算法与复杂性　作业十

516021910528 - SHEN Jiamin

2020 年 4 月 26 日

1　0420

1. 设计算法判定平面上 n 个点是否在一条直线上

> 任选一点 $p_0(x_0, y_0)$ 作为基准点。对其他 $n-1$ 个点，计算其相对于点 p_0 的斜率。若所有的斜率都相等，则这 n 个点在同一条直线上。
>
> 该算法具有线性时间复杂度。

2. 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。

> 将该多边形表示为用邻接表存储的图 $G = (V, E)$，设有 n 个顶点 n 条边，图中每个顶点的度数均为 2。以 q 为坐标原点建立极坐标系，求出每个顶点的极坐标 (ρ, θ)，并按极角 θ 的大小排序。应用扫描线算法，事件点进度表按极角排序的所有顶点，扫描线状态为与射线 $\theta = \alpha$ 相交的边集。
>
> 见算法1

算法 1 最小相交边数量

输入: $G=(V,E)$（给定多边形 P），q（给定矩形 R 内的一点）

输出: γ （射线 $\theta=\gamma$ 与 P 相交的边数量最少） ▷ 在该射线上任取 R 外的一点即满足题设

以 q 为坐标原点，x 轴正方向为极轴，建立极坐标系，并求各顶点的极坐标 ▷ $O(n)$

$Q \leftarrow \textsc{MinHeap}(V,\theta)$ ▷ 对顶点按极角构建最小化堆，$O(n)$

$S \leftarrow \Phi, count \leftarrow 0$

for (v,w) in E **do** ▷ 找出多边形 P 与射线 $\theta=0$ 相交的所有边，$O(n)$

  if $\sin(v.\theta)\sin(w.\theta)<0 \wedge q.x < \frac{q.y-w.y}{v.y-w.y}(v.x-w.x)+w.x$ **then**

    ▷ 两点在极轴上下两侧，且与直线 $y=q.y$ 的交点在 $q.x$ 的右侧

    $S \leftarrow S \cup \{(v,w)\}, count \leftarrow count+1$

  end if

end for

$minCount \leftarrow count, \alpha \leftarrow 0, \beta \leftarrow 0$

while $p \leftarrow \textsc{Heap.Pop}(Q)$ **do** ▷ 扫描线：按极角升序遍历所有顶点，$O(n\log n)$

  $s,t \leftarrow \textsc{Adjacency}(G,p)$ ▷ 从邻接矩阵中找 p 的两个相邻顶点，$O(1)$

  if $(p,s)\in S \wedge (p,t)\in S$ **then** ▷ 扫描线越过顶点 p 后，两条边都不与扫描线相交

    $S \leftarrow S-\{(p,s),(p,t)\}$

    $count \leftarrow count-2$

    if $count < minCount$ **then**

      $minCount \leftarrow count$

      $\alpha \leftarrow p.\theta, \beta \leftarrow p.\theta$

    end if

  else if $(p,s)\in S \wedge (p,t)\notin S$ **then** ▷ 扫描线越过顶点 p 后，与另一条边相交

    $S \leftarrow S \cup \{(p,t)\} - \{(p,s)\}$

  else if $(p,s)\notin S \wedge (p,t)\in S$ **then**

    $S \leftarrow S \cup \{(p,s)\} - \{(p,t)\}$

  else if $(p,s)\notin S \wedge (p,t)\notin S$ **then** ▷ 扫描线越过顶点 p 后，两条边都与扫描线相交

    $S \leftarrow S \cup \{(p,s),(p,t)\}$

    $count \leftarrow count+2$

    $\beta \leftarrow p.\theta$ if $\alpha=\beta$ else β

  end if

end while

$\gamma \leftarrow \frac{\alpha+\beta}{2}$ if $\alpha \neq \beta$ else $\frac{\alpha+2\pi}{2}$

2 0423

3. 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。（不得穷举，文献查阅，然后用自己的语言进行算法思想的描述，包括时间复杂性分析）

Shamos M I. Computational geometry[Ph. D. Thesis][J]. 1978.

引理 最远点对一定在这组点的凸包上。

证明 反证法。假设最远点对为 (p,q)，其中 q 不在这组点的凸包上，则根据凸包的定义 q 一定在凸包的内部。对凸包多边形进行三角形分划，一定能找到

- 凸包上一点 x，使点 q 在线段 px 上。此时一定有 $d_{px} > d_{pq}$，即 pq 不是最远点对。
- 或凸包上的两点 x, y，使点 q 在 $\triangle pxy$ 的内部。

  因为三角形内角和为 $180°$，所以在 $\triangle qxy$ 中，$\angle xqy < 180°$，所以 $\angle pqx + \angle yqp > 180°$。不妨设 $\angle pqx \leq \angle yqp$，则有 $2\angle yqp > 180°$，即 $\angle yqp > 90°$。

  所以在 $\triangle yqp$ 中，$\angle yqp$ 是最大的角，$\angle yqp > \angle pyq$。由正弦定理，$d_{yp} > d_{pq}$，所以 pq 不是最远点对。

因此最远点对一定都在这组点的凸包上。 □

应用 Graham 扫描算法可以找到这组点的凸包，时间复杂度为 $O(n \log n)$

作凸多边形两条平行的支撑线，并沿逆时针方向同时旋转两条平行支撑线。则若凸包上两点是最远点对，一定存在某一时刻，使两点均在平行线上。因此在旋转的过程中求出能同时出现在两平行线上的点对之间的距离，并找到最大值即可。

因为两条支撑线将共同遍历全部的点一次，所以算法的时间复杂度为 $O(n)$。

总的时间复杂度为 $O(n \log n)$

4. 给定测度空间中位于同一平面的 n 个点，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。

   该问题的直观解法就是把 D 扫描一遍，选择其中最大的元素即可。由于是在一个测度空间中，因此 d_{ij} 满足距离的基本要求，即非负性、对称性和三角不等式。我们就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。那么时间代价减小的同时，证明解不会小于最优值的一半。

证明 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 共线，则有 $d_{xp} + d_{xq} = d_{pq}$
  - 若 x, p, q 不共线，则三点构成平面上的一个三角形，有 $d_{xp} + d_{xq} > d_{pq}$

  即 $d_{xp} + d_{xq} \geq d_{pq}$。不妨设 $d_{xp} \leq d_{xq} \leq d_{pq}$，

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

  设选定的这一行中最大的距离（即算法的输出）为 d_{xy}，则

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

5. 在平面上给定一个有 n 个点的集合 S，求 S 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。

算法 2 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, descending)          ▷ 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
                                    ▷ 该点被之前遍历过的某个纵坐标为 maxY 的点支配
    end if
end for
```