A

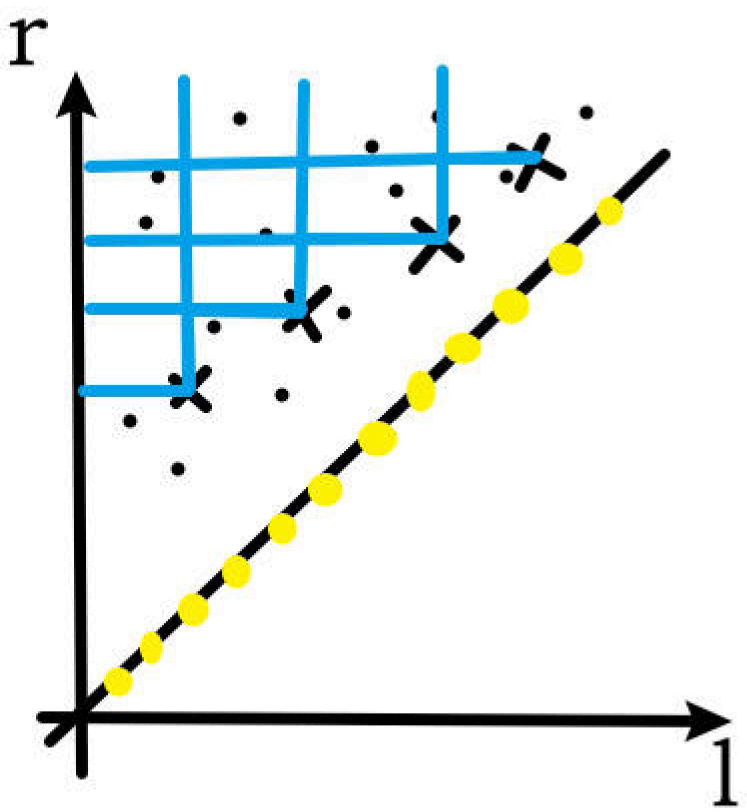
每个点问一次$d=1$，然后问一次$d=h+1$，这样可以得到根，第二层。同时目前已知和全局除了最底下一层和第一层系数都相同

取出前两层的点问一次$d=h$，得到根，再通过问一次$rt,2$和两次第二层，距离为1就可以算出答案。

B

我们把这个问题进行一个建模。具体地，对于 $a_i<b_i$ 的 (a_i,b_i)，我们视为一个"区间"(闭区间，左右端点为 a_i,b_i)。对于 $a_i>b_i$ 的 (a_i,b_i)，我们视为一个"反区间"。

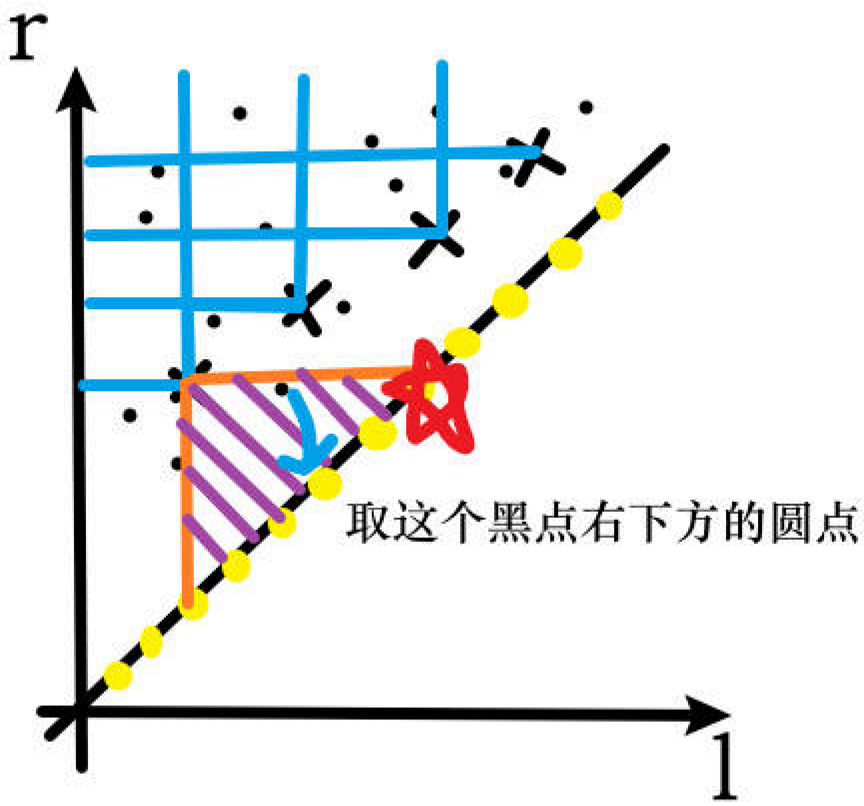
我们在二维平面上刻画该问题。我们用点表示每个"区间"，用 X 表示极短的"反区间"，用 O 表示横纵坐标相等的点，设第 i 个 X 左上方的点数为 c_i，第 i 个 O 左上方的点数为 d_i，那么答案就是 $\max(\max c_i+1,\max d_i)$。



我们考虑，添加一个 X 的时候，只有当阴影区域内没点，这个 X 才可能对答案造成贡献。

否则，取这个 X 右下方的 O，这个 O 目前左上方的点数一定不小于 X 左上方的点数 $+1$。而这个 O 覆盖范围严格包含 X，所以之后这个 O 一定一直优于 X。这个 X 没用，不需要加入。

那么只需要考虑阴影内没点的情况。此时投影到某个坐标上再弄一个线段树维护即可。



判断阴影区域有无点，再在按横坐标排序的 set 中查前驱即可。

时间复杂度 $O(n\log n)$。

C

首先随便定一个根，把 $\text{dis}(u,v)$ 转化为 $dep_u+dep_v-2\times dep_{\text{lca}(u,v)}$。

由于 $\sum dep_{p_i}$ 是固定的，所以我们只需要最小化 $\sum -dep_{\text{lca}(p_i,p_{i \bmod n+1})}$，即最大化 $\sum dep_{\text{lca}(p_i,p_{i \bmod n+1})}$。

考虑从下向上合并，从 u 的子树中会传上来若干个还没匹配的点，然后在 u 钦定若干个 pair 匹配，其他的继续往上传即可得到一个 $\mathcal{O}(Tn^3)$ 的 dp。

注意到转移方程的形式是 $f'_{i,j+k-t}=\max(t\times dep_i+f_{i,j}+f_{son,k})\ (0\le t\le 2\times\min(j,k))$，如果枚举 j,k,t 那么时间复杂度就是 $\mathcal{O}(Tn^3)$ 的，但我们可以给 f 数组的第二维做整体减，即将 $f_{i,x}$ 减去 $x\times dep_i$，这样新的 f' 数组的转移就没有了 $t\times dep_i$ 一项，是一个区间取 $\max$ 的形式，于是可以做到 $\mathcal{O}(Tn^2)$。

现在，我们需要考虑的是怎么快速计算 $h_{j+k-t}=\max(f_j+g_k)\ (0\le t\le 2\times\min(j,k))$。

考虑取出 f_j 和 g_k 的最大值位置，设为 x,y，不妨设 $x<y$，那么 h_{y-x} 至 h_{y+x} 的值都等于 f_x+g_y。

接下来我们考虑 $y-x$ 的左侧和 $y+x$ 的右侧。

这里有一个结论是，f 和 g 都是凸包，接下来会证明这个结论。

对于 $y+x$ 的右侧，转移式形如 $\max(f_j+g_k)\to h_{i,j+k}$，并且选择的 j,k 一定满足 $j\ge x$ 且 $k\ge y$，否则在凸包上一定能调整到更优，然后这就是一个 minkowski 和。$y-x$ 的左侧同理。

时间复杂度 $\mathcal{O}(Tn\log n)$。