算法与复杂性　作业十一

516021910528 - SHEN Jiamin

2020 年 5 月 3 日

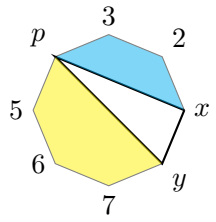
1　0426

1. 对凸多边形

   (a) 有多少种三角划分的方法?

任意一个三角形可以由一条边和不在这条边所在直线上的一个点构成。

凸多边形的任意一条边都一定属于划分后的一个三角形，任意一个顶点都一定是划分后三角形的一个顶点。在凸多边形中，确定一条边后，其他的顶点一定都不在这条边所在的直线上（否则该多边形一定不是凸多边形)。



所以凸 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 $T(n)$ 为一个 n 边形的三角形划分方法。则

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

   (b) 如何使对角线长度之和最小?

使用与上题相同的归纳方式。

在 $\triangle xpy$ 是三角形划分后结果中的一个三角形的条件下, 当多边形对角线长度之和最小时, 黄色和蓝色两个凸多边形一定达到了使其对角线长度之和分别最小的分划。

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 $Q\left(\langle p_1, p_2, \ldots, p_n\rangle\right)$。

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

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

由于子问题重叠，考虑使用动态规划求解。共有 $O(n^2)$ 个子问题需要求解，求解每个子问题所需的时间复杂度为 $O(n)$。总的时间复杂度为 $O(n^3)$

伪代码见算法1

2. 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。

使用扫描线算法，事件列表为线段的所有端点及所有交点按横坐标升序，当遇到第一个交点时算法结束。交点出现时，相交的两线段一定是相邻的。

该算法的时间复杂度为 $O((2n)\log(2n)) = O(n \log n)$。

伪代码见算法2

3. 用扫描线算法求解最近邻点对问题

事件为所有的点，按横坐标升序遍历。扫描线状态为已经扫描过且到扫描线的距离小于某个值所有点。

伪代码见算法3

4. 有 n 种液体 $S_1, S_2, \ldots, S_n$, 都含有 A,B 两种成分, 含量分别为 $(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)$, $a_i + b_i < 100\%$。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。

即找到一组数 $r_i : 0 \le r_i \le 1$ 满足 $\begin{cases} \sum_{i=1}^{n} r_i = 1 \\ \sum_{i=1}^{n} r_i a_i = x \\ \sum_{i=1}^{n} r_i b_i = y \end{cases}$

即求解非齐次线性方程组

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ a_1 & a_2 & \cdots & a_n \\ b_1 & b_2 & \cdots & b_n \end{bmatrix} \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{pmatrix} = \begin{bmatrix} 1 \\ x \\ y \end{bmatrix}$$

有增广矩阵

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{b} \end{bmatrix} = \left[\begin{array}{cccc|c} 1 & 1 & \cdots & 1 & 1 \\ a_1 & a_2 & \cdots & a_n & x \\ b_1 & b_2 & \cdots & b_n & y \end{array}\right]_{3\times n+1} = \begin{bmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \\ \boldsymbol{r}_3 \end{bmatrix}$$

对 $\begin{bmatrix} \boldsymbol{A} & \boldsymbol{b} \end{bmatrix}$ 进行行变换。行变换 $\boldsymbol{r}_i \leftarrow \alpha \boldsymbol{r}_i + \beta \boldsymbol{r}_j$ 的时间复杂度为 $O(n)$，经过最多 3 次行变换，然遍历三行即可求出 $\boldsymbol{A}$ 的秩 $r(A)$。

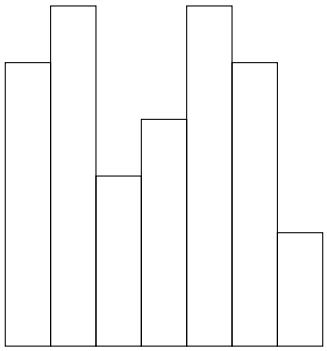
设变换后的增广矩阵为 $\begin{bmatrix} \boldsymbol{R} & \boldsymbol{d} \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{r}_3 \end{bmatrix}^T$

- 若 $r(A) = 3$，则再经过最多三次行变换即可使 $\boldsymbol{A}$ 中的三列构成三阶单位阵，进而容易求得线性无关的通解 $\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3$ 和特解 $\boldsymbol{\eta}$。因为三组通解是线性无关的，可以作为 $\mathbb{R}^3$ 的一组基，所以一定存在 $\boldsymbol{r} = \boldsymbol{\eta} + \sum_{i=1}^{3} \boldsymbol{\alpha}_i s_i$ 满足要求。
- 若 $r(A) = 2$，则方程有解当且仅当 $d_{3,1} = 0$
- 若 $r(A) = 1$，则方程有解当且仅当 $d_{3,1} = d_{2,1} = 0$

因此判别过程的时间复杂度不超过 $6n$ 即为 $O(n)$。

2 0427

5. 海报墙由 n 块宽度相同高度不同的木板组成，那么在此海报墙上能够张贴的最大海报面积是多少？设木板宽度为 1，高度为 $h_1, h_2, \ldots, h_n$，海报必须整体都粘贴在墙上，并且不能斜贴。



应用扫描线算法，事件点序列为木板高度升序。

$$S(m,n)=\begin{cases}\max\begin{cases}(n-m+1)\cdot\min_{m\le i\le n}h_i\\ S(m,\operatorname{argmin}_{m\le i\le n}h_i-1) & m<n\\ S(\operatorname{argmin}_{m\le i\le n}h_i+1,n)\end{cases}\\ h_m & m=n\\ 0 & m>n\end{cases}$$

子问题互不重叠，不需要使用动态规划。

伪代码见算法4

6. 平面有两组点，如何证明存在直线可以将这两组点分开?

1. 对两组点分别求凸包，$O(\log n)$
2. 判断两个凸包是否相交，$O(n)$

算法 1 最小对角线和

输入: $P = \langle p_i(x_i, y_i) \rangle_{1 \le i \le n}$（要求解的多边形的顶点序列）

输出: $S = \{\langle p_i, p_j \rangle\}$（长度和最小的对角线集合）

$Dist \leftarrow \{\mathsf{Null}\}_{n \times n}, Q \leftarrow \{\mathsf{Null}\}_{n \times n}$

$(minsum, S) \leftarrow \textsc{MinimizeDiagonal}(1, n)$

procedure $\textsc{Distance}(i, j)$ ▷ 获取对角线 $p_i - p_j$ 的长度

  if $Dist[i][j]$ is Null **then**

    $distance \leftarrow \|(x_i, y_i) - (x_j, y_j)\|$ if $|i - j| > 1$ else 0

    $Dist[i][j] \leftarrow distance, Dist[j][i] \leftarrow distance$

  end if

  return $Dist[i][j]$

end procedure

procedure $\textsc{MinimizeDiagonal}(m, n)$

▷ 求使多边形 $\langle p_m, \ldots, p_n \rangle$ 对角线长度和最短的对角线集合及其长度和

  if $Q[m][n]$ is Null **then**

    if $n - m + 1 \le 3$ **then**

      $Q[m][n] \leftarrow (0, \Phi)$

    else

      $min \leftarrow +\infty, Diag \leftarrow \Phi$

      for $i \leftarrow m + 1$ to $n - 1$ **do**

        $(sum_1, D_1) \leftarrow \textsc{MinimizeDiagonal}(m, i)$

        $(sum_2, D_2) \leftarrow \textsc{MinimizeDiagonal}(i, n)$

        $d_1 \leftarrow \textsc{Distance}(m, i), d_2 \leftarrow \textsc{Distance}(i, n)$

        $sum \leftarrow sum_1 + sum_2 + d_1 + d_2$

        if $sum < min$ **then**

          $min \leftarrow sum, Diag \leftarrow D_1 \cup D_2$

          $Diag \leftarrow Diag \cup \{(p_1, p_i)\}$ if $d_1 > 0$

          $Diag \leftarrow Diag \cup \{(p_i, p_n)\}$ if $d_2 > 0$

        end if

      end for

      $Q[m][n] \leftarrow (min, Diag)$

    end if

  end if

  return $Q[m][n]$

end procedure

算法 2 线段交点存在性判断

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 ▷ $O(n)$
$Segments \leftarrow$ Vector( )
$result \leftarrow$ Null
for (x, y, l) in H **do** ▷ 按横坐标升序遍历，$O(n \log n$
  if $l \notin Segments$ **then**
    $index \leftarrow$ Push(x, y, l) ▷ 将 l 插入 $Segments$ 中，使在 x 处纵坐标升序，$O(n)$
    return True if Intersect$(l, Segments[index - 1])$ ▷ 判断两线段是否相交，$O(1)$
    return True if Intersect$(l, Segments[index + 1])$
  else
    $index \leftarrow$ Vector.search$(Segments, l)$ ▷ 找到 l 在 $Segments$ 的索引，$O(n)$ 或 $O(\log n)$
    return True if Intersect$(Segments[index - 1], Segments[index + 1])$
    Remove(l) ▷ 从 $Segments$ 中移除 l，$O(n)$
  end if
end for

算法 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
$H_y \leftarrow$ RedBlackTree(key $= y$), $Q \leftarrow$ Queue( )
$d \leftarrow +\infty, p_1 \leftarrow$ 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.Top$(Q) \wedge |x' - x| > d$ **do** ▷ 最左侧的点到扫描线的距离超过了 d
    RBTree.Remove$((x', y'))$, Queue.Pop(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
  RBTree.Insert$((x, y))$, Queue.Push$((x, y))$
end for

算法 4 最大内接矩形

输入: $H[1 \dots n]$（n 块木板的高度）

输出: $MaxArea$（最大面积）

$I \leftarrow \textsc{SortedList}(H, \mathsf{key} = H[i], \mathsf{value} = i)$ ▷ 存储的是板子在 H 中的索引，只有顺序查找，用链表
$MaxArea \leftarrow -1$
$Q \leftarrow \textsc{Queue}(\ )$
$\textsc{Queue.Push}(Q, (1, n))$
while $(x, y) \leftarrow \textsc{Queue.Pop}(Q)$ **do**
    if $y > x$ **then**
        $idx \leftarrow \textsc{List.findFirst}(I, [x, y])$ ▷ 找到 $[x, y]$ 区间内最低的板子 idx
        $area \leftarrow H[idx] \cdot (y - x + 1)$ ▷ 算面积
        $MaxArea \leftarrow area$ if $MaxArea < area$
        $\textsc{Queue.Push}(Q, (x, idx - 1))$ ▷ 添加两个字问题
        $\textsc{Queue.Push}(Q, (idx + 1, y))$
        $\textsc{List.Remove}(I, idx)$ ▷ 移除这块板子
    else if $y = x$ **then**
        $area \leftarrow H[x]$ ▷ 算面积
        $MaxArea \leftarrow area$ if $MaxArea < area$
        $\textsc{List.Remove}(I, x)$ ▷ 移除这块板子
    end if
end while