示应用的个数和开发者的个数。第二行包含 m 个正整数，相邻两个数之间用一个空格分隔，表示每个开发者开发的应用个数，这些整数之和必然等于 n 。

输出描述

输出一个整数，表示图标穿插摆放效果的最大值。

样例输入：

```
8 3
3 3 2
```

样例输出

```
15
```

提示

对于 20% 的数据， $n \leq 10$ ；

对于 40% 的数据， $n \leq 100$ 。

对于 100% 的数据， $1 \leq m \leq n \leq 100,000$

1.2. 关键思路:

单独考虑每个开发者的图标，如下图 ABCD 为某开发者的图标，'.' 为其他开发者的图标。

...A....B..C....D...

可以发现这个开发者的分离度为 $\text{dis}(A, B) + \text{dis}(B, C) + \text{dis}(C, D) = \text{dis}(A, D)$ 。也就是说对于同一个开发者，只关心他最左和最右的位置。

然后对于不同的开发者，可以证明只要让前 m 个图标和后 m 个图标分别有所有人的图标即可。对于只有一个图标的开发者，把他的图标扔中间就可以了。

初赛第一场第二题：篮球场

2.1. 题目：

题目描述：

百度公司有一块长 a 米宽 b 米的矩形空地，空地的左上角坐标为 $(0, 0)$ ，右下角坐标为 (a, b) 。空地上长着 n 株灌木，每株灌木都非常小。现在百度公司准备清理掉其中的一些灌木，在空地上修建两个长 28 米宽 15 米的篮球场。

球场必须完全修建在空地内部（边缘可以和空地的边缘重合）且球场边缘必须与空地边缘平行，两个篮球场不允许重叠（不考虑边缘）。

在清理灌木的时候，只有球场内部的灌木需要清理掉，球场外部和边缘的灌木不用清理。请帮助百度公司找到一种球场的建设方案，使得需要清理的灌木最少。

注意：在最优方案中球场的左上角坐标可能是实数。

输入描述

输入包含多组数据。

每组数据的第一行包含两个整数 a 、 b ，表示空地的长和宽。

第二行包含一个整数 n ，表示空地上灌木的数量。

接下来 n 行表示所有灌木的坐标，其中第 i 行包含两个整数 x_i 、 y_i ，表示第 i 个灌木的坐标为 (x_i, y_i) 。

最后一组数据之后的一行为两个 0，表示输入结束。

输出描述

对于每组数据，输出一个整数，表示最少需要清理多少株灌木。

样例输入

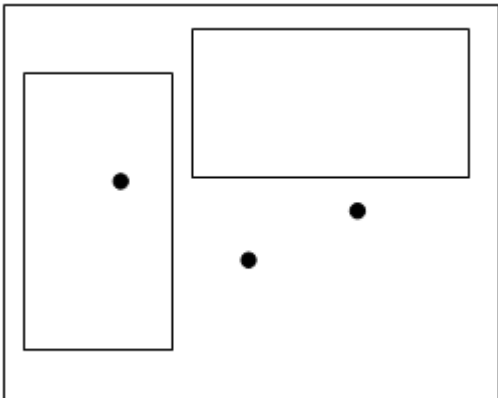
```
50 40
3
11 17
24 26
36 20
0 0
```

样例输出

```
1
```

提示

空地、灌木和最优的球场修建方案如下图所示。对于 100% 的数据， $30 \leq a, b \leq 100$ ，灌木的坐标都不相同。



2.2. 关键思路:

显然把矩形放在浮点数坐标点上并不会比放在整点上更优(可以旋转的话会更优), 所以只考虑整点的情况

记 $l[i]$ 表示放一个矩形使得矩形在 $x=i$ 的左边或线上, 最少要拔多少棵树。 $r[i], u[i], d[i]$ 类似

然后因为两个矩形不相交且均平行于坐标轴, 故一定可以用一条平行坐标轴的直线分开。

故 $ans = \min\{l[i] + r[i], u[i] + d[i]\}$ 即可

初赛第一场第三题: 度度熊大战僵尸

3.1. 题目:

僵尸最近老在百度大厦附近出没, 因此公司派出了度度熊去消灭他。

度度熊有 n 件武器, 第 i 件武器有物理攻击力 A_i 和魔法攻击力 B_i 。在某个时刻 t , 武器能造成的伤害为 $A_i + B_i * t$ 。僵尸有一个初始血量值 H , 受到武器的攻击后, 血量会减去武器的当前伤害值。如果某个时刻僵尸的血量值为负, 则僵尸将原地满血复活为血量值 H 。因此为了消灭僵尸, 度度熊的最后一击, 必须恰好使僵尸的血量为 0。

从时刻 1 开始的每个整数时刻, 度度熊可以从自己的武器中挑选一个武器攻击僵尸一次, 也可以不攻击僵尸。一件武器可以在不同的时刻使用多次。

由于度度熊武器的限制, 不是每个血量的僵尸都能杀死。度度熊希望能知道能杀死的僵尸中第 k 小的血量值是多少。

输入描述

输入的第一行包含两个整数 n, k , 分别表示度度熊拥有的武器数和要求的血量是第几小的。

接下来 n 行表示度度熊拥有的武器, 其中第 i 行包含两个整数 A_i, B_i , 表示第 i 个武器的物理和魔法攻击力。

输出描述

输出包含一个整数, 表示度度熊能杀死的僵尸中第 k 小的血量值。

样例输入

2 8

1 3

3 5

样例输出

15

提示

度度熊能杀死的僵尸中前 8 小的血量值依次为 4, 7, 8, 10, 11, 13, 14, 15。

对于 100% 的数据, $1 \leq n \leq 10, 1 \leq k \leq 50,000, 0 \leq A_i, B_i \leq 10,000$ 。

3.2. 关键思路:

BFS，状态为当前数，最后一次攻击的时间，最后一次用的哪个武器。

初赛第二场第一题 圆环

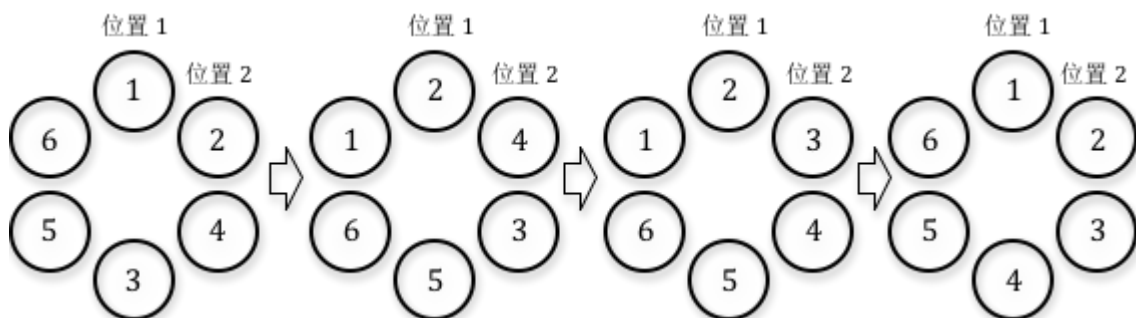
1.1. 题目:

一个圆环上有 n 个位置，这 n 个位置按顺时针依次标号为 $1, 2, \dots, n$ 。初始时圆环的每个位置上都有一个 1 至 n 之间的整数，且每个整数只出现一次。

任 何时刻，你可以将圆环上的数全部逆时针旋转一个位置，即第 i 个位置上的数变为原来第 $i + 1$ 个位置上的数，第 n 个位置上的数变为原来第 1 个位置上的数。也可以将圆环上的数全部顺时针旋转一个位置，即第 i 个位置上的数变为原来第 $i - 1$ 个位置上的数，第 1 个位置上的数变为原来第 n 个位置上的数。另有一个装置，可以交换圆环上第 a 个位置和第 b 个位置上的数。

下图给出了三种操作的 示例，圆环上有 6 个位置，初始数字分别为 $1, 2, 4, 3, 5, 6$ ，能交换第 2 个和第 3 个位置上的数。经过一次逆时针旋转后变为 $2, 4, 3, 5, 6, 1$ ，交换后变为 $2, 3, 4, 5, 6, 1$ ，再经过一次顺时针旋转后变为 $1, 2, 3, 4, 5, 6$ 。

请问通过旋转和交换，能否使得第 i 个位置上的数正好是 i 。



请问通过旋转和交换，能否使得第 i 个位置上的数正好是 i 。

输入

输入包含多组数据。

每组数据的第一行包含一个整数 n ，表示圆环上的数字个数。

第二行包含两个整数 a, b ($1 \leq a < b \leq n$)，表示可以交换圆环上第 a 个位置和第 b 个位置上的数。

接下来 n 行描述圆环上每个位置的初始值，其中第 i 行包含一个整数 a_i ，表示初始时刻第 i 个位置上的数。

最后一组数据之后的一行为一个 0 ，表示输入结束。

输出

对于每个测试用例，输出一行，如果能满足要求，这行中应只包含一个单词 Yes，如果不能满足要求，这行中应只包含一个单词 No。

样例输入

```
6
2 3
1
2
4
```

3
5
6
4
1 3
1
2
4
3
0

样例输出

Yes

No

提示

对于 100%的数据， $1 \leq n \leq 1,000$ 。

1.2. 关键思路：

相当于可以交换任意 i 和 $(i+b-a)\%n$ 的值，故只要看 $b-a$ 形成的全部剩余系是否可以排为有序即可

初赛第二场第二题：园艺布置

2.1. 题目：

题目描述：

近期，百度采纳了员工们的提议，计划在总部大楼内部种植园艺，以提供更加温馨的工作环境。公司将园艺设计的任务交给了度度熊同学。

公司总部大楼内部的构造可以分为 n 个区域，编号为 $0, 1, \dots, n-1$ ，其中区域 i 与 $i+1$ 是相邻的 ($0 \leq i < n-1$)。根据员工的投票和反馈，度度熊拿到了一份数据，表明在区域 i 种植园艺可以获得员工的满意度为 A_i 。度度熊希望园艺的布置方案满足条件：

1. 至少覆盖 m 个区域；
2. 布置园艺的区域是连续的。

请帮他找到一种满足条件的方案，使布置园艺区域的员工的满意度的平均值最大。

输入

输入的第一行包含两个整数 n 和 m ，分别表示总区域数和至少覆盖的区域数。

第二行包含 n 个整数 A_0, A_1, \dots, A_{n-1} ，依次表示在每个区域种植园艺可以获得员工的满意度。

输出

输出一行，表示员工的平均满意度的最大值。如果这个数是一个整数，则直接按整数格式输出；否则，请用最简分数表示，分子分母以 “/” 分割，格式见样例。

样例输入

样例输入 1

3 1
2 3 1

样例输入 2
5 3
1 8 2 4 8

样例输出
样例输出 1
3
样例输出 2
11/2

提示

样例 2 的正确答案为 11/2，尽管 22/4 数值也相同，但由于没有化简，所以是错误的。
对于 100% 的数据， $1 \leq m \leq n \leq 106$ ， $1 \leq A_i \leq 106$ 。

2.2. 关键思路：

首先考虑如下问题：

给定一个序列和一个数 x ，问能否从序列中找到连续一段使得平均值不小于 x 且这一段的长度不小于 m
只要将序列中每个数都减去 x ，就变为找长度不小于 m 的一段使得其和不少于 0

对于上述问题，可以用部分和的思想，一方面记录当前的前缀和为多少，以方便记录前面的数的前缀和的最小值是多少，只要其差大于等于 0 即可。

所以可以二分答案，然后用上述方法验证。

初赛第二场第三题：数据还原

3.1. 题目：

度度熊近日开发出一种新型随机数生成算法，方法是使用一个质数 P 和 n 个非负整数 A_0, A_1, \dots, A_{n-1} ，生成第 m 个随机数的公式为

$$rand_m = \left(\sum_{k=0}^{n-1} A_k \cdot m^k \right) \bmod P$$

通过适当的选取参数 A_i ，度度熊发现这种随机数生成的方法具备一种神秘的性质，并帮助他完成了多项研究。度度熊准备在一个新环境中进行他的下一次实验，他让他的助手去取他桌上写着 n 个整数 A_0, A_1, \dots, A_{n-1} 的纸条以产生新的随机数据，取回后度度熊发现助手取回的不是写着参数的纸条，而是他上一次实验时记录下来的随机数 $rand_s, rand_{s+1}, \dots, rand_{s+n-1}$ ，而数的个数正好也是 n 个。现在度度熊已经没有时间等他的助手再回去取写着参数的纸条了，你能帮度度熊生成接下来的 x 个随机数（即 $rand_{s+n}, rand_{s+n+1}, \dots, rand_{s+n+x-1}$ ）让他继续他的实验么？

输入

输入的第一行包含 4 个非负整数 n, P, s, x ，相邻两个整数间用一个空格分隔。
第二行包含 n 个整数 $rand_s, rand_{s+1}, \dots, rand_{s+n-1}$ ，表示度度熊上一次实验生成的随机数。

输出

输出一行，包含 x 个非负整数 $\text{rands}+n$, $\text{rands}+n+1$, \dots , $\text{rands}+n+x-1$ ，相邻的两个整数间用一个空格分隔，表示接下来生成的 x 个随机数。

样例输入

```
4 101 1 2
5 17 43 89
```

样例输出

```
60 63
```

提示

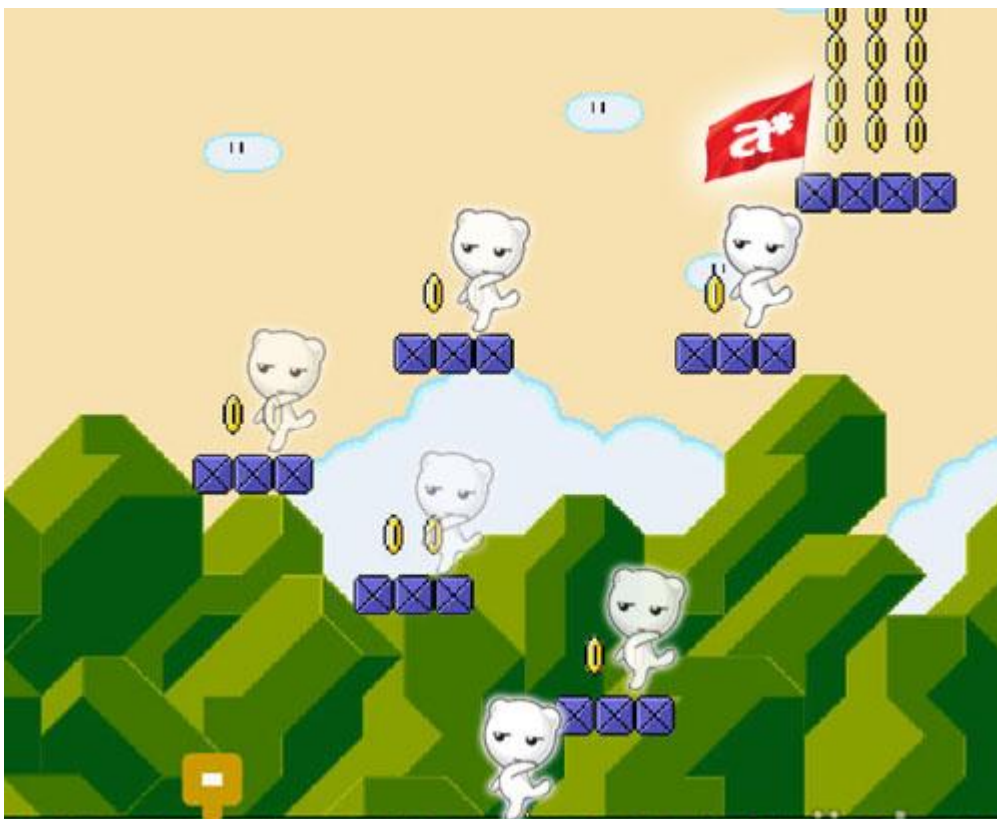
对于 100% 的数据， $1 \leq n, s, x \leq 1000$ ， $s + x + n \leq P < 10^9$ ， P 为质数。

3.2. 关键思路：

百度高阶等差数列 按定义逆向还原即可。

复赛第一题 跳跃

1.1. 题目：



度度熊最近迷上了一个电脑游戏，游戏中有这样一个场景：在一片高低错落的山地中，度度熊控制的人

物从左向右做若干次跳跃，由起点一直跳到终点。

经 过度度熊最近对这个游戏的研究，他发现游戏中能站立的地点只有 $n + 1$ 个，可以标号为 $0, 1, \dots, n$ 。其中 0 为起点， n 为终点。在跳跃时只能从左向右跳跃，从点 i 跳跃到点 j 的水平距离为 $j - i$ 。点 i 有高度值 h_i ，对于从点 i 到点 j 的直接跳跃，其水平距离不能超过点 i 的高度，而且在点 i 到点 j 之间不能出现高度值同时大于等于 h_i 和 h_j 的点（例如 $h_1=3, h_2=4, h_3=2$ ，则不能从点 1 直接跳到点 3 ，因为点 2 的高度值 h_2 既大于等于 h_1 ，又大于等于 h_3 ）。如果 $h_i > h_j$ ，则花费的体力值为 0 ，否则花费的体力值为 $h_j - h_i$ 。

由于体力值在游戏中很有用，度度熊希望尽量在这个场景节省体力。他已经知道了每个点的高度值，请你帮助度度熊计算一下最少需要多少体力。

输入描述

输入包含一组数据。

第一行包含一个正整数 n ，表示场景中起点与终点的水平距离。

第二行包含 $n + 1$ 个正整数，其中第 $i + 1$ 个整数为 h_i ，表示点 i 的高度值。

输出描述

输出一行，包含一个正整数，表示至少需要耗费的体力值。

样例输入

```
5
1 3 4 2 3 1
```

样例输出

```
3
```

提示

对于 30% 的数据， $1 \leq n \leq 3000$ 。

对于 100% 的数据， $1 \leq n \leq 100,000, 1 \leq h_i \leq 109$ 。

1.2. 关键思路：

复杂度 $N \log N$

$dp[i]$ 表示从 i 节点跳到 n 节点的最小花费，从后往前更新。 $dp[n]=0$ 。

对于任意一个 i ，如果存在 $i < j < k \leq i + h[i] \ \&\& \ h[i] < h[j] < h[k]$ 这样的情况，从 i 跳到 j 再跳到 k 的花费等于直接从 i 跳到 k 的花费，而且 j 点的可跳范围更大。所以在 $(i, i + h[i]]$ 范围内，所有比 i 点高的节点，选择第一个比 i 高的节点 j 是最优的。 j 是 i 右边第一个比 i 高的点，对于 $j < k \leq i + h[i] \ \&\& \ h[k] \leq h[j]$ ， k 不能跳。

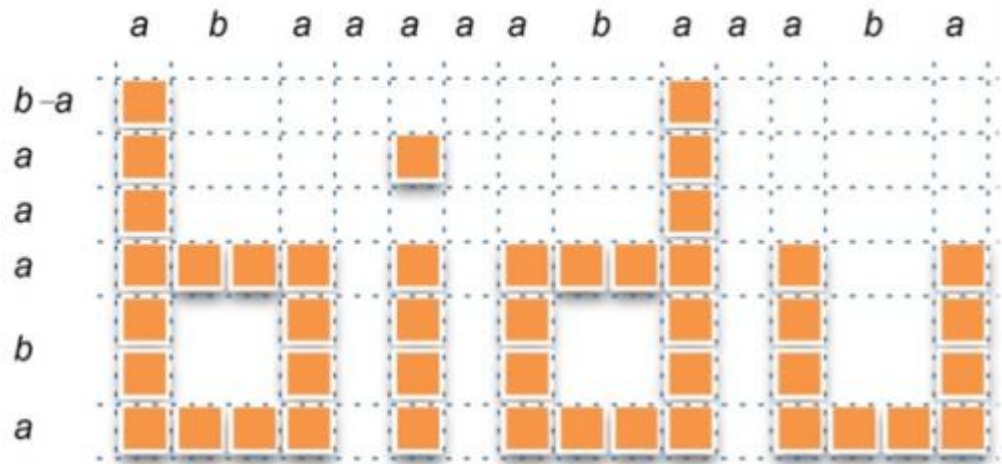
所以对于 i ，RMQ+二分找到 $(i, i + h[i]]$ 范围内的第一个比 i 高的点 j 。如果存在这样的 j ，更新 $dp[i] = dp[j] + h[j] - h[i]$ ，查找 (i, j) 内 dp 数组最小值 $dp[k]$ ，更新 $dp[i] = \min(dp[i], dp[k])$ 。如果不存在这样的 j ，查找 $(i, i + h[i])$ 内 dp 数组最小值 $dp[k]$ ，更新 $dp[i] = dp[k]$ 。用线段树实现点更新区间查询，复杂度为 $O(n \log n)$ 。输出 $dp[0]$ 。

复赛第二题：股票代码

2.1. 题目：

题目描述：

百度公司在纳斯达克上市的股票代码是 bidu，百度熊为了制作股票价格的显示屏，希望在一个矩形液晶屏幕上显示”bidu”字样的图标。图标可以看作由一个 $13 * 6$ 的矩形表格组成，其中每个单元格的尺寸以及颜色请参照下图，百度熊使用 a, b 两个参数来控制字体的粗细和大小，图中显示的是 $a=1, b=2$ 的例子，占用 $16*7$ 个单元格。



液晶屏的分辨率为 $w * h$ ，在摆放图标的时候，百度熊发现液晶屏有一个尺寸为 $p * q$ 像素的损坏部分，其中损坏的像素称为坏点。坏点无法点亮，只能显示黑色。为了不影响显示效果，他决定在摆放图标时一定要满足下述条件：

1. 图标中所有橙色区域必须正常显示，也就是不可与坏点有交集，并且不会超出液晶屏的显示范围。
2. 图标及显示屏均不可翻转或旋转。

请你实现一个程序，帮助百度熊计算满足条件的图标摆放位置一共有多少种。

输入描述

输入包含一组数据。

第一行包含 8 个整数 w, h, a, b, s, t, p, q ，以空格分割，表示液晶屏的宽（左右包含的像素数）为 w 、液晶屏的高（上下包含的像素数）为 h 、控制显示的两个参数为 a 和 b 、坏点区域距左边 s 个像素、坏点区域距上边 t 个像素、坏点区域宽 p 、坏点区域高 q 。

输出描述

第一行包含一个整数，表示图标可以摆放的位置个数

样例输入

17 8 1 2 8 4 1 1

样例输出

1

提示

以最左上角的单元格坐标为 $(0,0)$ ，最右下角的单元格坐标为 $(16,7)$ ，那么坏格的坐标为 $(8,4)$ 。图标以 $(0,0)$ 为左上角，恰好可以放进液晶屏。

对于 40% 的测试数据， $0 < w, h \leq 104$ ；

对于 100% 的测试数据， $0 < w, h \leq 109, b > a > 0, 10a + 3b \leq w, 3a + 2b \leq h, 0 \leq s < s + p \leq w, 0 \leq t < t + q \leq h$ 。

2.2. 关键思路:

由于 Bidu 图标格子数目较少, 可以采用如下思路:

首先计算出不能左右移动, 只能沿着左边缘上下移动的方案数;

在考虑左右移动时, 我们可以发现, 只有当坏点矩形边缘跨越 bidu 图标某个边缘时, 方案数才会发生改变, 否则均与上一状态方案数相同。因此当遇到这种跨越情况时, 我们重新计算方案数即可。Bidu 只有 45 个格子, 坏点矩形共有垂直方向边缘, 因此至多会跨越 $45 \times 4 = 180$ 次。

复赛第三题: 数据校验

3.1. 题目:

最近度度熊研究出了一种字符串压缩的方法, 基本想法是每个串, 都用其他串的片段的组合来表示。比如有三个串 abce, abcde, ced, 第一个串“abce”可以看成第二个串的 1 至 3 个字符“abc”加上第二个串的第 5 个字符“e”组成。第三个串“ced”可以看成第一个串的 3 至 4 个字符“ce”加上第二个串的第 4 个字符“d”组成。

如果把字符串从 1 开始编号, 每个字符串的字符位置也从 1 开始编号, 用三元组 (d, s, t) 表示编号为 d 的串中, 从 s 字符开始, 到 t 字符结束的子串, 那么每个串都可以用一系列三元组表示。例如, 对于上面的三个串, 第一个串可以表示为 (2, 1, 3) + (2, 5, 5), 第三个串可以表示为 (1, 3, 4) + (2, 4, 4)。

在使用这种方法表示了这些串之后, 度度熊还保存了这些串的一部分大小关系。如果串 A 在字典序中排在串 B 前面, 则认为 $A < B$ 。比较 A 和 B 的大小关系时, 可以从左到右同时扫描两个串, 知道找到第一个不同的字母, 此时字母小的比较小, 如果扫描到其中的一个串结束还没有找到不同的字母, 则长度短的比较小。比如: $abc < abd$, $ab < abcd$ 。

现在度度熊有压缩后的三元组信息, 以及一些比较的结果, 想校验这些信息是否正确。度度熊的校验方法很简单, 就是找是否存在一个字符串的序列, 能压缩成给定的三元组信息, 并且这些串的比较的结果和给定的比较结果一致。

输入描述

输入包含多组数据。

每组数据的第一行包含一个整数 n, 表示字符串的个数。

接下来 n 块, 每块描述一个压缩后的字符串。每块的第一行包含一个整数 K_i , 表示这个字符串由 K_i 个三元组表示。接下来 K_i 行描述这些三元组, 其中第 j 行包含三个整数 d_j, s_j, t_j , 表示第 j 个三元组为 (d_j, s_j, t_j) 。

接下来一行包含一个整数 m, 表示比较过的字符串对。接下来 m 行, 每行两个正整数 x, y, 表示第 x 个字符串小于第 y 个字符串。

相邻两组数据之间包含一个空行, 最后一组数据后包含一个整数 0, 表示输入结束。

输出描述

对于每组数据, 输出一行, 如果能满足要求, 这行中应只包含一个单词 Yes, 如果不能满足要求, 这行中应只包含一个单词 No。

样例输入

```
3
2
```

2 1 3
2 5 5
3
1 1 3
3 3 3
3 2 2
2
1 3 4
2 4 4
2
2 1
1 3
2
2
2 1 1
2 1 1
1
1 1 1
1
1 2
0

样例输出

Yes

No

提示

第一组样例，原来的 3 个字符串为 abce, abcde, ced。

由于字符串的内容可以是中文或其他文字，所以可以认为可用字符集足够大
每个测试点的 case 数 ≤ 10

对于 20% 的测试数据，所有串长度 ≤ 5 ， $n \leq 5$ ， $m \leq 10$ ；

对于 40% 的测试数据，所有串长度 ≤ 10 ， $n \leq 30$ ， $m \leq 100$ ；

对于 100% 的测试数据，所有串长度 ≤ 20 ， $n \leq 200$ ， $m \leq 5000$ ；

3.2. 关键思路：

通过三元组关系，用并查集维护字符间的相等关系；

根据字典序关系，从左至右扫描两个字符串，找到第一个不相等的字符，将这两个字符对应的并查集建立一条有向边；

求有向图求强连通分量，如果没有环则方案 Yes；有环则将同一连通分量内的字符合并为一个并查集，重新建图做强连通。

建图时出现矛盾则方案 No。

时间复杂度：共 4000 个字符，n 为 4000， $O(n^2)$

决赛第一题 德州扑克游戏

1.1. 题目：

一、背景

本次比赛的内容是在选手编写的程序之间进行德州扑克游戏。本文介绍德州扑克的游戏规则。

二、游戏工具

德州扑克使用一副普通的扑克牌去掉大小王牌之后的 52 张牌进行游戏。这 52 张牌分为 4 个花色：黑桃(S)、红桃(H)、梅花(C)、方块(D)，每个花色包含 13 张点数不同的牌，从小到大依次为 2、3、4、5、6、7、8、9、10、J、Q、K、A，其中 J、Q、K 分别表示 11、12、13 点，A 比较特殊，可以当作 14 点，也可以当作 1 点。注意花色没有大小顺序。为了描述方便，我们用“**点数+花色**”的形式来描述一张扑克，其中 10 用 T 表示。即 3S 表示黑桃 3，TC 表示梅花 10，AD 表示方块 A。

三、游戏玩家

游戏可以由 2 至 6 个玩家参与。所有玩家按照系统随机的顺序围坐在一张圆桌旁，从圆桌的正上方向下看，每个玩家的顺时针方向的玩家称为该玩家的“**下家**”，逆时针方向的玩家称为该玩家的“**上家**”。

从开始玩游戏一直到游戏结束称为“**一轮**”，每一轮的座位不变，不同轮的座位可能不同。在每一轮的开始，每个玩家都会得到数量为 1000 的筹码，之后玩家需要使用这些筹码下注以尽量赢取其他玩家的筹码，如果在一手(见“游戏进程”)结束后玩家手中的剩余筹码为 0 (见“游戏进程”)，则玩家无法继续进行游戏，被迫出局。每一轮游戏，先出局的玩家会获得较少的分数，后出局的玩家获得较多的分数，同时出局的玩家将获得相同的分数，当游戏中只剩下 1 个玩家时，一轮游戏结束。

每一阶段的比赛都根据选手的程序所获得的分数排名，详细说明见“赛制规则”部分。

四、游戏进程

每一轮游戏分为很多“**手**”，每手牌是一个相对独立的过程。

每手牌都会有一个“**庄家**”，第一手牌的庄家由系统随机分配，之后每一手的庄家都是上一手庄家的下家，即玩家按照顺时针顺序轮流担任庄家。

对于每手牌，都规定一个“**最小下注额**”，初始时为 2，当每个玩家都当过一次庄家时，最小下注额加 2，当每个玩家再当过一次庄家时，最小下注额再加 2，依此类推。

每手牌的过程如下：

1. **洗牌**：系统自动将所有 52 张牌打乱顺序。
2. **盲注**：庄家的下家和庄家下家的下家必须下“盲注”，即在得到牌之前下注。这是为了保证每个获胜的牌面都能赢得一些筹码。如果牌桌上玩家多于 2 人，则庄家的下家下“小盲注”，数额等于最小下注额。下“小盲注”玩家的下手下“大盲注”，数额等于最小下注额的 2 倍；如果牌桌上只有 2 人进行游戏，则庄家下“小盲注”，庄家的下家下“大盲注”。如果应当下盲注的玩家的剩余筹码不足上述的盲注数额，则该玩家必须下所有筹码。
3. **发底牌**：下盲注后，系统给每个玩家发两张牌，只有玩家自己可以看到是什么牌，别人无法看到。这些牌被称作为“**底牌**”。

4. **第一次下注：** 从下大盲注玩家的下家开始，沿顺时针方向依次下注。每个玩家可以选择“弃牌”、“让牌”、“跟注”或“加注”。对于各种下注方法以及每轮下注应当如何结束，将在后面的“下注系统”中详细介绍。
5. **发翻牌：** 第一次下注完成后，系统在圆桌中央发三张“公共牌”，所有人都可以看到这些牌是什么。这三张牌称为“翻牌”。
6. **第二次下注：** 发完翻牌后进行第二次下注。由庄家的下家首先下注。其他流程与第一次下注完全一样。
7. **发转牌：** 在第二次下注后，系统在圆桌中央发第四张“公共牌”，所有人能看到是什么牌，这张牌叫做“转牌”。
8. **第三次下注：** 这一次下注与第二次下注流程完全相同。
9. **发河牌：** 随后，系统在圆桌中央发第五张“公共牌”，所有人能看到是什么牌，这张牌叫做“河牌”。
10. **第四次下注：** 这是最后一次下注，和第二、三次下注流程一样。
11. **摊牌：** 在最后一次下注后，所有未弃牌（详见“下注系统”）的玩家翻开自己的牌，系统根据玩家的组合牌大小分配筹码，具体分配规则见下注系统的“筹码分配”。
12. **提前结束：** 在任何一次下注过程中，如果只剩下一个玩家未弃牌，则本手游戏提前结束，未弃牌的玩家获得所有筹码。

五、下注系统

1. 1. 基本概念

下注额： 每手游戏中，玩家会跟随游戏进程，分别逐步增加自己的下注筹码数量，玩家在一手游戏中下注的筹码总数量称为该玩家的“下注额”。

最高下注玩家： 下注额最高的第一个玩家称为“最高下注玩家”。玩家下盲注时不成为最高下注玩家，第一个跟注或加注的玩家成为最高下注玩家。

当前最高下注额： 在大、小盲注下完后，“当前最高下注额”为大盲注的筹码数。在出现了“最高下注玩家”之后，“当前最高下注额”为最高下注玩家的下注额。

初始筹码： 每手游戏开始前，玩家持有的筹码。

剩余筹码： 每手游戏中，玩家剩余筹码的数量。玩家的下注额+剩余筹码=初始筹码。

1. 2. 下注规则

下注顺序： 第一次下注，由下大盲注玩家的下家首先下注，然后其他玩家按照顺时针顺序依次下注。第二、三、四次下注，由庄家的下家首先下注，然后其他玩家按照顺时针顺序依次下注。

下注： 每个剩余筹码大于零的玩家在下注时可以选择“弃牌”、“让牌”、“跟注”或“加注”。

弃牌： 弃牌是决定放弃本手游戏，以后不再下注，直到新的一手牌开始。弃牌的玩家不参与筹码分配，弃牌前下注的筹码将用于筹码分配，玩家保留剩余筹码。

让牌： 玩家下注时，如果他当前的“下注额”已经等于“当前最高下注额”，则他可以选择不加注而留在游戏中。第一个让牌的玩家成为“最高下注玩家”。

跟注： 将自己的“下注额”提升至“当前最高下注额”，若“剩余筹码”不足，则将“下注额”提升至“初始筹码”的数量。

加注：将“下注额”提升至“当前最高下注额”，同时再追加下注一定数量的筹码。玩家在加注之后成为最新的“最高下注玩家”。追加的筹码数量需要大于或等于“最小下注额”，若“剩余筹码”不足，则不能选择加注。

下注结束条件：在重新轮转到“最高下注玩家”时，一次下注结束。即在最高下注玩家下注后一圈内无人加注时，此次下注结束。

选择权与下注结束：“选择权”是对下注结束条件的另一种理解方法。下注开始时，所有玩家具有“选择权”，当一个玩家弃牌、让牌或跟牌时，他失去选择权；当一个玩家加注时，他失去选择权，所有其他未弃牌并且无选择权的玩家重新获得选择权。当所有玩家都不具有选择权时，下注结束。这种理解方式与前面根据“最高下注玩家”定义的下注结束条件是没有矛盾的。

筹码分配：在第四次下注后，游戏会根据玩家的组合牌大小分配筹码，牌面大小见下一节描述。分配过程如下：

1. 所有未弃牌的玩家参与筹码分配。
2. 选择下注额最小的玩家（“最小玩家”），从所有玩家当前的下注筹码中分别取出与最小玩家的当前下注筹码相等的数量。
3. 将取出的这些筹码分配给当前参与分配的牌面最大的玩家，如果出现多个玩家牌面最大，则平均分配。对于不能均分的情况，在牌面最大的玩家中，由庄家下家位置开始，按顺时针顺序每人分配一个筹码，直到分完为止。
4. 当前剩下的下注筹码为 0 的玩家退出筹码分配。
5. 重复 2, 3, 4，直到所有下注的筹码都分配完毕。
6. 每手牌结束后，玩家持有的筹码数量为：剩余筹码+分配到的筹码。
- 7.

六、牌面大小

当一手牌结束，摊牌时，所有未弃牌的玩家将手中的牌翻开，比较牌的大小。系统自动从玩家手中的 2 张牌和 5 张公共牌中选出 5 张构成一幅最大的牌面。这 5 张牌可能构成不同的组合，组合的大小顺序依次为：同花顺、四张、葫芦、同花、顺子、三张、两对、一对、杂牌。组合的规则和相同组合中的牌的大小如下：

1. **同花顺：**五张牌花色相同，并且构成顺子（即点数相连）。例如：AD、KD、QD、JD、TD。由于 A 可以看成 14 点或 1 点，所以 5H、4H、3H、2H、AH 也是同花顺，但 4H、3H、2H、AH、KH 不是同花顺。两个同花顺比较时，只比较点数最大的牌，如 KD、QD、JD、TD、9D 比 5H、4H、3H、2H、AH 大，因为 K 比 5 大。而 AD、KD、QD、JD、TD 比 KD、QD、JD、TD、9D 大，因为 A 比 K 大。最大点数的牌相同时认为大小相同，如 5H、4H、3H、2H、AH 和 5C、4C、3C、2C、AC 大小相同。
2. **四张：**有四张点数相同的牌，再加上另一张牌。例如：9S、9H、9C、9D、5S。两个四张比大小时首先比较四张相同的牌的点数大小，如果点数相同再比较剩下的一张牌，此时如果剩下的一张牌大小相同就认为两个四张大小相同。
3. **葫芦：**三张点数相同的牌，加上另两张点数相同的牌。例如：9S、9H、9C、5H、5S。大小比较为首先比较三张的大小，再比较两张的大小。
4. **同花：**五张牌同花，但不是顺子。例如：AH、KH、TH、5H、2H。比较大小为按 5 张牌从大到小的顺序依次比较。
5. **顺子：**五张牌点数相连，但是不同花。例如：JH、TS、9S、8D、7D。比较大小的规则与同花顺相同。
6. **三张：**三张点数相同的牌，加上另两张点数不同的牌。例如：9S、9H、9C、TH、4D。比较大小时首先比较三张的大小，然后按从大到小的顺序依次比较剩下的两张牌。

7. **两对**：两对相同的牌加上一张散牌。例如：9H、9C、8S、8D、5S。比较大小时首先比较点数较大的对，再比较点数较小的对，最后比较散牌。
8. **一对**：一对相同的牌加上三张散牌。例如：6S、6C、QD、TD、7S。比较时首先比较对，然后按从大到小的顺序依次比较剩下的三张牌。
9. **杂牌**：五张散牌。例如：KH、QC、TD、8D、2H。比较大小时按从大到小的顺序依次比较。

1.2. 关键思路：

这题比较灵活，在比赛这么短的时间内也没办法实现太复杂的 AI，只能根据自己的经验和感觉设计策略。一个可行的思路是，我们可以根据手牌和公共牌计算一个分数，描述自己最终可能得到的组合的优劣程度的期望，根据这个分数我们去决定跟还是不跟。更细致我们还可以纳入当前有多少玩家还在跟牌，各个玩家的经济状况等条件综合考虑。事实上这不仅是写一个 AI，也是选手之间的博弈，揣度其它选手可能的策略。比如有人的策略是前 15 把直接弃牌存活下来，后面随机跟牌碰运气，这样简单的策略最终也取得了第五的成绩，当然这也和运气有关。在编码调试的过程中，百度还为大家提供了三个测试 AI，一个是全跟，一个是全弃，还有一个是百度员工实现的 AI。

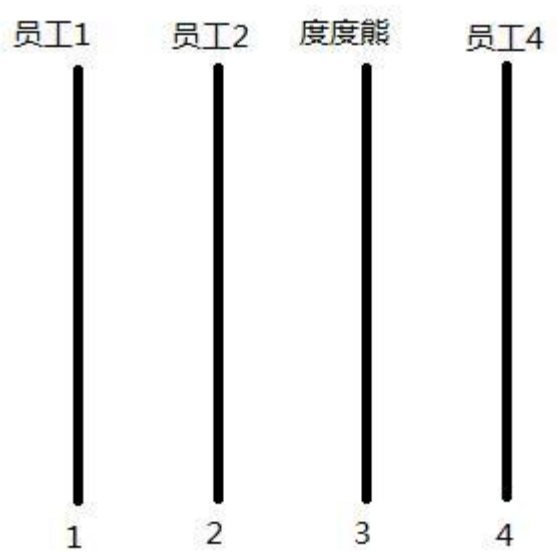
百度之星 2012

初赛第一场第一题：

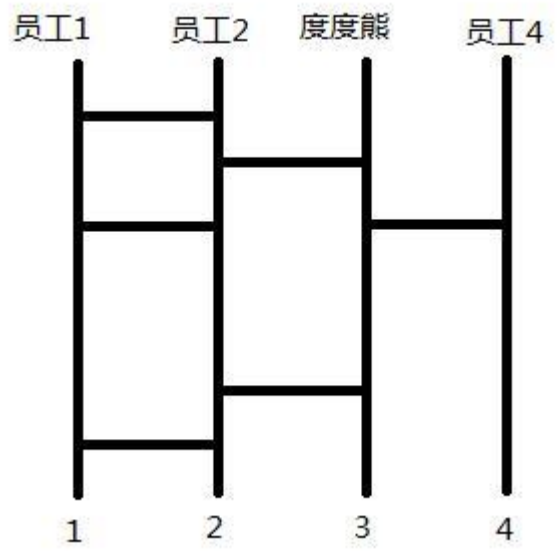
1.1. 题目：

Baidu 年会安排了一场时装秀节目。N 名员工将依次身穿盛装上台表演。表演的顺序是通过一种“画线”抽签的方式决定的。

首先，员工们在一张白纸上画下 N 条平行的竖线。在竖线的上方从左到右依次写下 1 至 N 代表员工的编号；在竖线的下方也从左到右依次写下 1 至 N 代表出场表演的次序。

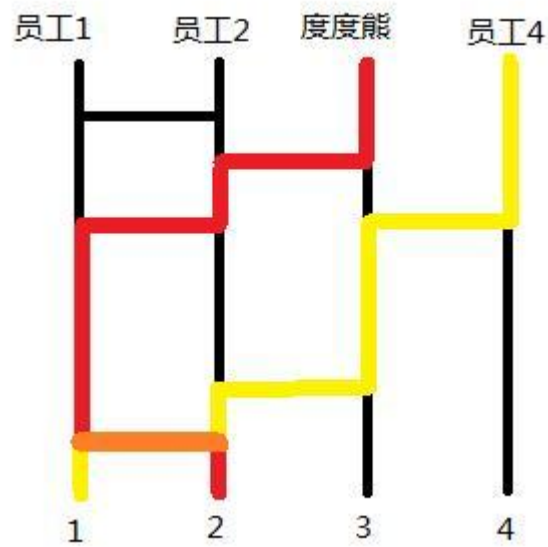


接着，员工们随意在两条相邻的竖线间添加垂直于竖线的横线段。

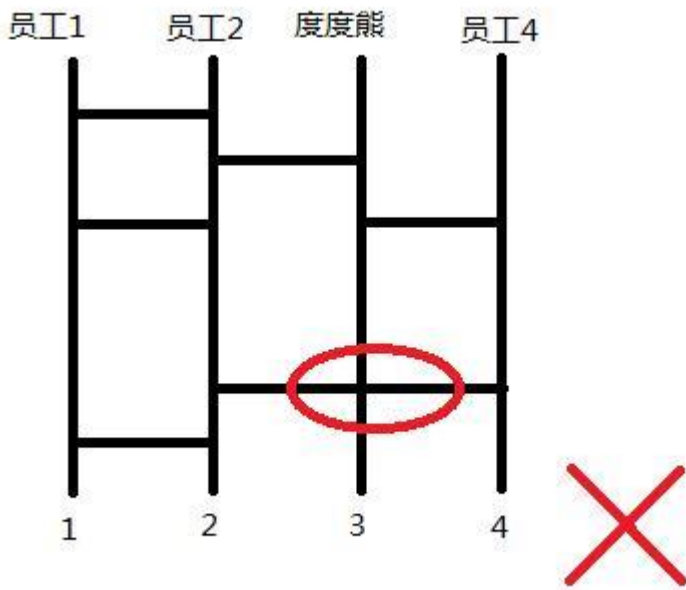


最后，每位员工的出场顺序是按如下规则决定的：每位员工从自己的编号开始用手指沿竖线向下划，每当遇到横线就沿横线移动到相邻的竖线上去，直到手指 到达竖线下方的出场次序编号。这时手指指向的

编号就是该员工的出场次序。例如在下图的例子中，度度熊将第二名出场，第一名出场的是员工 4。



员工在画横线时，会避免在同一位置重复画线，并且避免两条相邻的横线连在一起。即下图所示的情况是会出现的：



给定一种画线的方案，员工编号为 K 的度度熊想知道自己是不是第一位出场表演的。如果不是，度度熊想知道自己能不能通过增加一条横线段来使得自己变成第一位出场表演。

输入数据：

为了描述方便，我们规定写有员工编号的方向是 y 轴正方向（即上文中的竖线上方），写有出场次序的方向是 y 轴负方向（即上文中的竖线下方）。竖线沿 x 轴方向（即上文中从左到右）依次编号 1 至 N。于是，每条横线的位置都可以由一个三元组 $\langle x_l, x_r, y \rangle$ 确定，其中 x_l, x_r 是横线左右两个端点所在竖线的编号， y 是横线的高度。

输入第一行是一个整数 $T (T \leq 50)$ ，代表测试数据的组数。

每组数据的第一行包含三个整数 N, M, K ($1 \leq N \leq 100, 0 \leq M \leq 1000, 1 \leq K \leq N$), 分别代表参与表演的员工人数、画下的横线数目以及度度熊的员工编号。

每组数据的第 $2 \sim M+1$ 行每行包含 3 个整数, x_l, x_r, y , ($1 \leq x_l < N, x_r = x_l + 1, 0 \leq y \leq 1,000,000$), 描述了一条横线的位置。

输出数据:

对于每组数据输出一行 Yes 或者 No, 表示度度熊能否通过增加一条横线段来使得自己变成第一位出场表演。如果度度熊已经是第一位出场表演, 也输出 Yes。注意, 尽管输入数据中员工画的横线高度都是整数, 但是度度熊可以在任意实数高度画横线。此外, 度度熊和员工一样, 在画横线时需要避免在同一位置重复画线, 也要避免两条相邻的横线连在一起。

评分标准:

程序输出结果是否正确。

1.2. 关键思路:

根据作用效果易分析出, 题目中的横线起的作用是交换相邻两条竖线上的人的出场次序。

所以建造初始次序序列后, 根据横线的坐标从大到小模拟交换即可。

这样可以得到最终的次序序列即为结果, 如果度度熊是第一个出场自然输出 yes。

如果在出场次序的交换过程中俩个人相邻, 因为可以画实数高度的线, 必然添一条线可以交换两人的次序。

所以统计跟度度熊相邻的人中是否出现过“最终第一个出场的人”, 如果有也是 yes。

其他情况是 no。

初赛第一场第二题: 小小度刷礼品

2.1. 题目

一年一度的百度之星又开始了, 这次参赛人数创下了吉尼斯世界纪录, 于是百度之星决定奖励一部分人: 所有资格赛提交 ID 以 x 结尾的参赛选手将得到精美礼品一份。

小小度同学非常想得到这份礼品, 于是他就连续狂交了很多次, 提交 ID 从 a 连续到 b, 他想问问你他能得到多少份礼品, 你能帮帮他吗?

输入

第一行一个正整数 T 表示数据组数;

接下去 T 行, 每行三个正整数 x, a, b ($0 \leq x \leq 1018, 1 \leq a, b \leq 1018, a \leq b$)

输出

T 行, 每行为对应的数据情况下, 小小度得到的礼品数

样例输入

1

88888 88888 88888

样例输出

1

2.2. 思路:

先不妨设 x 有 len 位, 则对于符合要求的数字, 其最低 len 位是确定的 (即 x), 所以剩下位的每一种可能便对应一个符合要求的数字。如果 $a \% 10^{len} > x$, 则说明以 $a / 10^{len}$ 作为较高位不在范围内; 如果 $b \% 10^{len} < x$, 则说明以 $b / 10^{len}$ 作为较高位不在范围内。将 a 与 b 的较低 len 位截去, 判断边界条件后 (当前的 a, b 作为较高位是否在范围内, 不在则对应将其加减 1), 此时从 a 到 b 每一个数字都对应一个符合要求的数字, 所以 $b - a + 1$ 即为答案。

初赛第一场第三题: 集合的交并

3.1. 题目描述:

对于一个闭区间集合 $\{A_1, A_2, \dots, A_K\} (K > 1, A_i \neq A_j \{i \neq j\})$, 我们定义其权值

$$W = |A_1 \cup A_2 \cup \dots \cup A_K| \times |A_1 \cap A_2 \cap \dots \cap A_K|$$

其中 $|X|$ 表示 X 区间的长度; 如果 X 为空集 $|X| = 0$ 。

当然, 如果这些闭区间没有交集则权值为 0。

给定 N 个各不相同的闭区间, 请你从中找出若干个 (至少 2 个) 区间使其权值最大。

输入数据:

第一行一个整数 N ($2 \leq N \leq 10^5$)

接下来 N 行每行两个整数 l, r ($1 \leq l < r \leq 10^6$), 表示闭区间的两个端点。

输出数据:

最大权值

例如, 对于:

4

1 6

4 8

2 7

3 5

其输出结果是:

24

评分标准:

程序输出结果是否正确。

3.2. 关键思路:

首先, 显而易见地, 选出的所有区间必须有交集, 否则 W 必为0。

然后, 若对于多于两个相交区间的并, 其中必须有两个区间的并能够完全覆盖整个区间。若不然, 则这两个区间有未覆盖的子区间, 与所有集合相交矛盾。所以对于每种多于两个区间的选择, 只需选中其中两个区间便能获得同样的并集与交集。

第一题 度度熊就是要刷排名第一

1.1. 题目:

一天度度熊在 Baidu 游戏大厅中发现了一个隐藏的神奇游戏, 叫做“度度熊的逆袭”。度度熊很好奇到底是什么情况, 于是就进入了游戏。这个游戏很神奇, 游戏会给出 n 个数 A_i , 度度熊可以任意从中选取一些数, 一个数可以选任意多次。选好之后度度熊得到的分数为度度熊选出的数的 Xor (异或) 值。度度熊顿时产生了兴趣, 决心要刷至 Ranklist 的第一名。但是度度熊犯难了, 度度熊不知道自己给出的方案是不是最好的, 于是度度熊找到了你, 希望你告诉他对于某个回合, 度度熊能得到的最高分和第二高分是多少?

输入

第 1 行 1 个数 n , 接下来 1 行 n 个整数表示 A_i , ($0 \leq A_i < 231$)

$1 \leq n \leq 105$

输出

输出一行两个数, 表示度度熊能够得到的最高分和第二高分为多少

样例输入

2

5 3

样例输出

6 5

1.2. 关键思路:

显然, 题目中所说的一个数可以选任意多次是没有意义的, 因为一个数与自己的异或值为 0. 另外由于数组 A 中的数最大为 230, 换算成 2 进制最多 8 位, 也就是说最后的异或值不可能大于 255!

有以上分析, 有一个很显然的做法, 用一个数组 $mk[]$, $mk[i]$ 表示数字 i 可以用某些数的异或值来表示。那么初始的时候只有 0。然后, 每次扫一遍数组 A , 看看有没有新的值加入, 直到没有新的只加入, 循环结束!

然后从 255 到 0 扫一边找出最大值和次大值!

第二题：度度熊的礼物

2.1. 题目：

题目描述：

度度熊拥有一个自己的 Baidu 空间，度度熊时不时会给空间朋友赠送礼物，以增加度度熊与朋友之间的友谊值。度度熊在偶然的机会下得到了两种超级礼物，于是决定给每位朋友赠送一件超级礼物。不同类型的朋友在收到不同的礼物所能达到的开心值是不一样的。开心值衡量标准是这样的：每种超级礼物都拥有两个属性(A, B)，每个朋友也有两种属性(X, Y)，如果该朋友收到这个超级礼物，则这个朋友得到的开心值为 $A * X + B * Y$ 。

由于拥有超级礼物的个数限制，度度熊很好奇如何分配这些超级礼物，才能使好友的开心值总和最大呢？

输入

第一行 n 表示度度熊的好友个数。

接下来 n 行每行两个整数表示度度熊好朋友的两种属性值 X_i, Y_i 。

接下来 2 行，每行三个整数 k_i, A_i, B_i ，表示度度熊拥有第 i 种超级礼物的个数以及两个属性值。

$1 \leq n \leq 1000, 0 \leq X_i, Y_i, A_i, B_i, 1 + k_2 \leq n$

输出

输出一行一个值表示好友开心值总和的最大值

样例输入

```
4
3 6
7 4
1 5
2 4
3 3 4
3 4 3
```

样例输出

```
118
```

提示

送给第一种礼物的人有 1, 3, 4，送给第二种礼物的人有 2

保证数据不会爆 int

2.2. 关键思路：

贪心算法：在两种礼物都有的时候计算一下给朋友哪个开心值高就给哪一个；

在仅剩一种礼物的时候，就假设把这种礼物给这个朋友，然后再调整前 i 个朋友的礼物顺序，使得开心值增加最多。

①首先，显然在给第一个朋友的时候，两种礼物都有，哪个开心值高就给那个；

②然后，假设按照这种方法给了前 k 个朋友礼物，使得前 k 个朋友的开心值最高。那么，如果能证明按照这种方法给第 $k+1$ 个朋友的时候也能取得最优值，那么有归纳法就可以得知该算法的正确性！

1) 假设在给第 $k+1$ 个朋友的时候两种礼物都有，那么，由于前 k 个已经取得最优值，而且第 $k+1$ 个也取得最优值，显然可以取得前 $k+1$ 的最优值；

2) 假设在给第 $k+1$ 个朋友的时候仅剩一种礼物。其实我们是比较给第 $k+1$ 个朋友当前仅剩的一种礼物，和另一种礼物的开心值哪个比较高，但是由于没有另一种礼物了，所以唯一的可能性就是将当前的礼物与前面某个朋友的交换。然后，找到这样的一个最优值，这样就能保证前 $k+1$ 个朋友的开心值一定最高（其实有人还有一些疑问，会不会有连锁交换？绝对不会有！可以自己想一下）

综上，有归纳法可知贪心算法的正确性!!!

第三题：网页聚类

3.1. 题目：

描述

有 N ($N \leq 1000$) 个网页，我们想按照它们的相似度或差异度，把它们聚成 K ($2 \leq K \leq N$) 个类。每个网页都具有一些属性，简单起见我们认为每个网页只有三个属性： x 、 y 、 z 。归一化之后，这三个属性的取值范围都是 $[0, 1]$ 。两个网页 i 、 j 的差异度如下定义： $s(i, j) = (x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2$ 。请求出最大的 t ，每个类至少包含一个网页，并且其中任意两个位于不同类中的网页的差异度都至少为 t 。

输入

第一行包含两个整数 N 和 K ，后面 N 行每行三个实数，分别为 x 、 y 、 z

输出

最大的 t 的值，使用四舍五入在小数点后保留六位小数。

样例输入

```
5 3
0.1 0.2 0.4
0.2 0.8 0.7
0.3 0.4 0.5
0.0 0.5 0.0
0.3 0.3 0.2
```

样例输出

```
0.170000
```

3.2. 关键思路:

由题意可知两个网页的差异度的范围为 $[0, 3]$ ，且若存在 x 个聚类满足差异度条件，将它们中的任意几个合并后仍会满足条件，所以只需二分 t 的值，对于当前 t ，利用 bfs 将差异值小于 t 的归为一类，判断是否可以有 k 个不同聚类即可。

第四题 小王子的表演

4.1. 题目:

为了庆祝女王的生日，王宫前的广场上正举行着一场神枪手的表演赛。这些神枪手中包括军队里的射击天才，山中的顶级猎人，异国的神奇牛仔……来自五湖四海的高手汇聚一堂。在比赛中技压群雄的人，不仅仅能给女王的生日添上华丽的祝福，还能够获得无上的荣誉。

比赛的规则很简单。场中存在着 N 个靶子，每个枪手允许在场内任何一点向任意方向射击一次，穿透最多靶子数目的枪手就是胜利者。从广场的平面图来看，每个靶子都可以被认为是一个点，并且第 i 个靶子的运动轨迹是以点 (x_i, y_i) 为起点，点 $(x_i + a_i, y_i + b_i)$ 为终点的线段。发令枪响的那一刻，每个靶子同时从起点到终点开始匀速运动。虽然靶子各自的速度不尽相同，但是所有的靶子将会在 10 秒后同时到达终点，选手必须在这 10 秒之内（包含开始和结束的瞬间）进行射击。子弹的速度可以认为是无穷大并且射击场没有边界。

小王子偷偷地也报名参加的这次比赛，希望能在母亲的生日上表现出自己的成长。聪明的小王子早就通过观察把所有靶子的运动情况强记在心，那么，小王子最完美的射击究竟能够穿透多少靶子呢？

输入

第一行只有一个整数， N ，（ $1 \leq N \leq 50$ ）

之后每一行包含 4 个整数， x_i, y_i, a_i, b_i ，分别表示第 i 的靶子运动轨迹的起点 (x_i, y_i) ，以及方向 (a_i, b_i) ，假设这些整数的绝对值都不大于 1000。

输出

只需要输出一个整数，表示最优情况下小王子一发子弹能够击穿的靶子数目

样例输入

```
9
-14 -14 6 0
-12 -14 0 2
-10 -12 0 -2
-12 -12 2 0
-14 -14 0 6
-8 -14 0 6
-8 -8 -6 0
-13 -11 1 2
-9 -11 -1 2
```

样例输出

提示

两个靶子可能会在某些时刻重叠在一起，此时它们不会发生碰撞而是沿着各自的轨迹继续运动下去。数据中没有两个运动完全相同的靶子。

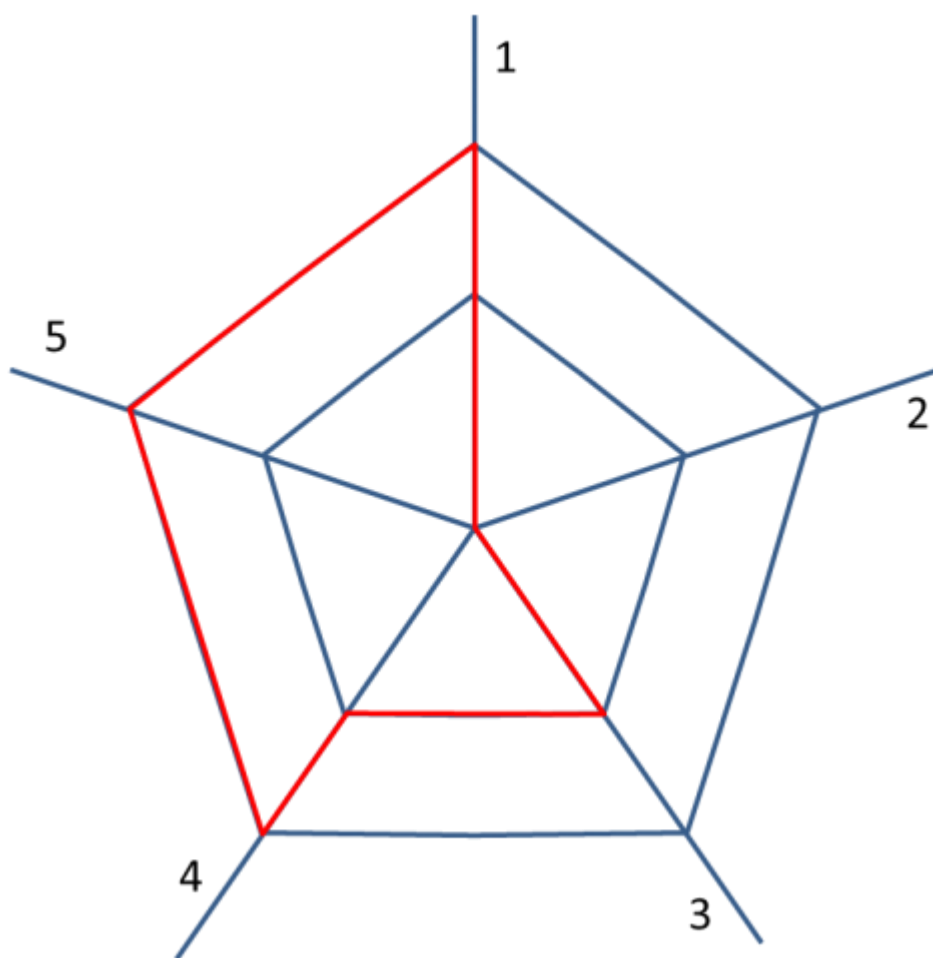
4.2. 关键思路：

N 只有 50，枚举 3 个点共线的情况，计算共线时间，及其他点的位置，统计有多少点共线应该就可以， n^4 。

复赛第一题、蛛网

1.1. 题目

存在如下图所示的蛛网，该蛛网有 N 条经线，顺时针标记为 $1..N$ 。相邻两条经线之间分布着 K 层纬线，从中心开始往外分别计为 $1..K$ ，每两层相邻纬线之间的距离固定为 D （此距离是指在经线上从一条纬线走到相邻纬线的距离，而非两条纬线在空间上的最短距离）。从蛛网中心到第 i 条经线第 1 层纬线的距离为 L_i ，经线 i 与经线 $i+1$ 之间的第 j 层纬线长度为 $X_{j,i}$ 。现在蛛网上有 M 个经纬交错的点上有食物（图中的小红点）。求问蜘蛛从图中任意一点出发，经过每条经线至多 1 次的情况下（即一旦离开了某条经线，就不能再次回到该经线上，除非出发点在该经线上，则可以在最后回到该经线上来；蛛网中心不属于任何经线的一部分），将所有食物全部收集起来并回到出发点需要经过的最短路程。在同一经线上移动时可以反复经过该经线上的每一条边无数次，蛛丝没有方向。下图中蓝色的边为经线。



输入/输出可能很大，推荐使用 `scanf/printf`

1.2. 思路：

根据题意，分析得知只能到达某一个经度一次并在该经度上一次性取完所有食物，具有很强的阶段性，采用动态规划算法解决该问题。

设状态 $f[i][j]$ 为在已经取完前 i 个经度上取完所有食物以后从第 j 个纬度出发到达下一个经度的最小代价，因为要形成一条环路，则枚举从第 n 个经度的第 s 个纬度出发到达第 1 个经度的。对于第 i 个经度而言，若该精度上没有食物，则从上一个经度上转移纬度和在当前经度上转移纬度等价，但是注意到每一个经度的 $L[i]$ 不同，所以可能导致从上个经度第 0 个纬度转移到当前经度某个纬度，或者从上个经度某个纬度转移到当前经度第 0 个纬度更优，所以要做针对于第 0 个纬度的最优化。若该经度上有至少一个食物，那么从上一个经度的任一纬度转移一定会保证取完所有食物的最小代价在纬度最高的食物或者纬度最低的食物上的一种情况，然后从这两个点转移到 $n+1$ 个纬度上，取两个点的最小值转移到每个纬度。最终状态即为 $f[n][s]$ ，取所有 s 的情况中 $f[n][s]$ 的最小值即为答案。

复赛第二题、改变数字

2.1. 题目

任意给定两个正整数 A, B ，试求一个正整数序列 $N[0], N[1], N[2], \dots, N[k]$ ，并且满足
 $0 \leq k < 500000$;
对任意的 $0 \leq i \leq k, 0 < N < 2^{62}$;
 $N[0]=A, N[k]=B$;
对任意的 $0 \leq i < k, N+N[i+1]$ 整除 $N*N[i+1]$ 。

输入

输入只有一行，是两个用空格隔开的正整数，分别表示 A, B ($3 < A, B < 30000$)。

输出

输出也只有一行。

如果有解，输出任意一组满足条件的的序列。每两个正整数之间用一个空格隔开。

否则，输出一个 0。

样例输入

28 12.

样例输出

28 168 24 12

提示

输入/输出可能很大，推荐使用 `scanf/printf`

思路：

思路一：搜索

考虑到每个数的范围 $0 \sim 2^{62}$ ，最多 500000 个数，所以搜索基本上是不可解的。

思路二：分解质因数

考虑样例：28 168 24 12

$$28 = 2 * 2 * 7$$

$$168 = 2 * 2 * 2 * 3 * 7$$

$$24 = 2 * 2 * 2 * 3$$

$$12 = 2 * 2 * 3$$

整个过程可以看作是把 7 替换为 3 的过程，首先乘上 $2*3$ ，然后换下 7，最后除掉多余的 2。从这个思路来考虑，情况很复杂，没有能理清楚。

思路三：逐一构造

考虑到如果 A 小于 B ，那个逐一在数列中构造 $A+1, A+2, A+3, \dots, B-2, B-1$ ，即可。如果 A 大于 B ，同理可以逐一构造。

性质 1: n 和 $n(n-1)$ 可以相邻。

证明：设 n 和 pn 可以相邻，则 $(n*n*p) \bmod (n+pn) = 0$ ，可得 $p = n-1$ 。

性质 2：如果 x 和 y 可以相邻，则 ax 和 ay 可以相邻（ a 是常数）。

证明： $(ax*ay) / (ax+ay) = (a^2*(x*y)) / (a(x+y))$

因为 $(x*y) \bmod (x+y) = 0$ ，所以 $(a^2*(x*y)) \bmod (a(x+y)) = a^2 \bmod a = 0$

性质 3： n 可以和 an 相邻，当且仅当 $n \bmod (a+1)$ 。（ a 是常数）

证明： n 和 an 相邻，可以推出 $n \bmod (a+1)$

性质 4： n 可以和 $n-1$ 相邻，当且仅当 n 是偶数。

证明： n 可以和 $n(n-1)$ 相邻， $n-1$ 可以和 $(n-1)(n-2)$ 相邻

可以推得 $n(n-1)$ 可以和 $n(n-1)(n-2)$ 相邻， $(n-1)(n-2)$ 可以和 $(n-1)(n-2)(n-2)$ 相邻

如果 $n(n-1)(n-2)$ 和 $(n-1)(n-2)(n-2)$ 相邻，则可以退出 n 是偶数。

性质 5： n 可以和 $n+1$ 相邻，当且仅当 n 是奇数。

证明：和性质 4 是等价的，证明同理。

除了以上性质之外，暂时没有发现其他比较好的性质。所以解题无法继续进行。

代码：

无

数据：

Input:

28 12

Output:

28 168 24 12

复赛第三题、消灭病毒

3.1. 题目

最近科学家发现了一种代号为 M 的奇怪病毒，感染 M 病毒的熊会止不住卖萌。经过不懈努力，科学家们研制出 N 种可以杀死 M 病毒的药品。

第 i 种药品在首次使用时可以杀死 D_i 个 M 病毒；以后每次使用，药品的效果会逐次递减，即第 2 次使用可以杀死 D_i-1 个病毒，第 3 次使用可以杀死 D_i-2 个病毒，第 4 次使用可以杀死 D_i-3 个病毒……直到最后不再能杀死 M 病毒（杀死 0 个）。

从现在开始，每秒钟科学家可以选择使用一种药品杀死 M 病毒。不过，目前研制出的药品还不够稳定：第 i 种药品会在从现在开始的第 C_i 秒后失去药效（不再能杀死 M 病毒），因此只能在前 C_i 秒使用（若 $C_i=0$ 则该药物在任何时候使用都没有效果）。科学家们想知道如何安排药品的使用，可以在所有药品失效前杀

死最多的 M 病毒。

输入

输入的第一行是一个正整数 $T(1 \leq T \leq 10)$ ，表示测试数据的组数。

每组测试数据的第一行是一个整数 $N(0 \leq N \leq 100000)$ ，表示药品的数目。

以下 N 行每行包含两个整数 $D_i, C_i(0 \leq D_i, C_i \leq 10^9)$ ，表示第 i 种药品首次使用能杀死的 M 病毒数和药效持续时间。

输出

每组数据输出一个整数，在所有药品失效前最多能杀死的 M 病毒的数目。

样例输入

```
3
0
1
100 1000
2
100 2
100 1
```

样例输出

```
0
5050
200
```

提示

$D_i=0$ 意味该药品对 M 病毒实际上没有药效，不能杀死任何 M 病毒。

样例第 3 组数据解释：第一秒用 2 号药品杀死 100 个病毒。第二秒的时候 2 号药品已经过期，这时用 1 号药品能再杀死 100 个病毒。第三秒的时候药品都过期了，总计可以杀死 200 个病毒。

3.2. 思路：

考虑将时间点按照 $C[i]$ 离散：

对于从第 i 个时间往后的最优策略，它的最优策略一定可以成为全局最优策略的一部分，也就是说它无后效性。

那么由归纳法得

假设第 $i+1$ 个点后的最优策略已知

对于第 i 个点后的最优策略，就是第 $i+1$ 个点后的最优策略加上 $C[i]$ 到 $C[i+1]$ 这个区间的最优策略。

第 0 个点后的最优策略即为问题的解。

所以我们的问题变为探究怎么求某个区间的最优策略。

设区间的长度为 len ，还剩下的药品个数为 m

区间的最优策略为一次次求最好的药品。

那么问题转化为求 len 次这样的操作{

取出 m 个数的最大值，将最大值减一，再放回。

往答案增加该最大值

```
}
```

len 次操作可以在 $O(n)$ 时间内完成，所以整体复杂度 $O(n^2)$

对于区间最优策略的求法还存在优化空间，可以考虑运用高级数据结构。

复赛第四题

1.3. 题目：最右交点

题目描述：

给定平面上 N 条直线 $l_1, l_2, l_3, \dots, l_N$ ，这些直线可能两两交出许多交点。请计算每条直线上最靠右(x 坐标最大)的交点的 x 坐标是多少。

输入数据：

第一行包含一个整数 N ($2 \leq N \leq 100000$)

第 2 至 $N+1$ 行每行包含 4 个整数 x_1, y_1, x_2, y_2 ($-100000 \leq x_1, y_1, x_2, y_2 \leq 100000$, $x_1 \neq x_2$)，表示该直线经过 (x_1, y_1) 和 (x_2, y_2) 两个点

输出数据：

输出 N 行，每行一个实数代表该直线上最靠右的交点的 x 坐标，保留 2 位小数。

输出 N 条直线的顺序与输入保持一致。

例如，对于：

3

0 1 2 1

1 0 0 1

2 1 1 0

其输出结果是：

2.00

1.00

2.00

评分标准：

程序输出结果是否正确。

1.4. 关键思路：

把直线的表达式转换成 $y = a * x - b$ (由于 $x_1 \neq x_2$, 所以转换一定成立), 那么两条直线的交点就是

$$x = (b_1 - b_2) / (a_1 - a_2);$$

如果把每条直线转换成 (a, b) 这样的点，就会有 n 个点，然后相当于对每个点求其他点到他的连线的斜率的最大值。

对于一个点 (a_0, b_0) ，先考虑 a 的值小于 a_0 的那些点。这里我们可以联想到 04 年 OI 论文《浅谈数形结合思想在信息学竞赛中的应用》(周源) 里提到的最大平均值问题。就是维护一个下凸的序列。

对于那些 a 值大于 a_0 的点，可以坐标变换一下，把所有的点变为 $(-a, -b)$ ，然后再处理一遍就行了。

复赛第五题

5.1. 题目：

百度楼下有一块很大很大的广场。广场上有许多轮滑爱好者，每天轮滑爱好者们都会在广场上做一种叫做平地花式轮滑的表演。度度熊也想像他们一样在轮上飞舞，所以也天天和他们练习。

因为度度熊的天赋，一下就学会了好多动作。但他觉得只是单独的做动作很没意思，动作的组合才更有欣赏性。

平地花式轮滑（简称平花），是穿轮滑鞋在固定数量的标准桩距间做无跳起动作的各式连续滑行。度度熊表演的舞台上总共有 N 个桩，而他也从自己会的动作中挑出 M 最好看的。

但事情并没有这么简单。首先，每个动作因为复杂度不同，所以经过的桩的个数、消耗的体力也不尽相同。然而度度熊的体力是有限的。

然后，为了保持连贯性，有些动作是接不起来的，所以每个动作都有他前面能接的一个动作的列表。更有甚者，有的动作要考虑前很多个动作才能确定是否能做出来。但度度熊这次把这些应该连接在一起的动作直接定为一个动作了。所以一个动作被描述为一个序列： $\{X_1, X_2, X_3, \dots, X_n\}$ 表示，做这个动作的时候会先前进 X_1 个桩，再前进 X_2 个桩（注意： X_i 可能是负数，这表示后退 $|X_i|$ 个桩）。度度熊的完整表演需要恰好停在最后一个桩后面，并且在表演过程中不允许有前进/后退方向上桩不够的情况发生。

最后，评分也很复杂。这次每个动作没有单独得分了，最终得分=组合得分+剩余体力。表演的时候，需要确定一个组合，表演过程中完成这组组合中所有的动作，同样的动作允许多次出现，但不能包含其他多余的动作。这个组合的分数就是最终得分中的“组合得分”。

举个例子，总共有 10 个桩，体力上限是 25，有以下几个动作：

动作 1: $\{1, 3, 1\}$ ，需要 5 的体力。

动作 2: $\{1, -3, 1\}$ ，需要 4 的体力。

动作 3: $\{3, 3\}$ ，需要 4 的体力。

动作 4: $\{5\}$ ，需要 3 的体力。

组合 1: (动作 1, 动作 2, 动作 4)，得分 15。

组合 2: (动作 1, 动作 3)，得分 10。

组合 3: (动作 2, 动作 3)，得分 5。

最优方案应该是动作 3+动作 2+动作 2+动作 3。最后得分：组合得分 5 分+剩余体力 9=14 分。虽然，度度熊一下就算出来自己应该怎么表演了。但是他还是想考考精通编程的你。

输入数据：

一开始一个整数 T ，表示有 T 组数据，每个数据如下格式：

第一行有四个整数， N, M, P, Q 。分别表示桩数、动作数、组合数和体力上限。

第二行 M 个整数，表示每个动作需要的体力。

接下来 M 行，每行描述一个动作。每行的第一个数 $X(1 \leq X \leq 10)$ ，表示动作分为 X 段。后面接 X 个数，这个长度为 X 的序列描述这个动作。

接下来 P 行，每行描述一个组合。每行的前两个数 $Z(1 \leq Z \leq M)$ ， Y ， Z 表示组合中总共有 Z 个动作， Y 表示组合能获得的分数。后面接 Z 个数，表示组合中包含的 Z 个动作的编号。

输出数据：

对于每个数据，输出度度熊能获得的最高分数。

评分标准：

程序输出结果是否正确。

提示

保证至少有一个方案满足要求。度度熊可以多次到达起始位置和终止位置。不存在两个组合包含完全一样的动作。

$1 \leq N \leq 1000$ ， $1 \leq M \leq 12$ ， $0 \leq P \leq 3000$ ， $1 \leq Q \leq 10000$ ，所有分数之和在 32 位有符号整数范围之内。每个动作至少需要 1 的体力。

5.2. 关键思路：

动态规划， $f[i][j]$ 表示前进 i 个桩子，当前动作状态为 j （二进制表示每个动作做了或者没做）。枚举每个动作选或者不选，根据 j 的状态确定当前剩余体力和选择一个组合来确定要求最大化的得分。具体做法用 spfa 递推就好。最终状态为 $\max\{f[n][*]\}$ ，时间复杂度为 $O(n*m*m)$ 。

百度之星 2013

区域赛第一场第一题

1.1. 题目描述：

你知道水果忍者吗？这是一个智能手机上面很流行的游戏。

这个出色的游戏迎合了人类最喜爱的运动轨迹：抛物线。就是各种各样物体被扔出去之后都会形成的曲线运动。现在这里的问题也是关于抛物线的。

现在有 N 个水果，假设它们就是一个个在平面上的圆，有着同样的半径。每个水果有着各自的位置和移动速度。由于重力的影响（这里假设重力常数 $g=10$ ），所有水果都从一开始就沿着抛物线运动。你的目标就是在给定的时间限制之内，一次切到最多的水果，即是找到在某一时刻的一条直线穿过最多的圆。

1.2. 关键思路:

可以知道肯定存在这样的最优解: 切出来的直线肯定和某两个圆相切, 否则我们可以旋转和移动直线使得它满足这个条件。于是我们可以枚举这两个圆, 然后枚举切线, 求出所有的其它圆被切线超过的时间区间, 最后做一下最大区间覆盖就可以求出在哪一个时刻这条切线切过最多水果。解方程可以用牛顿法来解。其实可以证明一个更强的结论, 就是一定存在一个最优解和某两个圆外相切。

区域赛第一场第二题

2.1. 题目描述:

Du 熊正在负责一个大型的项目, 目前有 K 台服务器, 有 N 个任务需要用这 K 台服务器来完成, 所以要把这些任务分成 K 个部分来完成, 在同上台服务器上执行的任务必须是连续的任务, 每个任务有各自需要的执行时间。

例如 $N=5, K=2$, 每个任务需要时间分别为 5, 3, 1, 4, 7 分钟, 那么我们可以分成 (5) (3 1 4 7) 两部分, 这样第一台服务器所花时间就是 5 分钟, 而第二台机器需要花 15 分钟, 当然, 所有任务完成的时间是按最迟完成的那台服务器的时间, 即这样划分的话完成所有任务所需要的时间就是 15 分钟。而另外一种划分方法是 (5 3 1) (4 7), 这种划分方案完成所有任务的时间就是 11 分钟, 也是最优的一种划分方案。

现在你的任务就是根据给定的 N, K 和每个任务要花费的时间, 找出使完成所有任务时间最短的方案。

输入数据:

多组输入。

第一行输入 N 和 K ($1 \leq K \leq N \leq 10000$)。

第二行输入 N 个不大于 1000 的正整数, 表示各个任务要花费的时间。

$N=K=0$ 表示输入结束。

输出数据:

每行输出一个整数, 对应对于每个数据 (除了 $N=K=0$ 不用输出)。

2.2. 关键思路:

此题可用二分答案 x 的方法解决, 即对每个时间长度 x , 判断是否有可能划分任务使得最长所需时间不超过 x 。由于在同一台服务器上执行的任务必须是连续的任务, 因此对每个时间

区域赛第二场第一题

1.1. 题目：

du 熊正在玩一个别人刚送给它的机器人。这个机器人只能在一个棋盘行走，棋盘的左上角格子为(0, 0)，右下角格子为(X, Y)。

du 熊控制这个机器人从棋盘的左上角，走到右下角，再从右下角回到左上角。当机器人从左上角走到右下角的过程中，如果它当前所在格子为(x, y)，则它只能走到(x+1, y)或(x, y+1)的格子；当机器人从右下角走回左上角的过程中，如果它当前所在的格子为(x, y)，则它只能走到(x-1, y)或(x, y-1)的格子。并且 du 熊要求机器人从左上角走到右下角再走回左上角的整个过程中，最多经过同一个格子一次。

请你帮 du 熊计算出这个机器人一共有多少种不同的路线。

Input

输入的第一行为一个正整数 T ($0 < T \leq 50$)，表示数据组数。

接下来的 T 行，每行有两个整数，分别表示 X 和 Y。 ($1 \leq X, Y \leq 1000$)

Output

对每组输入数据，输出一个整数，表示机器人不同的路线数量。

1.2. 关键思路：

区域赛第二场第二题

2.1. 题目：

aCYL 决定要教导 du 熊学习乘法。于是 CYL 需要生成两个整数来让 du 熊做乘法。于是 CYL 找到了他在镜面世界中的朋友-----LYC 来帮忙。CYL 有 m 根现状标有数字的棍子，一个在左边一个在右边。而 LYC 有 n 根这样的棍子，而镜面世界中的某些棍子和现实世界中的某些棍子存在着某种神秘相连的关系。

CYL 通过以下方式生成两个数字：

一开始，CYL 初始化两个累加器 A 和 B 为 0，然后重复以下步骤：

CYL 首先选取一根现实世界中的棍子，暂且称之为棍子 X。然后助手 LYC 会选取一根与 X 存在神秘相连关系的棍子 Y，如果 LYC 找不到任何一根与 X 神秘相连的棍子则本次过程跳过然后算法结束。否则棍子 X 左边的数字会加入到 A 中，右边的数字加入到 B 中，而棍子 Y 左边的数字会加入到 B 中，右边的数字加入 A 中。

最终算法结束的时候 CYL 就得到了两个整数 A 和 B。

为了考验 du 熊，CYL 和 LYC 将会选取尽量多的棍子，即如果最多可以选取 10 根棍子，那么 CYL 和 LYC 将不会考虑导致最后只能选取少于 10 根棍子的选取方式。

由于镜面世界中没有熊 ud，所以可怜的 du 熊没有助手，于是它只能算出所有可能的 $A*B$ 值中最小的一个。

现在要求你找出 du 熊可以算出来的唯一结果。

输入格式

第一行一个整数 T，代表测试数据个数。

接下来每个测试数据开头是两个整数 m 和 n ($1 < m, n \leq 30$)，然后接下来一行共有 $2*m$ 个整数，第 i 对整数分别代表现实世界中第 i 根棍子对应的左边和右边的数字。

接下来一行通过同样的格式给出了镜面世界中 n 根棍子上左边和右边的数字。

之后一个整数 C ($0 < C \leq 500$) 表示共有 C 个相连关系，接着 C 行每行包含两个整数: a 和 b，表示现实世界中第 a+1 根棍子和镜面世界中第 b+1 根棍子间存在相连关系(显然 a 的范围是 $0..m-1$, 依此类推)。

输出格式

对于每个测试数据输出一行，表求 du 熊可以求出的乘法结果。

2.2. 关键思路:

这题求的是所有可以得到的整数对 (A, B) 中乘积最小的一个。如果把所有可以得到的 (A, B) 都看作平面上的一点，可以发现乘积最小的 (A, B) 一定在所有这些点的下凸包上。即，一定存在实数对 $u \geq 0$ 和 $v \geq 0$ ，使得乘积最小的 (A, B) 具有最小的 $u*A + v*B$ 的取值。我们实际关心的是斜率 u/v 。

于是，我们首先需要枚举所有潜在的斜率 u/v ，对每个斜率，找出具有最小的 $u*A + v*B$ 取值的整数对 (A, B)，计算其乘积，从而找出具有最小乘积的最终答案。实际需要尝试的斜率个数最多只有 $O(n^2)$ 个，每两个棍子对应一个需要尝试的斜率。

对于每个需要尝试的斜率，我们把所有的棍子按照 $u*a+v*b$ 排序（ a 和 b 表示这个棍子对 A 和 B 所加的数），然后优先选择取值较小的棍子。由于同一个棍子不能重复选取，并且要选取尽可能多的棍子，因此需要使用二分图最大匹配算法来求出使 $u*a+v*b$ 总和最小的最大匹配。

区域赛第三场第一题

1.1. 题目：

a 度熊来到了一个陌生的城市，它想用最快的速度从 A 地到达 B 地，于是，他打开了百度地图。地图上有 N 个点，有若干道路，每条道路将连续的若干个连接起来。

在道路上，度熊可以采用步行的方式。除了步行这种方式外，地图上还有若干公交和地铁路线。

为方便模拟实际中公交或地铁运行的情况，我们假定这些路线都是环状的，车辆从路线起点开始，就沿着路线一直运行，直到回到起点才停止。

每条路线会在时刻 t_1, t_2, \dots, t_k 从路线起点分别发出一辆车，满足 $t_1 \leq t_2 \leq \dots \leq t_k$ ，当 t_i 的间隔足够小的时候，就会出现多辆车同时在运行的情况。

度熊如果选择乘坐公交或者地铁的话，它需要考虑候车时间。即，它通过步行到达某条公交或地铁路线的点之后，如果它想乘坐车辆，则必须等路线中的某一辆车到达该点之后才能乘车。

为了从 A 点到达 B 点，度熊可以采用步行，乘车，步行，乘车... 交替进行的方式，即，可以在某个点从地铁换乘公交，或者步行等等。

现在，给出地图中的步行路线和乘车路线，以及度熊所在的地点 A ，你能算出度熊从 A 点出发分别到达 N 个点所需要的最短时间吗？

Input

输入数据的第一行为一个整数 T ($1 \leq T \leq 100$)，表示有 T 组测试数据。

每组测试数据第一行为 5 个整数 N, M, K, A, S 。 N 代表地图上点的个数， M 代表道路的条数， K 代表公交或地铁路线的条数， A 是起点。

S 是度熊的出发时间。

$2 \leq N \leq 1000, 1 \leq M \leq 100000, 1 \leq K \leq 50, 1 \leq A, B \leq N, A \neq B, 0 \leq S \leq 100000$ 。

接下来是 M 行，每行三个整数 u, v, w 。表示从 u 点到达 v 点的步行时间为 w 。这是双向边， v 点到达 u 点的步行时间也为 w 。可能会有重边的情况，此时应考虑最小花费的边。

$1 \leq u, v \leq N, 1 \leq w \leq 10000$ 。

接下来是 $K * 4$ 行，每 4 行代表一条线路。

线路的第一行是两个整数 h, k 。 h 代表路线中点的个数， k 代表发车的数量。

$2 \leq h \leq N, 1 \leq k \leq 1000$

第二行是 h 个整数 p_1, p_2, \dots, p_h ，代表路线中按行车顺序排列的 h 个点， p_1 为发车地点。

$1 \leq p_i \leq N$

第三行是 h 个整数 d_1, d_2, \dots, d_h ， d_i 代表车辆从点 p_i 到达点 $p_{(i+1)}$ 所花费的时间（ $p_{(h+1)}$ 代表点 p_1 ）。

$1 \leq d_i \leq 10000$

第四行是 k 个整数 t_1, t_2, \dots, t_k ，代表 k 个发车的时间。

$0 \leq t_1 \leq t_2 \leq \dots \leq t_k \leq 100000$ 。

Output

对每个测数数据，在一行内输出 N 个数字，用空格分割（行末不要有空格），表示从 A 点出发，到点 $1, 2, \dots, N$ 分别所花费的最少时间。如果无法到达某个点，请输出 -1 。

1.2. 关键思路：

这题是简单的最短路问题，跟经典的最短路问题不同的是两点之间的距离（消耗时间）是不确定的，跟前一点的最早到达时间有关，但仍然可以通过简单的 Dijkstra 算法求解。

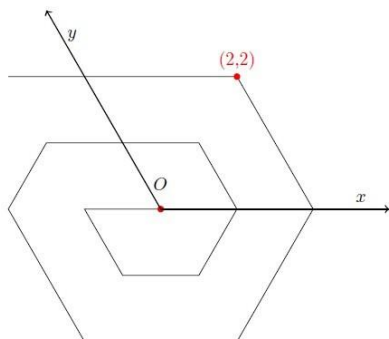
区域赛第三场第二题

2.1. 题目：

a 小 W 是一个宅男，喜欢发呆，并幻想一些不切实际的事情。

今天，小 W 又开始做他的白日梦了。他梦见他被困在了一条隧道里，周围漆黑一片。作为一个宅男，小 W 自然地掏出了手机，打开定位系统，确定了他的位置。又由此在网上搜索到了关于隧道的信息。

这条隧道是由一个点向外，呈六角螺旋形展开，并且没有其他的支路。最小的一圈每条边的长度都是 1 米，边长向外依次增大到 $2, 3, \dots$ 米，如下图所示。



如果以螺旋的中心为原点，按图中所示建立坐标系，那么隧道中的每一个点都可以用坐标来标定，如图中所指点的坐标为 $(2, 2)$ 。小 W 对这种方法了如指掌，轻易地算出了他所处位置的坐标为 $X1, Y1$ ，他还在网上查到了隧道的出口位置在 $X2, Y2$ ，这样他就可以逃出隧道了。但是，小 W 是个宅男，他可不急着出去，倒是想知道从他所在的点要走多远才能到出口。因为距离很远，他觉得这个问题有困难，你能帮他解决吗？

Input

输入数据以一个整数 $T < 104$ 开头，表示测试数据组数。以下每行为一组测试数据，包括 4 个整数 $X1, Y1, X2, Y2$ ，描述了小 W 所在的位置和出口位置的坐标。所有坐标的绝对值不超过 1018 ，并保证答案不超出 64 位有符号整数的表示范围。

Output

对于每组测试数据输出一行，一个整数，表示小 W 所在的位置和出口的距离。

2.2. 关键思路:

首先对两个坐标分别反解出它们在螺旋线上的位置，再计算位置的差值就可以了。因为一共只有六个不同方向的边，因此只要针对每个方向编写代码，计算坐标在某个方向时的位置就行了。

区域赛第三场第三题

3.1. 题目:

a 度度熊最近一直在陪小朋友玩，有一天，他放到桌子上的糖不见了，而屋里一共有 N 个小朋友，于是度度熊想知道是谁拿走了糖果，于是他就去问 N 个小朋友。

每个小朋友都会告诉他一句话。这句话的形式为糖果是 A 拿的，或者是糖果不是 A 拿的。

不幸的是一些小朋友会说谎，度度熊知道恰好有 K 个小朋友说谎，现在他想请你帮帮忙，帮他算出是谁拿走了糖果。

InputFormat

第一行为数据组数 $T(1 \leq T \leq 100)$ 。

每组数据第一行为整数 $N(1 \leq N \leq 10000)$ 和 $K(1 \leq K \leq N)$ 表示小朋友的个数和说谎的小朋友个数。之后 K 行每行由两个数字 a, b 组成， $a = 1$ 表示该小朋友告诉度度熊是 b 拿走了糖果， $a = 0$ 表示该小朋友告诉度度熊不是 b 拿走了糖果。小朋友的编号从 1 开始。

OutputFormat

每组数据输出一个整数，为拿走糖果的小朋友编号

如果没有符合条件的小朋友，就输出 "0"

如果有不止一个小朋友符合条件，那么就输出 "-1"

3.2. 关键思路:

只需要计算出当第 i 个小朋友拿走糖果时总的说谎的人数 $P[i]$ ，就可以得到答案了。可以在扫描一遍数据的过程中计算出所有的 $P[i]$ ，时间复杂度为线性。

月赛第一场第一题

1.1. 题目:

abaidu 熊最近在学习随机算法，于是他决定自己做一个随机数生成器。

这个随机数生成器通过三个参数 c, q, n 作为种子，然后它就可以通过以下方式生成伪随机数序列：

$$m_0 = c,$$

$$m_{i+1} = (q^2 m_i + 1) \bmod 2^n, \quad \text{for all } i \geq 0.$$

因为一些奇怪的原因， q 一定是奇数。现在 du 熊想知道对于一个给定的数 x ，是不是会出现在这个伪随机数序列里面，如果存在的话，他还想知道最早是在哪里出现，即给定一个整数 x ，要求找出一个最小的整数 k 满足 $m_k = x$ 。

输入格式

输入包含多组数据。

每个测试数据包含一行三个整数： c ， q ， n ， x 。

数据满足 $0 \leq c < 2^n$ ， $0 \leq q^2 < 2^{63}$ ， $0 \leq n \leq 63$ ， $0 \leq x < 2^{63}$ 。

输入以文件结束符结尾。

输出格式

对应每个测试数据输出满足条件的 k ，如果 x 不会出现在序列里面的话，就输出-1。

1.2. 关键思路：

a 设 $X(0)=0$ ， $X(i+1) = (X(i) * q^2 + 1)$ ，有 $X(n) = 1 + q^2 + q^4 + \dots + q^{2(n-1)}$ ($n \geq 1$)，即等比数列求和。

因为 q 是奇数，因此和 2^n 互质，于是 $X(n) \bmod (2^n)$ 的周期就是 2^n ，即 $X(0), X(1), \dots, X(2^n - 1)$ 是 $0 \sim (2^n - 1)$ 的一个排列，因此只要 $0 \leq x < 2^n$ 就必定有解，否则没有解。且若 $X(r) \bmod (2^n) = x$ ($0 \leq r < 2^n$)，那么对所有非负整数 k 有 $X(k * 2^n + r) \bmod (2^n) = x$ ，其中 r 就是最小的一个，对于其它的位置 i ， $X(i) \bmod (2^n)$ 都不等于 x 。

用 $L(x, y)$ 表示 x 的二进制表示的最低 y 位，即 $L(x, y) = x \bmod (2^y)$ 。

用 $R(x, y)$ 表示满足 $X(R(x, y)) \bmod (2^y) = L(x, y)$ 的最小位置。

那么由上面的过程可知只有 $R(x, y) + k * 2^y$ 的形式满足 $X(R(x, y) + k * 2^y) \bmod (2^y) = L(x, y)$ ，显然 $0 \leq R(x, y) < 2^y$

于是我们考虑 $y+1$ 的情况，由 $L(x, y)$ 定义可以知道， $L(x, y+1) \bmod (2^y) = L(x, y)$

因此， $X(R(x, y+1)) \bmod (2^y) = X(R(x, y)) \bmod (2^y)$ ，因此 $R(x, y+1)$ 满足 $R(x, y+1) = R(x, y) + k * 2^y$ ，而 $0 \leq R(x, y+1) < 2^{y+1}$ ，因此只有两种情况：

$R(x, y+1) = R(x, y)$ 或者 $R(x, y+1) = R(x, y) + 2^y$

由于 $X(n)$ 可以通过快速求等比数列的方式快速得到(例如用矩阵乘法)，因此我们可以这两种情况都求一下再比较一下就可以得到 $R(x, y+1)$ 了。

月赛第一场第二题

2.1. 题目：

a 小 H 是一个程序员。他很喜欢做各种各样的数学题，尤其喜欢做《水泥数学》。

在看了《水泥数学》的 2.5 章后，小 H 终于会用 9 种计算 $1^2+2^2+\dots+n^2$ 了！这两天，他一直在思考一个加强的问题。他想要计算 $1^k+\dots+n^k$ 。

通过思考，他发现对所有 k ， $P(n)=1^k+\dots+n^k$ 可以表示成一个最高次数为 $k+1$ 的有理系数多项式。比方说当 $k=1$ 时 $P(n)=n(n+1)/2$ 。

现在，对某个 k ，小 H 想知道 $P(n)$ 的系数。

Input

第一行为一个整数 t ($1 \leq t \leq 31$)，表示有 t 组测试数据；

下面 T 行每行一个正整数 k ($0 \leq k \leq 30$)，表示一组数据。

Output

对于每个输入 k ，输出一行 $k+2$ 个分数，依次给出此时 $P(n)=a_{k+1}n^{k+1}+\dots+a_1n+a_0$ 的系数 a_{k+1}, \dots, a_0 。所有分数必须以“ a/b ”的形式给出，其中 a 和 b 为整数且互质， $b>0$ ；如果某一项为 0，输出“ $0/1$ ”。

2.2. 关键思路：

因为一共有 $k+2$ 个参数，只需要构造 $k+2$ 个等式，然后求解线性方程就可以了。

月赛第一场第三题

3.1. 题目：

aCYL 决定要教导 du 熊学习乘法。于是 CYL 需要生成两个整数来让 du 熊做乘法。于是 CYL 找到了他在镜面世界中的朋友——LYC 来帮忙。CYL 有 m 根现状标有数字的棍子，一个在左边一个在右边。而 LYC 有 n 根这样的棍子，而镜面世界中的某些棍子和现实世界中的某些棍子存在着某种神秘相连的关系。

CYL 通过以下方式来生成两个数字：

一开始，CYL 初始化两个累加器 A 和 B 为 0，然后重复以下步骤：

CYL 首先选取一根现实世界中的棍子，暂且称之为棍子 X。然后助手 LYC 会选取一根与 X 存在神秘相连关系的棍子 Y，如果 LYC 找不到任何一根与 X 神秘相连的棍子则本次过程跳过然后算法结束。否则棍子 X 左边的数字会加入到 A 中，右边的数字加入到 B 中，而棍子 Y 左边的数字会加入到 B 中，右边的数字加入 A 中。

最终算法结束的时候 CYL 就得到了两个整数 A 和 B。

为了考验 du 熊，CYL 和 LYC 将会选取尽量多的棍子，即如果最多可以选取 10 根棍子，那么 CYL 和 LYC 将不会考虑导致最后只能选取少于 10 根棍子的选取方式。

由于镜面世界中没有熊 ud，所以可怜的 du 熊没有助手，于是它只能算出所有可能的 $A*B$ 值中最小的一个。

现在要求你找出 du 熊可以算出来的唯一结果。

输入格式

第一行一个整数 T，代表测试数据个数。

接下来每个测试数据开头是两个整数 m 和 n ($1 < m, n \leq 30$)，然后接下来一行共有 $2*m$ 个整数，第 i 对整数分别代表现实世界中第 i 根棍子对应的左边和右边的数字。

接下来一行通过同样的格式给出了镜面世界中 n 根棍子上左边和右边的数字。

之后一个整数 C ($0 < C \leq 500$) 表示共有 C 个相连关系，接着 C 行每行包含两个整数: a 和 b，表示现实世界中第 a+1 根棍子和镜面世界中第 b+1 根棍子间存在相连关系(显然 a 的范围是 $0..m-1$, 依此类推)。

输出格式

对于每个测试数据输出一行，表求 du 熊可以求出的乘法结果。

3.2. 关键思路:

跟之前的题是同一道。

月赛第二场第一题

3.3. 题目：

a 小 H 是一个程序员。但是他很喜欢一些新奇的东西。

有一次，他去找物理实验室的朋友玩。他见到了一串非常有意思的粒子。N 个粒子排成一排。每一秒中，每一段连续的粒子中会随意有一个爆炸，爆炸后该粒子就消失了，且将原来连续的一段粒子分隔成两段。

小 H 希望知道所有粒子都爆炸完的期望时间。

Input

第一行为一个整数 T ($1 \leq T \leq 400$)，表示有 T 组测试数据；

每组数据一个正整数 N ($1 \leq N \leq 400$)，表示一开始的粒子数。

Output

对于每组数据，输出期望时间（秒）。保留五位小数。

1.1. 关键思路：

简单的递推，只需要计算出 $P[i][j]$ ，即一段 i 个粒子在第 j 秒完全消失的概率，就可以了。时间复杂度 $O(N^3)$ 。

月赛第二场第二题

2.1. 题目：

小 H 是一个程序员，但他的生活不局限在写程序。

有一天，他走到公园散步。他见到公园的一棵苹果树上结满了苹果。他于是拿起石头，想砸几个苹果下来当第二天的早餐。突然他思考到了一个问题：怎样才能一次砸到最多的苹果呢？

考虑该局面是这样一个模型：所有东西位于二维笛卡尔坐标系，其中小 H 位于原点，苹果们分别在坐标系的整点上。石头飞出的轨迹是一条经过原点的抛物线，确切的说，经过的是 $y=ax^2+bx$ 的抛物线 ($a < 0$)。石头砸到一个苹果后，该苹果会落下，且石头不会改变运动轨迹。

现在小 H 希望求扔一个石头最多砸到的苹果数。

Input

第一行为一个整数 T ($1 \leq T \leq 10$)，表示有 T 组测试数据；

每组数据第一行一个正整数 N ($1 \leq N \leq 2000$)，表示苹果数。下面 N 行每行两个整数给出每个苹果的坐标 x_i, y_i ($1 \leq x_i \leq 1000, 1 \leq y_i \leq 1000000$)。

Output

对于每组数据，输出最多可能砸到的苹果数。

2.2. 关键思路:

把抛物线方程 $y=ax^2+bx$ 中的 a 和 b 作为平面的横坐标和纵坐标，则每个苹果决定了该平面上的一条可行直线，只需要计算出直线数量最多的交点就可以了。

月赛第三场第一题

1.1. 题目:

a 小 H 是一个程序员。

有一次他要用英语给 boss 写报告。他认为排版非常重要，所以他希望排出更好看的文章。在这里，我们考虑文章只有单词。我们需要插入空格将这些单词排版成若干行，且满足下面条件：

1. 一行包含恰好 W 个字符，字符是单词中的字母或空格；
2. 文本必须以原顺序出现；
3. 两个相邻单词至少由一个空格隔开；
4. 一个单词必须放在同一行，且不能被空格隔开；
5. 每一行的开头和结尾两个字符不能是空格，最后一行的末尾除外。

小 H 希望排版尽量美观，所以他希望令**最长的连续空格的长度**尽量短。这里忽略最后一个单词后面的所有空格（注意这个单词是不要求靠边的）。

Input

第一行为一个整数 T ($1 \leq T \leq 50$)，表示有 T 组测试数据；

每组数据第一行两个正整数 W, N ($3 \leq W \leq 80000, 2 \leq N \leq 50000$)，表示排版宽度和单词数。

数据第二行给出三个整数 X, Y, Z 。你需要根据三个数生成每个单词的长度 L ，其中，令：

$$P = \text{floor}((W-1)/2)$$

$$L[1] = (X \bmod P) + 1$$

$$L[i] = ((L[i-1] * Y + Z) \bmod P) + 1$$

Output

对于每组数据，输出最小可能的“最长连续空格长度”。

1.2. 关键思路:

可以二分答案 d ，然后对每个尝试 d 判断是否可以在最长连续空格长度不超过 d 的情况下排版。判断所需的时间是线性的。

月赛第三场第二题

2.1. 题目:

a 有一个数列 $a_k, k=1, 2, \dots, n$ ，出于某些原因，我们现在要对其进行如下操作：

$S_0 = \Phi$

若已求出 S_{k-1} ，则

$$b_k = (\dots(((S_{k-1,1}^2 S_{k-1,2} + S_{k-1,2}^2) S_{k-1,3} + S_{k-1,3}^2) \dots) S_{k-1,|S_{k-1}|} + S_{k-1,|S_{k-1}|}^2 + a_k) \mod p$$
$$S_k = S_{k-1} \cup \{b_k\}$$

其中 $S_{k-1, j}$ 表示 S_{k-1} 中第 j 小的元素，质数 $p=109+7$ 。

Input

测试数据仅包含一个测试点，第一行一个整数 n ， $1 \leq n \leq 105$ ，如题目描述中所述。

第二行 n 个整数，依次为 a_1, a_2, \dots, a_n 。

Output

输出 n 行，每行一个整数，表示 b_k 。

2.2. 关键思路:

维护一颗长度为 MOD 的线段树，每个区间维护两个状态： mul 和 add ，对于最细粒度的区间 $[x, x]$ ， mul 即为 x ， add 即为 $(x*x)\%MOD$ 。当求得一个区间 $[x, y]$ 的两个子区间 $[x, mid]$ 和 $[mid+1, y]$ 的 mul 和 val 后，进行合并即可，合并公式为： $MUL=MUL_L*MUL_R$ ； $ADD=ADD_L*MUL_R+ADD_R$ 。虽然线段树的总长度 MOD 很大，但是由于线段树初始为空，切修改操作为 n 次，所以总的时间和空间复杂度都是 $O(n \lg n)$ 。

决赛

1.1. 题目:

A. 语音识别系统简介:

(1) 语音识别系统:

语音识别系统的主要功能，是将语音转换成文字。目前，主流的语音识别系统都采用基于统计模式识别的方法。统计语音识别的目标是对给定的语音寻找最可能的词序列。

设 $O = \{O_1, O_2, \dots, O_n\}$ 是提供给语音识别系统的输入语音。 $W = \{W_1, W_2, \dots, W_m\}$ 为一个词序列。语音识别系统必须选择一个词序列 W 使得其对给定的语音序列 O ，在所有的词序列 W 中最优。

即:

$$W = \operatorname{argmax} (p(W|O))$$

$p(W|O)$ 被称为后验概率，它表示在已知声学观察 O 的情况下，词序列 W 的概率，如今绝大部分的语音识别系统采用上述方式，选择最优词序列假设，并得到相应的识别结果。

(2) 计算后验概率 $p(W|O)$:

对于任意一种语言来说，其可能的词序列是无穷的，要在其中选择最优词序列不太可能，直接根据上述公式得到识别结果非常困难。根据贝叶斯准则，

$$p(W|O) = p(O|W) * p(W) / p(O)$$

在观察序列 O 已知的情况下， $p(O)$ 可以看做为常数，因此公式可以化简为:

$$W = \operatorname{argmax} (p(O|W) * p(W))$$

其中， $p(O|W)$ 称为声学模型得分， $p(W)$ 称为语言模型得分。

在本题中，语音识别的公式正是上面的公式。

B. 词和词典

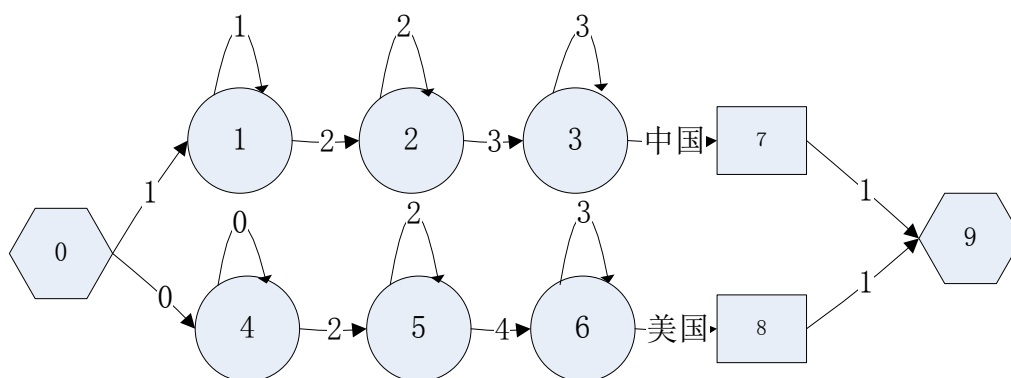
上面所述的词序列 $W = \{W_1, W_2, \dots, W_m\}$ ，其中的 W_i 在实际系统中不可能无限穷举，需要限制其在某一个有限的词集合内，称之为词典。

在中文里，由于词之间没有空格，并没有一个自然的“词”的定义。类似中文搜索系统，通常会采用某一个粒度的词来构造词典。词典规模通常在几万到几十万词之间。

C. 解码器与搜索空间

(1) 搜索空间:

搜索空间是根据声学模型得分的计算方式生成的一张图，一个简单的语音识别搜索空间如下



该图上，起点为 0，终点为 9。

图上的点，除起点和终点以外，分为两类，词尾节点以及非词尾节点；前者有 7、8，其他的都是非词尾节点。

图上的边都是有向边，通过这些边上的信息，可以计算得到一条路径对应的声学模型得分。边也相应的分两种：

一种是到达词尾节点的边。这样的边不会带来声学模型得分，而是会对应一个词。

其他的边会带来声学模型得分。但上图中所示的边上的数字不直接代表得分，而是代表对应的声学模型 ID。实际得分，它不仅跟声学模型 ID，还跟当前这条边对应的帧数有关。稍后会详细的介绍声学模型得分相关的内容。

(2) 解码器：

解码器的任务就是在搜索空间中，利用声学模型得分和语言模型得分，寻找最优路径。

这条最优路径就是从起点到终点的所有路径中，声学模型得分和语言模型得分之积最高的那条路径。这条路径上会有若干到达词尾节点的边，将这些边对应的词连起来，就是语音识别的结果。例如，还是对于上面的图：

如果最优路径序列为 0, 1, 1, 2, 2, 2, 3, 7, 9，那么最终识别结果为**中国**，如果最优路径序列为 0, 4, 5, 5, 5, 6, 6, 8, 9，那么最终识别结果为**美国**。

D. 语言模型得分的计算

(1) n-gram 语言模型：

在本题中，我们仅使用 n-gram 语言模型。该模型的假设是，每个词的出现概率只与它本身、以及它之前的至多 n-1 个词相关。

例如，我们现在使用 3-gram 语言模型。当前要计算语言模型得分的句子是“北京天安门广场升旗仪式”，根据联合概率的计算方法，以及 n-gram 假设，该句子的概率是 $p(\langle s \rangle) * p(\text{北京} | \langle s \rangle) * p(\text{天安门} | \langle s \rangle \text{北京}) * p(\text{广场} | \text{北京天安门}) * p(\text{升旗} | \text{天安门广场}) * p(\text{仪式} | \text{广场升旗}) * p(\langle /s \rangle | \text{升旗仪式})$

这里有两个特殊符号， $\langle s \rangle$ 表示句首， $\langle /s \rangle$ 表示句尾，计算语言模型得分的时候需要把这两个词添加上去。

为了表述方便，称 ABC 这样一个连续三个词的组合为一个三元文法，类似的可以定义 n 元文法；而 $P(C|AB)$ 称为这个三元文法的概率。一个 n-gram 模型需要包含很多的一元、二元、……、n 元文法。

(2) 回退及回退权：

如果不加以限制的话，n-gram 语言模型可以包含 V^n 个 n 元文法，其中 V 是词典大小。这在使用中是不可接受的，通常也没有足够的语料来训练这么大的模型。

解决方法：我们在模型中只保留最常出现的那些文法，对于不常出现的 i-gram 来说，回退到 i-1gram，并采用“回退”的方式来计算。例如 $p(\text{仪式} | \text{广场升旗})$ 假如属于不常出现的 3-gram，这样在语言模型中不能直接得到概率，则回退到 2-gram： $p(\text{仪式} | \text{广场升旗}) = p(\text{仪式} | \text{升旗}) * b(\text{广场升旗})$ 。如果 $p(\text{仪式} | \text{升旗})$ 还不能直接得到概率，则 $p(\text{仪式} | \text{广场升旗}) = p(\text{仪式}) * b(\text{升旗}) * b(\text{广场升旗})$ 。依次类推，任意的语言模型概率都可以如此求得。

这里的 $b(\text{广场升旗})$ 是这个二元文法的“回退权”，也在语言模型中一并有记录。回退权是和对应文法的概率一同被记录的，如果语言模型中不存在该文法，则对应的回退权为 1。

一元文法永远存在，不会使用回退方式进行计算。

(3) 语言模型权重：

由于实际中的语言模型得分和声学模型得分单位不同，需要在获取 n-gram 语言模型中得分之后，再给他一个权重，才是真正的语言模型得分。权重以幂的形式体现，即：

$$p(W) = (p_{n\text{-gram}}(W))^k$$

在本题中，k 取 12

(4) n-gram 的例子

假设是 4-gram 语言模型。

$p(\text{北京海淀区西二旗地铁站}) = p(\text{北京} | \langle s \rangle) * p(\text{海淀区} | \langle s \rangle \text{北京}) * p(\text{西二旗} | \langle s \rangle \text{北京海淀区}) * p(\text{地铁站} | \text{北京海淀区西二旗}) * p(\langle /s \rangle | \text{海淀区西二旗地铁站})$

假设四元文法北京海淀区西二旗地铁站不存在；三元文法海淀区西二旗地铁站也不存在；但二元文法西二旗地铁站存在，则

$$p(\text{地铁站}|\text{北京海淀区西二旗})=p(\text{地铁站}|\text{海淀区西二旗})*b(\text{北京海淀区西二旗})=p(\text{地铁站}|\text{西二旗})*b(\text{北京海淀区西二旗})*b(\text{海淀区西二旗})$$

E. 声学模型得分的计算

(1) 声学模型 ID:

之前已提到过，当搜索空间中的边到达的是非词尾节点时，对应的边上存储了声学模型 ID 信息，利用声学模型 ID 就可以实时计算出该边对应的声学模型得分。将一条路径上所有有声学模型得分的边的得分相乘，就得到了该条路径对应的声学模型得分。

(2) 声学特征及声学模型得分:

语音识别系统，会对输入语音序列，提取声学特征序列。这是一个时序的序列；每一时刻的特征，称为一帧特征。对于每一帧特征，如果已知当前的声学模型 ID，就可以计算出对应的声学模型得分。

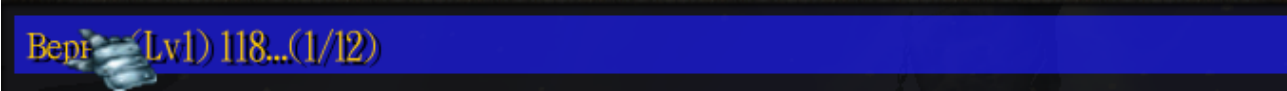
例如在上图中，如果语音包含 6 帧特征，对于每一帧特征，对于每个声学模型 ID，声学模型得分的 ln 对数，列表如下

声学 ID \ 帧 ID	0	1	2	3	4	5
0	-5	-1	-3	-3	-3	-5
1	-1	-2	-3	-3	-4	-4
2	-2	-3	-4	-4	-5	-3
3	-3	-4	-4	-4	-1	-2
4	-4	-5	-5	-5	0	-1

那么，如何确定在搜索的时候是使用哪一帧特征呢？很简单，所有起点出发的边，其帧数是 0。每经过一条到达普通节点的边，帧数就增加 1；经过一条到达词尾节点的边，帧数不变。

例如，还是在之前的那张图中。途径序列 0, 1, 2, 2, 2, 3, 7, 9 的路径得分的 ln 对数，如上表中飘红所示为-1, -3, -4, -4, -1, -4。那么最终的声学模型得分为 e^{-17} 。

在本题中，为了简化问题，在输入中略过前面的步骤，将直接给出帧 ID/声学 ID 和声学模型得分的对应矩阵，如上面已给出的矩阵那样。



F. 总结:

上面描述了搜索空间、语言模型以及声学模型。

而在搜索空间中，每一条从起点到终点的路径都对应一条声学模型序列和一条词序列，声学模型序列就是路径上所有经过的到达普通节点的边上的声学模型 ID，词序列就是所有经过的到达词尾节点的边上的词。路径的得分就是两条序列各自的得分的乘积。词序列的得分，也就是语言模型得分，使用 n-gram 语言模型计算。而声学模型序列的得分则根据其声学模型 ID 和帧数，通过查矩阵的方式来进行计算。有了这些信息，现在就可以构造一个解码器了，从输入数据得到输出的识别结果。

输入:

首先提供搜索空间、语言模型，格式在下面列出。

接着，对于每一组语音，提供其帧 ID/声学 ID 和声学模型得分的对应矩阵。

输出:

程序首先从标准输入依次读入搜索空间、语言模型，然后可在时间、内存限制下进行若干预处理操作，当初始化操作完毕以后，请立刻输出一行” ready”（检测预处理时间就是根据输出的时间来判定）。在预处理的时候不允许处理任何语音矩阵数据，否则视为犯规。

接着，读入语音的对应矩阵数据。对于每一组语音，输出一行，包含若干整数，整数间以空格分隔。这些整数按顺序代表从起点到终点的一条路径的点 ID（包括起点、终点），这条路径需要尽可能的接近最优结果。也就是说，该路径的语言模型得分*声学模型得分越高越好。

请在处理完一条语音之后立刻输出该条语音的对应路径（检测运行时间就是根据输出的时间来判定）。请输出到标准输出 stdout 中。

程序不允许读写文件，所有的读写操作都是标准输入输出。

不允许使用多线程编程。

评测环境说明：

评测时使用 windows 机器，使用 VC 编译器进行编译，并使用其 release mode。

评测时会使用多组 case 进行评测。每组 case 包含一个搜索空间、语言模型，以及若干条语音对应的矩阵。对于每组 case，会单独运行一遍程序进行评测。

时间限制：

有两个时间限制。从程序启动开始，到输出” ready”的时间视为预处理时间，这一步有单独的时间限制，为 30 分钟，对每一组数据都如此。

对于每一条语音，从读入矩阵，到输出路径，这又有一个时间限制。注意，时间限制是跟语音的帧数有关，总限制为帧数*0.01，单位为秒。例如，一个 300 帧的语音，时间限制为 3 秒。

对于所有语音的总处理时间，也有一个时间限制，为所有语音的帧数之和*0.1，单位为秒。这个限制比较宽松，主要是为了避免有运行时间超长或死循环的程序。

时间限制都是使用机器时间来评测，选手若要获取当前的机器时间，请使用 GetTickCount 函数（这是一个 windows 函数，稍后会给出其使用示例）。在评测时也使用该函数进行评测。

内存限制：

2G

评分标准：

(1) 0 分：

不能按照规定格式输出的，该条语音得 0 分。

预处理时间不允许超时，否则整组 case 的所有语音均记为 0 分。

处理每条语音的时间不允许超时，否则该条语音记为 0 分。

处理所有语音的时间不允许超时，否则整组 case 中，还未处理完毕的所有语音均记为 0 分。

内存不允许超限，否则整组 case 的所有语音均记为 0 分。

如有存在其他犯规情况（例如使用多线程编程），裁判有权判定该程序的部分语音甚至整组 case 得 0 分。

(2) 有效得分：

在所有满足格式、内存、时间等要求的提交程序中，对于每条语音，按照其输出的最优路径所对应的语言模型、声学模型总得分来进行排序，每条语音第一名的获得 50 分，第二名获得 49 分，依此类推。

(3) 排名规则:

统计所有语音的得分, 总计得分较高的选手排名靠前; 如果有得分相同的选手, 则比较其获得第一名的语音条数; 再相同, 则比较其获得第二名的语音条数; ……; 如果依然不能区别, 最后比较预处理时间+处理所有语音的时间, 总用时较少者获胜。

输入数据格式

搜索空间:

如上面所述, 在搜索空间中节点类型分为两个类型

1. 词尾节点
2. 普通节点

在本搜索空间中, 保证所有的节点 ID 都是从 0 开始连续编排的, 而且所有的词尾节点 ID 大于普通节点的 ID。

在本搜索空间中, 每条边上的元素包括 label 信息和目标节点信息。对于跳往词尾节点的边来讲, 边上的 label 信息存储了词 ID 信息, 对于到达普通节点的边来讲, 边上的 label 信息存储了声学 ID 信息。

输入数据首先包含四行, 每行一个整数, 依次为:

普通节点个数 v_t 、词尾节点个数 v_e 、图的起点的 ID v_s 、边数 e

接着有 $2 * e$ 行, 给出边信息。每条边对应连续的两行, 每行一个整数, 依次为:

该边的 label 信息、该边所到达的节点 ID

接着有一行, 该行的值等同于 $v_t + v_e + 1$ 。

接着有 $v_t + v_e + 1$ 行, 每行一个整数。设其中第 i 行为 s_i , 其含义为在上面给出的边信息中, 第 s_i 条边 (含, 边的编号从 0 开始) 到第 s_{i+1} 边 (不含) 之间的所有行数的边的起点都是 ID 为 i 的点。

图的起点已经给出, 图的终点请自行按如下规则寻找: 图中任意出度为 0 的点, 该点都是终点。这样的点可能不止一个。

语言模型:

语言模型通过文本格式提供。在输入中, 在搜索空间数据的最后一行之后, 紧接着就是语言模型, 格式为:

对于一个 n -gram 语言模型 ($n \leq 5$), 首先是一行, 为

`\data-n\`

这里的 n 就是 n -gram 中的 n 。

接着有 n 行, 为

`ngram 1=num1`

`ngram 2=num2`

`.....`

`ngram n=numn`

上面每行等号之后都是一个整数 num_i (其规模每组数据都不同, 会在下面的测试数据说明中给出), 其数值可变, 表示一元、二元、……、 n 元文法的个数。

接着, 有 n 段文本信息, 第 i 段文本信息代表 i 元文法的信息。

在 i 段文本信息中, 首先是一行空行, 然后是一行, 为

`\i-grams:`

接着有 num_i 行, 每行首先是一个浮点数, 接着是以空格分隔的 i 个词, 最后还有一个浮点数。两个浮点数和词之间都以 `\t` 分隔。第一个浮点数表示该 i 元文法的概率的 \log_{10} 对数 (该数 ≤ 0 , 且 ≥ -100), 中

间的 i 个词表示该 i 元文法是由这些词依次组成的，最后的一个浮点数表示该 i 元文法的回退权的 \log_{10} 对数（该数 ≥ -100 且 ≤ 100 ）。例如：

-4.351259 库存切 -0.440316

表示这个 2 元文法 $p(\text{切}|\text{库存})=10^{-4.351259}$ ，它的回退权为 $b(\text{库存切})=10^{-0.440316}$

在本题的所有数据中，对于词典中的所有词，都存在一元文法。

关于词 ID 和一元文法，需要再说明一点。如上所述，在搜索空间中，终点为词尾节点的边上存储有该边的词 ID 信息。这个词 ID 和具体的词是如何对应的呢？是按照 n -gram 模型中，一元文法的顺序。第一个出现的一元文法的词 ID 为 1，第二个出现的为 2，……，第 i 个出现的词 ID 为 i 。

详细的例子可参考样例数据。

每条语音对应的矩阵：

每一个语音对应一个声学得分矩阵。其中，得分矩阵是以声学得分的 \ln 对数的形式提供的。

矩阵以文本格式提供。在语言模型的最后一行结束之后，紧接着有一行，包含一个整数 t ，表明语音的条数。接着有 t 段文本，每一段包含一条语音对应的矩阵信息。每段的第一行包含两个整数，为语音帧数 d 、声学模型 ID 总个数 g 。

接着有 d 行，每行有 g 个浮点数。第 d_m 行、第 g_n 列代表语音帧数为 d_m 、声学模型 ID 为 g_n 时（语音帧数的下标从 0 开始，声学模型 ID 的下标从 0 开始），对应的声学模型得分的 \ln 对数。

特别提示，这个矩阵的输入数据很大，请使用字符串形式读入，以节约运行时间。可以参考下面的示例。

1.2. 关键思路：

百度之星 2014

资格赛 1: DiskSchedule

时间限制：1s 内存限制：65536K

1.1. 问题描述

有很多从磁盘读取数据的需求，包括顺序读取、随机读取。为了提高效率，需要人为安排磁盘读取。然而，在现实中，这种做法很复杂。我们考虑一个相对简单的场景。

磁盘有许多轨道，每个轨道有许多扇区，用于存储数据。当我们想在特定扇区来读取数据时，磁头需要跳转到特定的轨道、具体扇区进行读取操作。为了简单，我们假设磁头可以在某个轨道顺时针或逆时针匀速旋转，旋转一周的时间是 360 个单位时间。磁头也可以随意移动到某个轨道进行读取，每跳转到一个相邻轨道的时间为 400 个单位时间，跳转前后磁头所在扇区位置不变。一次读取数据的时间为 10 个单位时间，读取前后磁头所在的扇区位置不变。磁头同时只能做一件事：跳转轨道，旋转或读取。

现在，需要在磁盘读取一组数据，假设每个轨道至多有一个读取请求，这个读取的扇区是轨道上分布在 0 到 359 内的一个整数点扇区，即轨道的某个 360 等分点。磁头的起始点在 0 轨道 0 扇区，此时没有数据

读取。在完成所有读取后，磁头需要回到 0 轨道 0 扇区的始点位置。请问完成给定的读取所需的最小时间。

输入

输入的第一行包含一个整数 M ($0 < M \leq 100$)，表示测试数据的组数。

对于每组测试数据，第一行包含一个整数 N ($0 < N \leq 1000$)，表示要读取的数据的数量。之后每行包含两个整数 T 和 S ($0 < T \leq 1000$, $0 \leq S < 360$)，表示每个数据的磁道和扇区，磁道是按升序排列，并且没有重复。

输出

对于每组测试数据，输出一个整数，表示完成全部读取所需的时间。

样例输入

```
3
1
1 10
3
1 20
3 30
5 10
2
1 10
2 11
```

样例输出

```
830
4090
1642
```

1.2. 解题报告

由于磁头跳轨道所需时间是 400，大于旋转一周的时间 360，所以反证法容易证明磁头不应在读取时在磁道间往复移动。于是磁头移动的策略简化成从 0 道的起始位置开始依磁道升序移动到需读取的最远的磁道（记为去程），再从该磁道依磁道降序移动回 0 道的起始位置（记为返程）。另，由于每个磁道至多只有一个数据需要读取，所以磁头移动策略可以描述为，在需要读取的点之间选定若干个作为去程时读取，余下在返程时读取。去程和返程两个相邻读取点之间需要的旋转时间是固定的，即连接两点在 360 度轨道上的较短的弧对应的角度（或说夹角）。不相邻磁道之间的两次读取可以简化为“只跳磁道但不旋转”+“同磁道两次读取位置间的一次旋转”，于是磁道的具体信息可以简化掉，只需在最终结果上加上去返两程磁道跳转所需时间（最大磁道号*400*2）。读取本身的时间也可以简化掉，等于需读取数据的数量*10。所以只需要计算旋转所需的时间。

由于去程和返程时对称的，所以计算过程中互换对结果没有影响，于是得到如下的动态规划的解法：

（设需读取的数据个数 N ，需读取的最大磁道号 T ，最大磁道个数 $S=360$ ）

（设对于 i ($0 \leq i < N$)， b 表示输入中需要读取的扇区号；

$\text{dis}(x, y)$ 表示扇区号 x 和扇区号 y 之间旋转所需的时间)

对于 i, j ($0 \leq i < N, 0 \leq j < 360$)，令 $a[j]$ = 去程读取至第 i 个数据即磁道在 b ，并且返程当前在磁道 j 时，所需的最少时间。

状态转移为：

$$a[j] = \min \{ a[i-1][j] + \text{dis}(b, b[j]), a[i-1][k] + \text{dis}(b, k) \text{ if } k=b \text{ and } j=b[i-1] \}$$

初始条件为：

$$a[b[0]][0] = \text{dis}(0, b[0])$$

实现可以使用递推的方式，对于任意 b, j ：

$b, j \rightarrow b[i+1], j$ 表示第 $i+1$ 次读取与第 i 次读取是在同一程接连读取

$j, b \rightarrow b[i+1], b$ 表示第 $i+1$ 次读取与返程所在的 j 磁道是在同一程接连读取

伪代码如下：

$$a[0][0] = \text{getdis}(0, b[0])$$

for i in $0 \sim n$:

for j in $0 \sim 360$:

$$a[i+1][j] = \min(a[i+1][j], a[j] + \text{getdis}(b, b[i+1]))$$
$$a[i+1] = \min(a[i+1], a[j] + \text{getdis}(b[i+1], j))$$

算法复杂度为 $O(N*S)$ 。

另外，对于题目给出的数据范围，动态规划在 $N*N$ 或者 $T*T$ 的空间内进行也可以在时限内通过。

资格赛 2: Energy Conversion

2.1. 题目描述：

魔法师百小度也有遇到难题的时候——

现在，百小度正在一个古老的石门面前，石门上有一段古老的魔法文字，读懂这种魔法文字需要耗费大量的能量和大量的脑力。

过了许久，百小度终于读懂魔法文字的含义：石门里面有一个石盘，魔法师需要通过魔法将这个石盘旋转 X 度，以使上面的刻纹与天相对应，才能打开石门。

但是，旋转石盘需要 N 点能量值，而为了解读密文，百小度的能量值只剩 M 点了！破坏石门是不可能的，因为那将需要更多的能量。不过，幸运的是，作为魔法师的百小度可以耗费 V 点能量，使得自己的能量变为现在剩余能量的 K 倍（魔法师的世界你永远不懂，谁也不知道他是怎么做到的）。比如，现在百小度有 A 点能量，那么他可以使自己的能量变为 $(A-V)*K$ 点（能量在任何时候都不可以为负，即：如果 A 小于 V 的话，就不能够执行转换）。

然而，在解读密文的过程中，百小度预支了他的智商，所以他现在不知道自己是否能够旋转石盘，打开石门，你能帮帮他吗？

输入：输入数据第一行是一个整数 T ，表示包含 T 组测试样例；

接下来是 T 行数据，每行有 4 个自然数 N, M, V, K （字符含义见题目描述）； 数据范围：

$T \leq 100$ $N, M, V, K \leq 10^8$

输出：对于每组数据，请输出最少做几次能量转换才能够有足够的能量点开门；如果无法做到，请直接输出 -1。

样例输入：

```
4
10 3 1 2
10 2 1 2
10 9 7 3
10 10 10000 0
```

样例输出：

```
3
-1
-1
0
```

2.2. 解题报告：

```
01 #include <stdio.h>
02 #include <string.h>
03
04 const int maxn = 10000;
05
06 int dfs(long long n, long long m, long long v, long long k, int tim) {
07     if(n<=m) return tim;
08     if(m<v || (m-v)*k<=m) return -1;
09     return dfs(n, (m-v)*k, v, k, tim+1);
10 }
11
12 int main() {
13     int L, T;
14     long long n, m, v, k;
15     //freopen("input.txt", "r", stdin);
16     //freopen("output.txt", "w", stdout);
17     scanf("%d", &T);
18     for(L=1; L<=T; L++) {
19         scanf("%I64d%I64d%I64d%I64d", &n, &m, &v, &k);
20         printf("%d\n", dfs(n, m, v, k, 0));
21     }
```

```
22         return 0;
23 }
```

资格赛 3: Labyrinth

时间限制: 1s 内存限制: 65536K

3.1. 问题描述

度度熊是一只喜欢探险的熊，一次偶然落进了一个 $m \times n$ 矩阵的迷宫，该迷宫只能从矩阵左上角第一个方格开始走，只有走到右上角的第一个格子才算走出迷宫，每一次只能走一格，且只能向上向下向右走以前没有走过的格子，每一个格子中都有一些金币（或正或负，有可能遇到强盗拦路抢劫，**度度熊身上金币可以为负，需要给强盗写欠条**），度度熊刚开始时身上金币数为 0，问度度熊走出迷宫时候身上最多有多少金币？

输入

输入的第一行是一个整数 T ($T < 200$)，表示共有 T 组数据。

每组数据的第一行输入两个正整数 m, n ($m \leq 100, n \leq 100$)。接下来的 m 行，每行 n 个整数，分别代表相应格子中能得到金币的数量，每个整数都大于等于 -100 且小于等于 100。

输出

对于每组数据，首先需要输出单独一行 "Case #?:", 其中问号处应填入当前的数据组数，组数从 1 开始计算。

每组测试数据输出一行，输出一个整数，代表根据最优的打法，你走到右上角时可以获得的最大金币数目。

样例输入

```
2
3 4
1 -1 1 0
2 -2 4 2
3 5 1 -90
2 2
1 1
1 1
```

样例输出

```
Case #1:
18
Case #2:
```


3.2. 解题报告

考虑每一列可以向上向下走，但必须走走以前没有走过的格子，所以度度熊在每一列上都只能选一个方向走，加上可以往右走，这样在每个格子上的状态就只要 2 种：

1. $state[x][y][0]$ 表示之后只能向下或向右走；
2. $State[x][y][1]$ 表示之后只能向上或向右走；

动态规划的递推公式为：

1. $state[x][y][0] = \max \{ state[x-1][y][0], state[x][y-1][0], state[x][y-1][1] \} + value[x][y];$
2. $state[x][y][1] = \max \{ state[x+1][y][1], state[x][y-1][0], state[x][y-1][1] \} + value[x][y];$

最后只要看取右上角的两个状态的最大值即可，即 $\max \{ state[0][n-1][0], state[0, n-1][1] \}$ 。

动态规划的时间复杂度是 $O(N*N)$ 。

资格赛 4: Xor Sum

时间限制：1S 空间限制：64M

4.1. 问题描述

Zeus 和 Prometheus 做了一个游戏，Prometheus 给 Zeus 一个集合，集合中包含了 N 个正整数，随后 Prometheus 将向 Zeus 发起 M 次询问，每次询问中包含一个正整数 S ，之后 Zeus 需要在集合当中找出一个正整数 K ，使得 K 与 S 的异或结果最大。Prometheus 为了让 Zeus 看到人类的伟大，随即同意 Zeus 可以向人类求助。你能证明人类的智慧么？

输入

输入包含若干组测试数据，每组测试数据包含若干行。

输入的第一行是一个整数 T ($T < 10$)，表示共有 T 组数据。

每组数据的第一行输入两个正整数 N, M ($1 \leq N, M \leq 100000$)，接下来一行，包含 N 个正整数，代表 Zeus 的获得的集合，之后 M 行，每行一个正整数 S ，代表 Prometheus 询问的正整数。所有正整数均不超过 2^{32} 。

输出

对于每组数据，首先需要输出单独一行”Case #?:”，其中问号处应填入当前的数据组数，组数从 1 开始计算。

对于每个询问，输出一个正整数 K ，使得 K 与 S 异或值最大。

样例输入

2

3 2

3 4 5

1

5

4 1

4 6 5 6

3

样例输出

Case #1:

4

3

Case #2:

4

4.2. 解题报告

由于每个正整数都不超过 2 的 32 次方，所以将每个正整数的二进制数字作为字符串插入到字典树中，对于每一个询问，只需要按照贪心的做法搜索字典树就行了。

1. 将所有数字转为二进制。
2. 将集合中的所有二进制数变为一棵二叉树，左节点代表这一位为 0 ，右节点代表 1 。
3. 对每一个询问，在这棵二叉树上查询，如果这一位为 0 并且右孩子不为空，则往右走，否则往左走。最终到达叶子节点经过的路径就是所求的数的二进制表示。

初赛 1: BestFinancing

时间限制: 1s 内存限制: 65536K

1.1. 问题描述

小 A 想通过合理投资银行理财产品达到收益最大化。已知小 A 在未来一段时间中的收入情况，描述为两个长度为 n 的整数数组 `dates` 和 `earnings`，表示在第 `dates` 天小 A 收入 `earnings` 元 ($0 \leq i < n$)。银行推出的理财产品均为周期和收益确定的，可描述为长度为 m 的三个整数数组 `start`、`finish` 和 `interest_rates`，若购买理财产品 i ($0 \leq i < m$)，需要在第 `start` 天投入本金，在第 `finish` 天可取回本金和收益，在这期间本金和收益都无法取回，收益为 $\text{本金} \times \text{interest_rates} / 100.0$ 。当天取得的收入或理财产品到期取回的本金当天即可购买理财产品（注意：不考虑复利，即购买理财产品获得的收益不能用于购买后续的理财产品）。假定闲置的钱没有其他收益，如活期收益等，所有收益只能通过购买这些理

财产品获得。求小 A 可以获得的最大收益。

限制条件：

$1 \leq n \leq 2500$

$1 \leq m \leq 2500$

对于任意 i ($0 \leq i < n$) , $1 \leq \text{dates} \leq 100000$, $1 \leq \text{earnings} \leq 100000$, dates 中无重复元素。

对于任意 i ($0 \leq i < m$) , $1 \leq \text{start} < \text{finish} \leq 100000$, $1 \leq \text{interest_rates} \leq 100$ 。

限制条件

$1 \leq n \leq 2500$

$1 \leq m \leq 2500$

对于任意 i ($0 \leq i < n$) , $1 \leq \text{dates} \leq 100000$, $1 \leq \text{earnings} \leq 100000$, dates 中无重复元素。

对于任意 i ($0 \leq i < m$) , $1 \leq \text{start} < \text{finish} \leq 100000$, $1 \leq \text{interest_rates} \leq 100$ 。

输入

第一行为 T ($T \leq 200$)，表示输入数据组数。

每组数据格式如下：

第一行是 n m

之后连续 n 行，每行为两个以空格分隔的整数，依次为 date 和 earning

之后连续 m 行，每行为三个以空格分隔的整数，依次为 start , finish 和 interest_rate

输出

对第 i 组数据， i 从 1 开始计，输出

Case # i :

收益数值，保留小数点后两位，四舍五入。

样例输入

2

1 2

1 10000

1 100 5

50 200 10

2 2

1 10000

5 20000

1 5 6

5 9 7

样例输出

Case #1:

1000.00

Case #2:

2700.00

1.2. 解题报告

考虑所有理财产品在时间轴上已经确定分布，那么对于一个给定的时间点的单位投入，最优的收益率是一定的。另外，显然的，所有日期可以离散化成一个升序的序列。于是问题可以分解为对每个离散化后的日期，求得此时单位投入的最大收益率，然后若此日期有收入，将收入与收益率的乘积累加，即是最终的结果。

于是题目变成简单的一维的动态规划。对于某个理财产品，记为 $\text{profit}(\text{startdate}, \text{enddate})$ ，给定日期 date 的最大收益记为 $a[\text{date}]$ ，则

$$a[\text{date}] = \max \{ \\ a[\text{date}+1], \\ \text{profit}(\text{date}, \text{enddate}) + a[\text{enddate}] \text{ for any } \text{profit}(\text{startdate}, \text{enddate}) \text{ startdate} = \text{date} \\ \}$$

程序实现从后向前递推即可。由于不计算复利，所有操作可以简化为收益率的整数累加操作，在最后结果统一除以 100，保留小数点后两位即可。

动态规划的时间复杂度是 $O(N)$ ，排序的时间复杂度是 $O(N \log N)$ 。

对于题目给出的数据范围， $O(N*N)$ 的复杂度也可以通过。

初赛 2: Chess

时间限制: 3s 内存限制: 65536K

2.1. 问题描述

小度和小良最近又迷上了下棋。棋盘一共有 N 行 M 列，我们可以把左上角的格子定为 $(1, 1)$ ，右下角的格子定为 (N, M) 。在他们的规则中，“王”在棋盘上的走法遵循十字路线。也就是说，如果“王”当前在 (x, y) 点，小度在下一步可以移动到 $(x+1, y)$, $(x-1, y)$, $(x, y+1)$, $(x, y-1)$, $(x+2, y)$, $(x-2, y)$, $(x, y+2)$, $(x, y-2)$ 这八个点中的任意一个。

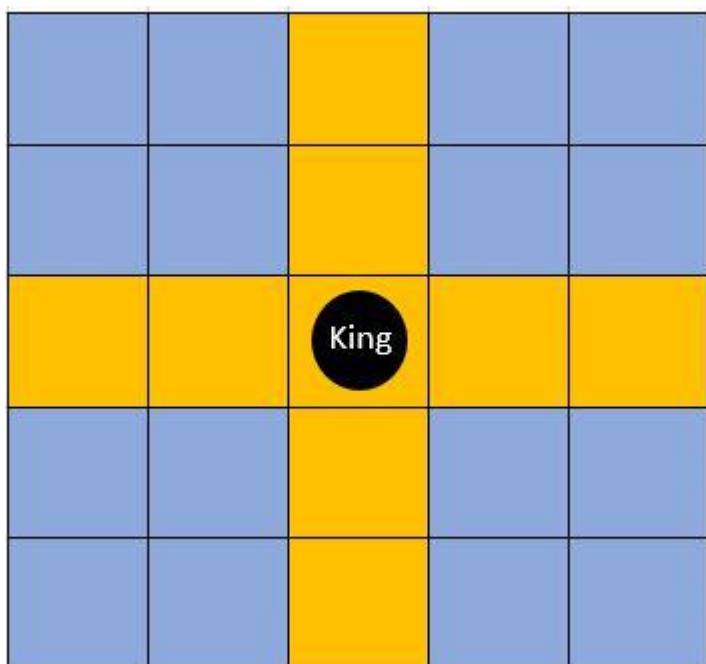


图 1 黄色部分为棋子所控制的范围

小度觉得每次都是小良赢，没意思。为了难倒小良，他想出了这样一个问题：如果一开始“王”在 (x_0, y_0) 点，小良对“王”连续移动恰好 K 步，一共可以有多少种不同的移动方案？两种方案相同，当且仅当它们的 K 次移动全部都是一样的。也就是说，先向左再向右移动，和先向右再向左移动被认为是不同的方案。

小良被难倒了。你能写程序解决这个问题吗？

输入：

输入包括多组数据。输入数据的第一行是一个整数 T ($T \leq 10$)，表示测试数据的组数。

每组测试数据只包括一行，为五个整数 N, M, K, x_0, y_0 。 ($1 \leq N, M, K \leq 1000, 1 \leq x_0 \leq N, 1 \leq y_0 \leq M$)

输出：

对于第 k 组数据，第一行输出 Case #k:，第二行输出所求的方案数。由于答案可能非常大，你只需要输出结果对 9999991 取模之后的值即可。

样例输入：

```
2
2 2 1 1 1
2 2 2 1 1
```

样例输出：

```
Case #1:
2
Case #2:
4
```

2.2. 解题报告:

最直接是使用 $O(NMK)$ 的 DP 方式, 用 $dp[x][y][k]$ 表示第 k 步跳转到 (x, y) 的方案数, 然后逐步递推。但是由于时间复杂度比较高, 所以需要考虑优化的方式。

考虑一次移动 k 步的方案, 可以发现行和列的移动方式是独立的。因此可以先分别求出在行和列上移动的方案数, 再枚举 k 步里有多少步分别是纵向和横向移动的

假设 $A[x][k]$ 表示起始横坐标为 x_0 , 移动 k 步到横坐标为 x 的方案数, 通过递归可求得每个 $A[x][k]$, 则可求得纵向移动 k 步的方案数 $V[k] = \sum(A[x][k] \mid 1 \leq x \leq n)$

假设 $B[y][k]$ 表示起始纵坐标为 y_0 , 移动 k 步到纵坐标为 y 的方案数, 通过递归可求得每个 $A[y][k]$, 则可求得横向移动 k 步的方案数 $H[k] = \sum(B[y][k] \mid 1 \leq y \leq m)$

最终答案为 $\sum(c(K, i) * V * H[K-i] \mid 0 \leq i \leq K)$ 。总体复杂度可以降低到 $O(NK) + O(MK)$

初赛 3: CycleCocycle

时间限制: 10s 内存限制: 64MB

3.1. 问题描述

有一个 n 个点 m 条边的图, 你要给每个点一个 0 或 1 的标号, 使得每个点与偶数个相同标号的点之间有边。如果有多解输出任意一组。

输入

第一行为 T , 表示输入数据组数。

下面 T 组数据。每组数据中:

第一行, n, m 。

下面 m 行, 每行两个数 x, y , 表示一条边。

输出

对第 i 组数据, 输出

Case # i :

然后输出一个长度为 n 的字符串, 表示每个点的标号。

限制条件

$1 \leq T \leq 100$

$1 \leq n \leq 1000$

$1 \leq m \leq 10000$

$1 \leq x, y \leq n$

图中无重边无自环。

样例输入

1

4 5

1 2

1 3

1 4

2 3

2 4

样例输出

Case #1:

0001

3.2. 解题报告:

建立 0/1 线性方程组，然后解方程就可以了。每个点对应一个等式。时间复杂度 $O(n^3)$ 。

初赛 4: Grids

时间限制: 5s 内存限制: 65536K

4.1. 问题描述

度度熊最近很喜欢玩游戏。这一天他在纸上画了一个 2 行 N 列的长方形格子。他想把 1 到 2N 这些数依次放进去，但是为了使格子看起来优美，他想找到使每行每列都递增的方案。不过画了很久，他发现方案数实在是太多了。度度熊想知道，有多少种放数字的方法能满足上面的条件？

输入

第一行为数据组数 T 。

然后 T 行，每行为一个数 N 表示长方形的大小。

输出

对于每组数据，输出符合题意的方案数。由于数字可能非常大，你只需要把最后的结果对 1000000007 取模即可。

样例输入

2

1

3

样例输出

Case #1:

1

Case #2:
5

提示

对于第二组样例，方案为：

1	2	3	1	2	4
4	5	6	3	5	6
1	2	5	1	3	4
3	4	6	2	5	6
1	3	5			
2	4	6			

共 5 种方案。

4.2. 解题报告

我们考虑将 $1, 2, \dots, 2n$ 依次填入 $2 \times n$ 的格子中，易见，1 只能填入左下角的格子中，2 可以填入 1 右侧或者 1 的上面，且不能填入其他位置。下面我们按照此法填充：

Step 1.

1

Step 2.

1 2

or

1

2

Step 3.

1 2 3

or

1 2

3

or

1 3

不难观察得出，在这两行格子中，填充的格子总是连续的，否则就会违反规则，并且，上行格子填充的进度必须小于等于下行格子填充的进度。这很类似于台阶问题：在一个 $n \times n$ 的格子中，我们从左下角出发，前往右上角，每一步，只能水平向右走一格，或者垂直向上走一格，并且不能越过对角线。如果我们将水平走一格对应为将当前的数填入下行，将垂直向上走一格对应为将当前的数填入上行，则可见这两个问题完全等价。那么填充格子的可行方案数等于台阶问题中从左下角到达右上角的路径数，等于 $c(2n, n) - c(2n, n-1) = c(2n, n) / (n+1)$ ，也就是著名的 Catalan 数。我们可以预先计算好 1 到 $2n$ 的阶乘模 1000000007，并且利用欧几里得算法计算它们对于 1000000007 的逆，那么 $cn = (2n)! / ((n+1)!n!) = (2n)! * \text{rev}((n+1)!) * \text{rev}(n!)$ ，其中 rev 表示逆。

初赛 5: Information

时间限制：1s 内存限制：65536K

5.1. 问题描述

军情紧急，我们需要立刻开发出一个程序去处理前线侦察兵发回的情报，并做出相应的分析。现在由你负责其中的一个子模块，你需要根据情报计算出敌方坦克的位置。

当敌方坦克静止时，侦察兵会尽力估算出它们之间的位置，而每当敌方坦克移动时，侦察兵都会记录下坦克新的位置并向你报告。每个坦克的位置可以由一个二维整数坐标来描述。

前线发回的情报有四种格式：

1 A B X Y

表示 A 坦克移动到了与 B 坦克的相对位置是 的地方，即 。

2 A X Y

表示 A 坦克移动到了绝对位置是 的地方，即 ， 。

3 A B X Y

表示发现了 A 坦克与 B 坦克的相对位置是 ，即 。

4 A X Y

表示发现了 A 坦克的绝对位置是 ，即 ， 。

我们需要你对于如下两种询问及时做出回应：

5 A B

表示询问 A 坦克与 B 坦克的相对位置是多少，即分别求出 以及 。

6 A

表示询问 A 坦克的绝对位置是多少，即求出 。

其中 A 和 B 代表的是任意的一个坦克的编号，(X, Y) 表示了坦克的二维坐标。你可以假设初始时刻我们对于敌方任何坦克的位置都一无所知，在此之后坦克的每一次移动都被侦察兵侦察到了。请注意两个坦克的坐标有可能相同。

输入

输入的第一行是一个整数 T ($T < 1000$)，表示共有 T 组数据。

对于每组数据，第一行有一个整数 N ，表示这组数据有 N 次查询。接下来的每行表示一次查询，每次查询第一个数字代表是哪种询问，询问的格式详见上文。

数据范围：

， X 和 Y 都是整数且

测试数据中 98% 的数据 N 不超过 50。

输出

对于每组数据，首先需要输出单独一行 "Case#?:"，其中问号处应填入当前的数据组数，组数从 1 开始计算。

对于每一个类型 (1) 或者 (2) 的询问，请把它们加入到你的记录中。

对于每一个类型 (3) 或者 (4) 的询问，如果与之前记录的内容有矛盾，请输出 "REJECT" 并将这个情报忽略掉，如没有矛盾，请把它们加入到你的记录中。

对于每一个类型 (5) 或者 (6) 的询问，如果根据之前的记录能推出结论，请输出两个整数 X 和 Y ，两个整数之间有一个空格；如果不能推出结论，请输出 "UNKNOWN"。输出的所有信息都不包括引号。

样例输入

2

7

1 1 2 3 4

2 3 4 5

3 4 5 2 1

4 6 2 2

3 2 4 6 2

5 4 1

6 3

6

6 3

4 3 2 2

6 3

2 4 2 3

5 3 4

3 3 4 1 2

样例输出

Case #1:

-9 -6

4 5

Case #2:

UNKNOWN

2 2

0 -1

REJECT

5.2. 解题报告 -Information

这题的算法并不复杂，主要就是一个并查集的使用，不过写的时候要注意很多细节。每个 Tank 的状态可以分 3 种情况：

1. 未初始化，不在任何集合里；
2. 在一个互相间已知相对位置的集合里；
3. 已知绝对位置。

具体的处理规则如下：

1. 1A B X Y

- 1) 如果 A 原来在一个集合里，把它从这个集合移除；
- 2) 根据 B 的情况：
 - a) 如果 B 已知绝对位置，设置 A 的绝对位置；
 - b) 如果 B 不在任何集合里，则创建一个集合，把 A、B 都加入，并设置相对位置；
 - c) 如果 B 已经在一个集合里，则把 A 加入这个集合，并设置相对位置；

2. 2 A X Y

- 1) 如果 A 原来在一个集合里，把它从这个集合移除；
- 2) 设置 A 的绝对位置；

3. 3 A B XY

- 1) 如果 A 和 B 其中一个不在任何集合里，则等于把其移到另一个的相对位置上，和类型 1 的处理流程一样；
- 2) 如果 A 和 B 都有绝对位置或者在同一个集合里，判断他们的相对位置是否和这次发现的一致，如果不一致，输出 REJECT，忽略本次操作；
- 3) 如果 A 和 B 其中一个有绝对位置，则可以把另外一个所在集合里的所有 Tank 都设置绝对位置；
- 4) 如果 A 和 B 在不同的集合里，则把其中一个集合合并到另一个集合，并更新所有 Tank 的相对位置；

4. 4 A X Y

- 1) 如果 A 已经有绝对位置，判断他的绝对位置是否和这次发现的一致，如果不一致，输出 REJECT，忽略本次操作；

- 2) 如果 A 不在任何集合内，设置 A 的绝对位置即可；
 - 3) 如果 A 在一个集合内，设置这个集合里的所有 Tank 的绝对位置；
5. 5 A B
- 1) 如果 A 和 B 都有绝对位置或者在同一个集合里，输出他们的相对位置；
 - 2) 否则输出 UNKNOWN；
6. 6 A
- 1) 如果 A 有绝对位置，输出 A 的绝对位置；
 - 2) 否则输出 UNKNOWN。

初赛 6: JZP Set

时间限制：5s 内存限制：256MB

6.1. 问题描述

一个 $\{1, \dots, n\}$ 的子集 S 被称为 JZP 集，当且仅当对于任意 S 中的两个数 x, y ，若 $(x+y)/2$ 为整数，那么 $(x+y)/2$ 也属于 S 。

例如， $n=3$ ， $S=\{1, 3\}$ 不是 JZP 集，因为 $(1+3)/2=2$ 不属于 S 。但是 $\{1, 2, 3\}$ 的其他子集都属于 S ，所以 $n=3$ 时有 7 个 JZP 集

给定 n ，求 JZP 集的个数。

输入

第一行为 T ，表示输入数据组数。

每组数据包含一行整数 n 。

输出

对第 i 组数据，输出

Case # i :

然后输出 JZP 集的个数。

限制条件

$$1 \leq T \leq 10^5$$

$$1 \leq n \leq 10^7$$

样例输入

3
1
2
3

样例输出

Case #1:

2

Case #2:

4

Case #3:

7

6.2. 解题报告:

Jzp 集的充分必要条件是差为奇数的等差数列，于是可以简单递推算出。

$q[n]=q[n-1]*2-q[n-2]+(n-1 \text{ 的奇约数的数量})$

初赛 7: party

时间限制:3s 内存限制: 65536K

7.1. 问题描述

B 公司共有 N 个员工，但是并不是所有人都能和睦相处。在每一个人的心中都有一个潜在的对手，任何人都不能接受和他的对手同时参加 B 公司的聚餐。然而这种关系并不一定是对称的，也就是说，A 把 B 视作自己的对手，而 B 所想的对手并不一定是 A。

现在，B 公司准备举办一次盛大的聚会，公司希望员工通过这次聚会获得尽可能多的快乐值。第 i 个员工的快乐值 $q[i]$ 是一个大于 0 不大于 100 的整数，如果他参加聚餐，他就会获得的快乐值，如果他的对手参加聚餐，他的快乐值就为 0。

但老板在安排聚餐时不知道如何解决这个问题，因此，他找到你帮忙计算这次聚会最多可以带来多少快乐值。

输入

输入数据的第一行是一个整数 T ，表示有 T 组测试数据。

每组数据的第一行包括一个整数 N ，表示共有 N 个员工。（约有 500 组数据 N 不大于 500，约有 10 组数据 N 不大于 100000）

第二行是 N 个用空格隔开的整数，第 i 个整数表示第 i 个员工的对手编号。数据保证，且。

第三行也包含 N 个用空格隔开的整数，表示第 i 个员工能够获得的快乐值。

输出

对于第 k 组数据，第一行输出 Case #k:，第二行输出仅包含一个数，表示这次聚会最多可以带来多少快乐值。

样例输入

1

8

2 7 1 8 4 2 3 5

50 30 40 40 50 10 70 60

样例输出

Case #1:

190

Hint

在样例中，应选择 1、6、7、8 号员工。

7.2. 解题报告:

将每个员工认为是图上的一个顶点，如果员工 s 的敌人是员工 t ，则 s 到 t 建立一条有向边。由于每个员工只有一个敌人，因此每个顶点的出度为 1。所以整个图可以被拆分成多个连通子图，每个连通子图上只有一个环，可以分别对每个连通子图单独处理。

考虑每个连通子图，找出环上的任意一条边 $s \rightarrow t$ ，去除这条边后，可以得到以 s 为根的一颗树，然后枚举 t 被选择或者不被选择的情况，分别做一轮树形 DP。

假设 $dp[k][0]$ 表示在节点 k 不被选择的情况下，以节点 k 为根节点的子树下能够选择出来的最大快乐值； $dp[k][1]$ 则表示节点 k 被选择的情况。那么可以得到

$dp[k][0] = \sum(\max(dp[son][0], dp[son][1]))$ (son 是节点 k 的儿子节点)

$dp[k][1] = \sum(dp[son][0]) + happy[k]$ ($happy[k]$ 表示节点 k 能够获得的快乐值)

当 t 被选择情况下 $max_value1 = dp[0]$; 当 t 不被选择情况下 $max_value2 = \max(dp[0], dp[1])$; 而每个连通子图的最优解是 $\max(max_value1, max_value2)$

整体处理的时间复杂度为 $O(N)$

初赛 8: Scenic Popularity

时间限制: 1s 内存限制: 65536K

8.1. 问题描述

临近节日，度度熊们最近计划到室外游玩公园，公园内部包括了许多的旅游景点区和休息区，由于旅游景点很热门，导致景区区和休息区都聚集了很多的人。所以度度熊在旅游之前想通过百度地图查看一下公园内各个地方的热门程度。

假设所有景点区和休息区都是 X 轴直线上的一系列顶点，所对应的坐标 X_i 保证唯一。每个景点区有个初始的热度值，而一个休息区(坐标为 X_i)的热度值等于离它距离最近的景点区 X_j 的热度值（距离定义为 $|X_i - X_j|$ ），如果此休息区与两个景点区的距离一样，则休息区的热度值选择两个景点区中的热度值最大值，如果两个热度值都一样，则随意选择其中一个。

度度熊在出门之前会经常去查看百度地图，每次查看前会有某些景点区的热度值已发生改变，从而也会导致周围的休息区的热度值发生改变，然后度度熊想知道当前热度值 $\leq R_k$ 的顶点(包括景点区和休息区)有多少个

输入

输入数据的第一行是测试 Case 的个数 ($T \leq 100$)。

每个 Case 的第一行是 N ($0 < N \leq 10000$)，表示景点区和休息区的总数。

接着会有 N 行数据，每一列首先是顶点的 X 坐标 X_i ($0 < X_i \leq 1e8$)，第二列是一个整数 H_i ($0 \leq H_i \leq 100000$)，如果 H_i 不为 0，则表示当前顶点为风景区且初始的热度值为 H_i ，否则表示当前顶点为休息区。这 N 行数据会按照坐标 X_i 递增的方式依次给出。

接着的一行数据是操作的次数 K ($K \leq 100$)，最后会有 K 行数据，每一行的第一列要么是 'U' 或者 'Q'，'U' 表示当前操作为更改热度操作，'Q' 表示当前操作为查询操作。如果是更改操作，接着会有两列数据，分别是热度值要改变的风景区的下标 L_k ($0 \leq L_k < N$) 以及改变后的热度值 V_k ($0 \leq V_k \leq 100000$)；如果是查询操作，第二列是要查询的热度范围 R_k ($0 < R_k \leq 100000$)

输出

对于第 k 组测试数据，第一行输出 Case #k:，接下来对每次查询操作(即 Q 操作)会输出一个整数，表示满足条件的顶点数有多少个

样例输入

```
1
4
10 0
20 3
30 0
40 2
3
Q 3
U 3 4
Q 3
```

样例输出

Case #1:

```
4
2
```

8.2. 解题报告:

整道题主要是考虑如何降低查询和更新的复杂度, 解决思路也是传统的树状数组统计方式。但是需要注意的是细节上的处理, 每次景区的热度值都会导致多个休息区的热度发生变更, 而需要特别处理的主要是一些特殊的休息区, 这种特殊的休息区与两个景区的距离一样, 因此需要引入额外的标志来记录当前休息区是与哪个景区的热度值是匹配的。

复赛 1: FindNumbers

时间限制: 5s 内存限制: 65536K

1.1. 问题描述:

给 n 个非负整数, 满足对于某正整数 k , $n=2^k-1$ 。从中选出 $(n+1)/2$ 个数, 使得它们的和是 $(n+1)/2$ 的倍数。

输入

第一行, T , 询问个数。

下面 $2T$ 行, 每两行是一个询问。对于每两行:

第一行, n 。

第二行, n 个整数, a_0, a_1, \dots, a_{n-1} 。

$$1 \leq T \leq 20$$

$$1 \leq n \leq 2^{15}$$

$$0 \leq a_i \leq 10^9$$

输出

对第 i 个 ($1 \leq i \leq T$) 询问的回答为两行, 第一行为编号:

Case # i :

第二行为结果:

如果不能选出这样的 $(n+1)/2$ 个数, 输出 -1;

否则输出一行, 用一个空格分隔的 $(n+1)/2$ 个数 $b_0, b_1, \dots, b_{(n+1)/2-1}$ 。满足 b_i 两两不同, 且 $\sum(a_{b_i})$ 是 $(n+1)/2$ 的倍数。如果有多解, 输出任意一个。

样例输入

```
2
3
1 3 5
7
0 1 2 3 4 5 6
```

样例输出：

Case #1:

```
0 2
```

Case #2:

```
0 2 4 6
```

1.2. 解题报告

鉴于 $n=2^k-1$ (k 为正整数)，因此由归纳法可证明必有解。

证明：

归纳基础：对于 $k=1$ ，显然成立，

归纳假设：假定对于任意正整数 k ，有解。

归纳推理：考虑对于 $n=2^{k+1}-1$ 的情形，我们可以将 $a[0..n-1]$ 分为 $a[0..2^k-2]$ 和 $a[2^k-1..n-2]$ ，由归纳假设可得：

从 $a[0..2^k-2]$ 中我们可以挑出 $(m+1)/2$ 个数 $b_1[0..(m-1)/2]$ ，满足 $\sum\{b_1[0..(m-1)]\}=r*(m+1)/2$ ，其中 $m=2^k-1$ ， r 为某个非负整数。

从 $a[2^k-1..n-2]$ 中我们可以挑出 $(m+1)/2$ 个数 $b_2[0..(m-1)/2]$ ，满足 $\sum\{b_2[0..(m-1)]\}=s*(m+1)/2$ ，其中 $m=2^k-1$ ， s 为某个非负整数。

从 $a-b_1-b_2$ 中（含 $|a-b_1-b_2|=m$ 个非负整数）我们可以挑出 $(m+1)/2$ 个数 $b_3[0..(m-1)/2]$ ，满足 $\sum\{b_3[0..(m-1)]\}=t*(m+1)/2$ ，其中 $m=2^k-1$ ， t 为某个非负整数。

由于 r, s, t 三个非负整数中必有两个奇偶性相同，从而 $\{(r+s), (s+t), (r+t)\}$ 中必有某个为 2 的倍数。若 $(r+s)\%2==0$ ，则可将 b_1 和 b_2 合并得到 $m+1=(n+1)/2$ 个非负整数且

$\sum\{b_1, b_2\}=(r+s)*(m+1)/2=(r+s)/2*(m+1)=(r+s)/2*(n+1)/2$ ，为 $(n+1)/2$ 的倍数。其他情况同理可得。

证毕

由上述证明，可得到分治算法，先将 a 分为 $a[0..(n-1)/2-1]$ 和 $a[(n-1)/2..n-2]$ ，递归分别求得 b_1 和 b_2 ，如果 $\sum\{b_1+b_2\}$ 满足条件，则返回，否则对 $a-b_1-b_2$ 递归，求得 b_3 ，如果 $\sum\{b_1+b_3\}$ 满足条件，则返回，否则 $\sum\{b_2+b_3\}$ 必满足条件，返回。易见该算法复杂度为 $O(n\log n)$ 。

复赛 2: Race

时间限制: 10s 内存限制: 64MB

2.1. 问题描述

度度熊最近参加了一场劲跑比赛，但是这个劲跑比赛的规则比较特殊。比赛方预先在地上画了一些横线和竖线（可以认为这些线为无限长的直线），要求选手从指定的位置出发，在最短时间内按照规定的顺序经过所有的直线（只要到达直线上的任意一点即为经过）。为了帮助度度熊获胜，你能为他规划一条最短的线路吗？

输入

第一行为 T ，表示输入数据组数。

下面 T 组数据。每组数据中：

第一行为一个整数 n ，表示有 n 条直线。接下来 n 行，每行包含两个整数，表示一条直线。如果直线为竖线，则第一个整数为 0，第二个整数为直线的横坐标 x 。如果直线为横线，则第一个整数为 1，第二个整数为直线的纵坐标 y 。度度熊将从平面的原点出发。

输出

对第 i 组数据，输出

Case # i :

然后输出一个实数，表示从原点出发按输入顺序经过所有直线的最短路径的长度。保留到小数点后 3 位。

限制条件

$$1 \leq T \leq 10$$

$$1 \leq n \leq 10^6$$

$$-500 \leq x, y \leq 500$$

样例输入

1

3

0 1

1 1

0 2

样例输出

Case #1:

2.236

提示:

度度熊需要按照规定的顺序经过所有的直线。如果度度熊不按照顺序经过直线，那么提前经过的直线是不算数的。例如，假设度度熊需要按顺序经过直线 1 和直线 2，那么如果度度熊在经过直线 2 之前经过了直线 1，这是不算数的，度度熊仍然需要在经过直线 2 后再次经过直线 1。但是度度熊可以经过直线 1 和直线 2 的交叉点而同时都算数。

2.2. 解题报告:

首先对横线和竖线的坐标分别做镜像翻装，使它们的坐标分别都是单调（非严格）递增的，并且相邻的横线和相邻的竖线的距离跟原始坐标的距离保持一致。这可以在线性时间内完成，并且不影响答案。然后按照顺序扫描所有直线，维护从原点出发的两条移动路径。一条移动路径尽可能往左走，一条移动路径尽可能往右走，但都保持最短路并且满足规定的直线到达顺序。这两条移动路径分别是上凸曲线和下凸曲线，每扫描到方向不同的两条直线时就根据交叉点更新其中一条移动路径。扫描到最后一条直线时就可以算出最短的移动路径长度了。

复赛 3: The Game of Coins

时间限制: 10s 内存限制: 64MB

3.1. 问题描述

Cirno 和 Marisa 经常在一起玩抛硬币的游戏，每次由 Cirno 先选择正面(H)或反面(T)，然后抛一次硬币，如果结果和 Cirno 的选择一致，则 Cirno 赢，如果结果相反，则 Marisa 赢。有一天，Marisa 对 Cirno 说，过去的玩法都弱爆了，给 Cirno 介绍了一种新的玩法。现在 Cirno 每次选择一个长度为 n 的由 H 和 T 组成的字符串，然后 Marisa 也选择一个不同的字符串，随后她们不断抛硬币，直到其中一个人选择的字符串和最后 n 次抛硬币的结果相匹配，匹配的人将获胜。玩了一段时间后，Cirno 发现自己的胜率下降了，但又不知道为什么。现在，Cirno 想知道在给定自己和 Marisa 所选的字符串时，自己获胜的概率。

输入

第一行为 T ，表示输入数据组数。

下面 T 组数据。每组数据都是两个空格分割的，长度均为 n 的，由 H 和 T 组成的字符串。

输出

对第 i 组数据，输出

Case # i :

然后输出 Cirno 获胜的概率，并以最简分数形式输出。

限制条件

$1 \leq T \leq 500$

$1 \leq n \leq 9+9+9$

样例输入

3

H T

HT TH

HH TH

样例输出

Case #1:

1/2

Case #2:

1/2

Case #3:

1/4

3.2. 解题报告:

input1/output1: $n \leq 3$

方法比较多，可以用和 input2/output2 类似的方法，以 2^n 为状态。

也可以用 Monte Carlo 求出浮点数解，反过来得到精确解（可以假设分母都不会太大）。

还可以利用对称性分类，再人肉求解后打表（有不少是 1/2）。

input2/output2: $n \leq 27$

以输入的字符串 a 和 b 的所有后缀作为状态，每抛一次硬币，就有一半的概率转移到两个新的状态。

状态 a 和 b 本身是终止状态，对应的胜率分别为 1 和 2。

$p_a = 1$

$p_b = 0$

而对于其余状态 i ，则有

$p_i = 0.5 * p_{\{s(i+H)\}} + 0.5 * p_{\{s(i+T)\}}$

于是可以得到一个 n 元线性方程组，用 Gauss 消元法可解。

```
# input3/output3: n <= 729
```

算法参考 solution3.rb。

记 $k_{ij} = \text{correlation}(i, j)$, $k_i = k_{ii} - k_{ij}$

a 胜利的概率是 $k_b / (k_a + k_b)$

b 胜利的概率是 $k_a / (k_a + k_b)$

参考: String overlaps, pattern matching, and nontransitive games。

解题代码:

```
001 #include <cstdio>
002 #include <string>
003 #include <vector>
004 #include <cstdlib>
005 #include <iostream>
006 #include <algorithm>
007
008 using namespace std;
009
010 typedef long long llint;
011
012 llint gcd(llint a, llint b) {
013     return b == 0 ? a : gcd(b, a % b);
014 }
015
016 struct Fraction {
017     llint num, den;
018
019     Fraction(llint n = 0, llint d = 1): num(n), den(d) {
020         llint g = gcd(num, den);
021         num /= g;
```

```

022         den /= g;
023         if (den < 0) {
024             num = -num;
025             den = -den;
026         }
027     }
028 };
029
030 Fraction operator+(const Fraction& lhs, const Fraction& rhs) {
031     llint g = gcd(lhs.den, rhs.den);
032     return Fraction(lhs.num * (rhs.den / g) + rhs.num * (lhs.den / g),
033                    lhs.den / g * rhs.den);
034 }
035
036 Fraction operator-(const Fraction& lhs, const Fraction& rhs) {
037     return lhs + Fraction(-rhs.num, rhs.den);
038 }
039
040 Fraction operator*(const Fraction& lhs, const Fraction& rhs) {
041     llint g1 = gcd(lhs.num, rhs.den);
042     llint g2 = gcd(rhs.num, lhs.den);
043     return Fraction((lhs.num / g1) * (rhs.num / g2), (lhs.den / g2) *
044                    (rhs.den / g1));
045 }
046
047 Fraction operator/(const Fraction& lhs, const Fraction& rhs) {
048     return lhs * Fraction(rhs.den, rhs.num);
049 }

```

```

049 Fraction operator==(const Fraction& lhs, const Fraction& rhs) {
050     return lhs.num == rhs.num && lhs.den == rhs.den;
051 }
052
053
054 const int MAXN = 64;
055
056 void LU(int n, Fraction a[MAXN][MAXN], int r[MAXN], int c[MAXN]) {
057     for (int i = 0; i < n; ++i) {
058         r[i] = c[i] = i;
059     }
060     for (int k = 0; k < n; ++k) {
061         int ii = -1, jj = -1;
062         for (int i = k; ii == -1 && i < n; ++i) {
063             for (int j = k; jj == -1 && j < n; ++j) {
064                 if (a[i][j].num != 0) {
065                     ii = i;
066                     jj = j;
067                 }
068             }
069         }
070         swap(r[k], r[ii]);
071         swap(c[k], c[jj]);
072         for (int i = 0; i < n; ++i) {
073             swap(a[i][k], a[i][jj]);
074         }
075         for (int j = 0; j < n; ++j) {
076             swap(a[k][j], a[ii][j]);

```

```

077     }
078     for (int i = k + 1; i < n; ++i) {
079         a[i][k] = a[i][k] / a[k][k];
080         for (int j = k + 1; j < n; ++j) {
081             a[i][j] = a[i][j] - a[i][k] * a[k][j];
082         }
083     }
084 }
085 }
086
087 void solve(int n, Fraction a[MAXN][MAXN], int r[MAXN], int c[MAXN], Fraction
b[MAXN]) {
088     static Fraction x[MAXN];
089     for (int i = 0; i < n; ++i) {
090         x[i] = b[r[i]];
091     }
092     for (int i = 0; i < n; ++i) {
093         for (int j = 0; j < i; ++j) {
094             x[i] = x[i] - a[i][j] * x[j];
095         }
096     }
097     for (int i = n - 1; i >= 0; --i) {
098         for (int j = n - 1; j > i; --j) {
099             x[i] = x[i] - a[i][j] * x[j];
100         }
101         x[i] = x[i] / a[i][i];
102     }
103     for (int i = 0; i < n; ++i) {
104         b[c[i]] = x[i];

```



```

105     }
106 }
107
108 Fraction a[MAXN][MAXN];
109 int r[MAXN];
110 int c[MAXN];
111 Fraction b[MAXN];
112
113 int main() {
114     int n, m;
115     string x, y, z;
116     vector<string> v;
117
118     int TT, NN;
119     cin >> NN;
120     for (TT = 1; TT <= NN; TT++) {
121         cin >> x >> y;
122         v.clear();
123         n = x.length();
124         for (int i = 0; i <= n; ++i) {
125             v.push_back(x.substr(0, i));
126             v.push_back(y.substr(0, i));
127         }
128         sort(v.begin(), v.end());
129         v.erase(unique(v.begin(), v.end()), v.end());
130         m = v.size();
131
132         for (int i = 0; i < m; ++i) {

```

```

133         fill(a[i], a[i] + m, 0);
134         a[i][i] = 1;
135         b[i] = v[i] == x ? 1 : 0;
136         if (v[i] == x || v[i] == y) {
137             continue;
138         }
139         const char char_list[2] = {'H', 'T'};
140         for (int list_idx = 0; list_idx < 2; list_idx++) {
141             char c = char_list[list_idx];
142             //for (char c: {'H', 'T'}) {
143                 z = v[i] + c;
144                 while (find(v.begin(), v.end(), z) == v.end()) {
145                     z = z.substr(1);
146                 }
147                 int j = find(v.begin(), v.end(), z) - v.begin();
148                 a[i][j] = a[i][j] - Fraction(1, 2);
149             }
150         }
151
152         LU(m, a, r, c);
153         solve(m, a, r, c, b);
154 cout<<"Case #"<<TT<<": "<<endl;
155 cout<<b[0].num<<"/"<<b[0].den<<endl;
156
157 //         printf("Case #d:\n", TT);
158 //         printf("%lld/%lld\n", b[0].num, b[0].den);
159     }
160

```

```

161     return 0;
162 }

```

复赛 4: The Patterns

时间限制：5s 内存限制：65536K

4.1. 问题描述：

对于一个 1 到 n 的全排列 Q，我们定义 Q 对应的字符串 SQ，其中

$$S_Q[i] = \begin{cases} U & Q[i] < Q[i+1] \\ D & \text{otherwise} \end{cases} \quad 0 \leq i < n-1$$

例如对于 n=5, Q=32154, SQ=DDUD。

本任务为给定一个仅包含字符 'U' 与 'D' 长度为 m 的模式串 P 以及一个正整数 n，求解 1 到 n 的任意全排列中模式串出现次数的期望值。例如某个 SQ=UUU, P=UU, 则 P 出现次数为 2 次。

例如 n=3, P=U, 则 1 到 n 的所有全排列方案对应的模式串 P 出现次数分别为

Q	SQ	P 出现次数
123	UU	2
132	UD	1
213	DU	1
231	UD	1
312	DU	1
321	DD	0

则 1 到 n 的任意全排列中模式串出现次数的期望值

$$E = \frac{2+1+1+1+1+0}{6} = 1.0$$

$$(E * n!) \bmod (10^9 + 7)$$

为了避免浮点数精度误差，请输出 $(E * n!) \bmod (10^9 + 7)$ ，其中 mod 为取模操作，具体定义以及运算规则参见：[链接地址](#)

输入

输入数据第一行为一个整数 T ($1 \leq T \leq 3333$)，表示有 T 组测试数据。

下面 T 组数据，每组数据第一行包含 2 个正整数 n, m ，接下来一行包含一个字符串 P (仅包含字符 'U' 与 'D')

数据规模: $1 \leq m \leq 1000$ $m \leq n \leq 1000000$

输出

对于第 k 组数据，第一行输出 Case #k:，第二行输出 $(E * n!) \bmod (10^9 + 7)$ 。

样例输入

```
5
4 3
UUU
4 2
UU
10 8
UUUUDDDD
2 1
U
2 1
D
```

样例输出

```
Case #1:
1
Case #2:
8
Case #3:
1400
Case #4:
1
Case #5:
1
```

4.2. 解题报告:

首先计算 $n=m+1$ 的情况下, 1 到 n 的所有全排列中能匹配 P 的次数。令其为 A , 则此时期望为

$$E_{m+1} = \frac{A}{(m+1)!}$$

令事件 X_i 表示在某个排列 Q 的第 i 个元素开始, 之后的 $m+1$ 个元素符合模式串 P 。则

$$E(X_i) = \frac{A}{(m+1)!}$$

根据期望的线性

$$E = \sum_{i=0}^{n-m-1} E(X_i) = (n-m)E_{m+1} = \frac{(n-m)A}{(m+1)!}$$

则

$$(E * n!) = \frac{(n-m)A}{(m+1)!} n!$$

因此只需求解得到 A 即可得到答案。

计算 A 采用动态规划算法

$f[i, j]$ 表示前 i 个数字, 最后一个数字所在前 i 个数字中的排名为 j (排名从 1 开始)

则有

$f[1, 1] = 1$

for $i = 2..m+1$

for $j = 1..i$

if $p[i-2] == 'U'$

$f[j] = f[i-1][j-1] + f[i-1][j-2] + \dots + f[i-1][1];$

else

$f[j] = f[i-1][j] + f[i-1][j+1] \dots f[i-1][i-1];$

时间复杂度为 $O(m^3)$

而通过部分和优化可以优化到 $O(m^2)$

复赛 5: Query on the tree

时间限制: 1s 内存限制: 65536K

5.1. 问题描述

度度熊最近沉迷在和树有关的游戏了，他一直认为树是最神奇的数据结构。一天他遇到这样一个问题：

有一棵树，树的每个点有点权，每次有三种操作：

1. Query x 表示查询以 x 为根的子树的权值和。
2. Change $x\ y$ 表示把 x 点的权值改为 y 。
3. Root x 表示把 x 变为根。

现在度度熊想请更聪明的你帮助解决这个问题。

输入

第一行为数据组数 T

每组数据第一行为 N ，表示树的节点数。

后面 $N-1$ 行每行有两个数 $u\ v$ ，表示 u 和 v 之间有一条边。初始时树是以 1 号节点为根节点。

之后的一行为 N 个数表示这 N 个点的点权。

然后为整数 Q 为操作次数。

之后的 Q 行为描述中的三种操作。

输出

对于第 k 组输入数据，第一行输出 Case # k 接下来对于每个 "Query x " 操作，输出以 x 为根的子数和。

样例输入

2

5

1 2

1 3

3 4

3 5

1 2 3 4 5

5

Query 1

Change 3 10

Query 1

Root 4

Query 3

8

```
1 2
1 3
3 4
4 5
5 6
5 7
4 8
1 2 3 4 5 6 7 8
5
Query 1
Query 3
Root 5
Query 3
Query 1
```

样例输出

Case #1:

```
15
22
18
```

Case #2:

```
36
33
6
3
```

5.2. 解题报告:

树上的查询一共有三种操作，如果只是考虑前两种操作 Query 和 Change，则需要一种高效的数据结构来支持子树和的查询和更新。

采取类似于 LCA 在线算法的方式，先 DFS 得到树上顶点序列，通过顶点序列可以知道每颗子树的范围。利用树状数组的方式来记录每个顶点的权值的变更。那么 Query 和 Change 的时间复杂度都为 $O(\lg n)$

当引入第三种操作-变更根节点后，并不需要调整树的结构，而只是要在查询操作的时候做些处理。

如果 Query x 的 x 为当前 root 节点，则直接输出当前所有节点的权值和，可以用一个变量 SumOfAllTree 来记录整颗树所有节点的权值和，查询为 $O(1)$ 的复杂度

如果 x 在原树上不为当前 $root$ 节点的祖先, 即 $lca(x, root) \neq x$, 那么直接输出 x 节点所在的子树和 $SubTree(x)$ 。

否则可找出 $root$ 到 x 这条路径上 x 的儿子节点 y , 那么在当前 $root$ 的条件下 x 对应的子树的和为 $SumOfAllTree - SubTree(y)$ 。

求 lca 和求 y 的时间复杂度都可以做到 $O(\lg n)$ 。因此加上预处理后整体的时间复杂度为 $O(n \lg n + Q \lg n)$

决赛

1.1. 题目

前言

从 2005 年至今, 百度之星已经走过了十个年头。伴随着百度的成长, 百度之星一路走来也承载和实现了大家的诸多希望。

在百度的核心产品----百度推广平台中, 推荐算法不仅帮助广大商户快速找到了最合适的买家; 也为广大消费者量身定做, 推荐了他们最需要的信息。

那么今年的百度之星决赛就让我们来触及百度最核心的算法, 来体验一下精准广告推荐的魅力。

背景

展示广告以联盟网站为推广平台, 通过分析网民的行为, 借助特有的定向技术帮助企业锁定目标人群, 当目标受众浏览联盟网站时, 以固定、贴片、悬浮等广告形式呈现企业的推广信息。

其中主题词定向, 实现了“观其所览, 予其所求”; 是基于对网页内容的语义分析、提取出页面的主题词; 同时, 系统自动匹配所有设置了相同主题词的客户, 将其推广信息呈现在页面上。



文档向量表示

网民当前的浏览内容, 即反映其当前所需; 需要对网页内容进行关键信息抽取、语义分析、提取出页面的主题词向量; 以抽象的主题词向量来代表网页。

例如：文档向量可表述为：{笔记本电脑，笔记本，家用本、联想 Y470N 全能本，联想，.....



相关性得分计算

相关性度量，旨在提取出网民的潜在需求，加以精确的表达。

所以相关性良好的广告，要求广告于网民浏览的页面的主题内容非常接近甚至完全一致。

例如“移民”主题的网页，呈现“移民服务”主题的广告。反之，则视作相关性差

例如“黄金周旅游”主题的网页，呈现“黄金买卖”主题的广告

题目

精准广告推荐

度度熊今年在南方承包的桂圆果林获得大丰收。然而，度度熊却高兴不起来。眼看着桂圆就要成熟了，但是销路还没有着落。如何才能找到合适的桂圆批发商或者代理商呢？度度熊想到了网络推广。到互联网上去推销，是一个好主意。但是推广经费有限，又不能滥发广告。只有精确地找到真正关心桂圆的商户或个人，才能事半功倍地把大量桂圆快速地销售出去。

现在度度熊联系到我们，想让我们帮忙编写一套普适的精准广告推荐算法。

请编写程序，根据网页内容，判断浏览该网页的商户或个人与候选广告的相关性，为每个网页推荐最相关的三个广告。

输入数据：

1、提供网页集合 $D=\{d_1, d_2, \dots, d_m\}$ ，数据路径和输入方式

- 每行表示一个网页，文件读取“Url.txt”
- 每行网页信息包括 Url、页面的标题特征 (Title)、页面的 Mypos 特征、页面的正文特征；特征列之间以 Tab 键分隔，每列的文本特征做了分词处理，词间以空格分隔
- 数据格式：Url \t Title 切词 \t Mypos 切词 \t 正文切词；
- Url 字段示例：
http://edu.ifeng.com/abroad/detail_2011_08/26/8697589_1.shtml
- Title 标题字段示例：
留学生 牛津“镀金”归来 月薪仅 1500 元 (组图)
- Mypos 字段示例：
凤凰网 教育 > 留学 > 正文
- 正文字段示例：

凤凰网 教育 > 留学 > 正文 留学生 牛津“镀金”归来 月薪仅 1500 元 (组图) 字号:T | T
2559 人参与 115 条评论 打印 转发 香港本硕博连读生当银行柜员今年 24 岁的庄佳化名)，
高考时顺利考入香港的一所大学。本科 4 年后，他选择继续留校攻读心理学硕士。毕业后，他在香港待了将近一年时间，但始终找不到中意的工作，于是想回家乡厦门寻找机会。原以为顶着境外本硕博学位的光环，到大陆找工作会容易些，没想到还是

碰了壁。一个月里，庄佳投了5份简历，但都杳无音讯。后来，他适当放宽了要求，投了几家非心理学领域的公司，但同样没有任何回音。眼看儿子回家当了两个月“海带”，庄爸爸赶紧托关系帮他在一家中资银行找了份差事。“我被安排从普通柜员做起，培训期间每个月的基本工资在2000元左右，培训期后算上绩效工资和福利，每个月估计能拿四五千元。”庄佳说。庄佳说，本硕连读，5年的学费和生活费花了大概60万元人民币，现在拿着2000元左右的月薪，心里多少都有些不甘。“不过，我还是打算先把这份工作做下去，毕竟现在工作不好找啊。”庄佳说，开一家心理咨询公司，将是他5年内的奋斗目标。记者调查发现，如果仅以第一份工作的待遇计算，很多海归要收回“留学成本”很难。以刘宁宁为例，以每个月工资5000元计算，即便不吃不喝，也要10年半才能收回留学成本。而庄佳，假设他转正后每个月能拿到4500元，也要至少11年才能赚回留学成本。【微调查】年轻海归平均月薪3800元近日，记者通过电话、邮件以及面对面采访等方式，对30位年龄在24岁~27岁、回国前无工作经验的海归进行调查。受访者在境外的学习时间在7个月至5年不等，留学的国家/地区包括美国、英国、法国、澳大利亚、新西兰、马来西亚、新加坡、菲律宾、日本、瑞士、瑞典、香港。根据调查，这些海归目前的月薪平均在3800元左右，最低为每月1500元，最高为每月6500元。其中，月薪为4000~5000元的最多，占56%；月薪为3000~4000元的紧随其后，占35%；月薪2000元以下的占6%，月薪6000元以上的占3%。从找到工作所花时间的长短以及所获得薪酬高低来看，室内设计、市场营销、工程机械这三个专业最吃香。从调查中记者发现，在出国前就有明确职业规划的海归，回国后在择业上更为主动，职业道路也走得更顺畅。受访者苏欣化名)，当年出国前就想好了自己的择业方向，因此在瑞士学习和实习时就有明确的目标，回国后不到两个星期，就找到了一份酒店管理的工作，而且做得特别顺手。“如果出国前就有明确的方向，并顺着它一直走下去，回国后就比较容易在自己喜欢的领域施展拳脚。”苏欣说。1234 相关新闻：·我国构建留学人员回国服务体系方便留学生回国·报告称中国留学生仅三成学成归国富豪移民海外相关图片：·盘点留美最易失业的九大专业组图)免责声明：本文仅代表作者个人观点，与凤凰网无关。其原创性以及文中陈述文字和内容未经本站证实，对本文以及其中全部或者部分内容、文字的真实性、完整性、及时性本站不作任何保证或承诺，请读者仅作参考，并请自行核实相关内容。[责任编辑：杨喆]标签：庄佳镀金留学生2559人参与115条评论查看)网友评论用户名密码文明上网，登录评论！所有评论仅代表网友意见，凤凰网保持中立同步到微博社会娱乐生活探索深圳砍人现场湖北一男子持刀拒捕捅伤多人被击毙东莞：孕妇在广场上被辅警打掉5颗牙图)04/21 07:02小偷偷山寨机后逃跑慌不择路丢失iPhone5图)04/21 07:0214岁自闭症少女被后妈安排与28岁男子结婚图)04/21 07:02常州一城管自费万元装备谷歌眼镜图)04

2、广告集合 $A=\{a_1,a_2,\dots,a_n\}$ ，数据路径和输入方式

- 每行表示一个广告，文件读取“Ad.txt”
- 每行广告信息包括广告Id，广告标题(Title)、广告描述1(Desc)、广告描述2(Desc)；特征列之间以Tab键分隔，每列的文本特征做了分词处理，词间以空格分隔
- 数据格式：广告Id\tTitle分词\t描述1分词\t描述2分词；
- 字段示例：

广告Id字段：100089188

Title字段：实木复合门十大品牌

描述1字段：唐噶尔,全国木门30强,绿色环保优质品牌

描述2字段：实木复合门十大品牌,就选唐噶尔.

输出数据：

- 为每个网页选取出最相关的三个广告
- 输出数据格式：

- a) 文件形式输出，文件名“Url_Adid_Rst”；
- b) 为目标文件的每个 Url 补充上最相关的三个 Adid 信息；
- c) 数据格式：Url \t Adid1 \t Adid2 \t Adid3 （至多三个广告）；

评分标准：

1、0 分：

不能按规定格式输出的，该条记录为 0 分

整体处理时间不允许超时，否则整体记录为 0 分

如有存在其他犯规情况（例如使用多线程编程），裁判有权判定该程序的部分甚至整组记录得 0 分

2、有效得分

在所有满足格式、内存、时间等要求的提交程序中，对于每条记录，基于评测结果，正确的一个 <Url, Adid>结果，计 1 分

3、排名规则

统计所有记录的得分，总计得分较高的选手排名靠前

如果依然不能区别，最后比较整体结果的处理时间，总用时较少者获胜

数据规模和样例

1、比赛选手测试数据和最终评测数据的规模和分布都基本相同，测试数据涉及 Url 492 条，Ad 5519 条，Good 标注数据 3829 个 Pair

2、数据文件说明：

- 1) Url_Info 文件，数据格式：url \t title 切词 \t mypos 切词 \t 正文切词；
- 2) Ad_Info 文件，数据格式：adid \t title 切词 \t 描述 1 切词 \t 描述 2 切词；
- 3) Url_Adid 文件，数据格式：url \t adid；

题目规则约束及补充

- 1、不允许多线程；
- 2、不允许修改 URL 字段的内容；

1.2. 解题思路

根据为所给 url 中的内容推荐一个广告，而这个题目中所给 url 中的信息中不含有用户信息，因此显然是根据网页内容去推荐。

而常见的推荐算法：比如协同过滤等算法，大都需要用户的一些行为信息以及一些典型的学习算法，而这里并没有提供这些信息及算法库什么的。又考虑到只有 6 个小时的时间，因此暂考虑不传统的这些方法。

如果把广告作为用户要找的内容，而网页中内容做为输入的关键词，那么这里的推荐从这个角度来看就是信息检索，因为信息检索的本质也是根据用户所给的关键词，找到与这个关键词最“相关”的内容，而在“找最相关”这一点，推荐与检索是相通的。因此可以用信息检索的思路来解决这个问题，且可以在搭建后检索的平台下调整排序所需的权重及策略来提高推荐质量。而信息检索比较典型且已证明较为有效的权值计算方法就是，逆文档频率，因此可以先用此权重方法做一个基本的算法。然后根据跑出的结果及时间进行调整。

而一个典型的信息检索分为两大块，一个是索引的建立，另一个则是检索的模型。考虑到所给时间有限及编码的方便性，这里选用 python 来做为编程语言。

以上分析部分大概花了约 1 个小时而具体实现时也分为两个模块，一个是索引的建立，建立词到广告的索引。

另一个则检索模型，根据一个网页内容进行拉链归并然后根据 $tf*idf$ 的值进行排序最后返回值最大的广告，从而做为推荐的广告。以上搭建检索框架大约用了 2 个小时。

a 完成两个模块后进行第一次的实验，发现有大量的 url 都指向了一个广告，显然不正常，经过分析发现，网页与广告中混有大量的中、英文标点符号。因此需要对这些字符进行过滤，且过滤时同时考虑到部分像 “的” “是” 等高频词。过滤完之后，结果相对较为正常。此部分也大概花了 1 个小时。

b 由于网页分为 title 与内容，广告也分为了若干项，而不同项的权值显然是不一样的。因此这里尝试了用两步检索的方法，即先用网页的 title 来进行检索，如果能检索到则返回权值最大的广告，否则则用其内容进行检索，然后再返回权值最大的结果。然后分析两步的方法与网页所有内容放在一起的方法的结果的对比。最后选用的是两步的方法。

此部分大概花了 2 个小时。

a 信息检索有很多模型，这里只用了一种，其它的模型并没有对比，并不知那种模型效果最好。

b 用于推荐的技术不仅仅是检索，还有其它的方法也并没有太多了解。

c 没有综合考虑网页所给不同属性与广告属性的权值，如果考虑这些，最终的效果应该会更好。

请大家帮忙审核一下这份题集，如果解题思路有问题，或是大家有更好的思路，请发邮件告知组委会。
联系邮箱: astar@baidu.com
