作业 2

Algorithm 1 切割木棍

思路: 用二分确定切割长度, 并且对于每个切割长度计算是否满足数量 = k.

时间复杂度: $O(n \cdot \log(max(L)))$

输入: 一组木棍长度 L, 切割数量 k

输出: 最大可能的切割长度 max_len

初始化最小值 $min\_len = 1$ 和最大值 $max\_len = \max(L)$

while $min\_len \leq max\_len$ **do**

  计算中间长度 $mid = \frac{min\_len + max\_len}{2}$

  初始化总段数 $total\_segments = 0$

  for 每根木棍长度 $length$ in L **do**

    增加 $total\_segments$ by $length//mid$ 计算可以切割的段数

  end for

  if $total\_segments \geq k$ **then**

    更新 $min\_len = mid + 1$

  else

    更新 $max\_len = mid - 1$

  end if

end while

return max_len 最大可能的切割长度

Algorithm 2 计算逆序对数量

思路是基于归并排序, 在归并时计算逆序对数量. 所以整体结构和归并排序类似, 只是加了计数部分.

时间复杂度为 $O(n \cdot \log(n))$

Function CountInversions(A)

if $length(A) \leq 1$ **then**

  return 0

end if

$L, R \leftarrow split(A)$ 将数组 A 分成左子数组 L 和右子数组 R

$leftInversions \leftarrow CountInversions(L)$

$rightInversions \leftarrow CountInversions(R)$

$acrossInversions \leftarrow MergeAndCount(L, R, A)$ 计算跨越左右子数组的逆序对数量

return $leftInversions + rightInversions + acrossInversions$

Function MergeAndCount(L, R, A)

$count \leftarrow 0$

$i, j \leftarrow 0, 0$ 初始化左子数组和右子数组的指针

for $k \leftarrow 0$ **to** $length(A) - 1$ **do**

  if $i < length(L)$ **and** ($j \geq length(R)$ **or** $L[i] \leq R[j]$) **then**

    $A[k] \leftarrow L[i]$

    $i \leftarrow i + 1$

  else

    $A[k] \leftarrow R[j]$

    $j \leftarrow j + 1$

    $count \leftarrow count + (length(L) - i)$ 计算逆序对数量

  end if

end for

return $count$

Algorithm 3 计算每个点的支配数量

时间复杂度为 $O(n \cdot \log(n))$

if $len(points) = 0$ **then**

空字典没有点，返回空字典

end if

对 $points$ 按照横坐标升序排序，如果横坐标相同则按纵坐标升序排序

将 $points$ 分成左子集 $left\_points$ 和右子集 $right\_points$

$left\_dom\_count \leftarrow CalculateDominance(left\_points)$ 计算左子集的支配数量

$right\_dom\_count \leftarrow CalculateDominance(right\_points)$ 计算右子集的支配数量

合并 $left\_dom\_count$ 和 $right\_dom\_count$ 成为总的支配数量字典 dom_count

初始化一个空的字典 $left\_max$，用于存储左子集中每个点的右侧支配点数量的最大值

for 每个点 P in $left\_points$ **do**

if P 不在 $left\_max$ 中 **then**

将 $left\_max[P]$ 初始化为 0

end if

for 每个点 Q in $right\_points$ **do**

if $Q.x \geq P.x$ 且 $Q.y \geq P.y$ **then**

增加 $left\_max[P]$ by 1

if $left\_max[P] \geq Q$ 的支配数量 **then**

停止循环

end if

end if

end for

end for

合并后的 dom_count 和 $left\_max$