distance

首先，对输入进行排序，考虑枚举两个位置，计算这两个位置相对差的贡献。
因此转变为要求在两个子集中，在这两个位置左边的元素个数相等，右边也是同理。

$\Sigma_{i=0}^{min(x,y)}C_x^iC_y^i=C_{x+y}^x$
预处理组合数，时间复杂度$O(n^2)$。

tree

因为要考虑路径中边的最大值，所以我们从小到大考虑每条边。
然后利用带权并查集建出重构树，然后设$f_{i,j}$表示在i子树内选取了j个黑点的答案。
按照树形背包的方法进行转移。
总时间复杂度为$O(n^2)$。

string

观察到数据范围很奇特，因此对于正则表达式的长度数据分治。
1.如果$|s_i|\leq B$，暴力枚举每一个长度不超过B的字符串并且检验是否合法，复杂度为$O(2^B\Sigma|s_i|)$。
2.对于$B\leq|s_i|$的情况，枚举至少有哪些字符串能匹配上最终的结果，把枚举的字符串合并后答案就是$2^{\text{问号个数}}$。然后容斥即可。复杂度为$O(2^{n/B}\Sigma|s_i|)$。
因此，当B取$\sqrt{n}$的时候复杂度最优，为$O(2^{\sqrt{n}}\Sigma|s_i|)$。

even

考虑所有数都是2的幂次，那么所有数在变成1之前都是偶数。因为每次操作的数都是区间内的最大值，一个数最多被操作log次，因此等价于区间第k大。

用主席树做，每次查询是在树上二分的过程。由于点的个数是$nlogn$个，总复杂度为$O(nlog^2n+Qlogn)$。

那如果序列里全是奇数会有什么区别?
假设当前操作了一个奇数，并把它操作一步后变成了一个偶数，那么下一步操作的数一定还是它 (因为不存在别的偶数)。这个过程直到他被操作到奇数为止。

所以我们发现一旦某一时刻全部数都是奇数。
问题可以转变成我们的序列时时刻刻都是奇数（当然算答案时不一定）。
我们可以把每个数从一个奇数变到下一个奇数的步数算出来当成一个值。
于是这个问题又变成了区间第 k大。

综合上面两部分：
首先处理有偶数的情况，

其次处理区间内全是奇数的情况。
时间复杂度:$O(nlog2n + Qlogn)$