A – 货物运输

题面：https://vjudge.net/contest/185024#problem/A

一开始以为是运输计划的链上版本，但是这个题有其特殊性，是任意两个城市之间修一条通道，那么可能导致路径根本不经过原来的路径，那么差分什么的自然是错误的，但是二分答案还是必要的；

我们二分答案之后，原本不能按时到达的计划的必然会经过新修的这一条道路才有可能变得合法，我们考虑假如在 x,y 之间修路，原本不合法的一个计划 l->r,的耗时(必须经过 x->y 这条新修路)

abs(l-x)+abs(r-x)<=mid;

如果我们把 l 看成横坐标，r 看成纵坐标，那么上面的式子为(l,r)与(x,y)两点的曼哈顿距离，然后与某点的曼哈顿距离<=mid 在二维平面上的图形是一个边长为 mid 的正方形去掉 4 个角。。。

那么对于一个点的可行域是这四条直线夹着的区域，然后我们要判断所有可行域是否有交，那么我们只需要分别算出 x+y(对应 y=-x+b 这条直线),x-y(对应 y=x+b 这条直线)的可行域，判断两个都有解则判断所有的可行域有交。

B - Little Pony and Alohomora Part III

题面：https://vjudge.net/contest/185024#problem/B

这个题题目中说明了一个箱子有且只有一把钥匙能够打开，然后那么每个点都有一个父亲，那么就是一个一个环形关系，然后我们发现一个环中只要有一个被打开了，那么环中的其余点都会被打开，那么我们定义 dp[i][j]表示到了第 i 个环，用了 j 次魔法的方案书，最后在除以 C(n,k)的总方案来得到概率，转移的话就是背包的转移，但是需要在转移时保证每个环至少要用一次魔法，300 的组合数不取模，竟然开 double 能过。。。

D – 部落信号

题面： https://vjudge.net/contest/185024#problem/D

一道单调栈的细节题，首先把最高的山峰隔离出来，把其余的点当做链来处理，因为不可能有两个点只能通过跨过最高的山峰才互相可见，然后我们最后再按照题目给出的能互相看见的定义，正反两边算出最高的山峰和那些点能互相看见；

如果是链的话，是一个很套路的单调栈的应用了，但是因为有重复元素的存在，还是比较细节的，首先我们通过单调栈求出第 i 个山峰，左边第一个大于他的山峰，右边第一个大于他的山峰，那么这他和左右两边的山峰就是互相可见的，因为在其路径上不可能存在大于两者的最小值的点，但是如果有重复元素的话，我们还需要特殊考虑，首先先按找正常的处理方法一边取=，一边不取=，然后我们对于不取等号的那边需要开一个数组记录，每个元素在单调栈中和他连着的并且和他高度相同的点的数目，显然每个元素和这些点的路径上的值不可能超过他们的高度，所以这些点是互相可见的，需要加入答案之中，貌似也没有别的细节了；

E – 交换机器的最小代价

题面：https://vjudge.net/contest/185024#problem/E

这题能一遍 A 的都是大佬，首先我们肯定要 sort 一遍求出每个 rnk 当前的位置 pos,然后 pos 就相当与是转移过程，我么发现转移是一个个环状的，只要环中的一个节点走到了原本的位置，那么所有的点都走到了应该的位置，所以我们肯定是要让最小的点一路交换过去即可，但是还有一种很骚的操作，就是先把全局中 rnk1 的元素与该环中 rnk1 的元素互换，然后把全局的 rnk1 一路交换，最后再把全局的 rnk1 换回去；

F – 稳定方块

题面：https://vjudge.net/contest/185024#problem/F

STL 大模拟题，因为是从高位往低位选，那么对于奇数轮肯定是选择能选择中的最大的，偶数轮肯定是选能选择中的最小的，我们可以用 set 来动态查询最大值和最小值，我们发现依赖关系是 DAG,能选择的点的满足的条件是：所有依赖于他的点要么已经被打掉了，要么就是入度>1,那么我们在删除的时候维护一下每一个点的入度，然后把满足条件的加入 set 中即可(已经用过的不要加了)，但是可能删掉一些方块后导致在 set 的某些点变得不合法，我们在查询的时候 while 弹掉即可，然后也没有太多的细节了；

H - Self-Driving III

题面：https://vjudge.net/contest/185024#problem/H

一道巨难实现的树型 Dp,首先一看到题目就是要设很多状态的；

dp[x][0],只有人走，人一定要回来；
dp[x][1],人车都走，人车都要回来；
dp[x][2],只有人走，人不一定回来；
dp[x][3],人车都走，人一定要回来，车不一定；
dp[x][4],人车都走，人车都不一定要回来；

对于那些不会来的操作，肯定只能选择一个子树然后不回来，其余的子树都是必须回来的；
直接说转移，因为不是很难想：

dp[x][0]+=dp[y][0]+2*v[i];
dp[x][1]+=min(dp[y][0]+2*v[i],dp[y][1]+2*w[i]);

然后对于 dp[x][2],每个子树有两种选择：

1.dp[y][2]+v[i];
2.dp[y][0]+2*v[i]；

第一种不会来的选择只能用一次，我们肯定是要使得用着一次最赚，即在两种决策差值最大的那一次用第一种决策；
这个实现的话可以假定都按照决策 2 进行决策来累加答案，然后用一个变量记录差值的最大，然后最后减去这个最大的差值即可；

对于 dp[x][3],同上，每个子树也是有两种选择的：

1.dp[y][3]+w[i]+v[i];
2.min(dp[y][0]+2*v[i]+dp[y][1]+2*w[i]);

第一种不把车开回来的只能用一次，那么具体做法和上一种类似，记录最大差值即可；

对于 dp[x][4],有两种情况：
1.最后一次有车，那么每棵子树有两种决策：
《1》min(dp[y][2]+v[i],dp[y][4]+w[i])
《2》min(dp[y][0]+2*v[i],dp[y][1]+2*w[i]);
同样第一种决策只能用一次，然后处理方法和其他一样

2.最后一次没车，那么我们假设倒数第二次人把车开走了，车没回来，人回来了，每个子树有三种决策：
《1》dp[y][2]+v[i],最后一次
《2》dp[y][3]+w[i]+v[i]，倒数第二次;
《3》min(dp[y][0]+2*v[i],dp[y][1]+2*w[i]);

处理方法就是我们还是假定按决策三操作，然后分别记录《3》与《1》的最大差值，《2》与《1》的最大差值，然后因为要保证《1》与《2》不能在同一棵子树中进行，那么我们还要记录次大的差值，以及每个差值代表的子树是什么，然后选取两个不在同一子树中的差值和最大的决策(4 选 1)即可；

对于 dp[x][4]，把这两种情况取 min 即可；

I – 布丁怪

https://vjudge.net/contest/185024#problem/I

一道非常具有难度和技巧的分治计数题，我们首先把问题简化为，给定一个序列，问有多少个连续的子序列的值域是连续的；

我们考虑用分治来解决这个问题，定义 solve(l,r),对于[l,r]中的子序列有一下几种情况：

1.完全在[l,mid]间；

2.完全在[mid+1,r]间；

3.会越过 mid;

对于第 1,2 种情况我们可以递归求解，我们重点要考虑越过 mid 的即可；

我们考虑如果要计算子序列的长度的话，我们需要知道值域的最大值和最小值，相减即为长度；

关于最大值和最小值只有以下的 4 种情况：

1.Max 在[l,mid],Min 在[l,mid];

2.Max 在[mid+1,r],Min 在[mid+1,r];

3.Max 在[l,mid],Min 在[mid+1,r];

4.Max 在[mid+1,r],Min 在[l,mid];

对于 1,2 种情况，处理方法是一样的，这里只说第一种。

对于[l,mid]的每个点是可以计算出以他为起点的子序列长度的，即 Maxi-Mini,(定义 Maxi 表示由 mid 到 i 的最大值，Mini 同理),然后我们得到了这个子序列在[mid+1,r]的终点 y，然后我们需要判断值域是否连续，就只要判断[Miny,Maxy]是否是[MiniMaxi]的子区间即可(因为元素没有重复)

对于 3,4 种情况，处理方法也是一样的，我们只考虑第三种；

对于起点 x 和终点 y，我们由已知：Maxx>Maxy,Minx>Miny,并且 Maxx-Miny=y-x,

第三个式子移项之后变为 Maxx+x=Miny+y,这都是只跟 x,y 有关的量，那么我们对于每一个起点 x 相当于是要统计用[mid+1,r]中满足 1,2 个条件的区间中 Miny+y=Maxx+x 的数量，我们容易知道从 mid->l 移动 x 的过程中，Maxx 不会变小，Minx 不会变大，并且从 mid+1->r,也是同理，那么我们发现在[mid+1,r]中的两个指针是单调往后移的，那么我们要统计区间中等于某个值的数量，开一个桶就行了，然后维护方法和莫队类似。