第5章 百度之星2017

5.1 资格赛

度度熊保护村庄

时间限制：2000/1000 毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

哗啦啦村袭击了喵哈哈村！

度度熊为了拯救喵哈哈村，带着自己的伙伴去救援喵哈哈村去了！度度熊与伙伴们很快的就过来占据了喵哈哈村的各个军事要地，牢牢守住了喵哈哈村。但是度度熊发现，这是一场旷日持久的战斗，所以度度熊决定要以逸待劳，保存尽量多的体力，去迎战哗啦啦村的战士。于是度度熊决定派尽量多的人去休息，但是同时也不能松懈对喵哈哈村的保护。换句话而言，度度熊希望尽量多的人休息，而且存在一个包围圈由剩下的人组成，且能够恰好包围住喵哈哈村的所有住房(包括边界)。

请问最多能让多少个人休息呢？

输入

本题包含若干组测试数据。

第一行一个整数 n，表示喵哈哈村的住房数量。

接下来 n 行，每行两个整数($x1[i]$,$y1[i]$)，表示喵哈哈村的住房坐标。

第 $n+1$ 行一个整数 m，表示度度熊的士兵数量。

接下来 m 行，每行两个整数($x2[i]$,$y2[i]$)，表示度度熊伙伴的坐标。

满足：$1 \leqslant n,m \leqslant 500$，$-10000 \leqslant x1[i],x2[i],y1[i],y2[i] \leqslant 10000$。

输出

请输出最多的人员休息的数目。如果无法保护整个村庄的话，输出 ToT。

输入样例

```
2
1 1
```

```
2 2
4
0 0
0 4
4 2
4 0
1
1 1
2
0 0
0 1
```

输出样例

```
1
ToT
```

关键思路

特判掉 $n=1$ 和 $n=2$ 的情况。

然后做一个由 n 个点组成的凸包，找到所有在凸包外面的边（由士兵组成）。

假设由第 i 个点和第 j 个点组成的边在凸包外的话，令 $mp[i][j]=1$。然后就可以跑 Floyd 了，表示第 i 个点到第 j 个点所能组成的最短边。最小的 $mp[i][i]$就是答案（实际上就是绕了一圈）。

度度熊的王国战略

时间限制：40000/20000 毫秒（Java/其他）

空间限制：32768/132768KB（Java/其他）

题目

度度熊国王率领着喵哈哈族的勇士，准备进攻哔啦啦族。哔啦啦族是一个强悍的民族，里面有充满智慧的谋士，拥有无穷力量的战士。所以这一场战争，将会十分艰难。为了更好地进攻哔啦啦族，度度熊决定首先从内部瓦解哔啦啦族。第一步就是使得哔啦啦族内部不能齐心协力，需要内部有间隙。哔啦啦族一共有 n 个将领，他们一共有 m 个强关系，摧毁每一个强关系都需要一定的代价。

现在度度熊命令你摧毁一些强关系，使得内部的将领，不能通过这些强关系，连成一个完整的连通块，以保证战争的顺利进行。

请问最少应该付出多少的代价。

输入

本题包含若干组测试数据。

第一行两个整数 n，m，表示有 n 个将领，m 个关系。

接下来 m 行，每行三个整数 u，v，w。表示 u 将领和 v 将领之间存在一个强关系，摧毁这个强关系需要代价 w。

数据范围：$2\leqslant n\leqslant 3000$，$1\leqslant m\leqslant 100000$，$1\leqslant u,v\leqslant n$，$1\leqslant w\leqslant 1000$

输出

对于每组测试数据，输出最小需要的代价。

输入样例

```
2 1
1 2 1
3 3
1 2 5
1 2 4
2 3 3
```

输出样例

```
1
3
```

关键思路

这道题是一个经典题了，是无向图的最小割。

采用 Stoer-Wagner 算法，再用堆优化，就可以将复杂度优化到 $n^2\log n$。

度度熊与邪恶大魔王

时间限制：2000/1000 毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

度度熊为了拯救可爱的公主，于是与邪恶大魔王战斗起来。邪恶大魔王的麾下有 n 个怪兽，每个怪兽有 $a[i]$的生命值，以及 $b[i]$的防御力。度度熊一共拥有 m 种攻击方式，第 i 种攻击方式，需要消耗 $k[i]$的晶石，造成 $p[i]$点伤害。

如果度度熊使用第 i 个技能打在第 j 个怪兽上面的话，会使得第 j 个怪兽的生命值减少 $p[i]-b[j]$，如果伤害小于防御，那么攻击就不会奏效。如果怪兽的生命值降为 0 或以下，那么怪兽就会被消灭。当然每个技能都可以使用无限次。

请问度度熊最少携带多少晶石，就可以消灭所有的怪兽。

输入

本题包含若干组测试数据。

第一行两个整数 n，m，表示有 n 个怪兽，m 种技能。

接下来 n 行，每行两个整数，$a[i]$，$b[i]$，分别表示怪兽的生命值和防御力。

再接下来 m 行，每行两个整数 $k[i]$和 $p[i]$，分别表示技能的消耗晶石数目和技能的伤害值。

数据范围：

$1\leqslant n\leqslant 100000$，$1\leqslant m\leqslant 1000$，$1\leqslant a[i]\leqslant 1000$，$0\leqslant b[i]\leqslant 10$，

$0\leqslant k[i]\leqslant 100000$，$0\leqslant p[i]\leqslant 1000$

输出

对于每组测试数据，输出最小的晶石消耗数量，如果不能击败所有的怪兽，输出 -1

输入样例

```
1 2
3 5
7 10
```

```
6 8
1 2
3 5
10 7
8 6
```

输出样例

```
6
18
```

关键思路

仔细观察，防御力实际上只有 11 种情况，对于每一种防御力，我们都做一次背包 DP，预处理出 DP 状态。然后对于每一个怪兽，我们 $O(1)$ 的进行回答即可。

度度熊的午饭时光

时间限制：2000/1000 毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

度度熊最期待每天的午饭时光，因为早饭菜品清淡，晚饭减肥不敢吃太多。

百度食堂的午餐超级丰富，祖国各大菜系应有尽有，度度熊在每个窗口都有爱吃的菜品，而且他还为喜爱的菜品打了分。

但是，好吃的饭菜总是很贵，每天的午饭预算有限，请帮度度熊算一算，怎样打饭才能买到最好吃的饭菜？（不超过预算、不重样、午餐等分最高的情况下，选择菜品序号加和最小，加和相等时字典序最小的组合）。

输入

第一行一个整数 T，表示 T 组数据。

每组测试数据将以如下格式从标准输入读入：

```
B
N
score_1 cost_1
score_2 cost_2
:
score_N cost_N
```

第一行，正整数 B($0 \leqslant B \leqslant 1000$)，代表午餐的预算。

第二行，正整数 N($0 \leqslant N \leqslant 100$)，代表午餐可选的菜品数量。

从第三行到第(N+2)行，每行两个正整数，以空格分隔，score_i 表示菜品的得分，cost_i 表示菜品的价格($0 \leqslant score\_i, cost\_i \leqslant 100$)。

输出

对于每组数据，输出两行：

第一行输出：Case #i:。i 代表第 i 组测试数据。

第二行输出菜品的总得分和总花费，以空格分隔。

第三行输出所选菜品的序号，菜品序号从 1 开始，以空格分隔。

输入样例

```
2
29
6
9 10
3 4
6 5
7 20
10 9
15 11
0
2
2 23
10 12
```

输出样例

```
Case #1:
34 29
2 3 5 6
Case #2:
0 0
```

关键思路

本题可以明显地发现是一个 dp 问题，考虑到较小的数据范围，可以使用比较直观的解法。既然判断题目是 dp，那么就应该有状转方程，这个状转方程和背包问题是相同的：

$$dp[i]=\max(dp[i],dp[j]+score[i-j])$$

为了直观，可以采用从 0 开始向后做 dp。因此外层循环从 0 到 b 枚举最多总花费，然后内层循环枚举菜品的价格，根据上述状转方程进行 dp。

但是题目条件还给出了一些限制。一个是不能重复。可以在更新状态时判断当前新加的菜品是否已被选取。另一个是对最优解的限定。如果遇到两个方案的得分是相同的，那么就需要用题中条件，即序号和和字典序来判断哪个方案更优。这两个条件可以分别写一个函数做判断。

寻找母串

时间限制：2000/1000 毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

对于一个串 S，当它同时满足如下条件时，它就是一个 01 偏串：

1. 只由 0 和 1 两种符组成；
2. 在 S 的每一个前缀中，0 的个数不超过 1 的个数；
3. S 中 0 的个数和 1 的个数相等。

现在给定 01 偏串 S，请计算一下 S 在所有长度为 n 的 01 偏串中作为子串出现的次数的总和。

由于结果比较大，结果对 1e9+7 取余后输出。

输入

第一行给出一个整数 $T(1\leqslant T\leqslant 40)$，表示测试数据的数目。

每一组测试包含一个整数 n 和字符串 S，中间用空格分开$(1\leqslant |S|\leqslant 100000, 1\leqslant n\leqslant 1000000000)$。

输入保证 S 是一个 01 偏串。

输出

对于每一组数据，输出一个整数占一行，表示答案。

输入样例

```
2
2 10
4 10
```

输出样例

```
1
3
```

样例解释

在第二个样例中，长度为 4 的偏串共两个 1010，1100。10 在 1010 中出现了两次，在 1100 中出现了 1 次。所以答案是 3。

关键思路

由于求的是在原串中出现次数的总和，那么可以枚举 S 在原串中出现的位置，然后求出原串有多少种。可以发现，如果把 S 从那个位置中拿掉之后，剩下来的串还是满足偏串定义的。这个是为什么呢？假设原串是 T，而 S 出现在 k 的位置，那么原串的形状是这样的 $T[0\cdots k-1]+S+T[k+\text{len}(s)\cdots n-1]$。

对于 $T[0\cdots k-1]$这个串，因为它的前缀是原串前缀的一部分，所以它的前缀中 0 和 1 的差是和原串一样的，而对于 $T[0\cdots k-1]+T[k+\text{len}(s)\cdots n-1]$这个串中结尾在$[k+\text{len}(s), n-1]$这个区间的前缀，因为 S 是一个偏串，里面的 0 和 1 数目相等，所以把它从原串中拿掉之后，并不影响到后面 0 和 1 数目的差，所以结尾在$[k+\text{len}(s), n-1]$这一区间内的前缀中，0 和 1 数目的差值是和原串一样的。因而 $T[0\cdots k-1]+T[k+\text{len}(s)\cdots n-1]$还是一个偏串。

所以只要求一下 $n-\text{len}(s)$的偏串有多少种即可了。记 $m=n-\text{len}(s)/2$。这个问题可以转化为在一个平面坐标系中，每次只向右走一步或者向上走一步，但是不能跨过对角线，从$(0,0)$走到(m,m)的方法数，这个答案是卡特兰数$\frac{1}{m+1}C_{2m}^{m}$。

又由于 S 在原串出现的位置有 $n-len(s)+1$ 种，所以最后答案是：

$$\frac{2m+1}{m+1}C_{2m}^{m}=C_{2m+1}^{m}=\frac{(2m+1)!}{(m+1)!\,m!}$$

这个问题是如何求阶乘，由于 m 是比较大的，对一个大素数取余，可以用分段打表法来求，按照 1 百万分段，先计算出 $fac[1000000*k]$的值，然后对于一个数字 n 从 $\text{fac}[n/1000000*1000000]$这个位置暴力求到 n，这样的复杂度是 $O(1000000)$。最后对分母部分求逆元即可。

另外注意一下边界情况。

5.2 初赛第一场

小C的倍数问题

时间限制：2000/1000毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

根据小学数学的知识，我们知道一个正整数 x 是3的倍数的条件是：x 每一位加起来的和是3的倍数。反之，如果一个数每一位加起来是3的倍数，则这个数肯定是3的倍数。

现在给定进制 P，求有多少个 B 满足 P 进制下，一个正整数是 B 的倍数的充分必要条件是每一位加起来的和是 B 的倍数。

输入

第一行一个正整数 T 表示数据组数($1 \leqslant T \leqslant 20$)。

接下来 T 行，每行一个正整数 P($2 < P < 1e9$)，表示一组询问。

输出

对于每组数据输出一行，每一行一个数表示答案。

输入样例

```
1
10
```

输出样例

```
3
```

关键思路

我们考虑一个位上的值是 d($d < B$)，那么 d 进到下一位的值是 $d * P$，考虑模拟取模运算，那对下一位的贡献是 $d * P \% B$。于是得到方程：$d * P \% B = d$。由于对所有 d 都满足，所以可以看成是 $P \% B = 1$。于是 B 是 $P-1$ 的约数，直接分解约数即可。

数据分割

时间限制：2000/1000毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)：

题目

小 w 来到百度之星的赛场上，准备开始实现一个程序自动分析系统。

这个程序接受一些形如 $x_i = x_j$ 或 $x_i \neq x_j$ 的相等/不等约束条件作为输入，判定是否可以通过给每个 w 赋适当的值，来满足这些条件。

输入包含多组数据。然而粗心的小w不幸地把每组数据之间的分隔符删掉了。他只知道每组数据都是不可满足的，且若把每组数据的最后一个约束条件去掉，则该组数据是可满足的。

请帮助他恢复这些分隔符。

输入

第1行：一个数字 L，表示后面输入的总行数。

之后 L 行，每行包含三个整数，i,j,e，描述一个相等/不等的约束条件，若 $e=1$，则该约束条件为 $x_i=x_j$，若 $e=0$，则该约束条件为 $x_i\neq x_j$。

$$i,j,L\leqslant 100000,\quad x_i,x_j\leqslant L$$

输出

输出共 $T+1$ 行。

第一行一个整数 T，表示数据组数。

接下来 T 行的第 i 行，一个整数，表示第 i 组数据中的约束条件个数。

输入样例

```
6
2 2 1
2 2 1
1 1 1
3 1 1
1 3 1
1 3 0
```

输出样例

```
1
6
```

关键思路

考虑用并查集维护，相等的变量在同一个连通块中。

维护每个不等关系(x,y)为以 x 为根的连通块与以 y 为根的连通块不等，这样合并以 x,y 为根的连通块的时候，我们以和每个连通块相关的不等关系个数作为它的 $rank$，使用按秩合并的思想合并。假设我们把以 y 为根的连通块合并到以 x 为根的连通块，我们枚举每一个与 y 相关的不等关系(y,z)，将其改为(x,z)，如果遇到(y,x)，则说明出现不合法情况，退出即可。不等关系(x,y)可以用链表维护与并查集一同合并。

复杂度 $O(N\log N)$。

路　径　交

时间限制：6000/3000 毫秒(Java/其他)

空间限制：132768/132768 K(Java/其他)

题目

给定一棵 n 个点的树，以及 m 条路径，每次询问第 L 条到第 R 条路径的交集部分的长度(如果一条边同时出现在2条路径上，那么它属于路径的交集)。

输入

第一行一个数 $n(n\leqslant 500000)$。

接下来 $n-1$ 行，每行三个数 x,y,z，表示一条从 x 到 y 并且长度为 z 的边。

第 $n+1$ 行一个数 m($m \leqslant 500000$)。

接下来 m 行,每行两个数 u,v,表示一条从 u 到 v 的路径。

接下来一行一个数 Q,表示询问次数($Q \leqslant 500000$)。

接下来 Q 行,每行两个数 L 和 R。

输出

Q 行,每行一个数表示答案。

输入样例

```
4
1 2 5
2 3 2
1 4 3
2
1 2
3 4
1
1 2
```

输出样例

```
5
```

关键思路

求两条路径的并可以通过四点间的 lca 讨论得到。

用 st 表预处理第 i 条到 $i+2^j-1$ 条路径的并如果用 rmq 问题的解法求 lca,那么这一步就是 $O(m\log m)$的。对于询问,用两条预处理好的路径求并即可。

总时间复杂度 $O(n\log n+m\log m)$。

迷宫出逃

时间限制:2000/1000 毫秒(Java/其他)

空间限制:32768/32768KB(Java/其他)

题目

小明又一次陷入了大魔王的迷宫,在无人机的帮忙下,小明获得了整个迷宫的草图。

不同于一般的迷宫,魔王在迷宫里安置了机关,一旦触碰,那么四个方向所在的格子,将翻转其可达性(原先可通过的格子不可通过,反之亦然,机关可以反复触发)。为了防止小明很容易地出逃,魔王在临走前把钥匙丢在了迷宫某处,只有拿到钥匙,小明才能打开门在出口处离开迷宫。

万般无奈之下,小明想借助聪明的你,帮忙计算是否有机会离开这个迷宫,最少需要多少时间。(每一单位时间只能向四邻方向走一步)

输入

第一行为 T,表示输入数据组数。

下面 T 组数据,对于每组数据:

第一行是两个数字 n,m($2<n*m\leqslant 64$),表示迷宫的长与宽。

接下来 n 行,每行 m 个字符,'.'表示空地可以通过,'x'表示陷阱,'*'表示机关,'S'代表起点,

‘E’代表出口，‘K’表示钥匙（保证存在且只有一个）。

输出

```
对第 i 组数据，输出
Case #i:
```

然后输出一行，仅包含一个整数，表示最少多少步能够拿到钥匙并走出迷魂阵，如果不能则打出-1。

输入样例

```
5
5 7
...*x..
...x...
xEx....
 *x...K.
.x*...S
5 7
K..*x..
...x...
xEx....
 *x.....
.x*...S
5 7
..K*x..
..*x*..
xEx....
 *x.....
.x*...S
5 7
..K*x..
. *xx*..
 *E*....
xx.....
.x*...S
4 4
S*..
 **..
...E
...K
```

输出样例

```
Case #1:
11
Case #2:
13
Case #3:
13
Case #4:
```

```
11
Case #5:
-1
```

关键思路

$2<n*m\leqslant 64$，范围很小，明显的广度优先搜索。

考虑实现一个带状态压缩的广度优先搜索，一个状态需要包含以下的信息：①所有机关的状态，用一个无符号 64 位整数来存储，某机关对应位置上为 1 表示机关被触发，为 0 则未被触发。②当前走到的格子位置。③已走的距离。④是否拿到了钥匙。记录某个状态是否被访问过可以用哈希表来实现。

在搜索时，每次从队列中取出一个状态并拓展出新的状态。如何拓展新的状态？枚举当前所在格子周围的四个格子，若是在操作完状态中记录的那些机关后，新的格子仍可通行，并且新的状态未被访问过，便可将新的状态扔到队列里继续搜索。注意判断是否取得了钥匙或是到达了出口。

提示：不可通行与陷阱不一样；机关触发后对它旁边的机关也有影响。

今 夕 何 夕

时间限制：2000/1000 毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

今天是 2017 年 8 月 6 日，农历闰六月十五。小度独自凭栏，望着一轮圆月，发出了“今夕何夕，见此良人”的寂寞感慨。为了排遣郁结，他决定思考一个数学问题：接下来最近的哪一年里的同一个日子，和今天的星期数一样？比如今天是 8 月 6 日，星期日。下一个也是星期日的 8 月 6 日发生在 2023 年。

小贴士：在公历中，能被 4 整除但不能被 100 整除，或能被 400 整除的年份即为闰年。

输入

第一行为 T，表示输入数据组数。

每组数据包含一个日期，格式为 YYYY-MM-DD。$1\leqslant T\leqslant 10000$，$YYYY\geqslant 2017$，日期一定是个合法的日期。

输出

对每组数据输出答案年份，题目保证答案不会超过四位数。

输入样例

```
3
2017-08-06
2017-08-07
2018-01-01
```

输出样例

```
2023
2023
2024
```

关键思路

简单模拟。由于答案不会超过四位数，按年枚举判断一下即可，间隔也不会很大。有一个小麻烦是

2月29号不是每一年都会有的。

度度熊的01世界

时间限制：2000/1000毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

度度熊是一个喜欢计算机的孩子，在计算机的世界中，所有事物实际上都只由0和1组成。现在给你一个 $n * m$ 的图像，你需要分辨它究竟是0，还是1，或者两者均不是。

图像0的定义：存在1字符且1字符只能是由一个连通块组成，存在且仅存在一个由0字符组成的连通块完全被1所包围。

图像1的定义：存在1字符且1字符只能是由一个连通块组成，不存在任何0字符组成的连通块被1所完全包围。

连通的含义是，只要连续两个方块有公共边，就看作是连通。

完全包围的意思是，该连通块不与边界相接触。

输入

本题包含若干组测试数据。

每组测试数据包含：

第一行两个整数 n，m 表示图像的长与宽。

接下来 n 行 m 列将会是只有01组成的字符画。

满足 $1 \leqslant n, m \leqslant 100$。

输出

如果这个图是1的话，输出1；如果是0的话，输出0，都不是输出-1。

输入样例

```
32 32
00000000000000000000000000000000
00000000000111111110000000000000
00000000001111111111100000000000
00000000001111111111110000000000
00000000011111111111111000000000
00000000011111100011111000000000
00000000111111000000111100000000
00000000111111000000111110000000
00000000111111000000011111000000
00000001111111000000011111000000
00000001111111000000001111000000
00000001111111000000000111100000
00000001111110000000000111100000
00000001111100000000000111100000
00000001111100000000000111100000
00000001111100000000000111100000
00000001111000000000000111100000
00000001111000000000000111100000
00000001111000000000000111000000
```

```
00000000111100000000000111000000
00000001111000000000011110000000
00000001111000000000011110000000
00000001111000000000011110000000
00000001111100000111111110000000
00000001111110001111111100000000
00000001111111111111111000000000
00000000111111111111111000000000
00000001111111111111100000000000
00000000111111111111000000000000
00000000001111111000000000000000
00000000001111100000000000000000
00000000000000000000000000000000
32 32
00000000000000000000000000000000
00000000000000001111110000000000
00000000000000001111111000000000
00000000000000011111111000000000
00000000000000111111110000000000
00000000000000111111110000000000
00000000000000111111110000000000
00000000000000111111110000000000
00000000000001111111100000000000
00000000000001111111100000000000
00000000000001111111100000000000
00000000000001111111100000000000
00000000000001111111100000000000
00000000000001111111100000000000
00000000000001111111100000000000
00000000000001111111100000000000
00000000000001111111100000000000
00000000000001111111100000000000
00000000000001111111100000000000
00000000000001111111100000000000
00000000000001111111100000000000
00000000000001111111100000000000
00000000000000111111110000000000
00000000000000001111111111000000
00000000000000001111111111000000
00000000000000000111111111000000
00000000000000000000000000000000
3 3
101
101
011
```

输出样例

```
0
1
-1
```

关键思路

如果没有1，直接输出－1。

如果有多个1的联通块，输出－1。

没有被包围的0联通块，输出1。

如果只有一个被包围的0联通块，输出0。

其他输出－1。

按着这个顺序判断即可。

5.3 初赛第二场

Chess

时间限制：2000/1000毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

車是中国象棋中的一种棋子，它能攻击同一行或同一列中没有其他棋子阻隔的棋子。一天，小度在棋盘上摆起了许多車……他想知道，在一共 $N \times M$ 个点的矩形棋盘中摆最多个数的車使其互不攻击的方案数。他经过思考，得出了答案。但他仍不满足，想增加一个条件：对于任何一个車 A，如果有其他一个車 B 在它的上方(車 B 行号小于車 A)，那么車 A 必须在車 B 的右边(車 A 列号大于車 B)。

现在要问问你，满足要求的方案数是多少。

输入

第一行一个正整数 T，表示数据组数。

对于每组数据：一行，两个正整数 N 和 M($N \leqslant 1000$，$M \leqslant 1000$)。

输出

对于每组数据输出一行，代表方案数模1000000007(1e9＋7)。

输入样例

```
1
1 1
```

输出样例

```
1
```

关键思路

求 $C(n,m)$ 是一个经典问题，有许多优秀的算法。这里笔者提供一种比较简单优美的做法。读者需对筛选质数、快速幂、高精度算法有所掌握。我们观察筛选质数算法：

```
i = 2 to n
   if i 是质数
j = n/i downto 2
将 i * j 标记为合数
```

这个算法可被证明为 $O(N\log N)$的，而且由于除法较少，实测效率可与其他线性的筛法不相上下。我们对这个算法进行改进，原理是质因数分解。我们对于每一个数再记录下最后答案要乘它的 tot 次方。开始对于 $tot[1$ 到 $m]=-1$，$tot[n-m+1$ 到 $n]=1$。那么我们每次将 $i*j$ 标记为合数时，顺便把 $i*j$ 拆分为 i 和 j。而且这样做可以保证最后所有的 tot 非负(因为答案为整数，除得尽)。然后最后将所有的数的 tot 次方相乘即可(这步可以用快速幂优化，但实测相差很小)，其中只需要高精度乘法。

时间复杂度：$O(N\log N$ * bignumer()[最后 50 位])。

Factory

时间限制：20000/10000 毫秒(Java/其他)

空间限制：132768/132768KB(Java/其他)

题目

我们将 A 省简化为由 N 个城市组成，某些城市之间存在双向道路，而且 A 省的交通有一个特点就是任意两个城市之间都能通过道路相互到达，且在不重复经过城市的情况下任意两个城市之间的到达方案都是唯一的。聪明的你一定已经发现，这些城市构成了树这样一个结构。

现在百度陆续开了许许多多的子公司。每家子公司又会在各城市中不断兴建属于该子公司的办公室。由于各个子公司之间经常有资源的流动，所以公司员工常常想知道，两家子公司间的最小距离。我们可以把子公司看成一个由办公室组成的集合。那么两个子公司 A 和 B 的最小距离定义为 $\min(dist(x,y))(x\in A,y\in B)$。其中 $dist(x,y)$表示两个办公室之间的最短路径长度。

现在共有 Q 个询问，每次询问分别在两个子公司间的最小距离。

输入

第一行一个正整数 T，表示数据组数。

对于每组数据：

第一行两个正整数 N 和 M。城市编号为 1 至 N，子公司编号为 1 至 M。

接下来 $N-1$ 行给定所有道路的两端城市编号和道路长度。

接下来 M 行，依次按编号顺序给出各子公司办公室所在位置，每行第一个整数 G，表示办公室数，接下来 G 个数为办公室所在位置。

接下来一个整数 Q，表示询问数。

接下来 Q 行，每行两个正整数 a，b(a 不等于 b)，表示询问的两个子公司。

【数据范围】

$0\leqslant$边权$\leqslant 100$，$1\leqslant N,M,Q,$工厂总数$\leqslant 100000$

输出

对于每个询问，输出一行，表示答案。

输入样例

```
1
3 3
```

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

输出样例

```
1
0
0
```

关键思路

记 N 为树节点数，M 为询问次数，C 为总工厂数，假定 N 和 C 同阶。

【算法一】

一个技巧是对于一个点建有多个工厂的情况，将其拆成多个点，连权为 0 的边。这样保证每个点至多有一个工厂。那么点数实际上变为 $O(N+C)$的了。

先对询问离线。考虑对树进行点分治。我们对于一次分治求解，把所有树里涉及到的询问用过根的路径信息更新一遍。设分治时树的大小为 S，那么这个一次显然是 $O(S+Q)$的。这样做总复杂度为 $O(N\log N*Q)$。

考虑另一种暴力，我们对于一棵树，暴力枚举根，求出到其他点的距离，那么根所在的询问就能全部更新了。设树的大小为 S，那么这一次显然是 $O(S\hat{}2)$的。这样做总复杂度为 $O(N\hat{}2+Q)$。

两者结合，树大的时候点分，小的时候暴力。

复杂度分析。设阈值为 S。由于每次点分能保证递归下去的树的大小减半。考虑做 T 层点分，那么树最大不超过 $N/(2\hat{}T)$。

当 $N/(2\hat{}T)=S$ 时 $T=\log(N/S)$，分治求解的总次数为 $2\hat{}T=2\hat{}\log(N/S)=N/S$。分治的复杂度为 $O(N/S*Q+N\log T)$，暴力的复杂度为 $O(S\hat{}2*(N/S))$。

两者均衡一下，不难得出 S 取 sqrt(Q)时最优，总复杂度为 $O(N\,\mathrm{sqrt}(Q))$。

【算法二】

考虑按公司的被涉及到的询问次数分类计算。

考虑两种公司，涉及到的询问次数大于等于 P 的为忙公司，否则为闲公司。

有忙公司的计算：暴力做，总复杂度为 $O(Q/P*N)$。

闲公司和闲公司的计算：整体点分治，每个闲公司的询问次数$<P$，每次分治计算时将所有询问更新即可。总复杂度为 $O(N\log N*P)$。

两者均衡一下，不难得出 P 取 sqrt(Q/log$N$)时最优，总复杂度为 $O(\mathrm{Nsqrt}(Q\log N))$。

【算法三】

考虑每次询问的两个公司。记工厂个数大于等于 D 的为大公司，否则为小公司。

有大公司的询问：枚举大公司暴力计算，总复杂度为 $O((N/D)*N)$。

小公司和小公司的询问：考虑对原树建出关于这两个公司工厂的虚树。注意到建虚树时需要排序。然后直接 dp 即可。单次复杂度为 $O(D\log N)$。总复杂度为 $O(Q*D\log N)$。

两者均衡一下，不难得出 D 取 $N/\mathrm{sqrt}(Q\log N)$时最优，总复杂度为 $O(N\,\mathrm{sqrt}(Q\log N))$。

注意到，这个算法是可以将询问做到在线的。

【算法四】

还是考虑对树进行点分治。我们可以先将点分治出来的树用 $O(N\log N)$空间记下来。对于所有分治的树根，记下一张所有涉及树里出现的工厂所在的公司到根最短距离的列表。考虑包含公司 x 的树根数量 $\mathrm{cnt}T$，由于一个工厂最多在 $\log N$ 个树根里出现，那么可以肯定公司 x 至少有 $\mathrm{cnt}T/\log N$ 个工厂。

考虑询问的两个公司，取 $\mathrm{cnt}T$ 的阈值为 V。如果其中至少有一个 $\mathrm{cnt}T$ 比较小，我们暴力将包含它的树根扫一遍，这个复杂度是 $O(V)$的，总复杂度为 $O(Q*V)$。

如果两个 $\mathrm{cnt}T$ 都比较大，考虑到它们都至少有 $\mathrm{cnt}T/\log N$ 个工厂，那么这样的公司最多有 $O(N/(V/\log N))=O(N\log N/V)$个，并且对数不超过 $O((N\log N/V)\hat{}2)$。每次遇到时直接 $O(N)$暴力做，并且将已算过的对数记下来。总复杂度为 $O(N\hat{}2\log N/V)$。

两者均衡一下，不难得出 V 取 $N\,\mathrm{sqrt}((\log N)/Q)$时最优，总复杂度为 $O(N\,\mathrm{sqrt}(Q\log N))$。

注意到，这个算法是可以将询问做到在线的。

这些算法都是基于对某个量的大小两种可能，设计不同的算法，且算法复杂度依赖的量并不一样，取一个均衡点从而达到优化算法的效果。

路 径 计 数

时间限制：16000/8000 毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

一个包含四个点的完全图，可以在任意节点出发，可以在任意节点结束，给出每个点被经过的次数，求有多少种合法的遍历序列。如果两个序列至少有一位是不同的，则认为它们不相同。

样例：

```
1 2 1 0
ABCB
BABC
BACB
BCAB
BCBA
CBAB
```

输入

多组数据。

对于每一组数据：

第一行四个数，分别表示 4 个点被经过的次数(每个数小于等于 1000，经过次数可以为 0)。

输出

一个表示答案，对 998244353 取模。

输入样例

```
2 3 3 3
```

输出样例

```
12336
```

关键思路

题目可以转化为，有多少种排列 a 个 A，b 个 B，c 个 C，d 个 D 的方法，使得任意两个相邻的字母不同。假设把 A 分为 i 堆，那么有 C_{a-1}^{i-1} 种分法。其他种类同理。假设我们 4 类分为 i，j，k，l 组，总共 n 个，则至少有 $n-i-j-k-l$ 对重复的字母。通过容斥原理我们知道，合法方案数为：

任意排列方案数－至少有一对重复字母的数量＋至少有两对重复字母的数量－……

对于分为 i，j，k，l 堆后，方案数为 $\frac{(i+j+k+l)!}{i!\ j!\ k!\ l!}\cdot C_{a-1}^{i-1}C_{b-1}^{j-1}C_{c-1}^{k-1}C_{d-1}^{l-1}$ 种。

故答案为：

$$\sum_{x=1}^{n}(-1)^{n-x}\cdot x!\sum_{i+j+k+l=x}\frac{C_{a-1}^{i-1}C_{b-1}^{j-1}C_{c-1}^{k-1}C_{d-1}^{l-1}}{i!\ j!\ k!\ l!}$$

维护 $C((a/b/c/d)-1,i-1)/i!$ 四个序列，做 NTT 求卷积即可。时间复杂度 $O(n\log n)$。

打怪兽 2

时间限制：8000/4000 毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

度度熊在玩一个叫做“打怪兽”的游戏。游戏的规则是这样的：度度熊一开始会有一个初始的能量值。每次遇到一个怪兽，若度度熊的能量值≥怪兽的能量值并且度度熊剩余血量≥怪兽的攻击力，那么怪兽将会被打败，度度熊的能量值增加 1，度度熊的血量减少该怪兽的攻击力，否则度度熊死亡(度度熊的血量刚好减到 0 时并不会死亡，还能继续战斗)，游戏结束。若怪兽全部打完，游戏也将会结束。

共有 n 个怪兽，由于度度熊比较弱，它一开始只有 1 点能量值。n 个怪兽排列随机，也就是说共有 $n!$种可能，度度熊想知道结束时它能量值的期望。注意这里怪兽的编号是从 1 开始到编号 n 为止且编号为 i 的怪兽能量值为 $i-1$。

由于小数点比较麻烦，所以你只需要输出期望 $*n!$ 关于 1000000007 取模后的值就可以了！

在样例中有 5 个怪兽，它们的能量分别为 0,1,2,3,4，其中每个怪兽的攻击力都为 1。

输入

多组数据。对于每一组数据：

第一行两个数 n，m 表示有 n 只怪兽，度度熊的初始血量($1\leqslant n\leqslant 500000$，$1\leqslant m\leqslant 10^9$)。

接下来一行 n 个数 ai 表示编号为 i 的怪兽的攻击力($0\leqslant ai\leqslant m$)。

输出

一行表示答案。

输入样例

```
5 4
1 1 1 1 1
```

输出样例

```
104
```

关键思路

引理：

假设一个序列 $c_i=0,1,2,\cdots,k-1$(角标从 1～k)，有 2^($k-1$)种排列方法使得 $c_i \leqslant i$

证明：

假设 0 在第 i 位，则 1～$i-1$ 位的数都大于 0，它们只能是 1,2,…,$i-1$，后面的第 $i+1$～k 位是 i～$k-1$，它们的排列数量是 $f(k-i)$，所以 $f(k)=f(k-1)+f(k-2)+\cdots f(k-k)$，其中 $f(0)=1$，$f(1)=1$。

数学归纳法证明略。

故 $f(k)=2^{(k-1)}$。

规定 $f(0)=1$。

能量值反映了我们可以打几只怪，我们可以统计当能量值达到 k 时，有多少种情况。

当能量值可以达到 $k(k \leqslant n)$，意味着至少杀了 $k-1$ 只怪，并且前 $k-1$ 只中，第 i 只的能量值不超过 i。这意味着 $k-1$ 只怪的能量值只能选取 0～$k-1$ 中的 $k-1$ 个不同的值，即从编号为 1～k 的怪中去除一只怪。

假设我们去掉了能量值为 q 的怪，并且保证 $a[1]+a[2]+\cdots a[k]-a[q+1] \leqslant m$ 时，那么顺序为第 $q+1$ 位到第 $k-1$ 位只能是 $q+1,q+2,\cdots,k-1$。第 1 到第 q 位是 0～$q-1$ 的排列，有 $f(q)$种方法。由于后面的 $n-k+1$ 只怪的能量值不限，故有$(n-k+1)!$ 种取法。经过计算，一共有 $f(q)*(n-k+1)!$ 种方法，其中 $0 \leqslant q<k$。

当能量值可以达到 $n+1$，意味着杀了 n 只怪，此时必须满足 $a[1]+a[2]+\cdots+a[n] \leqslant m$。并且这 n 只中，第 i 只的能量值不超过 i，故有 $f(n)$种可能。

题目所求即为：能达到 1 点能量值的数量＋能达到 2 点能量值的数量＋…＋能达到 n 点能量值的数量－能达到 1 点能量值的数量。

故总和即为满足 $0 \leqslant q<k$，且 $a[1]+a[2]+\cdots a[k]-a[q+1] \leqslant m$ 时的 $f(q)*(n-k+1)!$ 的总和，若 $a[1]+a[2]+\cdots+a[n] \leqslant m$，则再加上 $f(n)$。最后减去初始的一点能量值，即减去 $n!$

实现这个求和的朴素方法是 $O(n^2)$的，这个太慢，我们可以枚举 q 的值，然后确定满足 $a[1]+a[2]+\cdots a[k]-a[q+1]<=m$ 的 k 的范围。$k=q+1$ 是 k 的最小取值，由于 $ai \geqslant 0$，$a[1]+a[2]+\cdots a[k]-a[q+1]$随着 k 单增，可以二分出 k 的上界。

由于 $n-k+1$ 是一个连续值，可以根据 k 的范围，利用前缀和计算$(n-k+1)!$ 的和，再乘以 $f(q)$即可。之后再检查 $a[1]+a[2]+\cdots+a[n] \leqslant m$ 是否成立，若成立则再加 $f(n)$。最后再减去 $n!$ 即可。

度度熊的交易计划

时间限制：12000/6000 毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

度度熊参与了喵哈哈村的商业大会，但是这次商业大会遇到了一个难题：喵哈哈村以及周围的村庄可以看做是一共由 n 个片区，m 条公路组成的地区。

由于生产能力的区别，第 i 个片区能够花费 $a[i]$元生产 1 个商品，但是最多生产 $b[i]$个。同样的，由于每个片区的购买能力的区别，第 i 个片区也能够以 $c[i]$的价格出售最多 $d[i]$个物品。由于这些因素，度度熊觉得只有合理的调动物品，才能获得最大的利益。据测算，每一个商品运输 1 千米，将会花费 1 元。

那么喵哈哈村最多能够实现多少盈利呢？

输入

本题包含若干组测试数据。每组测试数据包含：

第一行两个整数 n，m 表示喵哈哈村有 n 个片区、m 条街道。

接下来 n 行，每行四个整数 $a[i]$，$b[i]$，$c[i]$，$d[i]$表示的第 i 个地区，能够以 $a[i]$的价格生产，最多生产 $b[i]$个，以 $c[i]$的价格出售，最多出售 $d[i]$个。

接下来 m 行，每行三个整数，$u[i]$，$v[i]$，$k[i]$，表示该条公路连接 $u[i]$，$v[i]$两个片区，距离为 $k[i]$

可能存在重边，也可能存在自环。

数据范围：

$1\leqslant n\leqslant 500$，$1\leqslant m\leqslant 1000$，$1\leqslant a[i]$，$b[i]$，$c[i]$，$d[i]$，$k[i]\leqslant 1000$，$1\leqslant u[i]$，$v[i]\leqslant n$

输出

输出最多能赚多少钱。

输入样例

```
2 1
5 5 6 1
3 5 7 7
1 2 1
```

输出样例

```
23
```

关键思路

首先，用 Floyd 求出任意两个片区之间的最短距离，因为运输肯定要走最短路径。我们要约束每个片区的生产和销售，可以使用网络流。我们把一个片区看成 2 个点，product，sale。从源点向 product 连一条容量为 $b[i]$，费用为 $a[i]$的边，从 sale 向汇点连一条容量为 $d[i]$，费用为 $-c[i]$的边。假如 x 片区到 y 片区的距离为 dis(x，y)，且 dis(x，y)$+a[x]<c[y]$，即生产单位货物与运费单价的总和小于售卖单价时，从 x 片区运到 y 片区是盈利的。那么从 x 片区的 product 到 y 片区的 sale 连一条容量为 inf，费用为 dis(x，y)的边。最后求解最小费用最大流，得到的结果就是总的支出的最小值，它的相反数即为所求。

小小粉丝度度熊

时间限制：2000/1000 毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

度度熊喜欢着喵哈哈村的大明星——星星小姐。为什么度度熊会喜欢星星小姐呢？首先星星小姐笑起来非常动人，其次星星小姐唱歌也非常好听。但这都不是最重要的，最重要的是，星星小姐写的一

手好代码！于是度度熊关注了星星小姐的贴吧。

一开始度度熊决定每天都在星星小姐的贴吧里面签到。但是度度熊是一个非常健忘的孩子，总有那么几天，度度熊忘记签到，于是就断掉了他的连续签到。不过度度熊并不是非常悲伤，因为他有 m 张补签卡，每一张补签卡可以使得某一忘签到的天，变成签到的状态。

那么问题来了，在使用最多 m 张补签卡的情况下，度度熊最多连续签到多少天呢？

输入

本题包含若干组测试数据。

第一行两个整数 n，m，表示有 n 个区间，这 n 个区间内的天数，度度熊都签到了；m 表示 m 张补签卡。

接下来 n 行，每行两个整数($l[i]$，$r[i]$)，表示度度熊从第 $l[i]$天到第 $r[i]$天，都进行了签到操作。

数据范围：

$1 \leqslant n \leqslant 100000$，$0 \leqslant m \leqslant 1000000000$，$0 \leqslant l[i] \leqslant r[i] \leqslant 1000000000$

注意，区间可能存在交叉的情况。

输出

输出度度熊最多连续签到多少天。

输入样例

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

输出样例

```
3
3
```

提示

样例一：度度熊补签第 2 天，然后第 1 天、第二天和第三天都进行了签到操作。

样例二：度度熊补签第 2 天和第 3 天。

关键思路

枚举每个区间的开头，然后二分这个区间最长延伸到哪个区间即可。用前缀和优化，复杂度 $O(n\log n)$。

5.4 复赛

Arithmetic of Bomb

时间限制：2000/1000 毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

图 5-1 Arithmetic of Bomb

题目

众所周知，度度熊非常喜欢数字。它最近在学习小学算术，第一次发现这个世界上居然存在两位数，三位数……甚至 N 位数！但是这回的算术题可并不简单，由于含有表示 Bomb 的 # 号，度度熊称之为 Arithmetic of Bomb(图 5-1)。

Bomb Number 中的 Bomb，也就是 # 号，会展开一些数字，这会导致最终展开的数字超出了度度熊所能理解的范畴。比如(1)#(3)表示 1 出现了 3 次，将会被展开为 111，同理，(12)#(2)4(2)#(3)将会被展开为 12124222。

为了方便理解，下面给出了 Bomb Number 的 BNF 表示。

```
<bomb number> ::= <bomb term> | <bomb number> <bomb term>
<bomb term> ::= <number> | '(' <number> ')' '#' '(' <non-zero-digit> ')'
<number> ::= <digit> | <digit> <number>
<digit> ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
<non-zero-digit> ::= '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
```

请将 Bomb Number 中所有的 # 号展开，由于数字可能很长，结果对 1000000007 取模。

输入

第一行为 T，表示输入数据组数。

每组数据包含一个 Bomb Expression。

$-1 \leqslant T \leqslant 100$，$-1 \leqslant$ length(Bomb Number) $\leqslant 1000$

输出

对每组数据输出表达式的结果，结果对 1000000007 取模。

输入样例

```
4
1
(1)#(3)
(12)#(2)4(2)#(3)
(12)#(5)
```

输出样例

```
1
111
12124222
212121205
```

关键思路

表达式的最大长度只有 1000，# 后面只会是一位的数字，也就是最大展开 9 次，因此直接展开的最大长度不会超过 9000，所以按照规则直接处理即可。

难度：2/5

Arithmetic of Bomb II

时间限制：30000/15000 毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

众所周知，度度熊非常喜欢数字。它最近在学习小学算术，沉迷于计算 $A+B$ 中不能自拔。但是这回的算术题可并不简单，由于含有表示 Bomb 的＃号，度度熊称之为 Arithmetic of Bomb(图 5-2)。

Arithmetic of Bomb 的目的与普通算术一样，就是计算一些 Bomb Expression 的结果。比如，1－2＋3 的结果为 2。然而，Bomb，也就是＃号，会展开一些普通表达式，这会导致需要计算的式子超出了度度熊所能理解的范畴。比如(1－2＋3)＃(3)表示 1－2＋3 出现了 3 次，将会被展开为 1－2＋31－2＋31－2＋3。

图 5-2 Arithmetic of Bomb

为了方便理解，下面给出了 Bomb Expression 的 BNF 表示。

```
<bomb expression> := <bomb term> | <bomb expression> <bomb term>
<bomb term> := <bomb statement> | '('<bomb statement> ')' '#' '('<number> ')'
<bomb statement> := <bomb element> | <bomb statement> <bomb element>
<bomb element> := <digit> | '+' | '-' | '*'
<normal expression> := <norm term> | <normal expression> '+' <norm term> | <normal expression> '-'
'<norm term>
<norm term> := <number> | <norm term> '*' <number>
<number> := <digit> | <non-zero-digit> <number>
<digit> := '0' | <non-zero-digit>
<non-zero-digit> := '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
```

请先将 Bomb Expression 中所有的＃号展开，使其成为 Normal Expression(题目的输入保证展开后是一个合法的 Normal Expression)，再来计算这个表达式的结果。

输入

第一行为 T，表示输入数据组数。

每组数据包含一个 Bomb Expression。

1≤T≤50，1≤length(Bomb Statement)≤10，1≤length(Number in Bomb term)≤10，

1≤length(Bomb Expression)≤300000

输出

对每组数据输出表达式的结果，结果对 1000000007 取模。

输入样例

```
6
1-2+3
(1-2+3)#(3)
(1)#(3)
(1+)#(2)1
(2*3+1)#(2)
(2)#(2)1+1(2)#(2)
```

输出样例

```
2
60
111
3
43
343
```

关键思路

（1）1≤length(Number in Bomb term)≤10，所以无法直接展开。

（2）包含+，-，* 等复杂运算。

一个可行的思路是把所有的操作都变成矩阵运算，即用当前结果与一个矩阵相乘，表示一种操作。操作包含：

（1）在当前表达式末尾添加一个数字。

（2）在当前表达式末尾添加一个运算符：+，-，*

那么一次 Bomb 操作就相当于对一个矩阵求幂。推导过程需要一些矩阵知识，以及比较复杂的演算，可以参考标程。

难度：5/5

Pokémon GO

时间限制：3000/1500 毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

众所周知，度度熊最近沉迷于 Pokémon GO(图 5-3)。今天它决定要抓住所有的精灵球！

为了不让度度熊失望，精灵球已经被事先放置在一个 2 * N 的格子上，每一个格子上都有一个精灵球。度度熊可以选择任意一个格子开始游戏，抓捕格子上的精灵球，然后移动到一个相邻的至少有一个公共点的格子上继续抓捕。例如，(2,2)的相邻格子有(1,1)，(2,1)和(1,2)等。

现在度度熊希望知道将所有精灵球都抓到并且步数最少的方案数目。两个方案被认为是不同，当且仅当两个方案至少有一步所在的格子是不同的。

图 5-3 Pokémon GO

输入

第一行为 T，表示输入数据组数。

每组数据包含一个数 N。$1 \leqslant T \leqslant 100$，$1 \leqslant N \leqslant 10000$

输出

对每组数据输出方案数目，结果对 1000000007 取模。

输入样例

```
3
1
2
3
```

输出样例

```
2
24
96
```

关键思路

一个可行的思路是考虑三个子问题：

(1) 全部走完 $2*N$ 个格子的方法总数 DP[N]。

(2) 全部走完 $2*N$ 个格子并且起点是最左边的两个格子之一的方法总数 DP2[N]。

(3) 全部走完 $2*N$ 个格子并且起点和终点分别是最左边的两个格子的方法总数 DP3[N]。

组合 2 和 3 的答案就可以得到 1 了。

难度：2.5/5

Pokémon GO II

时间限制：3000/1500 毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)。

题目

众所周知，度度熊最近沉迷于 Pokémon GO(图 5-4)。

由于太过沉迷，现在它只能按照游戏内置的指令行走了：对，简直就像一个现实中的 Pokémon!

游戏内置的指令实际上可以抽象成一种：保持现在的朝向前行 x 米，然后右转。度度熊相信，只要遵循这个指令，它就一定可以抓到最珍奇的精灵球。

但不幸的是，这个指令并不是很有可信度，有时会引导度度熊走回原来的位置。现在它想知道，在第几条指令时它第一次回到已经走过的位置？如果这种情况没有发生，请输出 Catch you。

图 5-4 Pokémon GO

输入

第一行为 T，表示输入数据组数。

每组数据的第一行包含一个数 N，表示指令长度。接着的一行包含 N 个数字 Xi，表示第 i 个指令中前行的距离。

$1 \leqslant T \leqslant 100$， $1 \leqslant N \leqslant 1000000$， $1 \leqslant Xi \leqslant 1000000000$

输出

对每组数据输出第一次回到已经走过的位置时的指令下标 $i(1 \leqslant i \leqslant N)$。

如果这种情况没有发生，请输出 Catch you。

输入样例

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

```
2 1 3 1
5
2 1 3 1 3
```

输出样例

```
4
Catch you
5
```

关键思路

轨迹要么一直扩大，要么一直缩小，记录四个维度的极值，模拟 Pokémon 的轨迹并判断是否有覆盖情况出现。

一个更简单的做法需要发现一个几何性质：第一次覆盖一定是与轨迹上的前 8 段（或者更少）中的某一段。

难度：3/5

Valley Numer

时间限制：2000/1000 毫秒（Java/其他）

空间限制：32768/32768KB（Java/其他）

题目

众所周知，度度熊非常喜欢数字。它最近发明了一种新的数字：Valley Number（图 5-5），像山谷一样的数字。

图 5-5 Valley Number

当一个数字，从左到右依次看过去数字没有出现先递增接着递减的“山峰”现象，就被称作 Valley Number。它可以递增，也可以递减，还可以先递减再递增。在递增或递减的过程中可以出现相等的情况。

比如，1，10，12，212，32122 都是 Valley Number。121，12331，21212 则不是。

度度熊想知道不大于 N 的 Valley Number 数有多少。注意，前导 0 是不合法的。

输入

第一行为 T，表示输入数据组数。

每组数据包含一个数 N。$1 \leqslant T \leqslant 200$，$1 \leqslant \text{length}(N) \leqslant 100$。

输出

对每组数据输出不大于 N 的 Valley Number 个数，结果对 1000000007 取模。

输入样例

```
3
3
14
120
```

输出样例

```
3
14
119
```

关键思路

非常经典的数位 DP，可以将状态设计成四维。

当前数字长度 len，最后一位数字 digit，是否已经在递增序列里 increase，是否和当前前缀相同 same_prefix。转移时处理好这些状态就好了。

难度：2/5

Valley Numer II

时间限制：12000/6000 毫秒(Java/其他)

空间限制：32768/32768KB(Java/其他)

题目

众所周知，度度熊非常喜欢图。它最近发现了图中也是可以出现 valley(山谷)的，如图 5-6 所示。

为了形成山谷，首先要将一个图的顶点标记为高点或者低点。标记完成后如果一个顶点三元组< X,Y,Z >中，X 和 Y 之间有边，Y 与 Z 之间也有边，同时 X 和 Z 是高点，Y 是低点，那么它们就构成一个 valley。

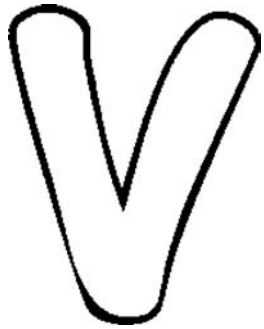

图 5-6 valley

度度熊想知道一个无向图中最多可以构成多少个 valley，一个顶点最多只能出现在一个 valley 中。

输入

第一行为 T，表示输入数据组数。

每组数据的第一行包含三个整数 N，M，K，分别表示顶点个数，边的个数，标记为高点的顶点个数。

接着的 M 行，每行包含两个两个整数 Xi，Yi，表示一条无向边。

最后一行包含 K 个整数 Vi，表示这些点被标记为高点，其他点则都为低点。

$1 \leqslant T \leqslant 20$，$1 \leqslant N \leqslant 30$，$1 \leqslant M \leqslant N*(N-1)/2$，$0 \leqslant K \leqslant \min(N,15)$，

$1 \leqslant Xi, Yi \leqslant N$，$Xi \neq Yi$，$1 \leqslant Vi \leqslant N$。

输出

对每组数据输出最多能构成的 valley Number。

输入样例

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

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

输出样例

```
1
0
2
```

关键思路

题目看上去是一道图论，然而问题并不像是二分匹配这种经典问题一样有一个多项式解法，只能借助题目数据的特性使用集合 DP 解决。由于高的顶点个数只有最多 15 个，记录这些点的匹配状态进行 DP。并且数据规模较大，可能需要优化一些写法。

难度：2.5/5

5.5 决赛

贪吃的度度蛇

题目

参赛选手需要实现“贪食蛇”游戏的 AI。在“贪食蛇”游戏中，选手编写的 AI 控制场上的蛇，在动态变化的场地中与对手比拼。在比赛中，AI 需要相互竞争，并尽可能多地积累分数。

赛程赛制

赛制为多次循环赛方式。比赛分为三阶段，第一阶段时间为 1.5 小时，第二阶段时间为 4 小时，第三阶段为 0.5 小时。比赛结束时，总得分最高的选手获胜。

第一阶段的比赛，即比赛的前 1.5 小时为编码及测试时间。期间选手可以提交代码至比赛平台。

第二阶段的比赛为分组循环赛。在比赛开始 1.5 小时后，进行第一次循环赛，之后每一小时进行一次，总共进行 4 次循环赛。

每次循环赛分 3 轮进行。该次循环赛开始时，暂停代码提交功能，选手最近一次成功编译的代码将被用于比赛。

在每一轮中，场上 50 名选手被随机分配到 5 组中，每组 10 人（如总人数小于 50 人，则尽量保证各组人数相等），并分别进行一局多人竞技赛。多人竞技赛的游戏规则将在后面详细描述。

选手在每轮比赛中的得分累加至该选手的总得分中。

每次循环赛的间隔时间选手可继续优化程序并将上传代码至比赛平台。在第二阶段结束时，平台将关闭代码上传功能。

第三阶段的多人竞技赛规则与第二阶段相同，分为 6 轮连续进行，其间选手无法调整代码。

选手总得分＝第二阶段得分＋第三阶段得分，即各轮比赛得分总和。

每轮比赛结束后，选手可通过录像回放即时观看每场比赛单组表现，也可以通过大屏幕观看所有选手的表现。

游戏规则

游戏借用一部分“贪食蛇”的设定，在一张二维地图上展开。每位选手的 AI 程序控制一条蛇，通过收集道具或“击杀”对手的方式积累分数。游戏引擎方面，采用实时计算的方式，每秒计算 30 帧。

详细规则如下。

数据结构

地图采用笛卡尔坐标系描述，长、宽均为 100。其上所有物体可用一个圆来表示。物体分为蛇、道具、障碍三类。每个物体采用该数据结构表示：$\{x_i, y_i, r_i, t_i, p_i\}$。其中 x, y, r 分别为坐标及半径，t 为物体的类型，p 为物体参数。x, y, r 均为正实数。

物体类型 t 的含义用枚举类型定义（见表 5-1）：

表 5-1 物体类型及含义

t	类 型	参数含义及取值范围
EntityTypeBody	蛇的身体（包括头）	当前比赛中的选手序号，（从 0 开始至[当前比赛选手数量－1]）
EntityTypeWall	障碍	无意义
EntityTypeFood	食物	奖励的食物量（1－3）
EntityTypeBoost	加速能量	奖励的能量（1－3）

蛇

每条蛇的身体由一组圆构成。如用 L 表示第 n 条蛇的长度，则从蛇头开始，构成蛇身的第 l 个圆可以表示为：$\{x_l^n, y_l^n, r_l^n\}$。其中相邻两个蛇身的圆心间的距离为常数，用 D 表示。

比赛中 D 为常数 0.5。

蛇的移动

每条蛇在游戏中均处于不停移动的状态。蛇的移动轨迹由蛇头确定，蛇身跟随蛇头，沿蛇头移动过的路径移动。每一节蛇身的圆心位置均处在蛇头的历史轨迹点构成的折线上。

服务器主程序控制蛇头移动的速率，选手的程序控制蛇头移动的角度以及是否使用加速道具。蛇头转动的角速度由服务器进行限制，每两帧间的角度之差无法超过 0.05π。即，当选手返回角度信息后，蛇头会朝向该角度，取旋转角度小的方向进行旋转。

具体解释如下

设蛇头移动过的轨迹 P 由一系列点$\{x0, y0\}, \{x1, y1\}, \cdots, \{xt, yt\}$及连接这些点的折线构成。在时刻 t，$l=0$ 的蛇头位置为 $p(t)=\{xt, yt\}$。此时，第一段蛇身（$l=1$）的位置为蛇头轨迹 P 上与点 $p(t)$ 距离为 D 的点。

例如：

```
p(t) = {0.3, 0.3}
p(t-1) = {0, 0.3}
p(t-2) = {0, 0}
D = 0.5
```

则 t 时刻，蛇头后面第一节蛇身（$l=1$）位置为{0，0.1}，如图 5-7 所示。

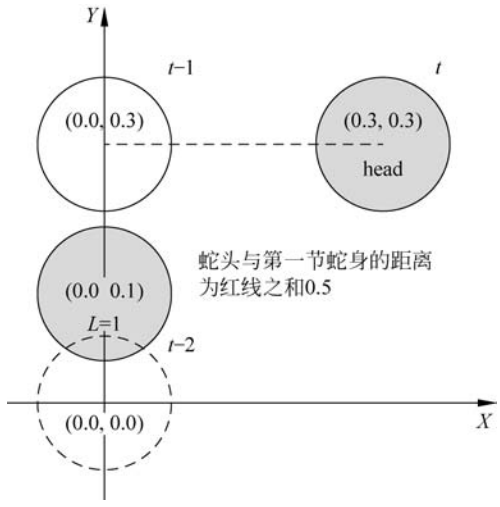

图 5-7　蛇头与蛇身距离

在时刻 t，蛇头位置 $p(t)=p(t-1)+v(t-1)$，其中 $v(t)$ 为时刻 t 时蛇头的速度向量。

服务器会将当前时刻 t 的非加速状态移动速率发送给选手。

加速

选手可以消耗能量获得短暂的加速能力，此时移动速率为正常速率的 1.5 倍。加速能量使用整型变量表示，选手可通过能量道具获取。每 1 点加速能量可提供 1 帧的加速时长，每条蛇最多拥有 30 点加速能量。

如选手在时刻 t 选择激活加速，需要返回相应参数给服务器。加速效果将在时刻 t 生效，持续直到能量耗尽为止。如当前时刻加速能量为 0，则无法激活加速。

初始加速能量为 0。

蛇的长度和直径

蛇的长度 L 和宽度 r（即构成蛇身的圆的半径）随着获取的食物量而增长，而移动速率 s 随宽度增长而下降。

以 f 表示某条蛇已经获取的食物数量（初始值 $f=10$），则具体公式为：

蛇身长度 $L=\text{int}(f/3)$，为蛇的身体段数。

蛇身直径 $d=\text{sigmoid}(f/250+1)$，$\text{sigmoid}(x)=\dfrac{1}{1+e^{-x}}$，全部蛇身（包括头部）的直径相同。

非加速状态移动速率 $s=0.1/r$。

当蛇的长度变化时，会在尾部加长一段身体，位置由前文所述规则确定。

碰撞检测与击杀

蛇与场上道具、其他蛇等物体的交互通过“碰撞”进行。当蛇头部“碰撞”到其他物体时，触发服务器的碰撞检测，并执行相应动作。

“碰撞”的定义为：当两物体之间的直线距离小于其半径之和时，判定为碰撞。

碰撞检测由服务器进行，玩家不需要进行计算。

当两条蛇的蛇头相互碰撞时，双方都被判为“击败”，击杀数不变。

当蛇头与障碍或其他蛇的蛇身碰撞时，该蛇被判定为“击败”，另一方的击杀数加 1。

注意：蛇 A 头部同时碰撞到蛇 B 的头部与身体，则蛇 A 与蛇 B 均判定为击败，且击杀数不变。

当蛇头与食物或能量道具碰撞时，该蛇获得该道具的效果，且该道具消失。

除上述规则外，蛇头以外的蛇身碰撞到物体时，不会触发碰撞判定。道具与道具之间不会触发碰撞判定。

蛇的死亡

蛇死亡时，其身体有一定概率转化为食物，且所有身体均从场上消失。

信息交互

每一帧中，选手的程序被服务器主程序调用，通过参数传递的方式获取以蛇头为中心一定范围内的物体数据。该范围称为视野，为一长、宽分别为 $Sx=Sy=20$ 的正方形。即：设当前帧中某物体中心坐标为(x,y)，蛇头中心坐标为$(x0,y0)$，则 $\text{abs}(x-x0)\leqslant 10 \&\& \text{abs}(y-y0)\leqslant 10$ 的物体信息将被传递给选手。其余的物体对选手不可见。

此外，选手也将获得当前帧中己方蛇的移动速率、长度、食物量、能量等信息。

选手的程序返回下一帧的速度方向矢量，以及是否使用加速道具。

选手程序每帧当中的执行时间不得超过 25ms，否则将被判当前帧返回结果无效。

计分

每局游戏开始时，每位选手初始分数为 0。

随着游戏进行，选手控制的蛇会积累食物和击杀数。

游戏结束时，该局游戏中每位选手的得分计算公式如下：

$$\text{score}=f+k*1000$$

其中 f，k 分别为食物量与击杀数。

比赛结束

每局比赛中，当所有选手的蛇均被判定击败，或比赛时间超过 9000 帧时，本局比赛结束。

注意事项

大赛鼓励选手独立解题，禁止进行任何形式的作弊、破解。我们会抽查选手的代码，一旦发现上述行为将取消比赛资格。

代码实现

选手自行在本机上实现代码，并通过网页方式提交 player_snake.cpp 至比赛平台。两次提交代码的时间间隔须大于 1 分钟。

选手可以修改示例代码中 player_snake.cpp 的 onFrame()函数来实现服务器对选手的 AI 逻辑的调用。该函数签名为

```
void onFrame(const PlayerDataOnFrameIn& data_in, PlayerDataOnFrameOut& data_out)
```

其中 PlayerDataOnFrameIn 为该帧的输入，PlayerDataOnFrameOut 为输出。二者的参数说明请参考示例代码。请不要修改除 player_snake.cpp 外的其他文件，否则会造成本地编译环境与服务器不一致，导致服务器端编译失败。

代码编译环境为 Centos 6u3、GCC 4.8.2。允许包含的头文件见附录 A。

禁止任何形式的破解、攻击、毁坏服务器及其他不正当竞争行为，包括在提交的代码中植入恶意代码等。违者将取消参赛资格。我们将人工检查选手们的代码。请选手自行实现自己的代码，以保证比

赛的公平性。

附录 A　允许引用的头文件列表

```
assert
ctype
string
utime
malloc
algorithm
array
cmath
cstring
ctime
deque
exception
list
map
memory
queue
random
set
string
stack
vector
tuple
iostream
stdlib
math
float
```