算法与复杂性 第五次课程作业

UNIkeEN

2023 年 4 月 26 日

问题解答

1. 设 P 是包围在给定矩形 R 中的一个简单多边形，q 为 R 中任意一点，设计高效算法寻找连接 q 和 R 外部一点的线段，使得该线段与 P 相交的边的数量最少

解 以 q 作为原点作与 x 轴的平行射线（即建立水平极轴极坐标系），求出 P 其余各顶点的极坐标并按照极角大小排列。容易判断出极角大小相邻的两个顶点之间以 q 为出发点的射线与 P 相交边的数量小于等于这两个顶点时的交点个数。因此我们只需从小遍历极角得到每一段内的交点个数，维护并不断比较刷新最小交点个数。完成全部旋转时得到与 P 相交的边数量最少的极角范围，在此区域外取一点与 q 相连，得到所求线段。

2. 给定平面上一组点，已知每个点的坐标，求最远点对之间的距离，即点集的直径

解 容易证明，最远点对位于这组点的凸包上，使用 Graham 扫描算法可在 $O(n\log n)$ 时间内计算凸包。

得到凸包后，我们首先从凸包中取一条边（假设为 AB），遍历除 AB 外各点得到距离此边最远的点（假设为点 D），分别计算 DA 和 DB 的距离，取最大值。

逆时针取凸包的下一条边（假设为 BC），此时若继续遍历其他所有点则时间复杂度为 $O(n^2)$，观察得到最远的点也会随取边的顺序进行逆时针移动，则此时只需从上次遍历到的最远点（D）继续逆时针遍历其他点，得到距离当前边最远的点（假设为点 E），计算 EB 和 EA 的距离，取最大值，与当前最大值相比较，以此类推。此时 $O(n^2)$ 优化为 $O(n)$，结果即最远点对距离，点集的直径。

3. 如果已知任意两点之间的距离 d_{ij}，存储在矩阵 D 中，求这组点的直径，此时一种直观的解法就是把 D 扫描一遍，选择其中最大的元素即可。由于距离 d_{ij} 满足非负性、对称性和三角不等式，我们可以给出一种时间亚线性的近似算法。算法很简单，由原来确定性算法的检查整个矩阵改为只随机检查 D 的某一行，这样时间复杂性就由原来的 $O(n^2)$ 减少为 $O(n)$，相对于输入规模 n^2 而言，这是一个时间亚线性的算法。证明时间代价减小的同时，解不会小于最优值的一半

解 记 (p,q) 为平面上最远点对，即 d_{pq} 为该点集的直径。

在矩阵 D 中任取一行，即在平面上任取一点 x，考察 x 与平面上其他点的距离。

- 若 $x = p \vee x = q$，则 d_{pq} 一定在选定的这一行中，所得解即为最优值。
- 若 $x \neq p \wedge x \neq q$，则 d_{xp}, d_{xq} 一定在选定的这一行中。
  - 若 x, p, q 三点在同一直线上，则有 $d_{xp} + d_{xq} = d_{pq}$
  - 否则，三点构成三角形，则有 $d_{xp} + d_{xq} > d_{pq}$

  不失一般性，我们可以设 $d_{xp} \leq d_{xq} \leq d_{pq}$，

$$d_{pq} \leq d_{xp} + d_{xq} \leq 2d_{xq} \Rightarrow d_{xq} \geq \frac{1}{2} d_{pq}$$

  设 d_{xy} 为选定这行的最大距离，则

$$d_{xy} \geq d_{xq} \geq \frac{1}{2} d_{pq}$$

4. 有 n 种液体 $S_1, S_2, ..., S_n$，都含有 A,B 两种成分，含量分别为 $a_1, b_1, a_2, b_2, ..., a_n, b_n$，其中 $a_i + b_i < 100\%$。现欲利用这 n 种液体配制目标液体 T，使之 A 和 B 的含量分别为 x 和 y。设计算法判别能否成功配制，并给出算法时间复杂性

解 判断点 (x, y) 是否落在 $(1_1, b_1), ...(a_n, b_n)$ 构成的凸包内即可。

证明如下，如果 (x, y) 落在 $(1_1, b_1), ...(a_n, b_n)$ 的凸包内，则其在原点水平极坐标系中的极角位于凸包的极角范围内，可以通过极角为 $(1_1, b_1), ...(a_n, b_n)$ 中各值的小段线段首尾相连构成 $O - (x, y)$ 的线段，即可以成功配制。落在凸包外则无法首尾相连构造。

构造凸包和二分法判断是否位于凸包内，时间复杂度均为 $O(n \log n)$，总时间复杂度 $O(n \log n)$

5. 在平面上给定一个有 n 个点的集合 S，求 S 所有的极大点。极大点的定义：设 $p_1 = (x_1, y_1)$ 和 $p_2 = (x_2, y_2)$ 是平面上的两个点，如果 $x_1 \leq x_2$ 并且 $y_1 \leq y_2$，则称 p_2 支配 p_1，记为 $p_1 \prec p_2$。点集 S 中的点 p 为极大点，意味着在 S 中找不到一个点 q，$q \neq p$ 并且 $p \prec q$，即 p 不被 S 中其它点支配

解 对横坐标进行降序遍历，维护一个最大值表示当前遇到过的纵坐标值。每遍历一个点，如果其纵坐标大于之前遇到过的最大纵坐标，则说明没有其他店可以支配该点。

值得注意的是极大点不是唯一的，伪代码如下

Algorithm 1 求平面的极大点

输入: $S = \{(x_i, y_i)\}$（平面上的 n 个点）

输出: $P = \{(x_i, y_i)\}$（平面上所有的极大点）

```
S ← Sort(S, x, reverse = True)          ▷ 对横坐标降序排序，O(n log n)
maxY ← −∞
for (x, y) in S do                       ▷ 按横坐标降序遍历
    if y > maxY then                     ▷ 横坐标大于 x 的点纵坐标都小于 y，没有其他点可以支配该点
        P ← P ∪ {(x, y)}
        maxY ← y
```

else
continue ▷ 该点被之前遍历过的某个纵坐标为 $maxY$ 的点支配
end if
end for

6. 对凸多边形

- 有多少种三角划分的方法?

解 凸 n 边形的一条边可以与其他的 $n-2$ 个点构成三角形，且该三角形将原凸多边形分成两个凸多边形，这两个凸多边形可以以相同的算法分划，直到其边数不超过 3。

设 $S(n)$ 为一个 n 边形的三角形划分方法。则

$$S(n)=\begin{cases}0 & ,n=2\\ 1 & ,n=3\\ \sum_{i=2}^{n-1} S(i)\times S(n-i+1) & ,n>3\end{cases}$$

递推可得三角划分方法数。

- 如何使对角线长度之和最小? 如果觉得递推式不好求，写出递推式就可以了

解

记凸 n 边形为 $\langle p_1,p_2,\ldots,p_n\rangle$，其对角线长度和最小为 $T\left(\langle p_1,p_2,\ldots,p_n\rangle\right)$。

$$T\left(\langle p_1,p_2,\ldots,p_n\rangle\right)=\begin{cases}0 & ,n\le 3\\ \min_{1<i<n}\left\{\begin{aligned}T\left(\langle p_1,\ldots,p_i\rangle\right)+T\left(\langle p_i,\ldots,p_n\rangle\right)\\ +D(p_1,p_i)+D(p_n,p_i)\end{aligned}\right\} & ,n>3\end{cases}$$

其中，$D(p_i,p_j)=\begin{cases}0 & ,|i-j|>1\\ ||p_i-p_j|| & ,\text{otherwise}\end{cases}$

7. 给定平面上 n 条线段，设计算法用 $O(n\log n)$ 时间确定其中是否有两条线段相交。

解 使用扫描线算法。将线段投影到 x 轴或 y 轴时，相交线段的投影是有重合的（或有共同端点）。利用这一思想，每次考虑与扫描线相交线段中各组相邻线段是否相交。维护一个数据结构，储存当前扫描线相交的线段（遇到上端点时插入，遇到下端点时删除）。

按横或纵坐标升序遍历结点的时间复杂度为 $O(n\log n)$，由端点判断线段是否相交的时间复杂度为 $O(1)$。该数据结构可使用平衡树等，以 $O(n\log n)$ 的最坏时间复杂度实现插入、删除、查询等操作。

在本题中，一旦发现相交线段算法即可结束，伪代码如下

Algorithm 2 判断是否存在相交线段

输入: $L=\{l_i\}$（线段集合）
输出: 是否存在交点

对所有线段的两个端点以横坐标为键，(x, y, l) 为值建立最小化堆 H ▷ $O(n)$
$Candidate$ ▷ 用于存储当前状态与扫描线相交的线段集合，按纵坐标升序存储
$result \leftarrow$ False
for (x, y, l) in H **do** ▷ 按横坐标升序遍历，$O(n \log n)$
  if $l \notin Candidate$ **then** ▷ 不在集合中，左端点
    Add(x, y, l) ▷ 将 l 插入 $Candidate$ 中，$O(n)$
    return True if Intersect$(l, neighbor)$ ▷ 以 x 为键，判断新加线段与左右两线段是否相交，$O(1)$
  else ▷ 已在集合中，右端点
    Search$(Candidate, l)$
    return True if Intersect$(leftofl, rightofl)$ ▷ 找到 l 在 $Candidate$ 的位置，$O(n)$ 或 $O(\log n)$，判断其原左右邻居是否相交
    $Candidate$.Remove(l) ▷ 从 $Candidate$ 中移除 l，$O(n)$
  end if
end for

8. 用扫描线算法求解最近邻点对问题

解 使用扫描线算法，记录扫描线左侧最小点对的距离，将各点按横坐标升序进行遍历，动态移除距扫描线距离大于 d 的点并检查距离当前点最近的点（由课本图 8.14 可知，最坏情况只有 6 个满足条件的点）

伪代码如下

Algorithm 3 扫描线算法求最近点对

输入: $P = \{(x_i, y_i)\}$（点集合）
输出: p_1, p_2（距离最近的两个点）
对所有点以横坐标为键，建立最小化堆 H_x，以纵坐标为键，建立搜索树 T_y
$Q \leftarrow$ Queue( ) ▷ 新建空队列
$d \leftarrow +\infty, p_1 \leftarrow$ Null$, p_2 \leftarrow$ Null ▷ d 为扫面线左侧所有点对之间的最小距离
for (x, y) in H_x **do**
  while $(x', y') \leftarrow$ Queue.Head$(Q) \wedge |x' - x| > d$ **do** ▷ 此时队列中最左侧的点到扫描线的距离超过了 d
    Tree.Remove$((x', y'))$, Queue.Dequeue(Q) ▷ 从当前状态中移除该元素
  end while
  for (x', y') in H_y where $|y' - y| < d$ **do** ▷ 最多只有 6 个满足该条件的点
    if $\|(x, y) - (x', y')\| < d$ **then**
      $d \leftarrow \|(x, y) - (x', y')\|$
      $p_1 \leftarrow (x, y), p_2 \leftarrow (x', y')$、
    end if
  end for
  Tree.Insert$((x, y))$, Queue.Enqueue$((x, y))$
end for

9. 给定正整数 $a_1, a_2, ..., a_n$，代表 n 条线段（由点 (i, a_i) 和 $(i, 0)$ 构成，$i = 1, 2, ..., n$），从中找出两条线段，使之与 x 轴构成的容器能够包含尽可能多的水

解 使用双指针法。两线段之间形成的区域总是会受到其中较短那条长度的限制。

最开始，考虑由最外围两条线段构成的区域，记录由最外围两条线段构成的容器容量作为最大值。将指向其中较短线段的指针向中央移动，尽管这造成了矩形宽度的减小，但却可能会有助于面积的增大，移动后与当前状态的最大值进行比较、然后继续移动，以此类推直至左右指针指向同一条线段。

移动较短线段的指针会得到一条相对较长的线段，这可以克服由宽度减小而引起的面积减小。（如果我们试图将指向较长线段的指针向内侧移动，矩形区域的面积将受限于较短的线段而不会获得任何增加）