

A

先考虑最靠左和最靠右的一系列同学，根据他们的行进方向照片的上下边界的长度会按照 $-2, -1, 0, 1, 2$ 增长，且每次增长的数值必然不小于上一次增长的数值（考虑边界上如果出现新同学或旧同学减少都不会使增长速度变慢）。

于是上下边界的长度随时间变化是一个凸函数，左右同理，相加还是凸函数（注意凸函数的性质比单峰函数更强，两个单峰相加不一定还是单峰），所以可以对时间 t 进行三分求函数极值。

B

如果 t 时刻阳光足够却没有种植向日葵，而在 $t + 1$ 时刻种植，显然会亏一轮阳光。

如果某时刻场上所有向日葵产出的阳光量大于两种向日葵加起来的花费，则可以在每一轮的开始直接购入这两种向日葵，直到最后发现买某种向日葵到回合结束时不能回本时停止购买此种向日葵，后续的代价可以 $O(1)$ 计算，我们将这种情况称为阳光自由。

考虑最劣的情况，即两种向日葵的花费均为 1023，产出阳光均为 1，大概估算一下需要至多 $\frac{2046}{1} + \frac{2046}{2} + \dots$ 轮达到阳光自由，约为 20000 轮左右，于是考虑动态规划，以 $f_{i,j}$ 代表第 i 轮场上向日葵的阳光产量为 j 时目前的最大阳光存量，枚举每种决策转移即可，当 j 超过 $c_1 + c_2$ 时 $O(1)$ 计算获取阳光自由后的收益即可。

C

深层的营养价值比浅层的低，而从树上某节点回到根的路径唯一，所以显然是从根节点开始贪心进行装入，直到作物装完或背包装满为止。

记 1 操作数量为 k_1 ，2 操作数量为 k_2 ，3 操作数量为 k_3 ，先考虑最简单的一种情况：

如果 1 操作在 2, 3 操作之前出现，则树的形态固定，策略肯定是在找到当前路径上离根最近的一个还有作物的节点后贪心取，如果取完后背包未满载则找到下一个离根最近的还有作物的节点继续贪心取，取的过程直接对 w_j 进行修改。

在这种策略下，贪心取的次数不会超过 $k_1 + k_2$ ，因为在树上的一个节点取时要么直接把背包取满，要么直接把这个节点的资源取空，复杂度均摊下来是正确的，考虑优化找到节点的过程，可以利用倍增，维护 i 的 2^j 级祖先，直接找到第一个 $w_j \neq 0$ 的节点即可。

对于操作 3 如果直接置零会影响到操作 2，所以先置为 -1 ，在这个点是当前要取到的节点时才重新置为 0。

每个操作 3 最多影响 1 个节点在倍增的时候被多做一次清空标记的动作，故复杂度为 $O(k_1 + k_2 + k_3) \log k_1$ 。

对于树形态不固定的情况，每次只加入一个叶子节点，不会影响前面点的倍增数组，所以可以每次添加进一个点后对这个点单独进行倍增的预处理即可，时间复杂度相同。

D

所有需要计算的 v 都与 1 号节点相连，故 $d(1, u) - 1 \leq d(u, v) \leq d(1, u) + 1$ 。

计算所有节点与 1 之间的距离，那么每条边只能连接两个到 1 的距离差值为 0 或 1 的节点，将这两种边记为 A 类和 B 类，且规定 B 类边为从距离大的点到距离小的点的有向边。

则 $d(u, v) = d(1, u)$ 当且仅当 u 可以通过一条 A 边和若干条 B 边到 v , $d(u, v) = d(1, u) - 1$ 当且仅当 u 可以通过若干条 B 边到 v , 其余情况都是 $d(u, v) = d(1, u) + 1$ 。

先将每个点的答案初始化为 $\max(a_u - b_u \times [d(1, u) + 1])$, 那么答案会在点 u 上增加要保证 u 到 v 最多经过一条 A 边。

构建分层图, 每一层都是 B 边组成的 DAG, 层与层之间通过 A 边连接, 只能从下层走到上层, 意义是经过了一条 A 边。相当于分层图也是一个 DAG。

在 DAG 上进行 DP, 初始化下层点的 $f(u) = a_u - b_u \times [d(1, u) - 1]$, 上层点的 $f(u) = a_u - b_u \times d(1, u)$, 相当于对每个 v 找能够通过 $d(1, u) - 1$ 或 $d(1, u)$ 到 v 的 u 中的 $f(u)$ 的最小值, 即自底向上更新:

$$f_v = \max_{u \rightarrow v} f_u$$

最后直接输出与 1 号点相连的 v 的 f 值和 $\max(a_u - b_u \times [d(1, u) + 1])$ 之中的最大值即可。