算法设计与分析

第2章 递归与分治策略

(4)

最接近点对问题

- 空中交通控制系统应用
  - 具有最大碰撞危险的两架飞机是空间中最接近的一对点
  - 马航MH370失联、MH17被击坠毁
- 温州动车碰撞事件

最接近点对问题

- 图片比较问题
  - 假设图片以像素点矩阵方式存储。对于大小为28*28的图片，共有784个像素点，每个像素点取值在0-255之间（灰度图）。假设班里有72名同学，给每名同学拍一幅28x28的头像，将头像图片存储于计算机内。
  - 请设计算法来自动确定进来某名同学是否是班里学生
    - 为该同学拍一张28x28的头像 I
    - 将I和存储的72位同学的头像比较
    - 找出最接近的一个头像C
    - 判断C和I的距离是否小于一定范围，是则认定为I=C，否则I！=C
- 问题推广：
  - 每幅图片包含784个数据，可以看作784维空间中的一个点
  - 头像比较问题转化为：给定空间的n个点，找出这些点中欧式距离最小的一对点

最接近点对问题

- 问题：给定平面上n个点的集合S，找其中的一对点，使得在n个点组成的所有点对中，该点对间的距离最小。
- 分析
  - 最近指欧几里得距离,即

    $s1=(x_1,y_1)$, $s2=(x_2,y_2)$

    Distance=

$$\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$$

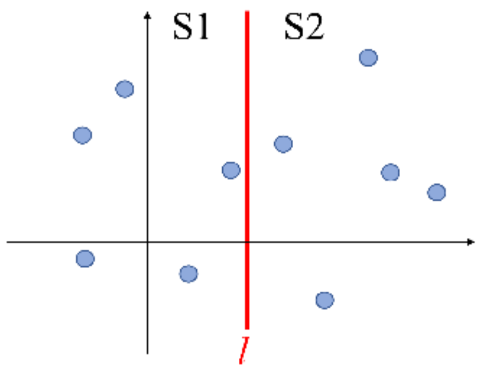
  - 蛮力算法
    - 简单地检验所有可能的n(n-1)/2个距离并返回最短间距的点对
    - **$\Theta(n^2)$**

最接近点对问题

- 分治法所能解决的问题一般具有以下几个特征：
  - 可解性：该问题的规模缩小到一定的程度就可以**容易地解决**；
  - 递归性：该问题可以分解为若干个规模**较小的相同问题**，即该问题具有最优子结构性质
  - 合并性：利用该问题分解出的子问题的解可以**合并**为该问题的解；
  - 独立性：该问题所分解出的各个子问题是相互独立的，即子问题之间不包含公共的子问题。

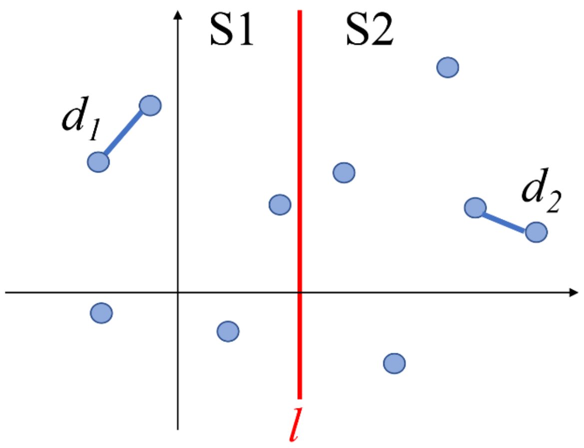
最接近点对问题

- 输入: 点集合S, 数组X和Y, X中点按其x坐标单调递增序排序,Y中点按其y坐标单调递增序排列.
- 输出: S中具有最小距离的两点及其距离
- 分治算法
  - 分解(Divide):找出一条垂直线l,把点集S划分为两个集合S_1和S_2.S_1中的所有点在线l上或左侧,S_2中的所有点在线l上或右侧. 数组X(Y)划分为两个数组X1,X2(Y1,Y2),分别包含S1和S2中的点.



最接近点对问题

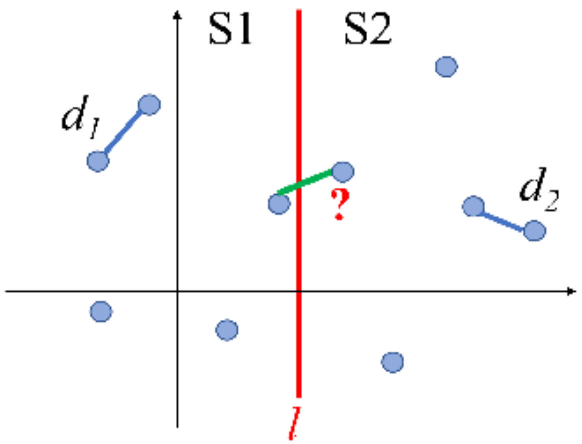
- 解决(Conquer)
  - 进行两次**递归调用**,一次找出S_1中的最近点对,一次找出S_2中的最近点对.第1次调用的输入为点集S_l,数组X_1和Y_1，第2次调用的输入为点集S_2,数组X_2和Y_2.
  - 设对S_1和S_2返回的最近点对距离为d_1 和d_2, 令$d=\min(d_1 ,d_2)$

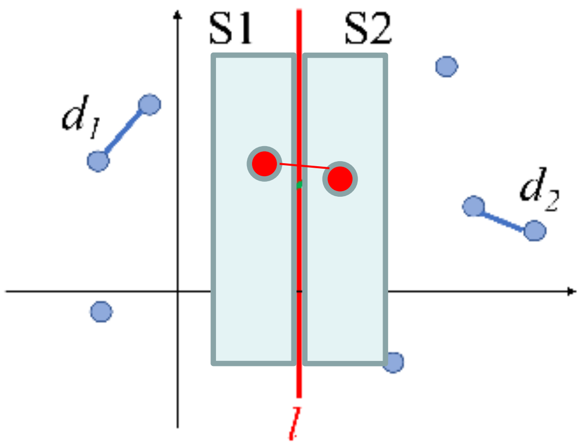


最接近点对问题

- 合并(Merge)
  - 最近点对要么是某次递归调用找出的距离为d的点对
  - 要么是S_1中的一个点与S_2中的一个点组成的点对,算法确定是否存在其距离小于d的一个点对.
    - 若存在,则点对中的两个点必定都在距离直线l的d单位之内

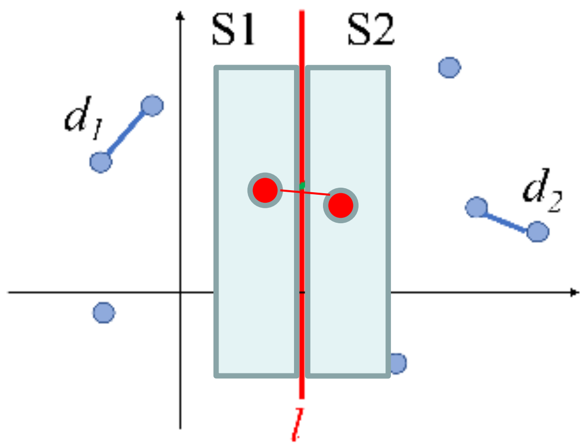
$d=\min(d_1, d_2)$





最接近点对问题

- 输入:S
- 输出:最接近点对(p,q) 及其距离
- 设计
  - 分: S 分为S1 和S2
  - 解：
    - 递归调用算法分别对S1和S2两个子问题进行求解
    - d = min(d1,d2)
  - 合：
    - min(d,d’),
    - d’ 是点对(p3,q3)距离，p3在S1， q3在 S2



特点：重在合，怎么合？ **(p3, q3)?**

最接近点对问题

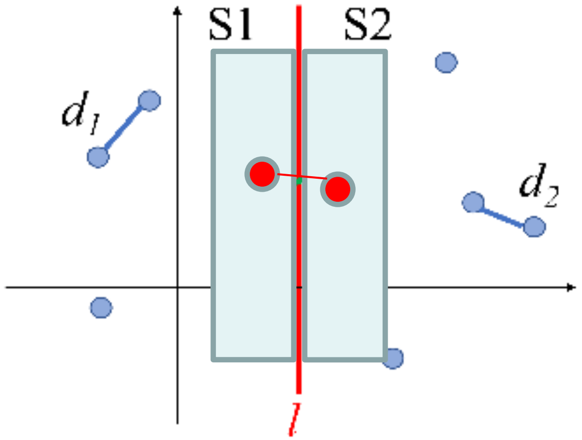
- 输入:S
- 输出:最接近点对(p,q) 及其距离
- 设计
  - 分: S 分为S1 和S2
  - 解：
    - 递归调用算法分别对S1和S2两个子问题进行求解
    - d = min(d1,d2)
  - 合：
    - min(d,d’),
    - d’ 是点对(p3,q3)距离，p3在S1， q3在 S2

T(n)

O(1)

2T(n/2)

O(n)? O(n^2)?



特点：重在合，怎么合？ **(p3, q3)?**

最接近点对问题

- 分析
  - 查找一个点在S1、一个点在S2中的最接近点对的工作量是(n/2)*(n/2)=O(n^2)

- $T(n)=2T(n/2)+\underline{O(n^2)}$

$\Rightarrow T(n)=O(n^2)$

？效率怎么改进

最接近点对问题

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

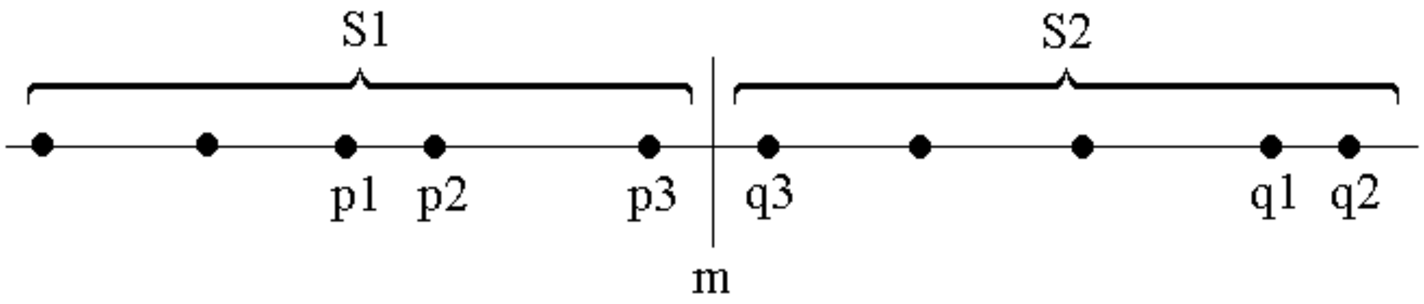
$$T(n)=n^{\log_m k}+\sum_{j=0}^{\log_m n-1} k^j f(n/m^j)$$

- T(n)=2T(n/2)+O(n²)

  T(n)=O(n²)

最接近点对问题

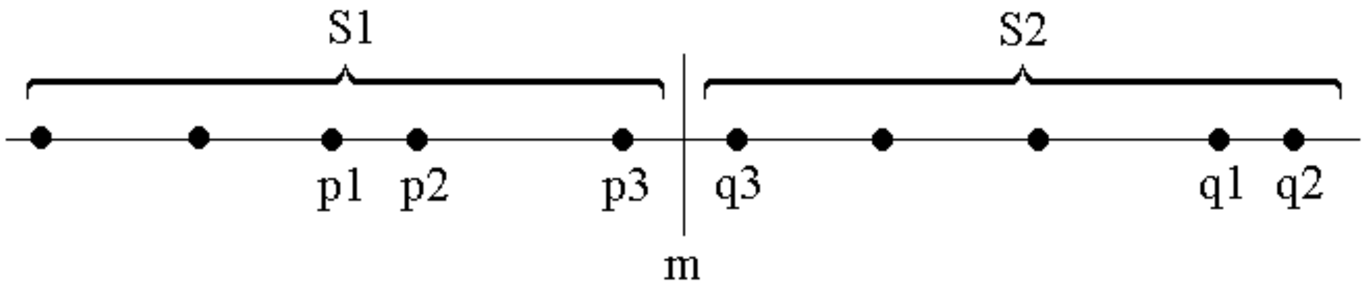
- 为了使问题易于理解和分析，先来考虑一维的情形。此时，S中的n个点退化为x轴上的n个实数 $x_1,x_2,\ldots,x_n$。最接近点对即为这n个实数中相差最小的2个实数。



最接近点对问题

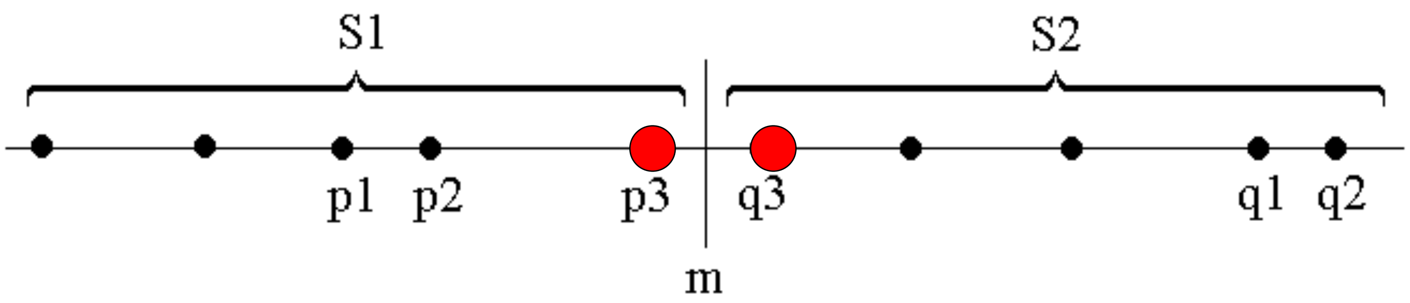
- 为了使问题易于理解和分析，先来考虑一维的情形。此时，S中的n个点退化为x轴上的n个实数 x1,x2,…,xn。最接近点对即为这n个实数中相差最小的2个实数。

> ➢假设我们用x轴上某个点m将S划分为2个子集S1和S2 ，基于**平衡子问题**的思想，用S中各点坐标的中位数来作分割点。
>
> ➢递归地在S1和S2上找出其最接近点对{p1,p2}和{q1,q2}，并设**d=min{|p1-p2|,|q1-q2|}**，S中的最接近点对或者是{p1,p2}，或者是{q1,q2}，或者是某个{p3,q3}，其中p3∈S1且q3∈S2。
>
> ➢**能否在线性时间内找到p3,q3？**



最接近点对问题

能否在线性时间内找到p3,q3？



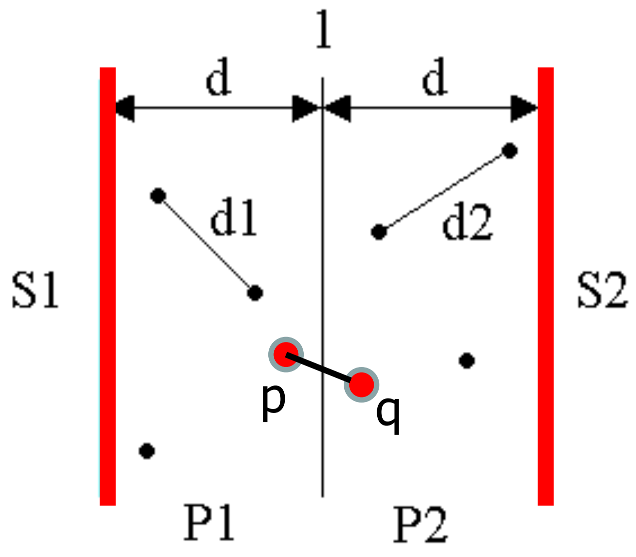
❖如果S的最接近点对是{p3,q3}，即|p3-q3|<d，则p3和q3两者与m的距离不超过d，即**p3∈(m-d,m]，q3∈(m,m+d]**。

❖由于在S1中，每个长度为d的半闭区间至多包含一个点（否则必有两点距离小于d），并且m是S1和S2的分割点，因此(m-d,m]中**至多**包含S中的一个点。由图可以看出，**如果(m-d,m]中有S中的点，则此点就是S1中最大点。**

❖因此，我们用线性时间就能找到区间(m-d,m]和(m,m+d]中所有点，即p3和q3。**从而我们用线性时间就可以将S1的解和S2的解合并成为S的解。**

最接近点对问题

- 下面来考虑二维的情形

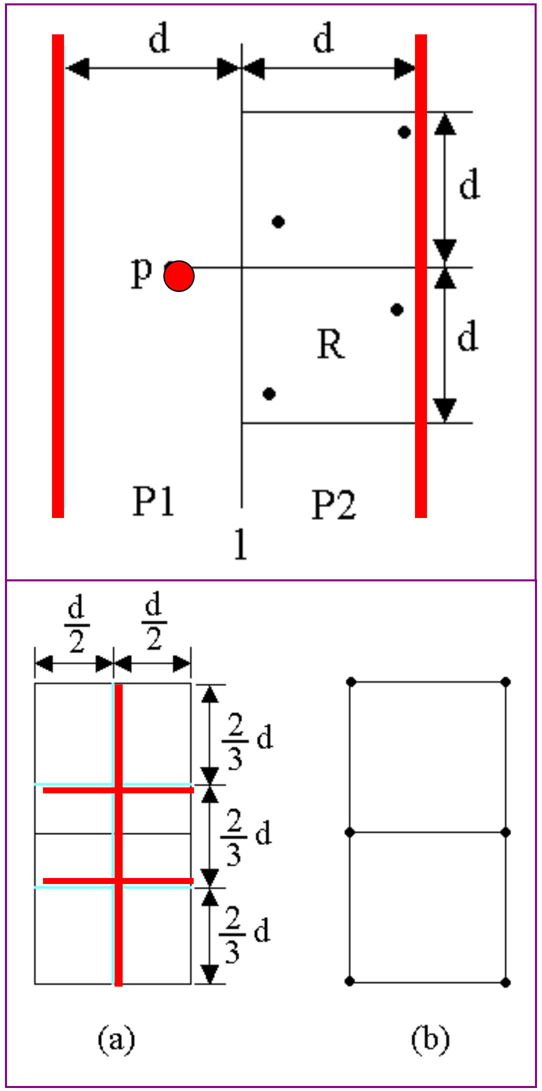


➢选取一垂直线**l:x=m**来作为分割直线。其中m为S中各点x坐标的中位数。由此将S分割为S1和S2。

➢递归地在S1和S2上找出其最小距离d1和d2，并设**d=min{d1,d2}**，S中的最接近点对或者是d，或者是某个{p,q}，其中p∈P1且q∈P2。

➢**能否在线性时间内找到p,q？**

最接近点对问题



能否在线性时间内找到p3,q3？

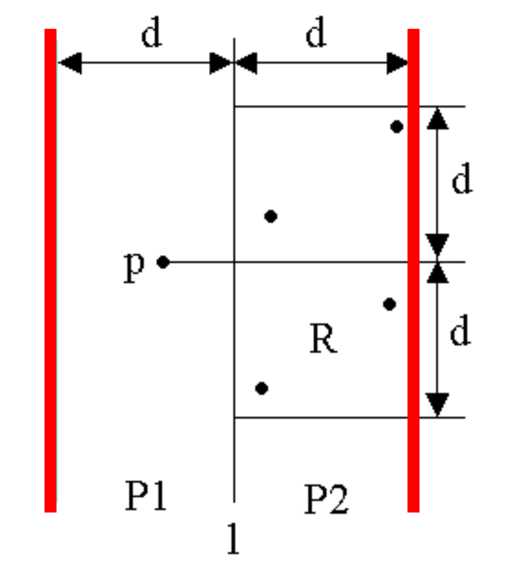
- 考虑P1中任意一点p，它若与P2中的点q构成最接近点对的候选者，则必有distance(p，q) < d。**满足这个条件的P2中的点一定落在一个d×2d的矩形R中**
- 由d的意义可知，P2中任何2个S中的点的距离都不小于d。由此可以推出**矩形R中最多只有6个S中的点。**
- 因此，在分治法的合并步骤中**最多只需要检查6×n/2=3n个候选者**

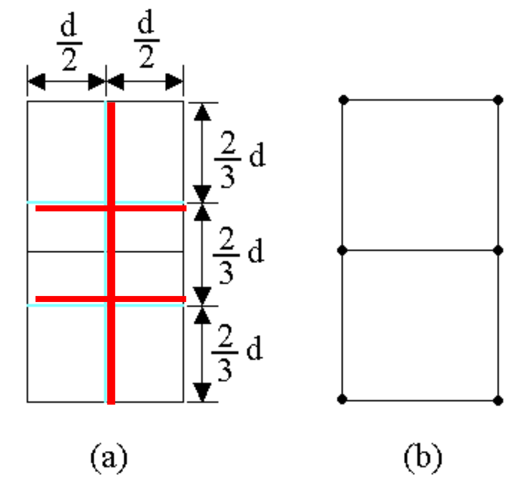
最接近点对问题

- 鸽巢定理
  - 若有n个鸽子巢,n+1只鸽子,则至少有一个鸽子巢里至少有两只鸽子.

- 例:　1年365天,今有366人,则至少有两个人生日相同

最接近点对问题





证明:将矩形R的长为2d的边3等分，将它的长为d的边2等分，由此导出6个(d/2)×(2d/3)的矩形。若矩形R中有多于6个S中的点，则由鸽舍原理易知至少有一个(d/2)×(2d/3)的小矩形中有2个以上S中的点。设u，v是位于同一小矩形中的2个点，则

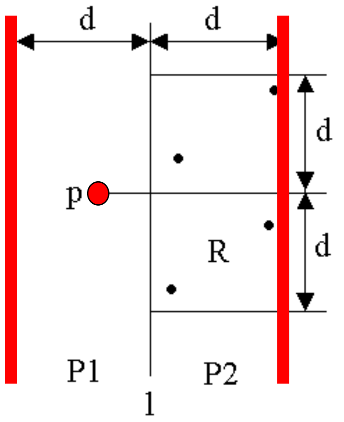
$$(x(u)-x(v))^2+(y(u)-y(v))^2 \le (d/2)^2+(2d/3)^2=\frac{25}{36}d^2$$

distance(u,v)<d。这与d的意义相矛盾。

$d=\min(d_1, d_2)$

最接近点对问题

- 为了确切地知道要检查哪6个点，可以将p和P2中所有S2的点投影到垂直线l上
  - 由于能与p点一起构成最接近点对候选者的S2中点一定在矩形R中，所以它们在直线l上的投影点距p在l上投影点的距离小于d
  - 由上面的分析可知，这种投影点最多只有6个
- 因此，若将P1和P2中所有S中点按其y坐标排好序，则对P1中所有点，对排好序的点列作一次扫描，就可以找出所有最接近点对的候选者。对P1中每一点最多只要检查P2中排好序的相继6个点



最接近点对问题

预排序

double **cpair2**(S)

{

n=|S|;

if (n < 2) **return** -1;

1、m=S中各点x间坐标的中位数; O(n)

构造S1和S2；

//S1={p∈S|x(p)<=m},

S2={p∈S|x(p)>m}

2、d1=**cpair2**(S1); 2T(n/2)

d2=**cpair2**(S2);

3、dm=**min**(d1,d2); O(1)

4、设P1是S1中距垂直分割线l的距离在d之内的所有点组成的集合；

P2是S2中距分割线l的距离在d之内所有点组成的集合；

将P1和P2中点依其y坐标值排序； O(nlogn)

并设X和Y是相应的已排好序的点列；

5、通过扫描X以及对于X中每个点检查Y中与其距离在d之内的所有点(最多6个)可以完成合并； O(n)

当X中的扫描指针逐次向上移动时，Y中的扫描指针可在宽为2d的区间内移动；

设dl是按这种扫描方式找到的点对间的最小距离；

6、d=**min**(dm,dl); O(1)

return d;

}

最接近点对问题

double **cpair2**(S)

{

n=|S

if (n

1、m=S

构造

//S1={p∈S|x(p)<=m},

S2={p∈S|x(p)>m}

2、d1=**cpair2**(S1);

d2=**cpair2**(S2);

3、dm=**min**(d1,d2);

4、设P1是S1中距垂直分割线l的距离在dm之内的所有点组成的集合；

P2是S2中距分割线l的距离在dm之内所有

；

Y中与

以完成

合并；

当X中的扫描指针逐次向上移动时，Y中的扫描指针可在宽为2dm的区间内移动；

设dl是按这种扫描方式找到的点对间的最小距离；

6、d=**min**(dm,dl);

return d;

}

复杂度分析

$$T(n)=\begin{cases} O(1) & n<4 \\ 2T(n/2)+O(n) & n\geq 4 \end{cases}$$

T(n)=O(nlogn)

渐近意义下的最优算法

最接近点对问题小结

- 分治法求解最接近点对问题
  - 问题分析过程（降维，理论到实践）
  - 问题求解算法的三步骤（分、解、合）
  - 问题的复杂度分析

  - 为提高效率，对算法合并部分进行了优化: 从O(n^2)⇒O(n)

分治策略总结

- 分：分割成p个子问题。子问题间相互独立(才能分别求解)。
  - 问题：应该把原问题分割成多少个子问题才合适？子问题规模是否应该相同？
  - 答：需具体问题具体分析。一般来讲，均匀获得较好效果。
- 解：如果子问题大于预定阈值m，则递归应用算法求解，否则直接对子问题求解。
  - 问题：一般递归代价较大，故而有阈值m，那m如何确定？
  - 答：具体问题具体分析。严格数学分析，经验等。
- 合并：具体问题具体分析

小结

- 重点：
  - 递归的概念。
  - 设计有效算法的分治策略，分治算法框架
  - 经典问题案例分析
- 难点
  - 分治算法框架
  - 递归方程式的获取

本章总结

- 理解递归的概念
- 掌握设计有效算法的分治策略，分治算法框架
- 通过下面的范例学习分治策略设计与分析技巧
  - 二分搜索技术
  - 大整数乘法
  - Strassen矩阵乘法
  - 合并排序和快速排序
  - 线性时间选择
  - 最接近点对问题