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 ← CalculateDominance(left_points) 计算左子集的支配数量
right_dom_count ← 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 ≥ P.x 且 Q.y ≥ P.y then
      增加 left_max[P] by 1
      if left_max[P] ≥ 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
```

4

（1）若被分为每组7个，则大于中位数的中位数x的元素个数至少为：

$4(\lceil \frac{1}{2} \lceil n/7 \rceil \rceil - 2) \geq \frac{2n}{7} - 8$

类似的，可以推出：

$$T(n) \leq \begin{cases} \$O(1)\$, & \text{if } n < 126 \\ T(\lceil \frac{n}{7} \rceil) + T(\frac{5n}{7} + 8) + O(n), & \text{if } n \geq 126 \end{cases}$$

下面给出126的推导过程：

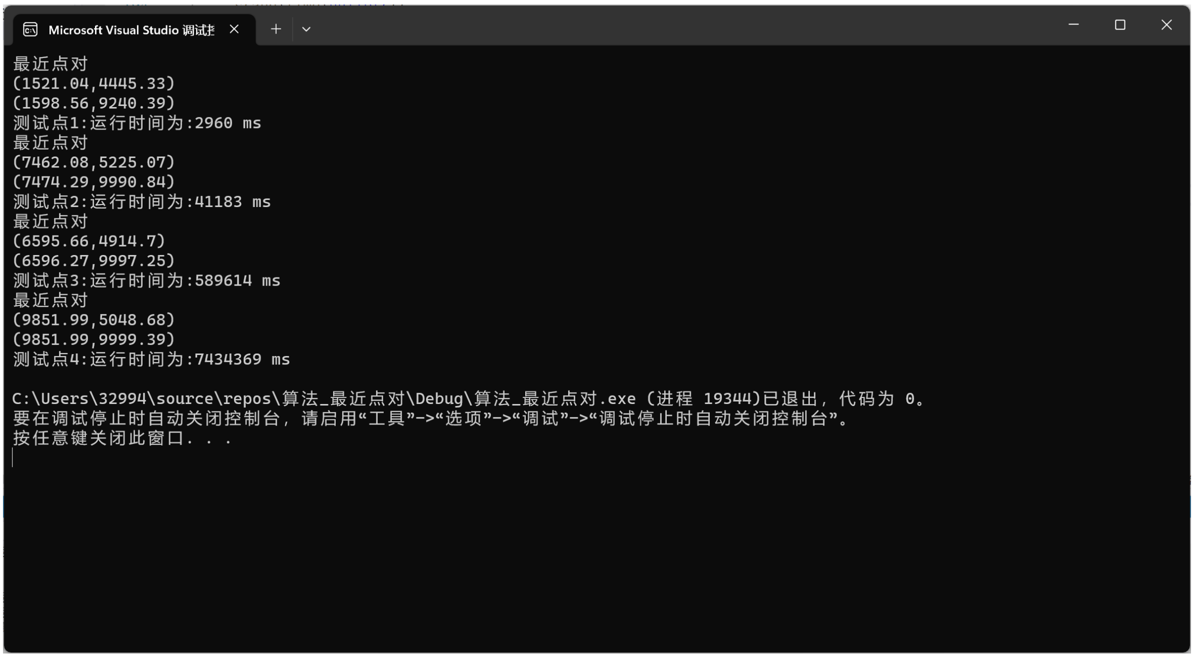
$$\begin{gathered} \text{对于足够大的常数} c\text{，和适当的常数} a\text{：} \\ T(n) \leq c\lceil \frac{n}{7} \rceil + c(\frac{5n}{7} + 8) + an \\ \leq cn/7 + c + 5cn/7 + 8c + an \\ = 6cn/7 + 9c + an \\ = cn + (-cn/7 + 9c + an) \\ \text{若下式成立，则上式最多为} cn\text{：} \\ -cn/7 + 9c + an <= 0 \\ \text{当} n > 63, \text{上式等价于} c \geq 7a(n/(n-63)), \\ \text{假设} n > 126, \text{所以} n/(n-63) \leq 2, \text{所以选择} c \geq 14a \text{即可满足要求。} \end{gathered}$$

（2）若为每组3个，$T(n) = T(\lceil n/3 \rceil) + T(4n/6) + O(n) \geq T(n/3) + T(2n/3) + O(n)$
$\geq T(2n/3) + O(n)$

用主方法:

a=1,b=3/2,所以复杂度$\geq O(n^{1.5})$,非线性

5



```
最近点对
(1521.04,4445.33)
(1598.56,9240.39)
测试点1:运行时间为:2960 ms
最近点对
(7462.08,5225.07)
(7474.29,9990.84)
测试点2:运行时间为:41183 ms
最近点对
(6595.66,4914.7)
(6596.27,9997.25)
测试点3:运行时间为:589614 ms
最近点对
(9851.99,5048.68)
(9851.99,9999.39)
测试点4:运行时间为:7434369 ms

C:\Users\32994\source\repos\算法_最近点对\Debug\算法_最近点对.exe (进程 19344)已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用"工具"->"选项"->"调试"->"调试停止时自动关闭控制台"。
按任意键关闭此窗口. . .
```