西北工业大学

课程大作业答题册

学　　号: 2015300005

姓　　名: 黄晓阳

课　　程: 计算几何算法与应用

得　　分:

日　　期: 2018.11.10

西北工业大学

作业名称：	计算凸包的分治式算法

基本策略及其原理（30分）：

1.凸包问题概述

如下图所示，在给定点集中选取最少量的点，作一多边形，使所有点都包含在构造的凸多边形中的问题即凸包问题。

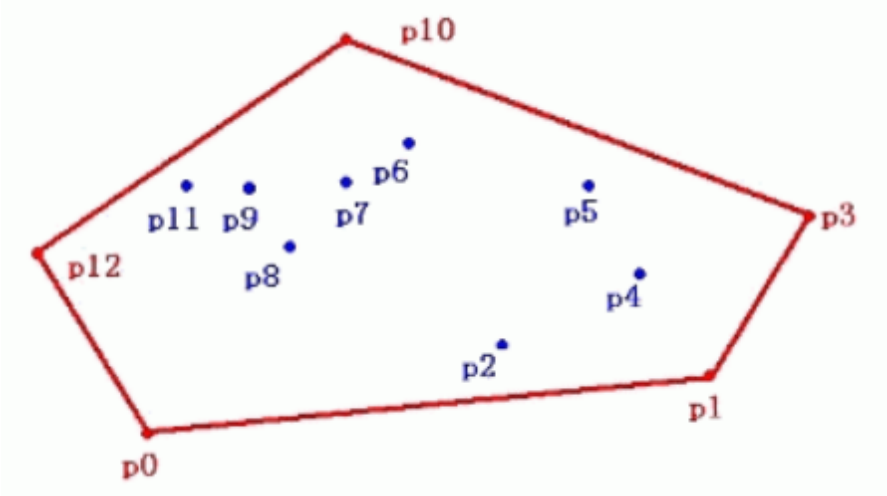

图 1: 凸包范例

2.分治法的基本原理

通过建立递归方程的方式，将整个问题划分为多个无关联的子问题，这些子问题相互独立且与原问题一致。分而治之，递归的求解这些子问题并将这些子问题的解进行合并就能得到最终的原问题的解。

3.用分治法求解凸包问题策略

将分治法的思想应用于凸包问题，我们将得到如下的求解方法。首先将点集映射到一个二维坐标系中，如下图所示。

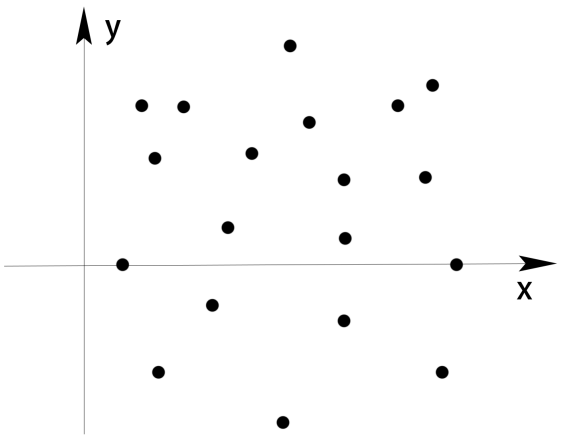

图 2: 点集的映射

由于点集中的所有点都在凸包内部或者是凸包的顶点，因此我们可以知道x方向或者y方向的最大与最小值点都是凸包的顶点。因此我们可以从x方向开始，用最大x值点P_1与最小x值点P_2将点集进行分割，分割成上半部分与下半部分，如下图所示。

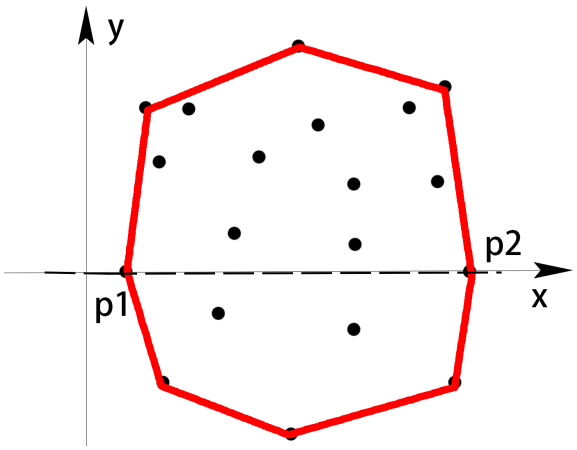

图 3: 第一次分割

用同样的思想，我们可以确定知道上半部分中距离分割线最远的点P_3与下半部分距离分割线最远的点P_4也一定是凸包上的点，因此我们可以通过连接$P_1\ P_3$，$P_1\ P_4$，$P_2\ P_3$，$P_2\ P_4$四条线段将点集进一步进行分割，如下图所示。

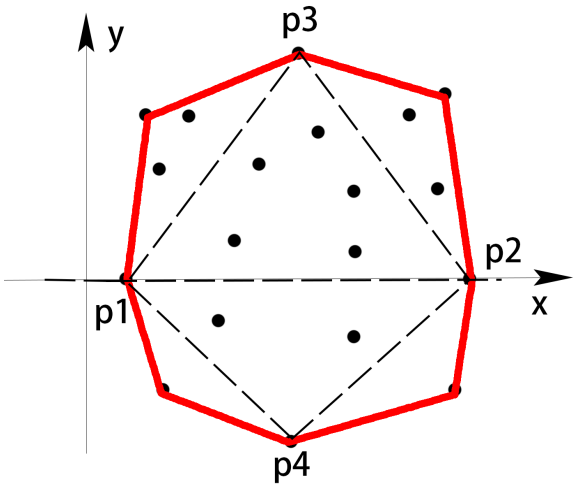

图 4: 第二次分割

由于$P_1\ P_2\ P_3$围成的上三角以及$P_1\ P_2\ P_4$围城的下三角内的点既然可以被已经选择的点包围起来那么，这些点一定不会是凸包上的点，因此我们可以再次分割的时候对其不再进行考虑。进而只对虚线分割包围三角形外的点进行枚举，递归求P_{max}，最终将凸包中的所有点找出来。

算法伪代码（30分）：

算法 CONVEXHULL(P)
输入： 平面点集P
输出： 由CH(P)的所有顶点沿顺时针方向组成的一个列表

```
1.   根据x-坐标，对所有点进行排序，得到序列P1,...,Pn
2.   在l_upper中加入P1 Pn，在l_downner中加入Pn P1
3.   将P1 Pn连线上方的点计入U_pper，将P1 Pn连线下方的点计入D_ownner
4.   Recursion(S[ ],P1,P2,l[ ])//参数表示划分点集、分割线端点端点、存储的端点组
5.   {
6.     IF S[ ]==null THEN
7.       Return;
8.     ELSE
9.       P3←Max(S[ ],P1,P2)//求划分点集内距分割线最远的点
10.      AddPoint(l[ ],P1,P2,P3)//根据P1,P2的位置将P3插入两者之间
11.      A[ ]←Get(S[ ],P1,P3);//求划分点集中被P1,P3分割出的点集
12.      B[ ]←Get(S[ ],P2,P3);//求划分点集中被P2,P3分割出的点集
13.      Recursion(A[ ],P1,P3,l_upper);
14.      Recursion(B[ ],P3,P2,l_upper);
15.      Return;
16.    END
17.  }
18.  Recursion(U_pper,P1,Pn,l_upper);
19.  Recursion(D_ownner,Pn,P1,l_downner);
20.  最后将l_downner与l_upper相连接得到顺时针方向组成的凸包顶点集合
```

退化情况的考虑和处理（20分）：

可能出现的退化情况有三点。

第一个问题是当点存在于分割线上的时候该点应属于哪一部分的问题，对于这个问题，我们可以考虑不管属于哪一部分这个点都不可能是凸包上的端点，因此在具体算法中可以将该点分到两部分的任意一部分均可。

第二个问题是多个点到分割线的距离相同时的选取问题，这时候从算法本身考虑，选取其中任意一个点都能够将其他点在最后筛选出来，但是处理方式的不同会带来算法的平均时间复杂度不同。由于计算机本身有计算误差，因此这种可能性比较小，因此以下两种处理方式都较优，第一种是选取这些点中从分割线一端到另一端的中间位置的点，这样递归深度会比选取处于两端的点的递归深度低；第二种处理方式是首先将所有点安顺序插入凸包端点序列，再分别将这些点两端的点与对应较近的分割线两端的点带入下一步的递归，这样虽然处理起来看似比较麻烦，但是也消除了后续对这些距离相同点的筛选时的时间消耗问题。

第三个可能的问题是当点集的总数小于或等于三时，这时候为了避免出现内存溢出或者是空指针调用的情况，可以对其进行提前的判断，然后一步到位的返回对应的结果。

算法时间复杂度分析（20分）：

分治算法的时间复杂度可以用以下的式子来进行表示：

$$T(n) = \begin{cases} O(1) & n = 1 \\ aT(n/b) + f(n) & n > 1 \end{cases}$$

因此采用递推求解法就能够通过写出的递推式求出结果：

$$\begin{aligned} T(n) &= 2T(n/2) + O(n) \\ &= 2(2T(n/4) + O(n/2)) + O(n) \\ &= 4T(n/4) + 2O(n) \\ &= 8T(n/8) + 3O(n) \\ &= nT(1) + log_2 n\ O(n) \\ &= nO(1) + log_2 n\ O(n) \end{aligned}$$

因此使用分治算法最终的时间复杂度为$O(nlogn)$.