A

字符串比较大小。数字小于字母，数字按大小排，字母按ASCII排。连续多个数字的话把它们的值转化为十进制数保存在一起，字母一个一个存。注意相等时输出+。

B

由于要找的等差数列一定是单调的，因此可以先将所给数排序，那么找出的数在约数列中的相对位置与目标数列一致。

$D[i][k]$表示以i结尾的第k种等差数列的公差，$f[i][k]$表示以i结尾的第k种等差数列的长度。对于i之前的每一个数j，去找到k使$D[j][k] = v[i] - v[j]$，若能找到,相应的$f[i][k] = f[j][k] + 1$，否则，$f[i][k] = 2$.

由于v是排好序的，因此$D[i]$也是有序的，则可以使用二分来找到k，所以，需要枚举i，j，再二分查找得到k，时间复杂度为$O(n^2logn)$。

C

计算出每个乘客出去的本来步数，如果该步数已经有乘客，就考虑下一步。注意远离过道的乘客应该比过道处的后离开。
共有$2 * r * s$个乘客，最多$2 * r * s$步就能全部撤离，这样复杂度为$O(4 * r * r * s * s)$，但并不是每一个乘客都需要遍历所有的步数,所以实际远小于上述复杂度。

D

$f[i][j]$表示a串前i个，b串前j个的最短不公共序列，保证$a[i+1] == b[j+1]$,从$i+1$后找到a串第一个0的位置x，从$j+1$后找到b串第一个0的位置y，那么，$f[x-1][y-1] = f[i][j] +' 0'$，同理，1也一样。这样广搜转移状态，第一次给$f[x][y]$赋值的一定是字典序最小且最短的不公共序列。
每种状态最多进队出队一次，时间复杂度$O(len(a) * len(b))$。注意MLE，要在队列里记录每次的i, j, str，不能直接开 `strirng f[][]` .

E

首先很显然，要构成欧拉回路，必须所有点的度数都是偶数。显然点数为奇数，直接构造完全图就是答案；点数为偶数就比较复杂了，需要让补图的每一个点的度数都变成奇数。之所以考虑补图，是最开始的想法：补图能够形成$n/2$对匹配，这样必然形成一组合法解，但反例证明这只是充分不必要条件。真正的充要条件是：补图的每个连通块删去若干条边后，每个点的度数都是奇数，且没有满度数点（即原图的空点）。如果原图有空点，可以通过以下特判解决：除这个空点p外的子图如果不是完全图，即$g[i][j]=0$，那么将除点p和边(i,j)外的子图补成完全图，再连接(p,i)、(p,j)，即可得到欧拉图；而没有空点的情况，实际上等价于补图的每个连通块的节点数都是偶数。

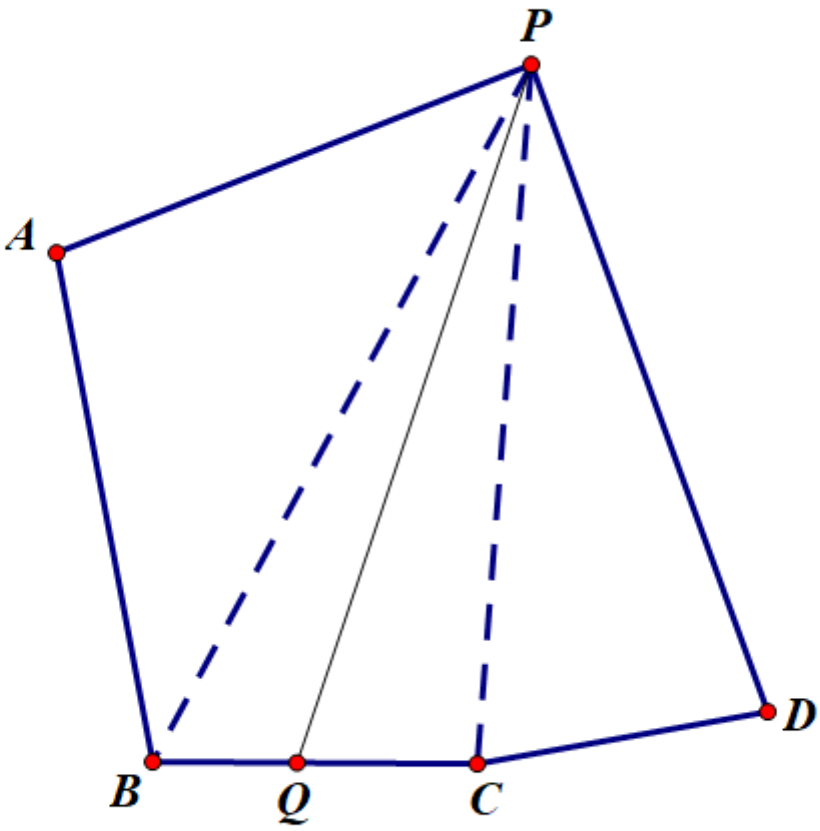
必要性证明：如果某个连通块包含奇数个点，假如删去若干条边后每个点的度数为奇数，那么这个连通块所有点的度数和为奇数，这和图的性质矛盾，因此假设不成立，必要性得证。

充分性证明：对这个连通块构造任意生成树（即先删去不属于生成树的所有边），然后跑*dfs*，对于从下到上顺序（即回溯顺序）的每一个点，如果它当前度数为偶数，那么删去它和它父亲相连的那条边，使它的度数变为奇数。这样可以保证除根节点外所有点的度数都是奇数；而由于除根节点外其余节点数量为奇数，因此除根节点外，所有点度数和为奇数。又由于所有点的度数和始终为偶数，因此根节点的度数也为奇数。显然在证明充分性的时候，构造答案的方法也出来了。注意输出答案有格式的限制。

F

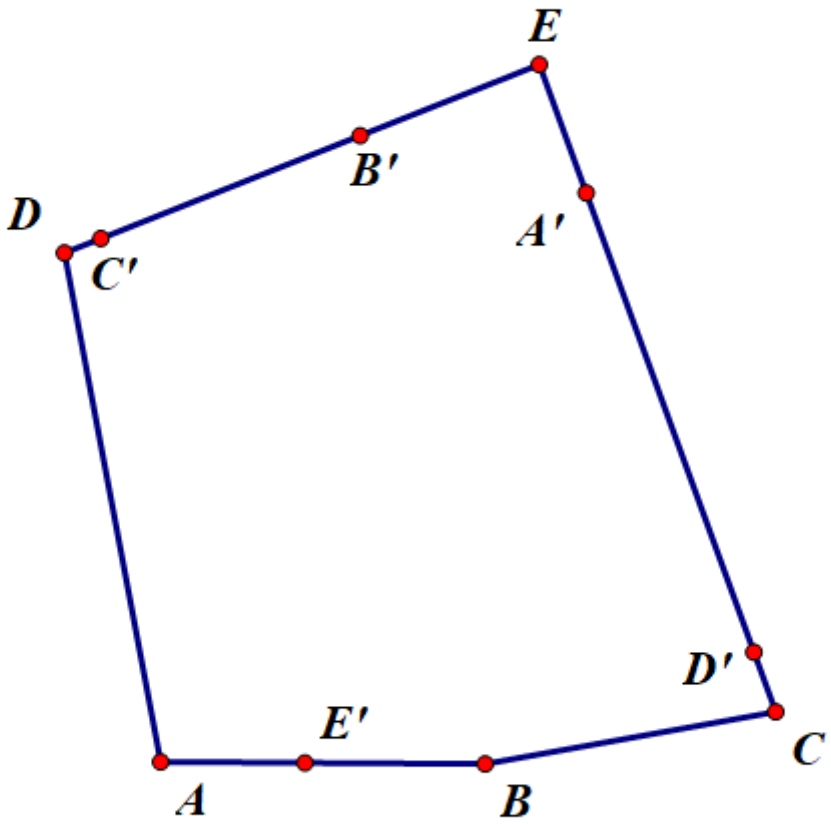
题目要求平分凸多边形面积的最长线段和最短线段，因此考虑枚举所有端点可能所在的线段。如果P、Q在多边形的边（包括顶点）上，且线段PQ平分多边形面积，则称P、Q互为对应点。

枚举凸多边形的顶点P，通过三角剖分求凸多边形面积的方法（判断当前面积是否到达总面积的一半）找到P的对应点所在线段。使用面积法，求出P的对应点的坐标。枚举所有顶点后，得到n个对应点的坐标。

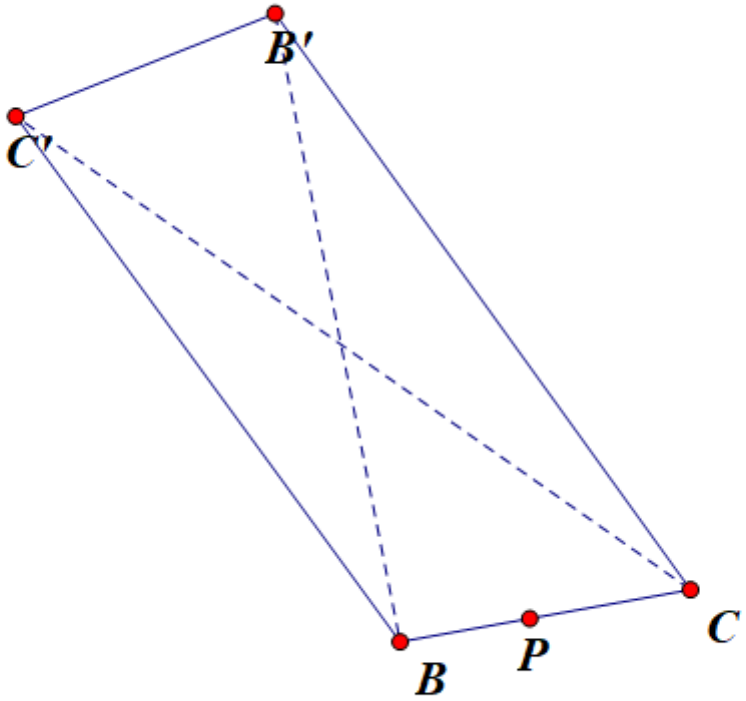


（如图，$S_{\triangle PAB} < \frac{\sum S}{2}$，$S_{\triangle PAB} + S_{\triangle PBC} > \frac{\sum S}{2}$，故$S_{\triangle PBQ} = \frac{\sum S}{2} - S_{\triangle PAB}$，从而求出$Q$点坐标）

将n个顶点和n个对应点按逆时针顺序重新排列，得到$2n$个边上的点：

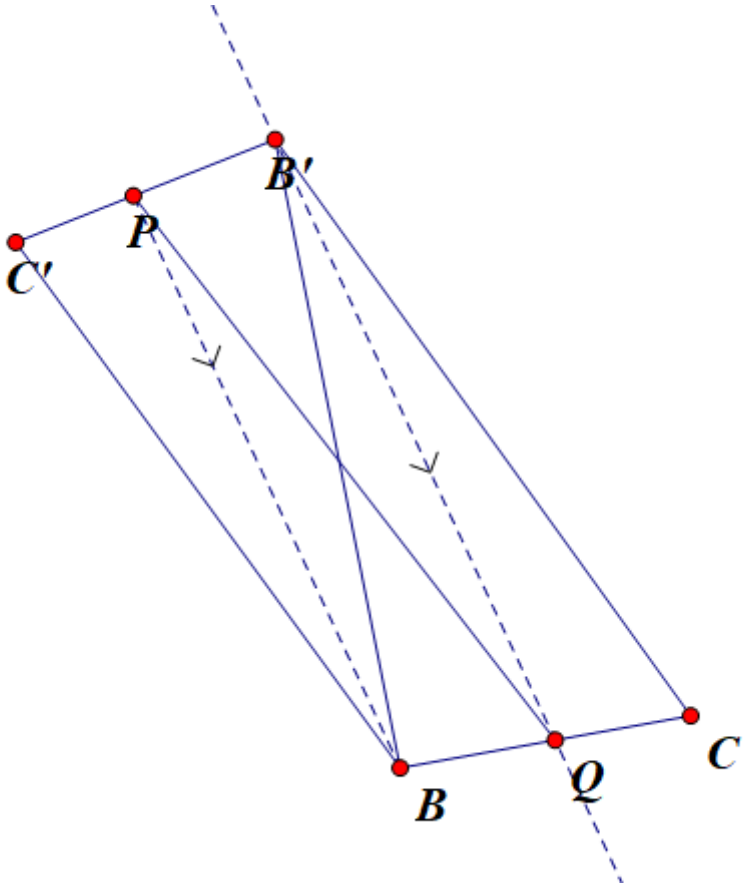


这些点将凸多边形分割为$2n$条线段，而任意一条多边形的面积平分线的端点必然在这些相邻点组成的线段上。我们枚举这些相邻点组成的线段，并找到当前枚举相邻点的两个对应点。可以证明，这两个对应点也是相邻的，因此它们相连形成的线段必然也在多边形上。



（如图，当前枚举到线段$B'C'$，其对应线段为BC，BB'、CC'都是多边形的面积平分线，假设BC之间有存在顶点或顶点的对应点P，那么与BB'、CC'比较后发现，P的对应点P'既不可能在B'右侧，也不可能在C'左侧，否则都无法平分多边形面积，因此P'必在线段$B'C'$上，但这与枚举的B'、C'相邻矛盾。故点P不存在。）

因此，对于任意一个在当前线段上的点，其对应点必然在端点对应点组成的线段上，所以问题可以简化到当前线段与对应线段组成的四边形上。那么假设已知当前点P的坐标，如何快速求解其对应点的坐标呢？



如图，由于BB'平分多边形面积，显然PQ也平分面积的充要条件是$S_{\triangle BB'C'} = S_{BQPC'}$。连接$BP$，去掉公共面积$\triangle BPC'$，原式变为$S_{\triangle BB'P} = S_{\triangle BQP}$。由于两三角形的$BP$边同底，故三角形的高相等，因此$B'Q//BP$，所以过点$B'$作$BP$的平行线，与$BC$的交点就是$Q$点。另一种做法是同样使用面积法，已知$S_{\triangle BQP}$和点$P$到$BC$的距离，求出线段$BQ$的长度，进而得到$Q$点坐标。

当然，我们实际上不可能枚举点P的坐标。我们发现，当P移动时，Q始终在线段BC上，故$|PQ|$（线段PQ的长度）关于点P位置的函数是光滑的。观察发现，对于P所在的每段线段，$|PQ|$关于P的位置是凹函数，即至多只有一个极小值点（极大值点必然在端点处），因此设（以上图为例）$\vec{B'P} = k\vec{B'C'}(k \in [0,1])$，则$P,Q$坐标仅与$k$有关，问题转换为函数$f(k) = |PQ|$关于$k$的极小值。由于$f$是单峰函数，因此用三分法/折半搜索法/爬山/模拟退火之类算法乱搞就行了。

实现起来还是有一点难度的。由于求顶点的对应点时，顶点是按逆时针排序的，因此求得的对应点序列也是按逆时针排序的，故将两个数组合并时，可以以任意一个顶点为起点，按顺序找到第一个在当前顶点和下一个顶点组成的线段上的对应点，并依次将对应点加入新的点集数组，直到下一个对应点不在这条线段上为止。注意保存点的对应信息，可以使用指针或数组下标模拟指针来减少复杂度。

G

如果第i个数a_i最终被放在左半部分，则对于它至少要交换的次数，应是左边比它大的数的个数，同理，右边也一样。考虑以下几种种情况（假设相邻的两个数是$a_i, a_j, i < j, L_i$为左边比a_i大的，R_i同理

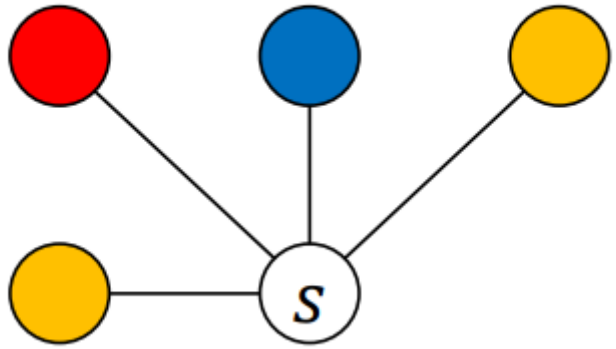
）：

- $a_i == a_j$，无论，二者分别放在哪边都不用交换。
- $a_i < a_j$并且a_i最终在左半部分，a_j也在左半部分，二者不用交换，L_i与R_i中也没有。
- $a_i < a_j$并且a_i最终在右半部分，a_j也在左半部分，二者需要交换，交换次数算在L_i中。

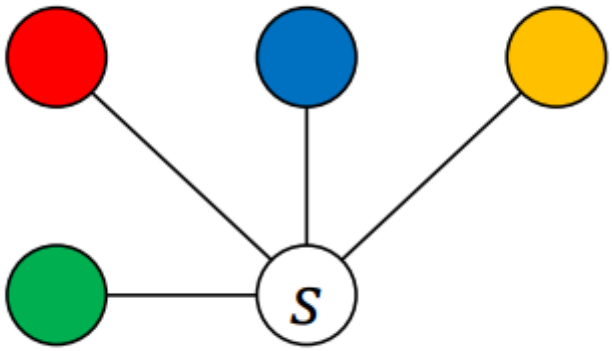
综上，交换次数恰为那一段比该数大的个数。因此，计算出每个数的L_i, R_i，再将小的加入答案就好。计算iL_i, R_i时，用树状数组，复杂度就为O(n*logn);

H

显然，每个点最多连接8条边。如果按从左上到右下、先行后列的顺序构造每个点的颜色，那么与新加入的点相邻且已被染色的点，至多只有4个，分别位于该点的左上、正上、右上、左侧。如果这四个点不都存在，或者四个点的颜色有重复的，那么当前点的颜色可以直接指定为未被占用的颜色。

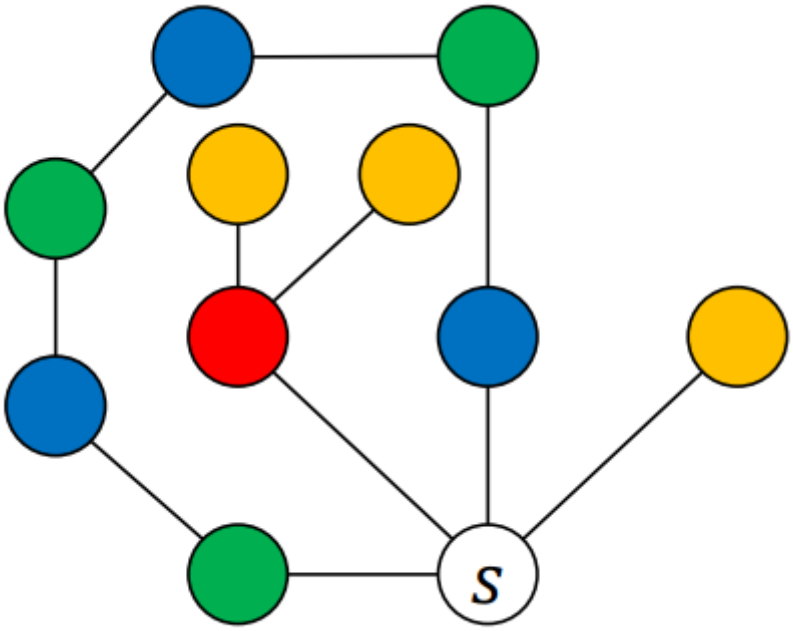


否则，当前点相连的四个点颜色互不相同：



此时考虑如何“反悔”之前的选择。如图，如果只考虑改变左侧绿色点，可能进一步影响到与这个点相连的其他点的选择。但如果只考虑两种颜色，如蓝色与绿色，并试图将所有蓝色点与绿色点颜色互换，那么另外两种颜色的点是无影响的。所以，考虑左侧点所在的、仅由蓝色点和绿色点组成的连通块，将其颜色互换。如果当前点上方的蓝点不在这个连通块内，那么转换完成后，当前点左侧的点变为蓝色点，当前点就可以涂绿了。

但如果上方的点在连通块内呢？



此时，在蓝色和绿色之间互换颜色显然是无效的。但注意到，此时蓝色点和绿色点组成了环形，并将左上的红色点包围了。由于点的连线不交叉，在仅考虑红点和黄点的情况下，右上的黄点和左上的红点必然不在一个连通块内。因此，此时只需把左上的红点所在的连通块，将红色与黄色的点颜色互换即可。这样当前点被涂为红色。于是，无论之前的点怎样涂色，必然有一种方案，使得经过“反悔”步骤的交换颜色后，当前点可以被染为某种颜色。因此按顺序枚举点并进行相应操作，即可完成染色。

I

这道题是让你求01异或矩阵的秩，而且还问每一位翻转后矩阵的秩会不会增加。刚开始的想法是对列向量$O(m)$遍历一下，然后每次用m-1个向量构成一个线性基，然后将剩下的向量每一位都翻转一下在线性基里筛一下看看筛出来是不是0向量，但是这个不管是查询还是构造的时间复杂度是$O\left(\frac{nm^3}{w}\right)$（如果是nm遍历然后高斯消元那更刺激，直接$O\left(\frac{nm^4}{w}\right)$），直接爆炸了。

但是我们如果这么处理：将m-1个向量构成的线性基变成高斯消元后的形式（具体看代码实现，与原线性基等价），然后把剩下的一个向量放线性基里筛一下，我们可以得到一个筛后的向量，然后我们考虑该某一位修改的影响，如果这一位发生了改变，那么这个向量在线性基原来是被这一位主元的向量筛/不

筛的就要变得不筛/筛的，相当于该向量再异或上这个主元向量，就可以把查询的时间复杂度优化到$O(\frac{nm^2}{w})$但是构造复杂度依旧很头疼，因为一个向量对最多两个区间有影响，所以可以考虑线段树分治，把修改放在线段树上然后遍历一整颗线段树就好了最终复杂度为$O(\frac{nm^2logm}{w})$

(英文题解莫名其妙看不懂，只能写写线段树分治，而且我还是以行向量为基准写的. --gsd)

J

首先，考虑一个结论：在一棵树上，选一些点，那么，想要联通这些点的最小边数emin可以如下计算：先将这些点集$s[]$按**dfs序**排序，再求出 $dist(s[1],s[2])+dist(s[2],s[3])+..+dist(s[n],s[1])=2*emin$。简单说明一下该结论的正确性：

1. n=2时，显然成立；
2. n>=3时，对于最终选中的每一条边,假设该边的父亲为f，该边的儿子为son，那么在需要连接的点集s中一定存在 $s[i]$和$s[i+1]$满足$dfn[s[i]]<=dfn[f]<dfn[s]<=dfn[s[i+1]]$,再讨论 点$s[j],j>i+1$，若点$s[j]$不存在或与点$s[i+1]$在同一棵子树，那么 $dist(s[n],s[1])$一定会访问且仅访问该边第二次；若点$s[j]$与点$s[i+1]$在不同子树，$dist(s[i+1],s[j])$一定会访问且仅访问该边第二次

根据以上结论，用set维护每种颜色的点序列，处理出初始状态时每个set所需边数，时间复杂度为$O(n*logn)$；每次U操作，需要从原颜色集合删除并在新元素集合添加，因此，修改复杂度为$O(logn)$；每次Q的时间复杂度是$O(1)$

K

先贪心求出能赢的局数，对对手的每一张牌选择相应的牌，尽可能的大，还要保证后面的牌能赢够足够多的局数。

暴力的话就是，枚举对手的牌，用自己还有的每一张牌去尝试，每次尝试又需要枚举对手剩下的牌，因此复杂度为$O(n^3)$

自己手中的牌排好序，假设对手当前牌为$p[i]$，二分地从自己手中选择最大的牌$f[j]$。check时，如果当前自己的牌输了，但是需要这张牌赢才能赢够局数，那就区间右移；如果这张牌输了而且本来不能赢，但是赢的局数还不够，那就区间左移；如果这张牌赢了，但是后面赢得局数还不够，就区间左移，其余情况就右移。
最终时间复杂度就为$O(n^2*logn)$