基因编辑(gene)

考虑什么样的 (i,j) 可以做出贡献，首先有 $i<l$ 且 $r<j$。同时要求 (i,j) 是唯一的 (s_i,s_j)，那么有：

1. i 是第一个颜色为 s_i 的位置，j 是最后一个颜色为 s_j 的位置。
2. j 在 i 的下一个颜色为 s_i 的位置的左侧。
3. i 在 j 的上一个颜色为 s_j 的位置的右侧。

发现对于第一个限制，就是确定了哪些点能够做 i，哪些点能够做 j。

对于第三个限制，决定了 j 只对一段区间内的 i 有用，所以我们从 1 扫到 l，那么每一个 j 都会在某一个时刻开始可以被选；由于 j 在 i 的下一个同色位置的左侧，所以只需要检验最小的 j 是否比它小即可。

时间复杂度 $O(n\log n)$ 或者 $O(n)$。

长野原龙势流星群(tree)

发现有一个很显然的二分答案做法，我们先二分答案 X，然后把 i 点的权值变成 w_i-X，就是检验是否存在以这个点为根的权值和 ≥ 0 的连通块了。这个问题是有一个很显然的贪心的的：f_i 表示以 i 为根的答案，那么就有 $f_i=w_i-X+\sum\limits_{y\in son(i)}\max(f_v,0)$，可以认为是它只选择那些 >0 的儿子。

但是发现二分没有前途，所以我们考虑对于 X 进行扫描，直接从大到小，那么所有点的权值也都从 w_i-X 开始从小往大增大。

如果我们能在这个过程中动态维护所有的 f_i，那么我们就只需要关注每一个 f_i 变成 0 的时刻的 X 即可。

而对于一个 f_i，其变成 0 之后，随着 X 的增大，其必然一直保持 $f_i>0$，那么在其父亲的决策中就必然会选择这个点。那么我们就可以让其和其父亲所在的连通块合并。

那么我们就变成了合并树上的两个连通块，以及查询当前所有 <0 的 f_i 中最早变成 0 的 f_i（也就是找最小的 $\frac{f_i}{siz_i}$）。发现可以直接使用并查集和可删堆分别处理，时间复杂度 $O(n\log n)$。

京都观光(kyoto)

行走的路径是一条折线，因此，问题的重点就在于什么情况下会选择转弯。

例如考虑 l,r 两行，以及 x,y,z 三列，什么时候会选择路径 $(l,x)\to(l,y)\to(r,y)\to(r,z)$，而不是 $(l,x)\to(r,x)\to(r,z)$ 或 $(l,x)\to(l,z)\to(r,z)$。

分别计算三种情况的代价：

- $(l,x)\to(l,y)\to(r,y)\to(r,z)$：$A_l(y-x)+A_r(z-y)+B_y(r-l)$。
- $(l,x)\to(r,x)\to(r,z)$：$B_x(r-l)+A_r(z-x)$。
- $(l,x)\to(l,z)\to(r,z)$：$A_l(z-x)+B_z(r-l)$。

求解不等式组 $\begin{cases} A_l(y-x)+A_r(z-y)+B_y(r-l) \le B_x(r-l)+A_r(z-x) \\ A_l(y-x)+A_r(z-y)+B_y(r-l) \le A_l(z-x)+B_z(r-l) \end{cases}$ 有:

$$\frac{B_y-B_x}{y-x} \le \frac{A_r-A_l}{r-l} \le \frac{B_z-B_y}{z-y}$$

发现如果是从 y 列转弯，就需要上述不等式成立，而上述不等式可能成立的前提条件为 $\frac{B_y-B_x}{y-x} \le \frac{B_z-B_y}{z-y}$。这一个对于 B 单独的斜率型的限制，说明了，只有所有 (i,B_i) 构成的下凸包上的点对应的列上，才有可能纵向移动。A 序列同理。

使用单调栈对 A 和 B 建立凸包，而根据上页的不等式可知，实际移动的路线会选择斜率更小一侧移动，而在 A 和 B 的凸包上，斜率是分别单调递增的。

对于 A 和 B 同时维护两个指针，使用类似归并的方式处理，就可以得到移动的路径，进而计算出答案。时间复杂度 $O(H+W)$。

求和避免(sum)

首先考虑求解 N 的值。

如果 $a \in A$，则必然有 $(S-a) \notin A$。因此将 $[1,S)$ 中和为 S 的数两两匹配，每一对数中只能够选择一个。因此有 $N \le \left\lfloor \frac{S-1}{2} \right\rfloor$。

同时这个上界也是可以取到的，具体的，取每一对数中较大的那个，也就是所有 $> \frac{S}{2}$ 的数。此时，任意两个数的和都 $> S$，满足第二条限制。

再考虑如何最小化字典序。

从小到大枚举 t，如果将 t 加入 A 中仍然合法，就直接加入。可以证明这样的方法仍然能够满足 N 最大：

如果 t 不能加入 A，说明对于当前已经加入 $A=\{a_1,a_2,\dots a_{N'}\}$，不存在非负整数列 $\{x_1,x_2\dots x_{N'}\}$ 使得 $\sum_{i=1}^{N'} x_i a_i = t$。

证明考虑反证法，如果存在一组 $\{x\}$，使得 $\sum_{i=1}^{N'} x_i a_i = t$，由于 t 不能加入，说明存在非负整数列 $\{x'_1,x'_2\dots x'_N,y\}(y\neq 0)$，使得 $\sum_{i=1}^{N'} x'_i a_i + yt = S$，将前式带入可以得到 $\sum_{i=1}^{N'}(x_i+yx'_i)a_i = S$。这与当前的 A 合法矛盾。

从此以后加入的元素都 $> t$，因此无论如何 A 中元素无法组成 t，说明将 $S-t$ 加入 A 中将是合法的。

这也就是说明，这样贪心处理所有 $t \le \frac{S}{2}$ 的正整数，即可直接唯一确定 $> \frac{S}{2}$ 部分的选择，使得 A 数列的大小达到理论最大值。

但是由于 S 太大了，需要尝试加速上面贪心的过程。

对于 A 中的第一个元素，其应当是最小的满足 $t \nmid S$ 的正整数 d。而由于 $\operatorname{lcm}(1,2,\dots 50) > 10^{18}$，所以有 $d < 50$。

仿照上页的证明方式，我们可以证明，对于任意 $a, b \in A$，若 $a+b<S$，则有 $a+b \in S$。特别地，取 $b=d$，能够说明只需要对于每一个 $r=0,1 \ldots d-1$，确定最小的正整数 $a \bmod d=r$，使得有 $a \in S$。

设 f_i 表示当前已经确定了最小的满足 $x_0 \bmod d=i$ 且能够加入 A 中的 x_0 是多少。

那么初始时就有 $f_i=\begin{cases} d & ,i=0 \\ +\infty & ,i \neq 0 \end{cases}$。

考虑当前能够新加入的最小的 v 是多少：假设一组方案中选择了 i 个 v（$i=1,2 \ldots d-1$），且希望 v 能够合法，就必然有 $f_{(S-iv) \bmod d}+iv>S$。

移项后得到：$iv>S-f_{(S-iv) \bmod d}$，即 $v \geq\left\lfloor\frac{S-f_{(S-iv) \bmod d}}{i}\right\rfloor+1$。那么也就有 $v \geq \max\limits_{i=1}^{d-1}(\left\lfloor\frac{S-f_{(S-iv) \bmod d}}{i}\right\rfloor+1)$。

后半部分取 $\max$ 的式子只与 $v \bmod d$ 的值有关，因此对于还没有确定最小值的 f_r，计算出在 $v \bmod d=r$ 的时候 v 的最小值 v_r。在所有的 v_r 中选择最小的那一个更新。

更新时，首先有 $f_r \leftarrow v_r$。然后需要进行一个类似于同余最短路的操作，对于所有 i, j 进行 $f_{(i+rj) \bmod d} \leftarrow \min(f_i+v_r j)$ 的操作。

由于每一个 $f_r \leftarrow v_r$ 只会进行一次，所以这一部分的复杂度为 $O(d^3)$。

对于找到第 K 小的，可以通过二分答案来确定。这一部分的时间复杂度为 $O(d \log n)$。