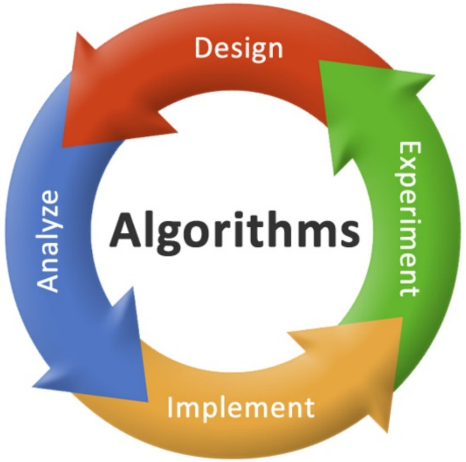


Algorithm Design and Analysis

算法设计与分析

▌Chapter 4-1: Divide And Conquer ▌ 张乾坤

课程提要

- 归并排序(Merge sort)
- 逆序对计算 （ Counting inversions ）
- 最近点对 （ closest pair of points ）

分治范式（Divide-and-conquer paradigm）

分而治之（Divide-and-conquer）

- 将问题分成几个子问题（同类）
- 递归地解决（处理）每一个子问题
- 将子问题的解决方案合并为整体解决方案

最常见的用法

- 将大小为n的问题分成两个大小为$n/2$的子问题 ⟵ $O(n)$ time
- 递归地解决（处理）两个子问题
- 将两个解决方案合并为整体解决方案 ⟵ $O(n)$ time

结果

- 暴力法：$\Theta(n^2)$
- 分而治之策略：$O(n\log n)$

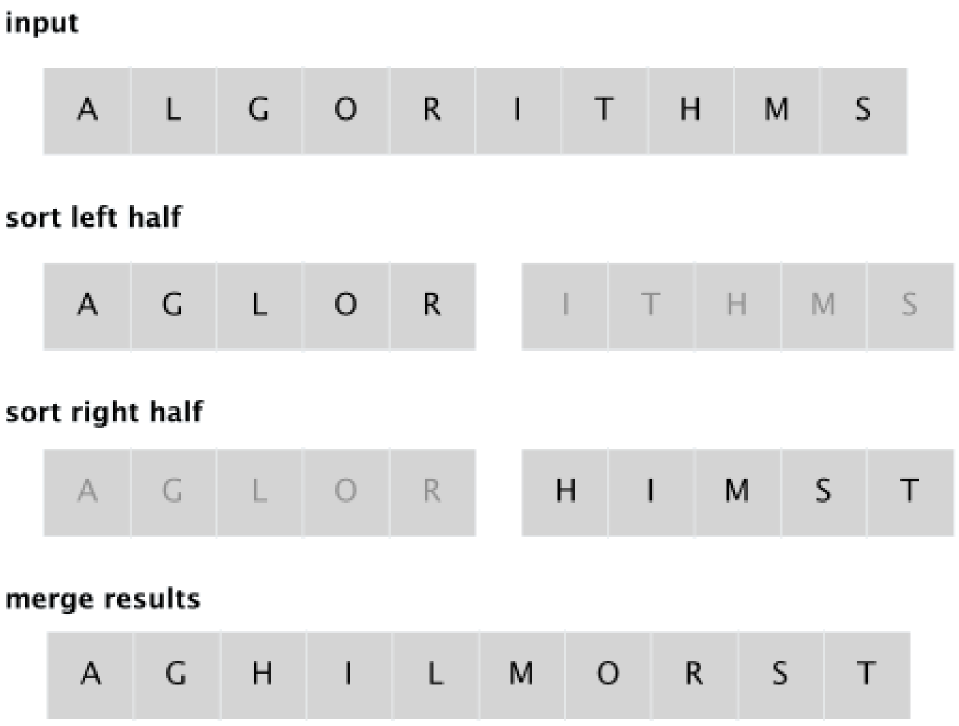
attributed to Julius Caesar

课程提要

- 归并排序(Merge sort)
- 逆序对计算 （ Counting inversions ）
- 最近点对 （ closest pair of points ）

归并排序（Mergesort）

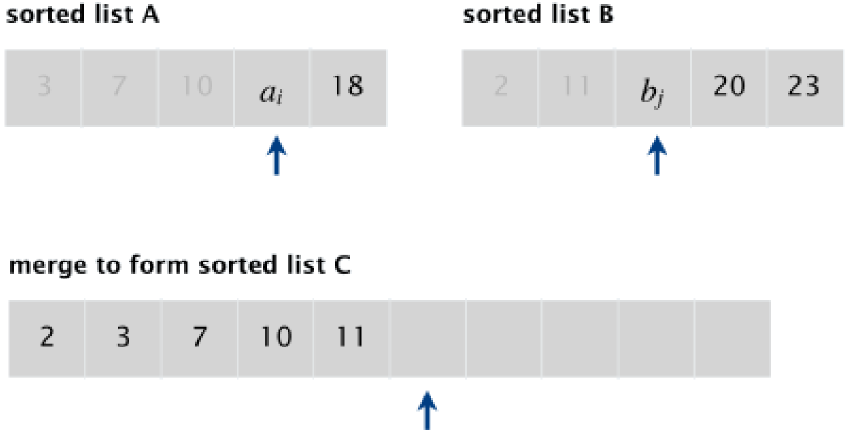
- 问题：给出一个由完全有序的域中的 n 个元素组成的列表 L，按升序重新排列它们。
- 对左半部分进行递归排序
- 对右半部分进行递归排序
- 合并左右两半部分，使其成为已排序的整体



归并（Merging）

目标：将两个排序列表 A 和 B 组合成一个排序的整体 C

- 从左向右扫描 A 和 B
- 比较 a_i 和 b_j
- 如果 $a_i \le b_j$，将 a_i 加到 C（不大于 B 中的任何剩余元素）
- 如果 $a_i > b_j$，将 b_j加到 C（不大于 B 中的任何剩余元素）



归并排序实现

输入：一个由完全有序的域中的 n 个元素组成的列表 L

输出：升序排列的 n 个元素

```
MERGE-SORT(L)

IF (list L has one element)
    RETURN L.

Divide the list into two halves A and B.
A ← MERGE-SORT(A).   ⟵ T(n / 2)
B ← MERGE-SORT(B).   ⟵ T(n / 2)
L ← MERGE(A, B).     ⟵ Θ(n)
RETURN L.
```

一个有用的递归关系

定义：$T(n)$ = 归并排序一个长度为n的列表的最大比较数

递归：

$$T(n) \le \begin{cases} 0 & if\ n = 1 \\ T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + n & if\ n > 1 \end{cases}$$

在$\lfloor n/2 \rfloor$和$n-1$之间进行比较

结果：$T(n)$是$O(n \log_2 n)$

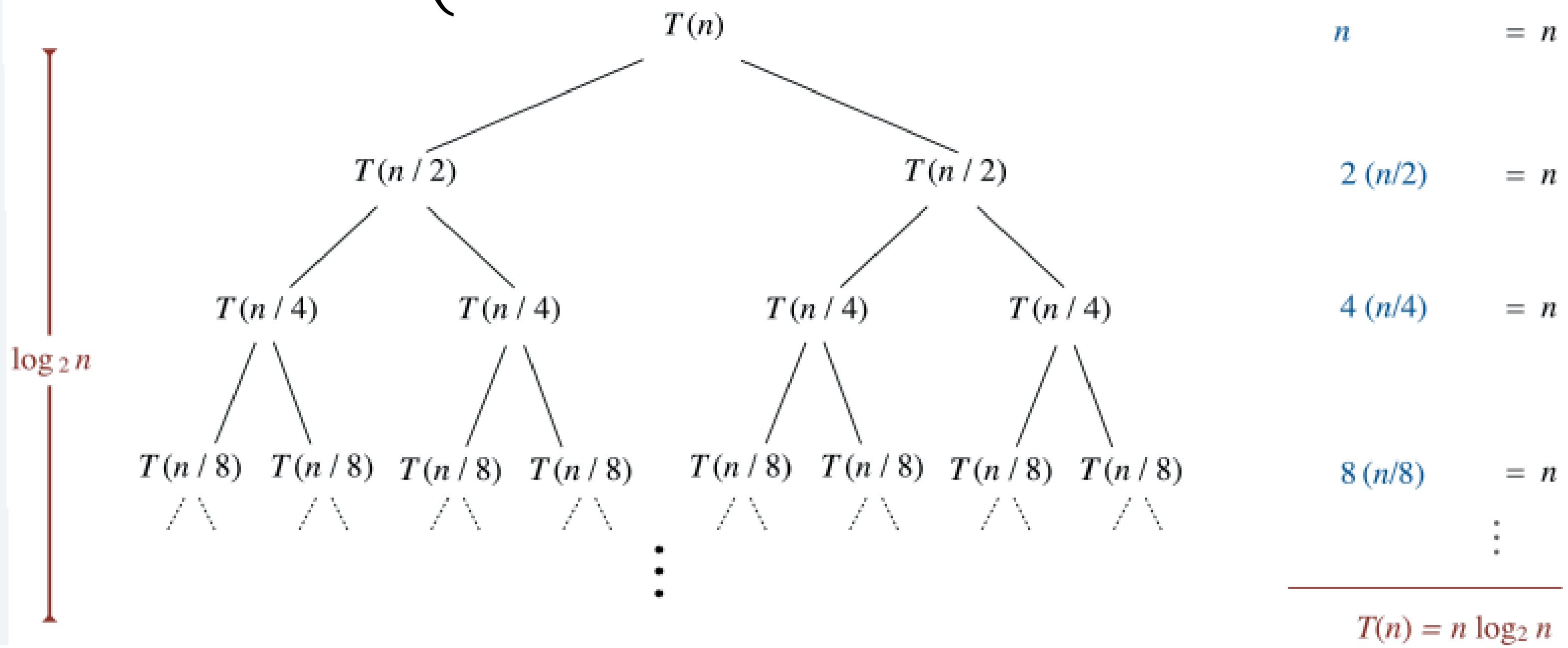
各种证明：我们阐述几种解决这种递归的方法。最初我们假设 n 是 2 的幂，并用 = 代替递归中的 $\le$

分治递归：递归树(recursion tree)

命题：如果$T(n)$满足如下的递归式，那么$T(n) = n\log_2 n$

假设n是2的幂

$$T(n) \le \begin{cases} 0 & if\ n = 1 \\ 2T(n/2) + n & if\ n > 1 \end{cases}$$



归纳法证明

命题：如果$T(n)$满足如下的递归式，那么$T(n) = n\log_2 n$

假设n是2的幂

$$T(n) \leq \begin{cases} 0 & if\ n = 1 \\ 2T(n/2) + n & if\ n > 1 \end{cases}$$

证明：[通过基于n的归纳]

- 基本情况：当$n = 1, T(1) = 0 = n\log_2 n$
- 归纳假设：假设$T(n) = n\log_2 n$
- 目标：证明$T(2n) = 2n\log_2 2n$

递归

$$T(2n) = 2T(n) + 2n$$

归纳假设

$$= 2n\log_2 n + 2n$$

$$= 2n(\log_2(2n) - 1) + 2n$$

$$= 2n\log_2(2n)$$

归并排序递归分析

以下递归的精确解是什么？

$$T(n) \leq \begin{cases} 0 & if\ n = 1 \\ T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + n - 1 & if\ n > 1 \end{cases}$$

不再假设n是2的幂

证明：[通过基于n的归纳]

基本情况：当$n = 1$

定义$n_1 = \lfloor n/2 \rfloor$和$n_2 = \lceil n/2 \rceil$并且注意到$n = n_1 + n_2$

归纳步骤：假设上述对$1,2,\ldots,n-1$均成立

归并排序递归分析

以下递归的精确解是什么?

$$T(n) \le \begin{cases} 0 & if\ n = 1 \\ T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + n - 1 & if\ n > 1 \end{cases}$$

不再假设n是2的幂

$$\begin{aligned} T(n) &\le T(n_1) + T(n_2) + n \\ &\le n_1 \lceil \log_2 n_1 \rceil + n_2 \lceil \log_2 n_2 \rceil + n \\ &\le n_1 \lceil \log_2 n_2 \rceil + n_2 \lceil \log_2 n_2 \rceil + n \\ &= n \lceil \log_2 n_2 \rceil + n \\ &\le n(\lceil \log_2 n \rceil - 1) + n \\ &= n \lceil \log_2 n \rceil \end{aligned}$$

归纳假设

$$\begin{aligned} n_2 &= \lceil n/2 \rceil \\ &\le \left\lceil 2^{\lceil \log_2 n \rceil} / 2 \right\rceil \\ &= 2^{\lceil \log_2 n \rceil} / 2 \end{aligned}$$

$$\log_2 n_2 \le \lceil \log_2 n \rceil - 1$$

an integer

课程提要

- 归并排序(Merge sort)
- 逆序对计算 （ Counting inversions ）
- 最近点对 （ closest pair of points ）

逆序对计算

- 音乐网站试图匹配你与其他人的歌曲偏好
  - 你给出n首歌排名
  - 音乐网站查询数据库，寻找有相似品味的人
- 相似性度量：两个排名之间逆序对的数量
  - 我的排名：$1, 2, \ldots, n$
  - 你的排名：$a_1, a_2, \ldots, a_n$
  - 如果$i < j$，但$a_i > a_j$，歌曲i与j为逆序对
- 暴力法：检查所有$\Theta(n^2)$个对

	A	B	C	D	E
me	1	2	3	4	5
you	1	3	4	2	5

2 inversions: 3-2, 4-2

逆序对计算：应用

- 投票理论
- 协同过滤
- 测量数组的“排序性”
- 谷歌排名函数的敏感性分析
- 用于Web上元宇宙搜索的秩聚合
- 非参数统计

Rank Aggregation Methods for the Web

Cynthia Dwork* Ravi Kumar† Moni Naor‡ D. Sivakumar§

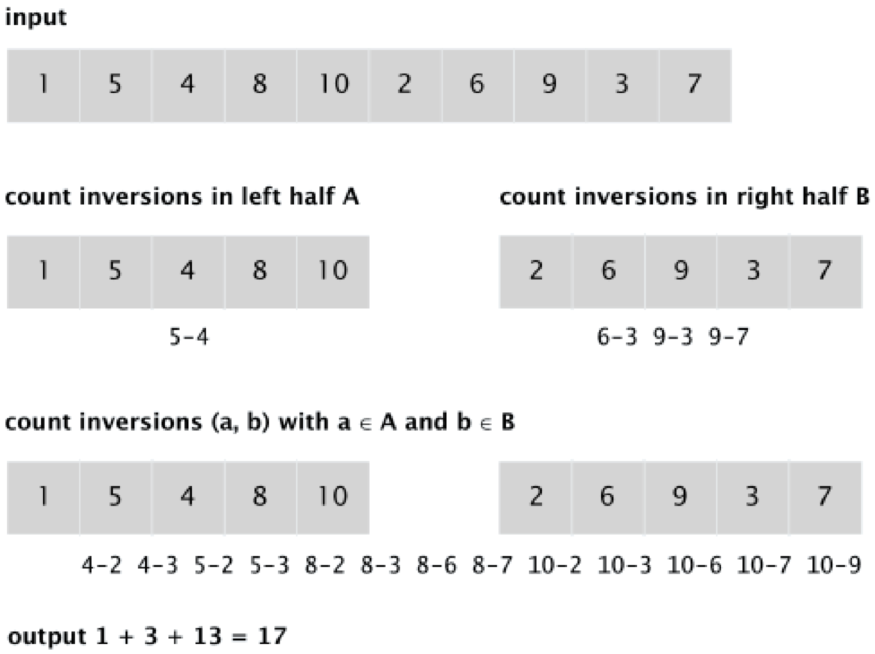
ABSTRACT

We consider the problem of combining ranking results from various sources. In the context of the Web, the main applications include building meta-search engines, combining ranking functions, selecting documents based on multiple criteria, and improving search precision through word associations. We develop a set of techniques for the rank aggregation problem and compare their performance to that of well-known methods. A primary goal of our work is to design rank aggregation techniques that can effectively combat "spam," a serious problem in Web searches. Experiments show that our methods are simple, efficient, and effective.

Keywords: rank aggregation, ranking functions, meta-search, multi-word queries, spam

逆序对计算：分治

- 划分：将列表分成A和B两部分。
- 处理：递归地计算每个列表中的逆序
- 合并：计算逆序对(a, b)，其中$a \in A$和$b \in B$。
- 返回三个计数的和



逆序对计算：如何合并两个子问题

Q：如何计算逆序对(a, b)，其中$a \in A$和$b \in B$？

A：如果A和B是有序的，那就简单多了！

热身算法：

- 将A和B排序
- 对每一个元素$b \in B$，
  - 对A进行二分查找，找出A中的元素如何大于b

list A

7	10	18	3	14

list B

20	23	2	11	16

sort A

3	7	10	14	18

sort B

2	11	16	20	23

binary search to count inversions (a, b) with a ∈ A and b ∈ B

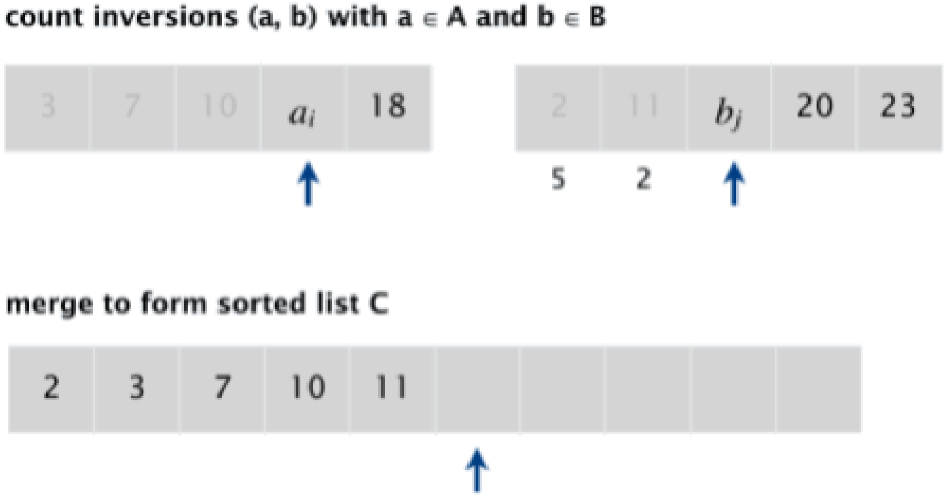
3	7	10	14	18

2	11	16	20	23
5	2	1	0	0

逆序对计算：如何合并两个子问题

计算逆序对(a, b)，其中$a \in A$和$b \in B$ 且假设A与B都已经排好序

- 从左向右扫描A和B
- 比较a_i和b_j
- 如果$a_i < b_j$，那么a_i对于B中的任意剩下元素不是逆序的
- 如果$a_i > b_j$，那么b_j对于A中的任意剩下元素都是逆序的
- 向已排序列表C中添加较小的元素



逆序对计算：分治算法实现

输入：列表L

输出：排好序的L和L的逆序数对

SORT-AND-COUNT(L)

IF (list L has one element)

  RETURN $(0, L)$.

Divide the list into two halves A and B.

$(r_A, A) \leftarrow$ SORT-AND-COUNT(A). ⟵ $T(n / 2)$

$(r_B, B) \leftarrow$ SORT-AND-COUNT(B). ⟵ $T(n / 2)$

$(r_{AB}, L) \leftarrow$ MERGE-AND-COUNT(A, B). ⟵ $\Theta(n)$

RETURN $(r_A + r_B + r_{AB}, L)$.

逆序对计算：分治算法分析

命题：排序计数算法在计算大小为n的排列中的逆序队个数的时间复杂度为$O(n\log n)$。

证明：最坏情况运行时间$T(n)$满足递归式：

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 0 \\ T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + \Theta(n) & \text{if } n > 1 \end{cases}$$

课程提要

- 归并排序(Merge sort)
- 逆序对计算 （ Counting inversions ）
- 最近点对 （ closest pair of points ）

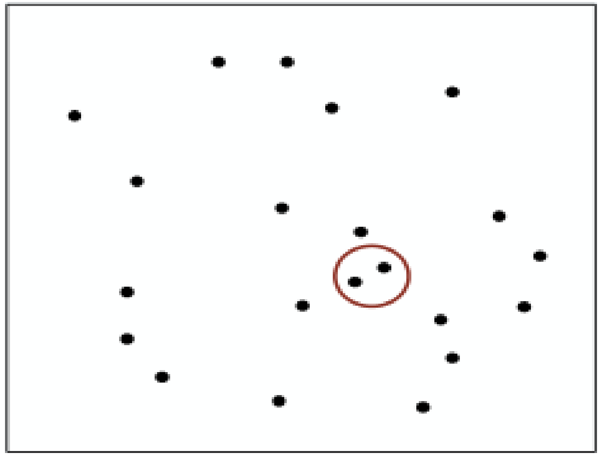
最近点对（closest pair of points）

最近点对问题：给定平面上的n个点，找出它们之间的欧氏距离最小的点对

基本几何元素：

- 图形学，计算机视觉，地理信息系统，分子建模，空中交通管制
- 最近邻居的特殊情况，欧几里得MST，泰森多边形法

fast closest pair inspired fast algorithms for these problems



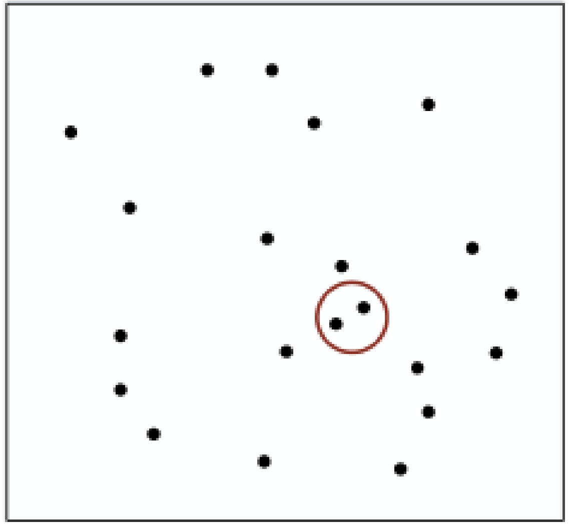
最近点对（closest pair of points）

最近点对问题：给定平面上的n个点，找出它们之间的欧氏距离最小的点对

暴力解法：以$\Theta(n^2)$的距离计算复杂度检查所有点对

一维版本：如果所有点在一条直线上，很容易实现复杂度为$O(n\log n)$的算法

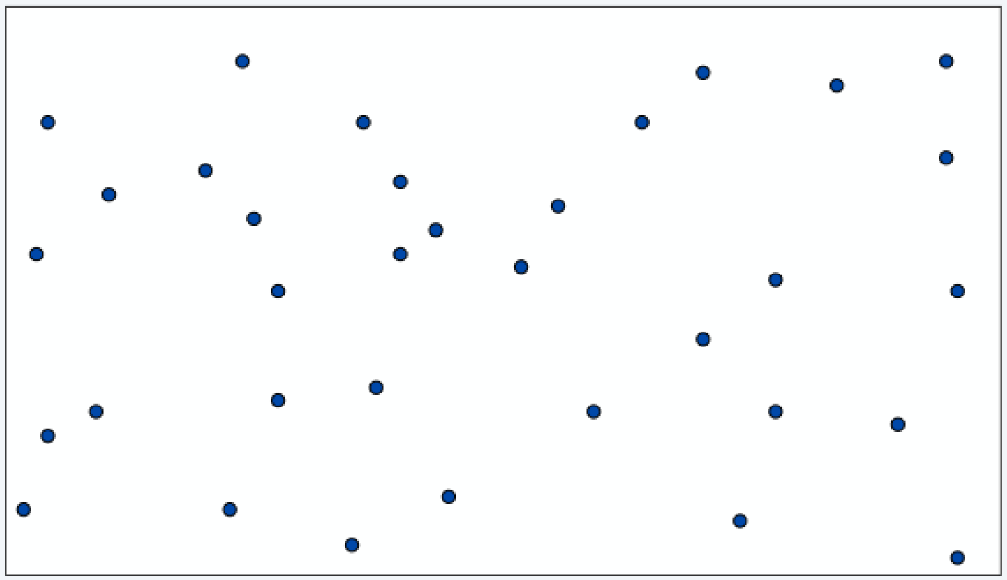
假设：没有任何两个点的x坐标相同



最近点对：第一次尝试

排序解法：

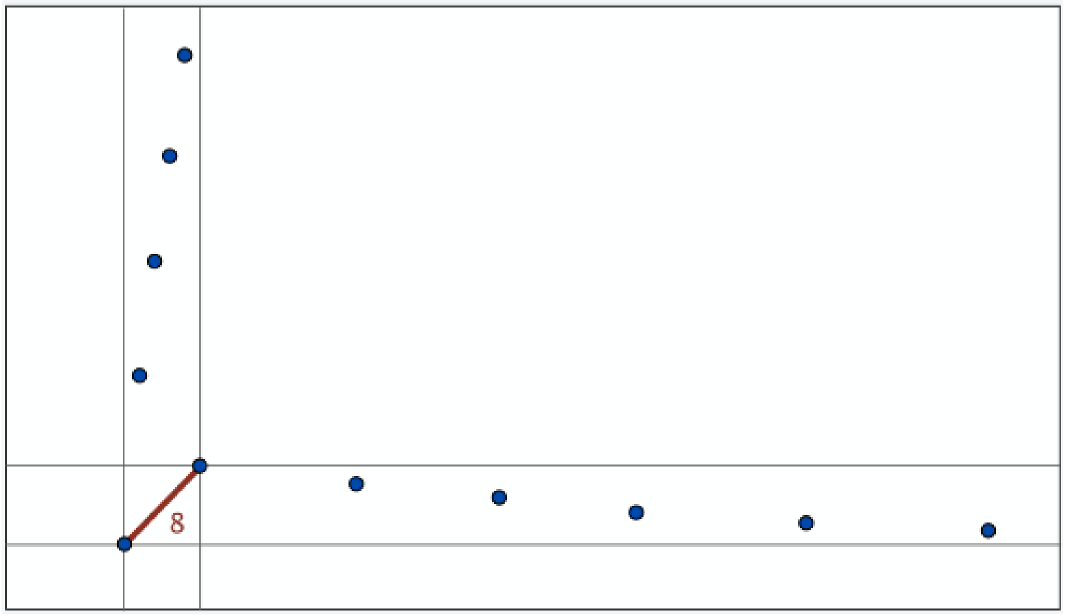
- 根据x坐标排序并考虑附近的点
- 根据y坐标排序并考虑附近的点



最近点对：第一次尝试

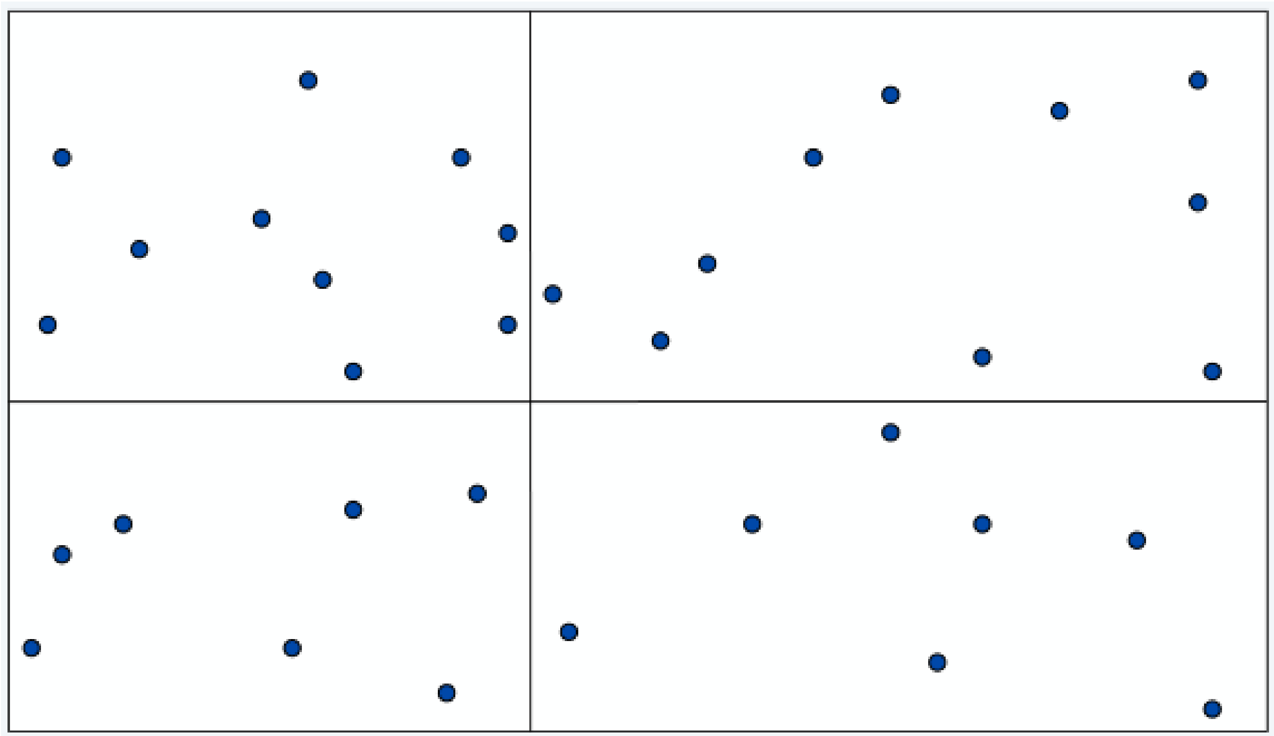
排序解法：

- 根据x坐标排序并考虑附近的点
- 根据y坐标排序并考虑附近的点



最近点对：第二次尝试

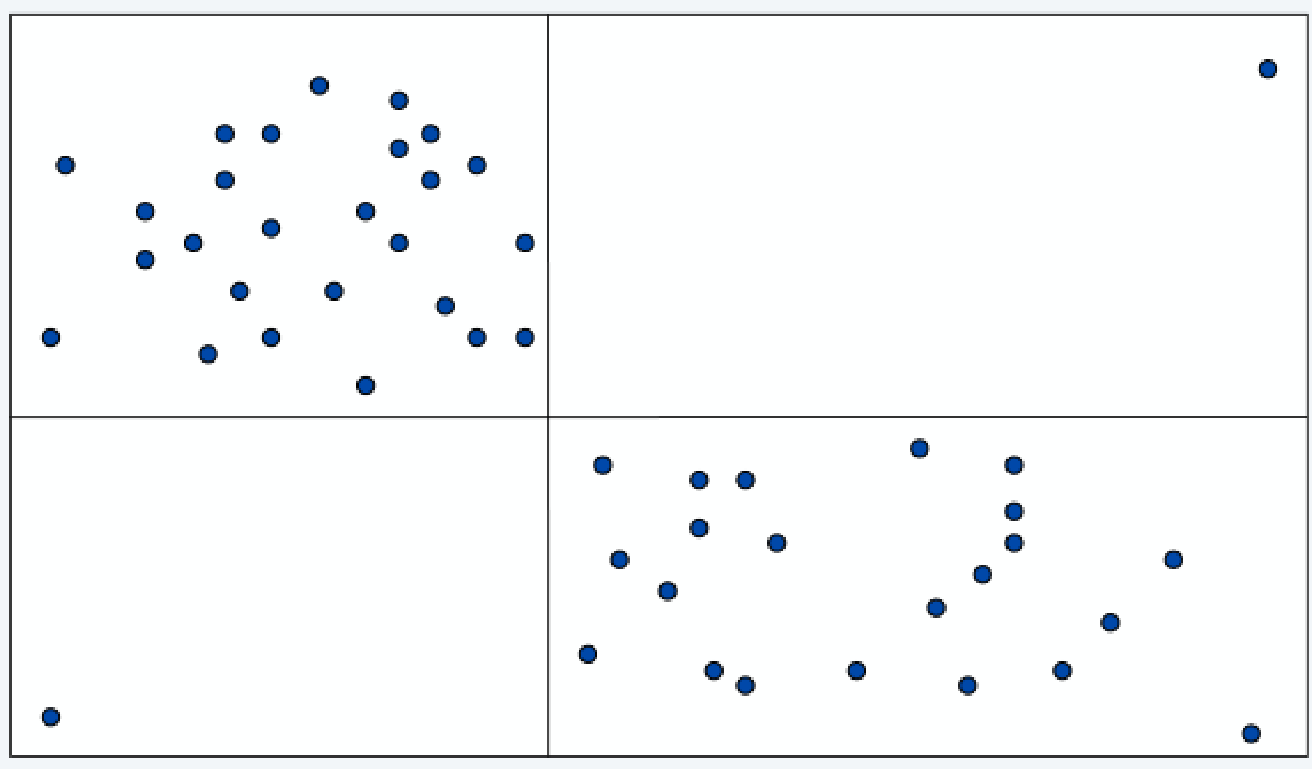
划分：将区域细分为4个象限



最近点对：第二次尝试

划分：将区域细分为4个象限

阻碍：不可能保证每个象限都有$n/4$个点



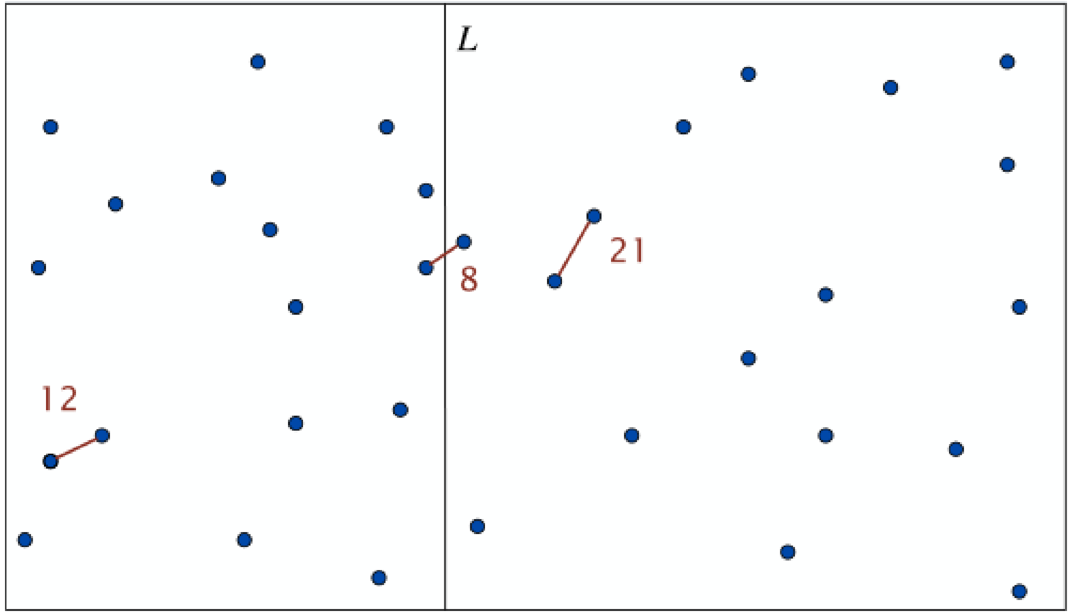
最近点对：分治算法

划分：画一条垂直线L，这样每边有$n/2$个点

处理：递归地在每边找到最近的一对

结合：在两边各找一点，其距离最近

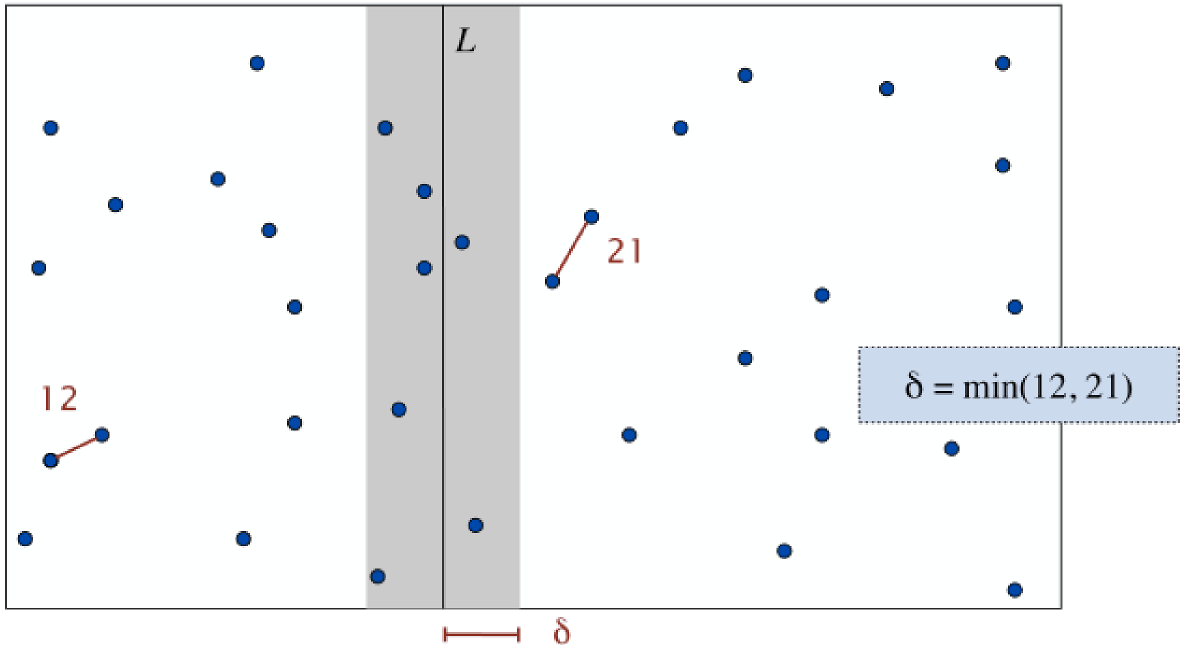
返回第三步中的最优解



如何找到两边各有一个点的最接近点对?

找出最接近的一对，两边各有一个点，假设距离< δ

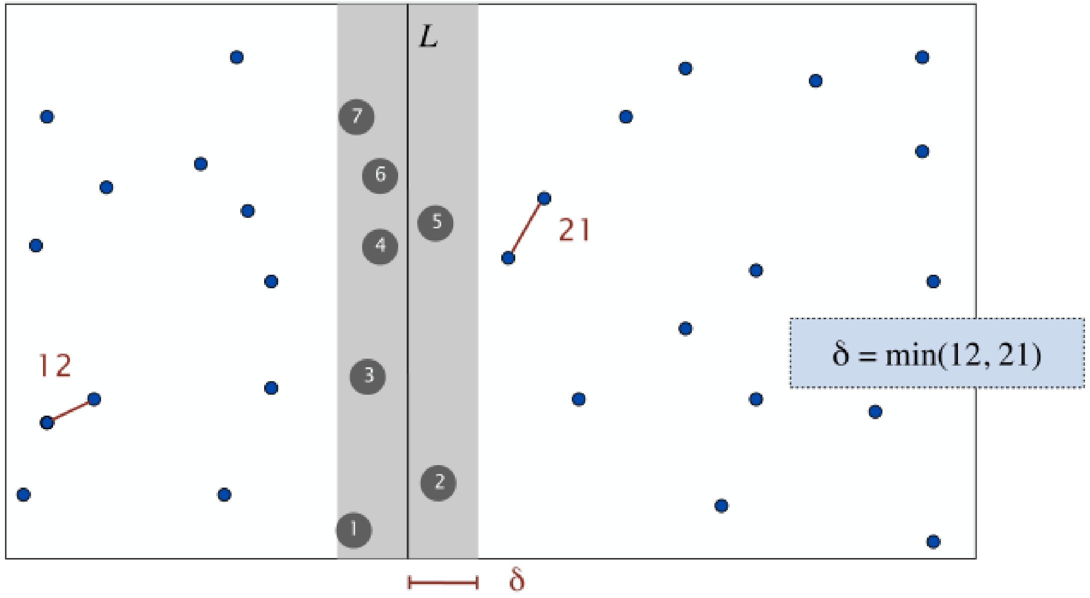
- 观察：只需考虑直线L距离δ内的那些点就足够了



如何找到两边各有一个点的最接近点对?

找出最接近的一对，两边各有一个点，假设距离< δ

- 观察：只需考虑直线L距离δ内的那些点就足够了
- 按Y坐标对2个长度为δ的带中的点进行排序
- 只检查排序列表中7个位置内的点的距离(why?)



如何找到两边各有一个点的最接近点对?

定义：假设s_i是2δ宽的带中y坐标第i小的一个点

声明：如果$|j-i|>7$，那么s_i和s_j之间的距离至少是δ

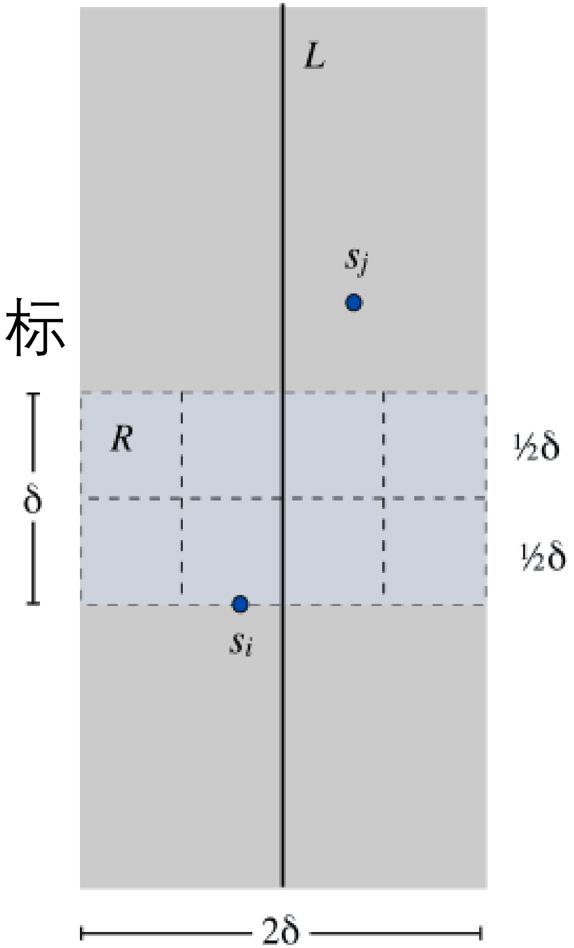
证明：

- 考虑带中$2\delta\times\delta$的矩形R，其中点的最小y坐标就是s_i的y坐标
- R中任意点s_j到s_i的距离$\geq\delta$
- 将R划分为8个方块
- 每个方块内至多有一个点

直径为
$\delta/\sqrt{2}<\delta$

- 最多可以有7个其他点可以在R中

常数可以通过更精细的
几何包装论证来改善



最近点对：分治算法

CLOSEST-PAIR($p_1, p_2, \ldots, p_n$)

Compute vertical line L such that half the points are on each side of the line.	← $O(n)$
$\delta_1 \leftarrow$ CLOSEST-PAIR(points in left half).	← $T(n / 2)$
$\delta_2 \leftarrow$ CLOSEST-PAIR(points in right half).	← $T(n / 2)$
$\delta \leftarrow \min \{ \delta_1, \delta_2 \}$.	
Delete all points further than δ from line L.	← $O(n)$
Sort remaining points by y-coordinate.	← $O(n \log n)$
Scan points in y-order and compare distance between each point and next 7 neighbors. If any of these distances is less than δ, update δ.	← $O(n)$
RETURN δ.	

分治策略：问题 6

以下递归的解是什么？

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + \Theta(n \log n) & \text{if } n > 1 \end{cases}$$

A. $T(n) = \Theta(n)$

B. $T(n) = \Theta(n \log n)$

C. $T(n) = \Theta(nlog^2 n)$

D. $T(n) = \Theta(n^2)$

最近点对算法的精炼版本

Q: 怎么改善到$O(n\log n)$?

A: 不要每次都从头开始对条带中的点进行排序

- 每个递归调用返回两个列表:所有按x坐标排序的点，和所有按y坐标排序的点
- 通过合并两个预先排序的列表进行排序

定理: [Shamos 1975]在平面上寻找最近点对的分治算法可在$O(n\log n)$时间内实现

证明:
$$T(n)=\begin{cases}\Theta(1) & \text{if } n=1\\ T(\lfloor n/2\rfloor)+T(\lceil n/2\rceil)+\Theta(n) & \text{if } n>1\end{cases}$$

分治策略：课堂测验7

二维最近点对问题的复杂度是多少？

$A.\ \Theta(n)$

$B.\ T(n) = \Theta(n \log^* n)$

$C.\ T(n) = \Theta(n \log\log n)$

$D.\ T(n) = \Theta(n \log n)$

E. 就连Tarjan也不知道

最近点对算法的计算复杂度

定理： [Ben-Or 1983, Yao 1989]在二次决策树模型中，任何最接近对（即使在一维情形中）的算法都需要$\Omega(n \log n)$二次检验

Lower Bounds for Algebraic Computation Trees

with Integer Inputs*

Andrew Chi-Chih Yao

Department of Computer Science

Princeton University

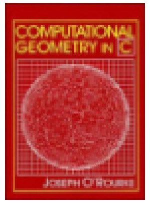
Princeton, New Jersey 08544

题外话： 计算几何

核心几何问题的巧妙分治算法：

problem	brute	clever
closest pair	$O(n^2)$	$O(n \log n)$
farthest pair	$O(n^2)$	$O(n \log n)$
convex hull	$O(n^2)$	$O(n \log n)$
Delaunay/Voronoi	$O(n^4)$	$O(n \log n)$
Euclidean MST	$O(n^2)$	$O(n \log n)$

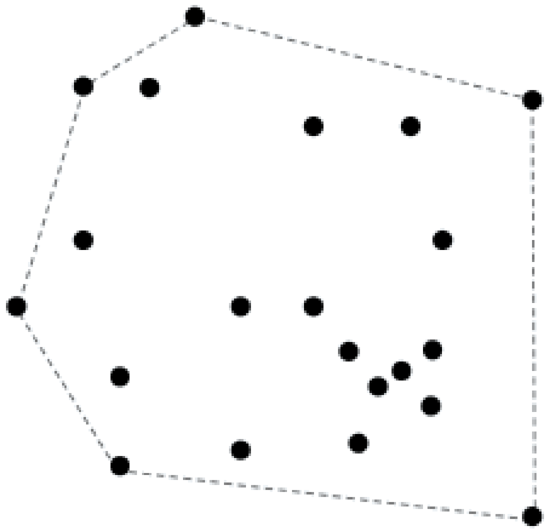
running time to solve a 2D problem with n points



注释： 三维和更高维度测试了我们创造力的极限

凸包（Convex hull）

一个包含n个点的组的凸包是最小的能将这些点围起来的周围边栏：

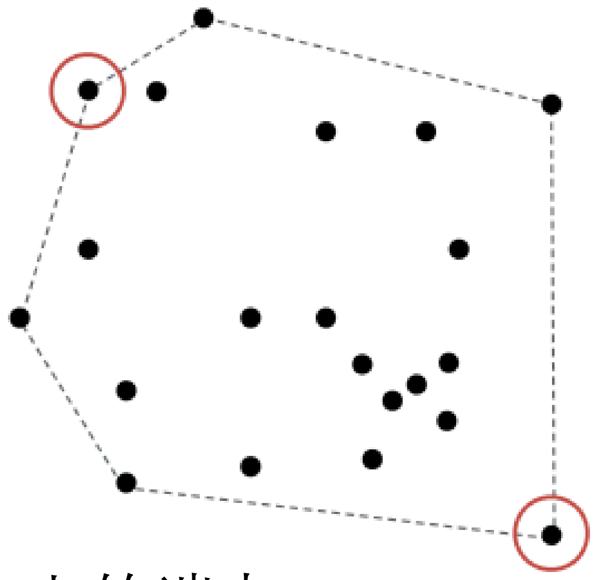


等效定义：

- 包围点的最小面积凸多边形
- 包含所有点的所有凸集的交集

最远对（Farthest pair）

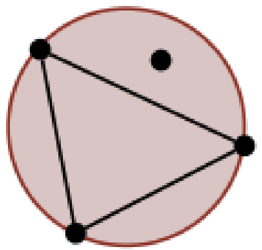
给出平面内的n个点，找出一对之间欧氏距离最大的点。

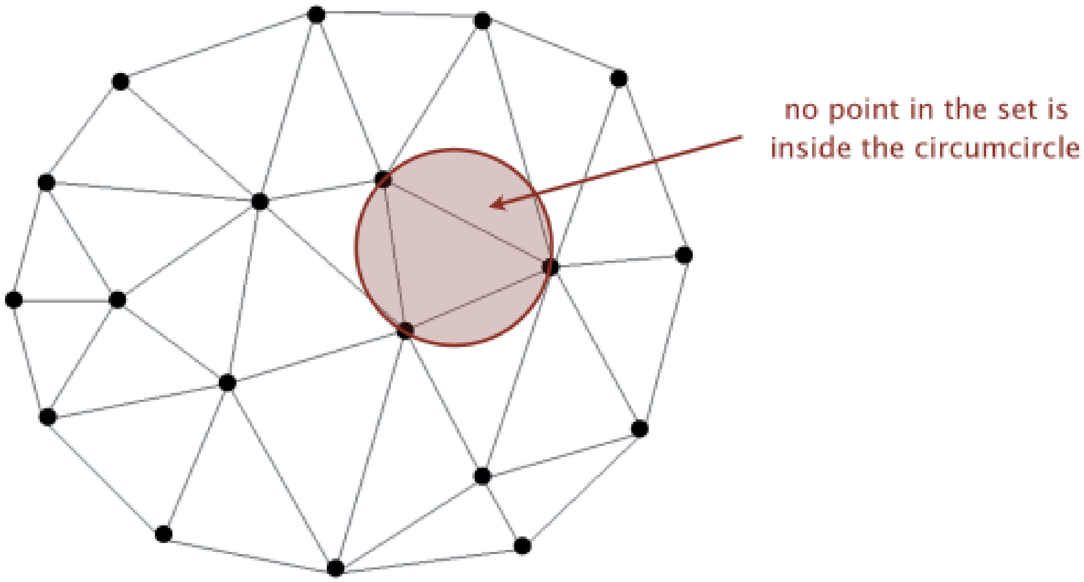


事实：最远对中的点是凸包上的端点。

Delaunay三角剖分

Delaunay三角剖分是对平面内n个点的三角划分，使得没有一个点在任何三角形的外接圆内





Delaunay triangulation of 19 points

欧几里得MST

给出平面内的n个点，找出连接它们的最小生成树
[点对之间的距离是欧几里得距离]

事实：欧几里得MST是Delaunay三角划分的子图

实现：欧几里得MST可以在$O(n\log n)$时间复杂度内完成计算

- 计算Delaunay三角划分
- 计算Delaunay三角划分的最小生成树(MST)