《算法设计与分析》

第三章 分治法

马丙鹏

2024年10月14日

中国科学院大学
University of Chinese Academy of Sciences

第三章 分治法

- 3.1 一般方法
- 3.2 二分检索
- 3.3 找最大和最小元素
- 3.4 归并排序
- 3.5 快速排序
- 3.6 选择问题
- 3.7 最接近点对问题
- 3.8 斯特拉森矩阵乘法
- 3.9 大整数乘法

3.7 最接近点对问题

- 问题描述
  - 在应用中，常用诸如点、圆等简单的几何对象来代表现实世界中的实体。在涉及这些几何对象的问题中，常需要了解其邻域中其他几何对象的信息。
  - 例如，在空中交通控制问题中，若将飞机作为空间中移动的一个点来看，则具有最大碰撞危险的两架飞机，就是这个空间中最接近的一对点。
  - 这类问题是计算几何学中研究的基本问题之一。我们着重考虑平面上的最接近点对问题。

3.7 最接近点对问题

- 问题描述
  - 给定平面上n个点，找其中的一对点，使得在n个点所组成的所有点对中，该点对间的距离最小。
  - 说明：
    - 严格来讲，最接近点对可能多于一对，为简便起见，我们只找其中的一对作为问题的解。

3.7 最接近点对问题

- 蛮力算法
  - 将每一个点与其他n-1个点的距离算出，找出最小距离的点对即可。
  - 该方法的时间复杂性是$T(n)=n(n-1)/2+n=O(n^2)$，效率较低。
  - 已经证明，该算法的计算时间下界是$\Omega(n\log n)$。

3.7 最接近点对问题

■ 蛮力算法

算法 BruteForceClosestPoints(P)

//输入:一个n(n≥2)个点的列表P, $P_1=(x1, y1)$ …., $P_n=(x_n, y_n)$

//输出:两个最近点的下标,index1和index2

dmin=∞

for i= 1 to n-1 do

    for j= i+1 to n do

        d=sqrt($(x_i-x_j)^2 + (y_i-y_j)^2$)

        if d<dmin {

            dmin=d; index1=i; index2=j; }

return index1, index2

总结:此算法的时间复杂度是$O(n^2)$

计算距离：n(n-1)/2次

if 语句最多执行n次

n(n-1)/2+n

3.7 最接近点对问题

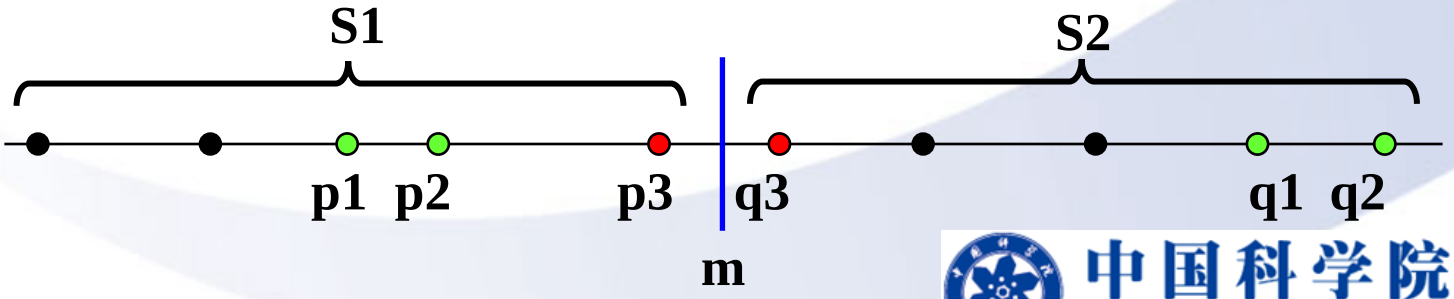
- 一维空间找最接近点对
  - 怎么样在一条线上找最邻近的点对?
    - ① 用O(nlogn)时间对它们排序。
    - ② 对排好序的表，计算每一个点到跟在它后面的点的距离，容易看出这些距离的最小值。时间复杂性为：n-1
    - ➢ 故T (n) = O(nlogn) + (n-1) = O(nlogn)
  - 显然，这种方法不能推广到二维的情形。
  - 故尝试用分治法来求解，并希望推广到二维的情形。

3.7 最接近点对问题

■ 一维空间找最接近点对

□分治策略下一维的情形

- 先把$x_1, x_2, \ldots, x_n$排好序，再进行一次线性扫描就可以找出最接近点对，T(n)=O(nlogn)。
- 假设用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$\{p_1, p_2\}$和$\{q_1, q_2\}$，并设$d=\min\{|p_1-p_2|, |q_1-q_2|\}$，S中的最接近点对或者是$\{p_1, p_2\}$，或者是$\{q_1, q_2\}$，或者是某个$\{p_3, q_3\}$，其中$p_3 \in S1$且$q_3 \in S2$。



3.7 最接近点对问题

■ 一维空间找最接近点对

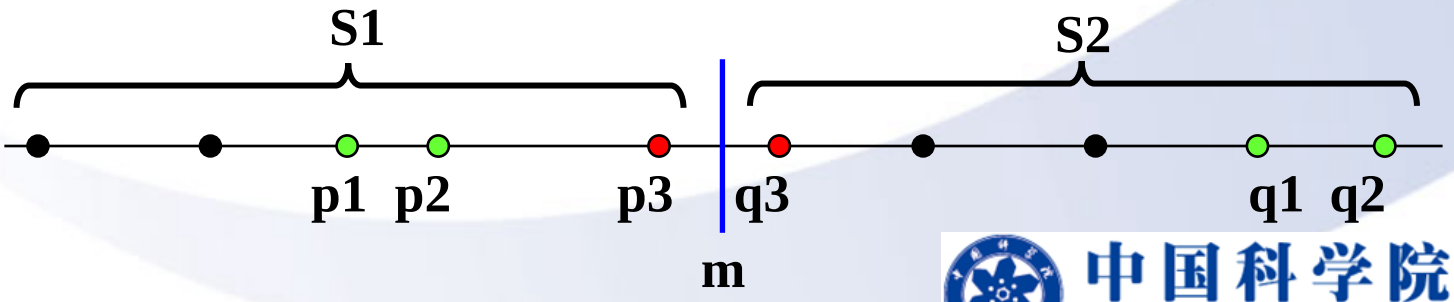
□分治策略下一维的情形

➢大致算法：

① 用S中各点坐标的中位数来作分割点，将S分成S1和S2 。

② 递归地在S1和S2上找出其最接近点对$\{p_1, p_2\}$和$\{q_1, q_2\}$。

③ 合并：S中的最接近点对在 $\{p_1, p_2\}$、$\{q_1, q_2\}$、$\{p_3, q_3\}$中，$p_3 \in S1$且$q_3 \in S2$。



3.7 最接近点对问题

- 一维空间找最接近点对
  - 分治策略下一维的情形
    - 如果S的最接近点对是$\{p_3, q_3\}$，即$|p_3-q_3|<d$，则p_3和q_3两者与m的距离不超过d，即$p_3-m<d$，$q_3-m<d$,即$p_3 \in (m-d, m]$，$q_3 \in (m, m+d]$ 。
    - 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。同理，如果(m, m+d]中有S中的点，则此点就是S2中最小点。
    - 因此用线性时间就能找到区间(m-d, m]和(m, m+d]中所有点，即p_3和q_3。从而用线性时间就可以将S1的解和S2的解合并成为S的解。

3.7 最接近点对问题

T(n)=O(nlogn)

- 一维空间找最接近点对
  - 分治策略下一维的情形

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

```
public static double Cpair1(S, d) //找S中最接近点对的距离d
{
    n=|S|; //S中点的个数
    if(n<2)    d=∞;
    m=S中各点坐标的中位数;
    构造S1和S2;    //S1={x∈S|x<=m}, S2={x∈S|x>m}
    Cpair1(S1, d1);  p=max(S1);
    Cpair1(S2, d2);  q=min(S2);
    d=min(d1, d2, q-p);
    return true;
}
```

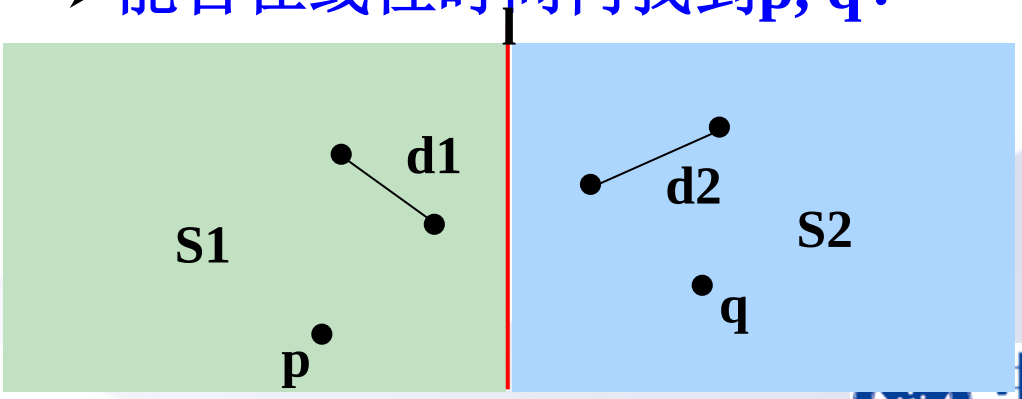
O(n)

2T(n/2)

O(n)

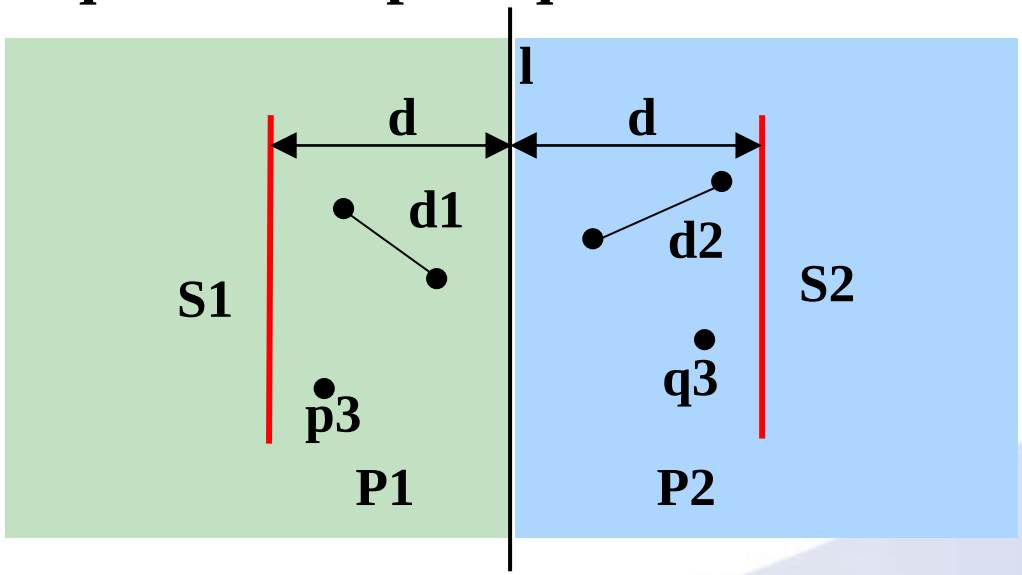
3.7 最接近点对问题

- 二维空间找最接近点对
  - 分治策略下二维的情形
    - 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∈S1且q∈S2。
    - 能否在线性时间内找到p, q？



3.7 最接近点对问题

- 二维空间找最接近点对
  - 在线性时间内找到p3, q3
    - 第一步筛选：如果最近点对由S1中的p3和S2中的q3组成，则p3和q3一定在划分线L的距离d内。

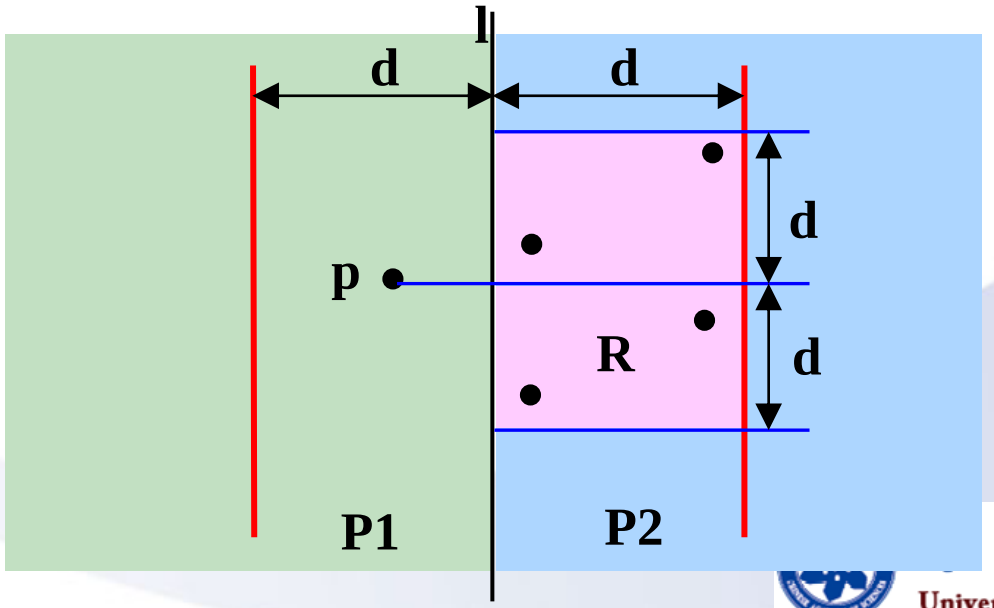


需要计算P1中的每个点与P2中的每个点的距离？$O(n^2)$

不需要！

3.7 最接近点对问题

- 二维空间找最接近点对
  - 在线性时间内找到p3, q3
    - 第二步筛选：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。

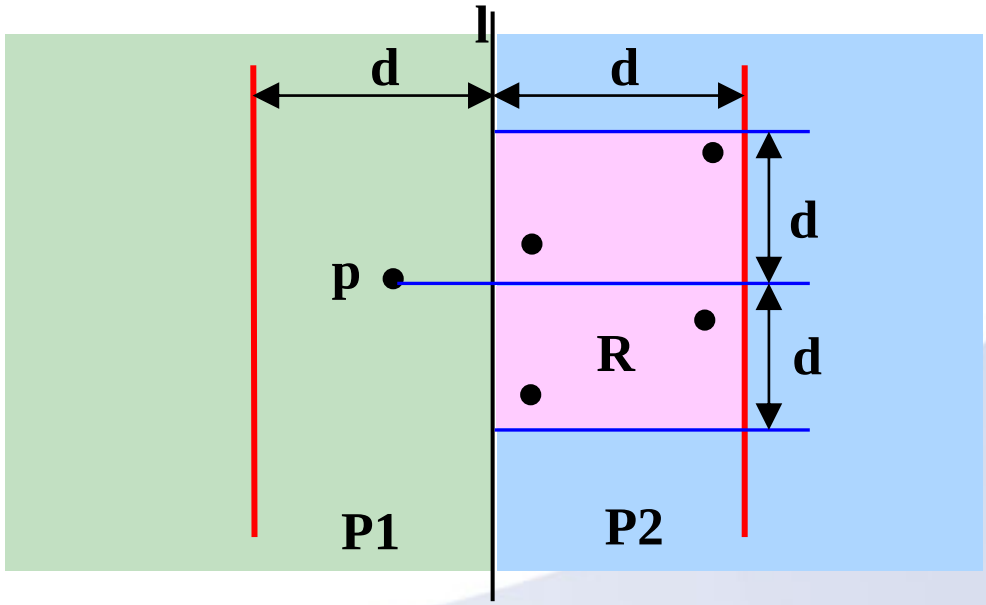


3.7 最接近点对问题

■ 二维空间找最接近点对

□R中的点具有稀疏性

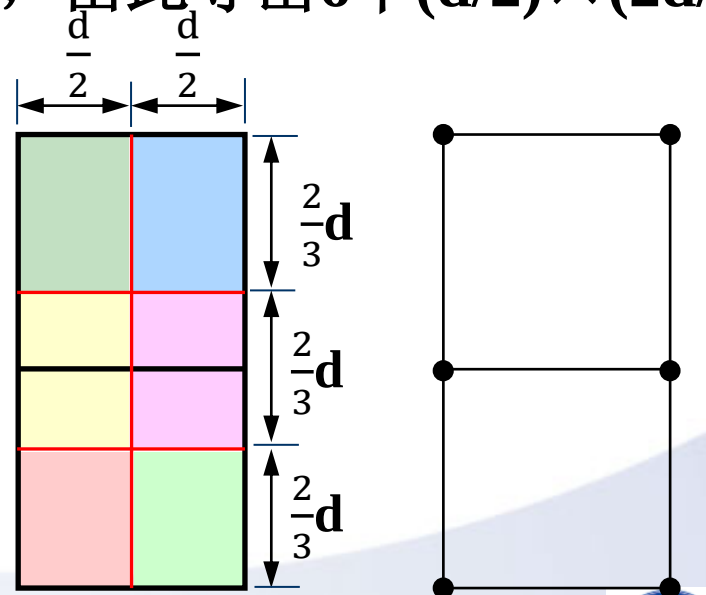
- 重要观察结论：P2中任何2个S中的点的距离都不小于d。由此可以推出矩形R中最多只有6个S中的点。



- 重要结论：在分治法的合并步骤中最多只需要检查6×n/2=3n个候选点对！

3.7 最接近点对问题

- 二维空间找最接近点对
  - R中最多只有6个S中的点
  - 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。



(a)　　(b)

3.7 最接近点对问题

* 鸽舍原理（也称抽屉原理）
把n+1个球，放入n个抽屉，则一定有一个抽屉内至少有2个球。

- 二维空间找最接近点对
  - R中最多只有6个S中的点
  - 证明:
    - 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。

3.7 最接近点对问题

- 二维空间找最接近点对
  - 如何确定要检查哪6个点
    - P2中与点p最接近这6个候选点的纵坐标与p的纵坐标相差不超过d。
    - 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。

```
double cpair2(S)
{ n=|S|; if (n < 2) return;
1、m=S中各点x间坐标的中位数;
    构造S1和S2;      //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之内所有点组成的集合;
    将P1和P2中点依其y坐标值排序;
    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dm之内的所有点
        (最多6个)可以完成合并;
    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; }
```

O(n)

2T(n/2)

O(n)

O(n)

3.7 最接近点对问题

- 二维空间找最接近点对
  - 复杂度分析
    - ①、⑤用了**O(n)**时间；
    - ②用了**2T(n/2)**时间
    - ③、⑥用了常数时间
    - ④在预排序的情况下用时**O(n)**

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

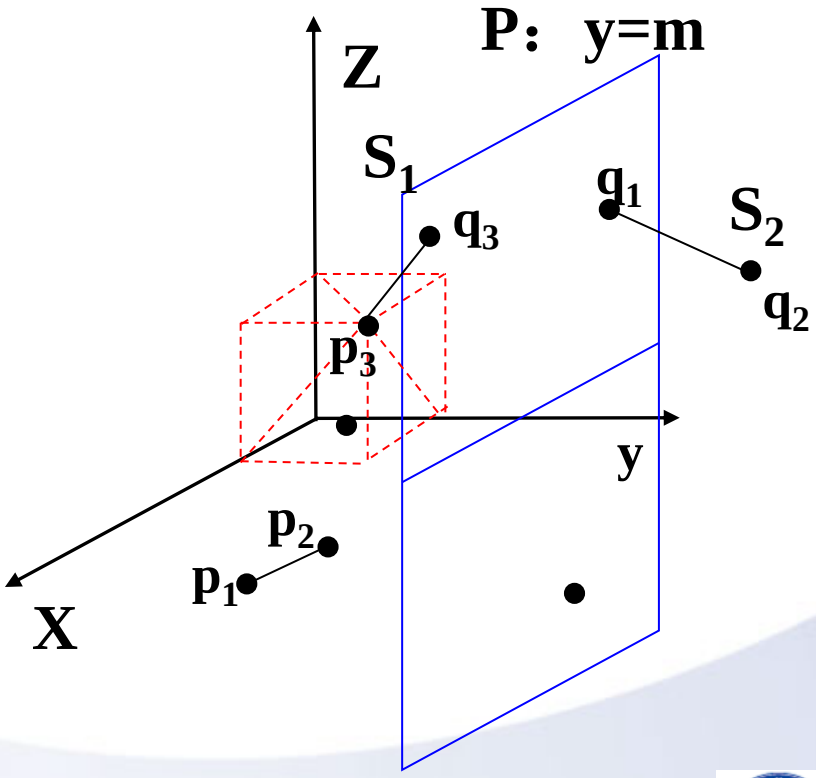
$$T(n)=O(n\log n)$$

3.7 最接近点对问题

- 三维空间找最接近点对
  - 分治策略下三维的情形
    - 选取一垂平面l:y=m来作为分割平面。其中m为S中各点y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∈S1且q∈S2。

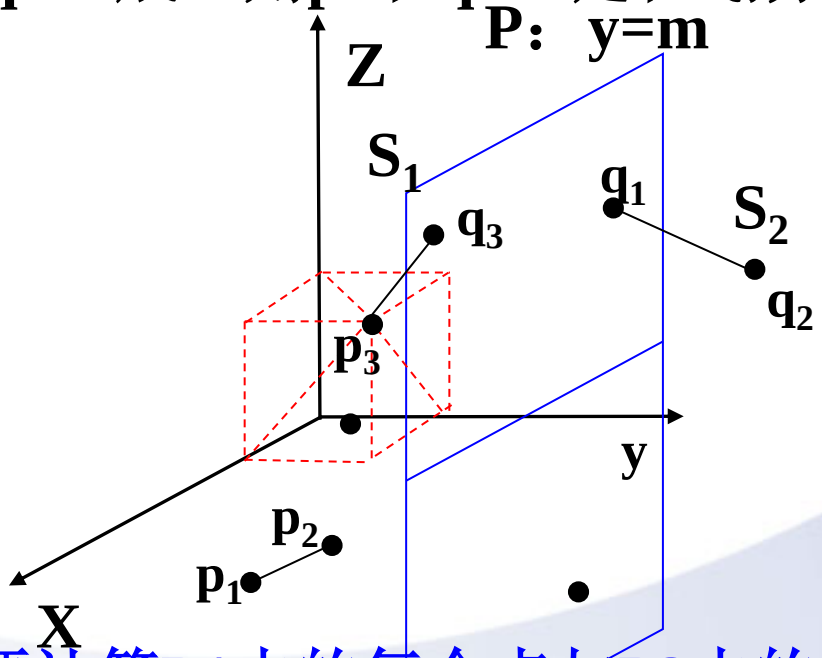
3.7 最接近点对问题

- 三维空间找最接近点对
  - 分治策略下三维的情形



3.7 最接近点对问题

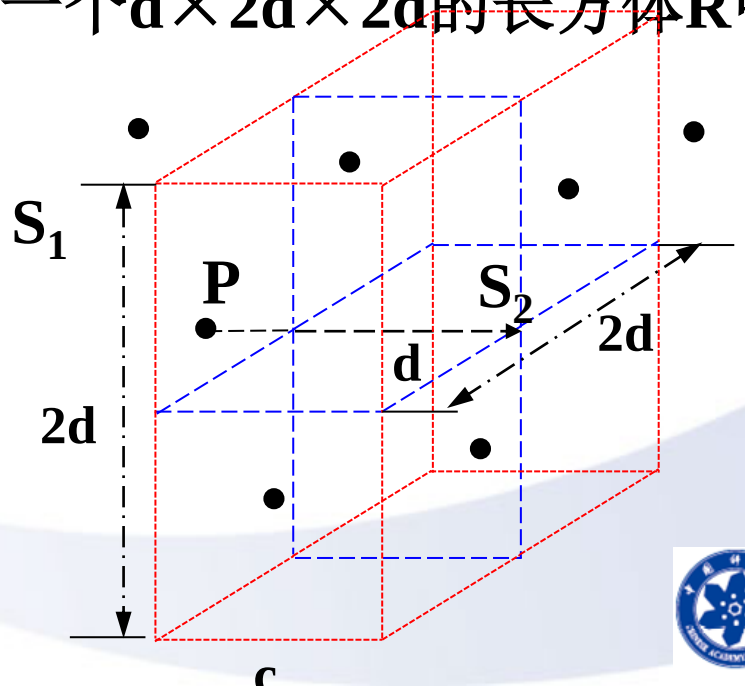
- 三维空间找最接近点对
  - 分治策略下三维的情形
    - 第一步筛选：如果最近点对由S1中的p3和S2中的q3组成，则p3和q3一定在划分平面L的距离d内。



需要计算P1中的每个点与P2中的每个点的距离？

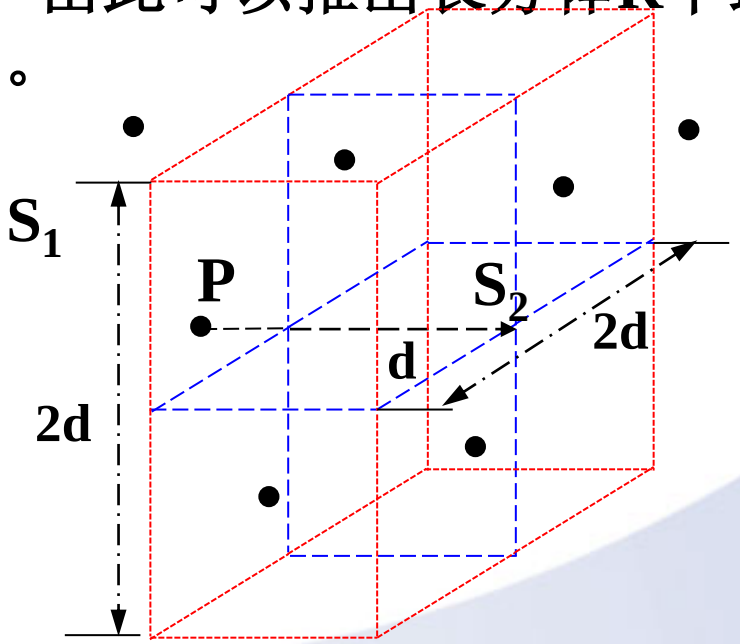
3.7 最接近点对问题

- 三维空间找最接近点对
  - 分治策略下三维的情形
    - 第二步筛选：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中的点定落在一个d×2d×2d的长方体R中。



3.7 最接近点对问题

- 三维空间找最接近点对
  - 分治策略下三维的情形
    - 重要观察结论：P2中任何2个S中的点的距离都不小于d。由此可以推出长方体R中最多只有24个S中的点。



重要结论：在分治法的合并步骤中最多只需要检查24×n/2=12n个候选点对！

3.7 最接近点对问题

- 三维空间找最接近点对
  - 分治策略下三维的情形
    - P2 中任何2个S中的点的距离都不小于d。
    - 由此可以推出长方体R中最多只有24个S 中的点.
    - 可以将长方体C的长为2d的两条边分别3等分和4等分，将它的长为d的边2等分，由此导出24个大小相等的(d /2) ×(d /2) ×(2d /3) 的小长方体。
    - 如图3所示:若长方体C中有多于24个S中的点,则由鸽舍原理易知至少有一个(d /2) ×(d /2) ×(2d /3) 的小长方体中有2个以上S中的点. 设u, v是这样2个点,它们位于同一小长方体中, distance (u, v)

$$(x(u)-x(v))^2+(y(u)-y(v))^2+(z(u)-z(v))^2 \le (d/2)^2+(d/2)^2+(2d/3)^2=\frac{17}{18}d^2$$

    - distance(u,v)<d。这与d的意义相矛盾。

3.7 最接近点对问题

- 三维空间找最接近点对
  - 如何确定要检查哪24个点
    - 为了确切地知道对于P1 中每个点p最多检查P2 中的哪24个点,我们可以将点p和P2 中所有S2的点投影到平面P: y = m上.
    - 由于能与p点一起构成最接近点对候选者的S2 中点一定在长方体R中,所以它们在平面P上的投影点与点p在P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24个.
    - 因此,若将P1 和P2 中所有S的点依次按其x坐标和z坐标排好序,则对P1 中任意点p而言,对已经按照x坐标和z坐标排好序的点列作一次线性扫描,就可以找出所有最接近点对的候选者.

3.7 最接近点对问题

- 三维空间找最接近点对
  - 如何确定要检查哪24个点
    - 设点q (x, y, z) 为P2 中可以与P1 中的一点p (x0 , y0 , z0) 构成候选点对的排好序的24个点中的一点, 则满足x ∈ (x0 - d, x0 + d) , z ∈ (z0 - d, z0 +d) ,即投影点在以p的投影点为中心的2d ×2d的正方形区域中的点是我们要考察的候选点.
    - 这就意味着我们可以在O (n) 时间内完成分治法的合并步骤.

```
double spair (S )
{ n =| S | ;  //| S | 表示S中点的个数3 /
   if ( n < 2) return ∞;
1. m = S中各点y坐标的中位数;
   利用平面P: y = m 划分构造子集S1 和S2 ;
   S1 = { p ∈ S | y ( p) < = m } , S2 = { p ∈ S| y ( p) > m }
2. d1 = spair (S1 ) ; d2 = spair (S2 ) ;
3. dm = min ( d1 , d2 ) ;
4. 设P1 是S1 中距垂直分割面P的距离在dm之内的所有点组成的集合;
    P2 是S2 中距垂直分割面的距离在dm 之内的所有点组成的集合;
    将P1 和P2 中点依其依次按照其x坐标值和z坐标值排序;
    并设X1、X2 是P1、P2 依据x坐标值排好序的点列;
     Z1、Z2是X1、X2再依据z坐标值排好序的点列;
5. 通过扫描Z1 以及对于Z1 中每个点检查Z2中相继的最多24个点完成合并;
   当Z1 中的扫描指针沿着某一个方向移动时,
   Z2中的扫描指针可2dm×2dm的方形区间内移动;
   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;}
```

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

$$T(n)=O(n\log n)$$

第三章 分治法

- 3.1 一般方法
- 3.2 二分检索
- 3.3 找最大和最小元素
- 3.4 归并排序
- 3.5 快速排序
- 3.6 选择问题
- 3.7 最接近点对问题
- 3.8 斯特拉森矩阵乘法
- 3.9 大整数乘法

3.8 斯特拉森矩阵乘法

- 问题描述
  - 矩阵的加法
    - 若A和B是2个n×n的矩阵，则它们的加法C=A+B同样是一个n×n的矩阵。
    - A和B的和矩阵C中的元素C[i, j]定义为:

      $c_{ij} = a_{ij} + b_{ij}, \quad i, j = 1, 2, \ldots, n$

    - 则计算C的任意一个元素C[i, j]，需要做1次加法。
    - 因此求矩阵C的n^2个元素所需的计算时间为$O(n^2)$。

3.8 斯特拉森矩阵乘法

■ 问题描述

□矩阵的乘法

- ➢若A和B是n×n的矩阵，则A和B的乘积矩阵C=AB同样是n×n的矩阵。C中的元素C[i, j]定义为:

$$C(i,j) = \sum_{1 \le k \le n} A(i,k)B(k,j) \qquad 1 \le i,j \le n$$

- ➢计算C的任意一个元素C[i, j]，需要做n次乘法和n-1次加法。
- ➢因此求矩阵C的n^2个元素所需的计算时间为$O(n^3)$。
- ➢问：是否可以用少于n^3次乘法完成C的计算?
- ➢60年代末，Strassen采用了分治技术，将计算2个n阶矩阵乘积所需的计算时间改进到$O(n^{\log 7})=O(n^{2.81})$。

3.8 斯特拉森矩阵乘法

- 解法介绍
  - 假设n是2的幂。将矩阵A，B和C中每一矩阵都分块成为4个大小相等的子矩阵，每个子矩阵都是n/2×n/2的方阵。由此可将方程C=AB重写为:

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

  - 由此可得:

$$C_{11}=A_{11}B_{11}+A_{12}B_{21}$$

$$C_{12}=A_{11}B_{12}+A_{12}B_{22}$$

$$C_{21}=A_{21}B_{11}+A_{22}B_{21}$$

$$C_{22}=A_{21}B_{12}+A_{22}B_{22}$$

3.8 斯特拉森矩阵乘法

- 解法介绍
  - 复杂度分析
    - 如果n=2，则2阶方阵的乘积可以直接用上式计算出来，共需8次乘法和4次加法。
    - 当子矩阵的阶大于2时，为求2个子矩阵的积，可以继续将子矩阵分块，直到子矩阵的阶降为2。这样，就产生了一个分治降阶的递归算法。

3.8 斯特拉森矩阵乘法

■ 解法介绍

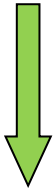
□复杂度分析

➢令T(n)表示两个n×n矩阵相乘的计算时间。

① 8次(n/2) ×(n/2) 矩阵乘 ——————>8T(n/2)

② 4次(n/2) ×(n/2) 矩阵加 ——————>dn^2

$$T(n)=\begin{cases} b & n\le 2 \\ 8T(n/2)+dn^2 & n>2 \end{cases}$$



渐近复杂度

$$T(n)=\mathrm{O}(n^3)$$

➢其中，b，d是常数。

3.8 斯特拉森矩阵乘法

- 解法介绍
  - 复杂度分析
    - 该方法并不比用原始定义直接计算更有效。
    - 原因
      - ✓没有减少矩阵的乘法次数。
    - 观察：
      - ✓矩阵乘法的花费比矩阵加法大
      - ✓$O(n^3)$对$O(n^2)$
    - 要想改进矩阵乘法的计算时间复杂性，必须减少子矩阵乘法运算的次数。

3.8 斯特拉森矩阵乘法

- 解法描述
  - Strassen提出了一种新的算法来计算2个2阶方阵的乘积。他的算法只用了7次乘法运算，但增加了加、减法的运算次数
    - $P=(A_{11}+A_{22})(B_{11}+B_{22})$
    - $Q=(A_{21}+A_{22})B_{11}$
    - $R=A_{11}(B_{12}-B_{22})$
    - $S=A_{22}(B_{21}-B_{11})$
    - $T=(A_{11}+A_{12})B_{22}$
    - $U=(A_{21}-A_{11})(B_{11}+B_{12})$
    - $V=(A_{12}-A_{22})(B_{21}+B_{22})$

7个乘法和10个加(减)法

3.8 斯特拉森矩阵乘法

- 解法描述
  - 用八个加减法计算C_{ij}

8个加(减)法

    - $C_{11}=P+S-T+V$
    - $C_{12}=R+T$
    - $C_{21}=Q+S$
    - $C_{22}=P+R-Q+U$
    - 共用7次乘法和18次加减法

3.8 斯特拉森矩阵乘法

- 解法描述
  - 用八个加减法计算C_{ij}
    - $C_{11}=P+S-T+V$

      $=(A_{11}+A_{22})(B_{11}+B_{22}) + A_{22}(B_{21}-B_{11}) - (A_{11}+A_{12})B_{22}$

      $+ (A_{12}-A_{22})(B_{21}+ B_{22})$

      $= A_{11}B_{11}+A_{11}B_{22}+A_{22}B_{11}+A_{22}B_{22} + A_{22}B_{21}-A_{22}B_{11}$

      $- A_{11}B_{22} -A_{12}B_{22} + A_{12}B_{21}+ A_{12}B_{22}-A_{22}B_{21}-A_{22}B_{22}$

      $=A_{11}B_{11}+A_{12}B_{21}$
    - $C_{12}=R+T$

      $= A_{11}(B_{12}-B_{22}) + (A_{11}+A_{12})B_{22}$

      $= A_{11}B_{12}-A_{11}B_{22} + A_{11}B_{22}+A_{12}B_{22}$

      $= A_{11}B_{12}+A_{12}B_{22}$

3.8 斯特拉森矩阵乘法

- 解法描述
  - 斯特拉森时间复杂度

$$T(2) = b$$

$$T(n) = 7T(n/2) + an^2 \qquad n > 2$$

  - 按照解递归方程的套用公式法，其解为

$$T(n) = an^2(1 + 7/4 + (7/4)^2 + \cdots + (7/4)^{k-1}) + 7^k T(1)$$

$$\leq cn^2(7/4)^{\log n} + 7^{\log n}$$

$$= cn^{\log 4 + \log 7 - \log 4} + n^{\log 7}$$

$$= (c+1)n^{\log 7} = O(n^{\log 7}) \approx O(n^{2.81})$$

3.8 斯特拉森矩阵乘法

- 其他矩阵乘法
  - 有人曾列举了计算2个2阶矩阵乘法的36种不同方法。但所有的方法都要做7次乘法。
  - 除非能找到一种计算2阶方阵乘积的算法，使乘法的计算次数少于7次，按上述思路才有可能进一步改进矩阵乘积的计算时间的上界。
  - 但是Hopcroft 和 Kerr(197l) 已经证明，计算2个2×2矩阵的乘积，7次乘法是必要的。

3.8 斯特拉森矩阵乘法

- 其他矩阵乘法
  - 因此要想进一步改进矩阵乘法的时间复杂性，就不能再寄希望于计算2×2矩阵的乘法次数的减少。或许应当研究3×3或5×5矩阵的更好算法。
  - 在Strassen之后又有许多算法改进了矩阵乘法的计算时间复杂性。目前最好的计算时间上界是$O(n^{2.367})$。而目前所知道的矩阵乘法的最好下界仍是它的平凡下界$\Omega(n^2)$。

3.8 斯特拉森矩阵乘法

- 实际性能分析
  - 斯特拉森矩阵乘法目前还只具有理论意义，因为只有当n相当大时他才优于通常的矩阵乘法。
  - 经验表明，当n=120时，斯特拉森矩阵乘法与通常的矩阵乘法在计算上无显著差别。
  - 有益的启示
    - 由定义出发所直接给出的明显算法并非总是最好的。

第三章 分治法

- 3.1 一般方法
- 3.2 二分检索
- 3.3 找最大和最小元素
- 3.4 归并排序
- 3.5 快速排序
- 3.6 选择问题
- 3.7 最接近点对问题
- 3.8 斯特拉森矩阵乘法
- 3.9 大整数乘法

3.9 大整数乘法

- 问题描述
  - X和Y都是n位的二进制整数，要计算他们的乘积XY。
- 直接求解
  - Y的n位分别和X的n位相乘
  - 计算复杂性： $O(n^2)$
- 分治法：
  - 将n位的二进制整数X和Y都分成2段，每段的长为n/2位。

X= | A | B |　　Y= | C | D |

  - 所以， $X = A2^{n/2} + B$, $Y = C2^{n/2} + D$
  - $XY = AC\ 2^n + (AD+BC)\ 2^{n/2} + BD$

3.9 大整数乘法

- 复杂性分析
  - $XY = AC\ 2^n + (AD+BC)\ 2^{n/2} + BD$
    - 乘法次数：4次n/2位的整数乘法；
    - 加法次数：3次整数加法；
    - 移位次数：2次；
    - 当n>1时，有$T(n)=4T(n/2)+O(n)$
  - $T(n)=O(n^2)$

3.9 大整数乘法

- 其他解法
  - $XY = AC\ 2^n + (AD+BC)\ 2^{n/2} + BD$
  - $XY = AC\ 2^n + ((A-B)(D-C)+AC+BD)\ 2^{n/2} + BD$
    - 乘法次数：3次n/2位的整数乘法；
    - 加法：6次整数加法；
    - 移位：2次；
    - 当n>1时，有$T(n)=3T(n/2)+O(n)$
  - $T(n)=O(n^{\log 3})=O(n^{1.59})$

3.9 大整数乘法

- 问题描述
  - 请设计一个有效的算法，可以进行两个n位大整数的乘法运算
- 解法
  - 小学的方法：$O(n^2)$
  - 分治法：

$$T(n)=\begin{cases}O(1) & n=1\\ 3T(n/2)+O(n) & n>2\end{cases}$$

$$T(n)=O(n^{\log 3})=O(n^{1.59})$$

  - $XY = AC\ 2^n + (AD+BC)\ 2^{n/2} + BD$
  - 1. $XY = AC\ 2^n + ((A-B)(D-C)+AC+BD)\ 2^{n/2} + BD$
  - 2. $XY = AC\ 2^n + ((A+C)(B+D)-AC-BD)\ 2^{n/2} + BD$
    - ✓细节问题：两个XY的复杂度都是$O(n^{\log 3})$，但考虑到A+C, B+D可能得到m+1位的结果，使问题的规模变大，故不选择第2种方案。

3.9 大整数乘法

- 问题描述
  - 请设计一个有效的算法，可以进行两个n位大整数的乘法运算
  - 小学的方法：$O(n^2)$ ✘ 效率太低
  - 分治法: $O(n^{1.59})$ ✓ 较大的改进
    - 如果将大整数分成更多段，用更复杂的方式把它们组合起来， 将有可能得到更优的算法。
    - 最终的，这个思想导致了快速傅利叶变换(Fast Fourier Transform)的产生。该方法也可以看作是一个复杂的分治算法，对于大整数乘法，它能在O(nlogn)时间内解决。
    - 是否能找到线性时间的算法？？？目前为止还没有结果。

作业-算法实现2

- 棋盘覆盖问题
  - 在一个$2^k \times 2^k$个方格组成的棋盘中，恰有一个方格与其他方格不同，称该方格为一特殊方格，且称该棋盘为一特殊棋盘。如图1所示，蓝色的为特殊方格：

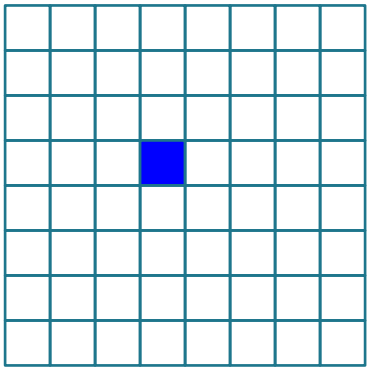


图1 特殊棋盘，蓝色的为特殊方格

图2 四种L形骨牌

作业-算法实现2

- 棋盘覆盖问题
  - 棋盘覆盖问题是指，要用图2中的4种不同形态的L型骨牌覆盖给定的特殊棋盘上除特殊方格以外的所有方格，且任何2个L型骨牌不得重叠覆盖。
- 作业
  - 用分治法设计一个求解棋盘覆盖问题的算法。
  - 用C(C++)，Matlab，python，pytorch语言实现。
  - 有求解思路的简单说明。
  - 上载到课程网站上。

■ End