算法分析与设计 第四讲

分治法及相关实例分析

主要内容

- 快速排序的改进
- Fibonacci数列
- Strassen矩阵乘法
- 最大元、最小元
- 最近点对问题

快速排序的分析

- 最坏情况划分
- 最好情况划分
- 平衡的划分

```
QUICKSORT(A,p,r)
    if p<r
        then   q=PARTITIION(A,p,r)
               QUICKSORT(A,p,q-1)
               QUICKSORT(A,q+1,r)
```

PARTITION(A, p, r)

$x \leftarrow A[r]$

$i \leftarrow p-1$

for $j \leftarrow p$ to $r-1$

  do if $A[j] \leq x$

    then $i \leftarrow i+1$

      exchange $A[i] \leftrightarrow A[j]$

exchange $A[i+1] \leftrightarrow A[r]$

return $i+1$

快速排序的随机化版本

- 主要区别在于主元的选择
  - 不总是选择A[r]作为主元，而是从A[p…r]中随机选择一个元素作为主元
  - 具体操作方法是将A[r]与A[p…r]中的随机选中的一个元素交换

```
RANDOMIZED-PARTITION(A, p, r)
i ← RANDOM(p, r)
exchange A[r] ↔ A[i]
return PARTITION(A, p, r)
```

```
RANDOMIZED-QUICKSORT(A, p, r)
if p < r
   then q ← RANDOMIZED-PARTITION(A, p, r)
        RANDOMIZED-QUICKSORT(A, p, q - 1)
        RANDOMIZED-QUICKSORT(A, q + 1, r)
```

快速排序的实际应用

- 通常是用于排序的最佳实用选择
- 原地排序，在虚存环境中也可以很好工作

Fibonacci数列

$$F(n)=\begin{cases}1 & n=0\\ 1 & n=1\\ F(n-1)+F(n-2) & n>1\end{cases}$$

- 递归算法：$\Omega(\varphi^n), \varphi=(1+\sqrt{5})/2$
- 自底向上，依次计算
  - $\Theta(n)$
- 有更好的方法吗？

Fibonacci数列

- 矩阵法

$$\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n$$

矩阵乘法

- 矩阵乘法

$$\left.\begin{array}{l} A=\left[a_{ij}\right], B=\left[b_{ij}\right] \\ C=A \bullet B=[c_{ij}] \end{array}\right\} i, j=1,2,\ldots,n$$

$$c_{ij}=\sum_{k=1}^{n} a_{ik} \bullet b_{kj}$$

矩阵乘法

● 直接解法

for $i \leftarrow 1$ **to** n

do for $j \leftarrow 1$ **to** n

do $cij \leftarrow 0$

for $k \leftarrow 1$ **to** n

do $cij \leftarrow cij + aik \cdot bkj$

● 时间复杂度

$\Theta(n^3)$

矩阵乘法

- 分治策略
  - 假设n为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

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

$$C = A \bullet B$$

矩阵乘法

- 分治策略分析

$$T(n) = 8T(n/2) + \Theta(n^2)$$
$$T(n) = \Theta(n^3)$$

- 直接分治的时间复杂度并不比直接计算好

Strassen的策略

- 只需要7次子矩阵的乘法
  - **引入M_i(i=1,2,…,7)，如下**

$$M_1 = A_{11}(B_{12} - B_{22})$$

$$M_2 = (A_{11} + A_{12})B_{22}$$

$$M_3 = (A_{21} + A_{22})B_{11}$$

$$M_4 = A_{22}(B_{21} - B_{11})$$

$$M_5 = (A_{11} + A_{22})(B_{11} + B_{22})$$

$$M_6 = (A_{12} - A_{22})(B_{21} + B_{22})$$

$$M_7 = (A_{11} - A_{21})(B_{11} + B_{12})$$

$$C_{11} = M_5 + M_4 - M_2 + M_6$$

$$C_{12} = M_1 + M_2$$

$$C_{21} = M_3 + M_4$$

$$C_{22} = M_5 + M_1 - M_3 - M_7$$

Strassen矩阵乘法分析

- $T(n) = 7T(n/2) + \Theta(n^2)$

  $T(n) \approx \Theta(n^{2.81})$

- 更好的算法？

  $T(n) = \Theta(n^{2.376})$

快速排序的稳定性分析

- 2 2 2 2 2
- 49 38 49 76 13 27

最大元、最小元

- 给定n个数据元素，找出其中的最大元和最小元
  - 直接解法：逐个找，用n-1次比较来找出最大元，再用n-2次比较来找出最小元，比较次数（基本运算）为2n-3次
  - 可以用分治法吗？

最大元、最小元

- 分治法
  - 当n=2时，一次比较就可以找出两个数据元素的最大元和最小元
  - 当n>2时，可以把n个数据元素分为大致相等的两半
  - 求数组最大元、最小元的算法下界

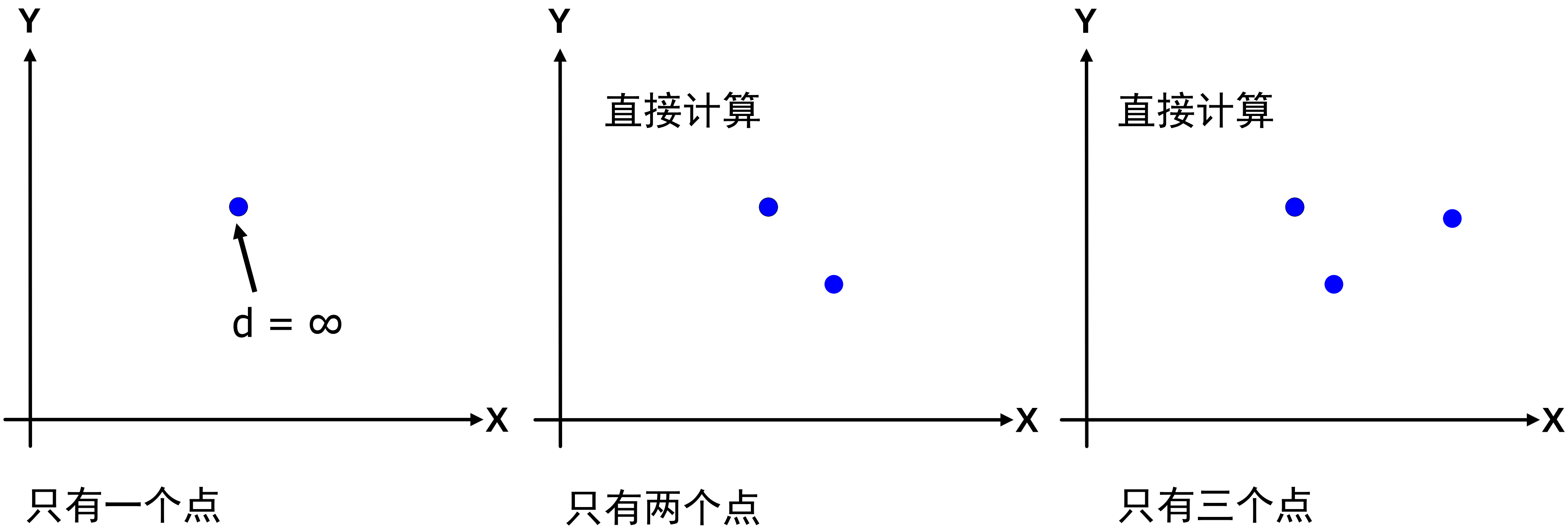
$$\lceil 3n/2-2 \rceil$$

最近点对

- 对于平面上给定的N个点，给出距离最近的两个点
  - Brute force法：把所有点对逐一检查一遍
  - 分治策略
    - 如何分解？
    - 如何合并？

最近点对

- 点数较少时的情形



最近点对

- 分解
  - 对所有的点按照x坐标从小到大排序
  - 根据下标进行分割，使点集分成两个集合

最近点对

- 解决
  - 递归寻找P_L和P_R中的最近点对
  - 设其找到的最近点对的距离分别是δ_L和 δ_R
  - 置$\delta=\min(\delta_L, \delta_R)$

最近点对

- 合并
  - 可能并不是δ，存在这样的情况，一个点在P_L中，另一个点在P_R中，而这两点之间的距离小于δ
  - 如何检查呢？
  - 只考虑分割线两侧距离各为δ的点
  - 继续压缩点的范围

最近点对

- 难点
  - 如何在线性时间内获得

    Y_L, Y_R, X_L, X_R, Y'

分治法小结

- 二分搜索
- 幂乘
- 合并排序
- 快速排序
- Fibonacci数列
- Strassen矩阵乘法
- 最大元、最小元
- 最近点对问题
- ……