这个问题可以通过计算直线的**上限包络线 (Upper Envelope)** 来在 $O(n \log n)$ 时间内解决。上限包络线是由所有直线中在每个 x 坐标处 y 值最大的那部分直线段组成的分段线性函数。上限包络线上的所有线段就是可见的线段。

算法思路

该算法采用分治策略，其递归关系类似于归并排序。

1. **分解 (Divide):** 将 n 条输入直线集合 S 分成两个大小近似相等的子集 S_1 和 S_2。例如，可以简单地将前 $n/2$ 条直线分给 S_1，后 $n/2$ 条直线分给 S_2。
2. **解决 (Conquer):** 递归地调用算法来计算 S_1 的上限包络线 U_1 和 S_2 的上限包络线 U_2。
3. **合并 (Combine):** 将 U_1 和 U_2 合并，以得到原始集合 S 的上限包络线 U_{12}。这是算法的关键步骤。
   - 由于 U_1 和 U_2 都是分段线性的函数，它们可以表示为一系列按 x 坐标排序的线段。
   - 合并过程类似于合并两个已排序的列表。我们遍历 U_1 和 U_2 的线段，找到它们的交点。这些交点是 U_1 和 U_2 之间切换上方直线的位置。
   - 设 U_1 由 k_1 条线段组成，U_2 由 k_2 条线段组成。由于上限包络线的性质，最多有 $k_1 + k_2$ 个交点。因此，合并可以在 $O(k_1 + k_2)$ 时间内完成。由于 $k_1, k_2 \leq n/2$，合并步骤的时间复杂度为 $O(n)$。

递归关系与时间复杂度

该分治算法的递归关系为：

$$T(n) = 2T(n/2) + O(n)$$

其中 $T(n)$ 是处理 n 条直线所需的时间。根据**主定理 (Master Theorem)**，该递归关系的时间复杂度为 $O(n \log n)$。

算法步骤概述

1. 定义函数 ‘ComputeUpperEnvelope(Lines)‘：
2. 如果 ‘Lines‘ 只有一个元素 L_i，返回包含 L_i 的列表。
3. 如果 ‘Lines‘ 为空，返回空列表。
4. 将 ‘Lines‘ 分为 ‘Lines$_Left$‘ ‘$Lines_Right$‘
4. 递归调用 ‘U$_Left = ComputeUpperEnvelope(Lines_Left)$‘
4. 递归调用 ‘U$_Right = ComputeUpperEnvelope(Lines_Right)$‘
4. 调用 ‘MergeEnvelopes(U$_Left, U_Right)$‘ ‘$U_Combined$‘
4. 返回 ‘U$_Combined$‘