x

我们化一下前面的绝对值, 就会发现两个端点分别是x - y, x + y.

所以我们每个点看成区间(x - y, x + y) 两个区间没有交集代表他们有一条边.

所以我们直接贪心求一个区间不相交的覆盖问题就可以了.

时间复杂度O(n log n)

h

考虑在序列上分治, 每次计算跨过分治中心的对数.

考虑点对(i, j), j > i, 那么$dis = h_i + h_j + j - i - 2 * Mn \le k$, `Mn`为两点之间最低的高度.

那么也可以写成这样:$dis = h_i + h_j + j - i - 2 * min(Mn_i, Mn_j) \le k$, `Mn[i]` 为$i$到分治中心之间最低的高度

我们假设$Mn_j \ge Mn_i$, 另一种情况直接再做一次来统计.

那么显然要满足: $h_i - i - 2 * Mn_i \le k - h_j - j$

那么问题就很简单了, 我们扫描一边, 对于每个点, 用树状数组维护当前插入了多少个左边的柿子, 对于当前点, 直接用树状数组查询有多少个数小于右边的柿子, 这就是它的贡献.

$O(m \log n \log k)$

c

容易发现数字变成相同的值的过程是独立的, 那么a_i变成数字p($p > i$)的花费就是`__builtin_popcount(p - a[i] = t + mx - a[i])`

我们令$b[i] = mx - a[i]$, 那么问题就变成了, 找一个t, 使得$\sum_{i=1}^{n} __builtin_popcount(t + b[i])$最小.

我们从低位开始到高位dp, 那么对于当前这一位我们就要知道: 1. b[i]在这一位上的值. 2. 上一位的进位 3. t在这一位填一个什么数.

所以我们设`Dp[i][s]`表示作到从低到高第i位, 现在向下一位的进位情况为S, 转移的时候只要判一下填1还是填0大就知道t当前的取值.

这样做是$O(2^n \log n)$.

考虑优化, 我们发现前n位加的数字是相同的, 那么越大的数字加上t的前n位后越有可能进位, 所以我们每次将整个数据按照前n位排序,只要记`dp[i][j]`表示计算到第i位, 有j个进位, 我们就具体知道有哪些进位了.

时间复杂度$O(n \log n \log max(a[i]))$, 使用基数排序可以优化到 $O(n \log max(a[i]))$