



哈尔滨工业大学

海量数据计算研究中心

Massive Data Computing Lab @ HIT

# 大数据算法

## 第三讲 亚线性算法例析

哈尔滨工业大学

王宏志

wangzh@hit.edu.cn

# 本讲内容

**3.1 数据流中频繁元素**

3.2 最小生成树

3.3 序列有序的判定



# 大数据的数据流模型

- 数据只能顺序扫描1次或几次
- 能够使用的内存是有限的
- 希望通过维护一个内存结果(概要)来给出相关性质的一个有效估计
- 数据流模型适用于大数据
  - 顺序扫描数据仅一次
  - 内存亚线性



# 数据流模型

- 来自某个域中的元素序列

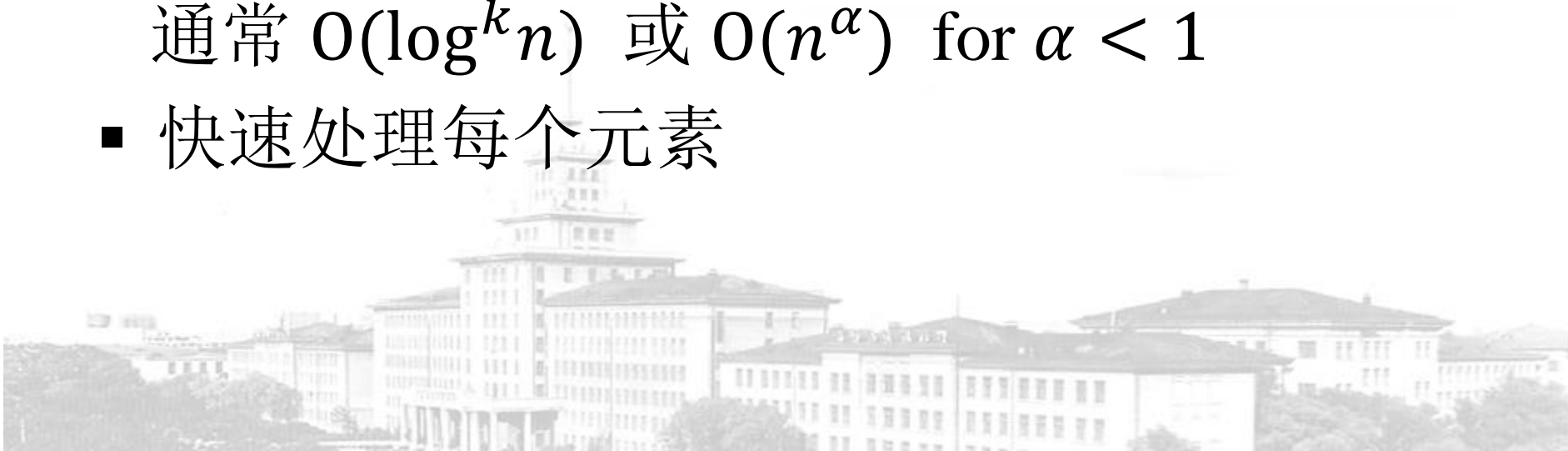
$\langle x_1, x_2, x_3, x_4, \dots \rangle$

- 有限的内存:

内存  $\ll$  数据的规模

通常  $O(\log^k n)$  或  $O(n^\alpha)$  for  $\alpha < 1$

- 快速处理每个元素



# 从数据流中计算什么？

32, 112, 14, 9, 37, 83, 115, 2,

容易计算的函数: min, max, sum, ...

使用单个寄存器  $s$ , 直接更新:

- max: 初始化  $s \leftarrow 0$

对于元素  $x$ ,  $s \leftarrow \max\{s, x\}$

- sum: 初始化  $s \leftarrow 0$

对于元素  $x$ ,  $s \leftarrow s + x$

➤ “概要”是单个值

➤ 是可合并的

# 频繁元素

32, 12, 14, 32, 7, 12, 32, 7, 6, 12, 4,

- 元素出现多次，希望找到出现最频繁的元素
- $n$ : 不同元素的数量
- $m$ : 数据流中元素个数



# 频繁元素

32, 12, 14, 32, 7, 12, 32, 7, 6, 12, 4,

应用:

- 网络: 找到 “elephant flow”
- 搜索: 找到频繁查询

**Zipf原则:** 典型的频率分布是高度偏斜的, 只有少数频繁元素.

最多10%的元素占元素总个数的 90%.

我们发现出现次数最多的元素

# 频繁元素: 精确解

32, 12, 14, 32, 7, 12, 32, 7, 6, 12, 4,

精确解:

- 对每一个单独元素设置一个计数器
- 当处理一个元素时, 增加相应计数器

32	12	14	7	6	4
●	●	●	●	●	●
●	●		●		
●	●				

**问题:** 需要维护  $n$  个计数器  
但只能有  $k \ll n$  个计数器



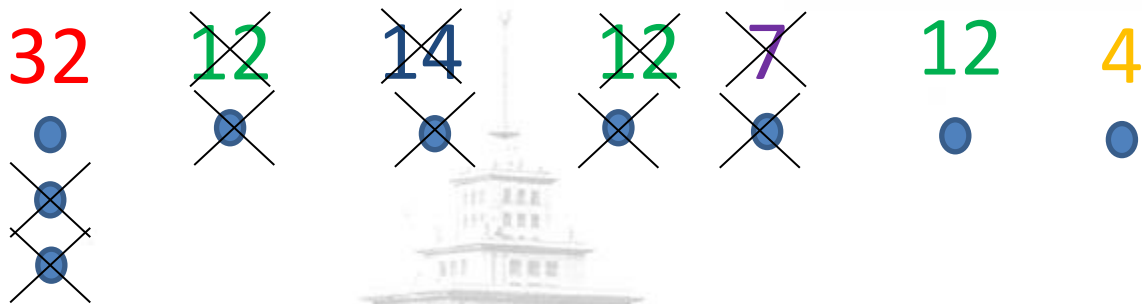
# 频繁元素计算算法

## Misra Gries(MG)算法

32, 12, 14, 32, 7, 12, 32, 7, 6, 12, 4,

处理元素  $x$

- If 已经为  $x$  分配计数器, 增加之
- Else If 没有相应计数器, 但计数器个数少于  $k$ , 为  $x$  分配计数器, 并设为 1.
- Else, 所有计数器减 1. 删除值为 0 的计数器.



$$\begin{aligned} n &= 6 \\ k &= 3 \\ m &= 11 \end{aligned}$$

# 频繁元素算法

32, 12, 14, 32, 7, 12, 32, 7, 6, 12, 4,

处理元素  $x$

- If 已经为  $x$  分配计数器, 增加之
- Else If 没有相应计数器, 但计数器个数少于  $k$ , 为  $x$  分配计数器, 并设为 1.
- Else, 所有计数器减 1. 删除值为 0 的计数器.

$x$  出现几次?

- If 我们有一个  $x$  的计数器, 返回其值
- Else, 返回 0.

该估计显然过低  
如何精确估计?

# 分析

## 一个计数器 $x$ 减少了几次?

$\Leftrightarrow$  我们有几个减少计数器的步骤?

- 整个结构的权重(计数器的和)记作 $m'$
- 整个数据流的权重(全部元素的数量)是 $m$
- 每一个计数器降低的步骤减少 $k$ 个计数, 但是并未计入输入元素的此次出现, 即 $k + 1$ 次未计入的元素出现.

$\Rightarrow$  最多有 $\frac{m-m'}{k+1}$  个减少步骤

$\Rightarrow$  估计值和真实值相差最多  $\frac{m-m'}{k+1}$

# 分析

估计值与真实值相差最多  $\frac{m-m'}{k+1}$

⇒当数据流中元素的总数  $\gg \frac{m-m'}{k+1}$  时，得到  $x$  的一个好的估计

- 错误的界限和  $k$  成反比
- 利用概要计算错误的界限：记录  $m$ ，计算  $m'$  和  $k$ .
- 该算法有效的原因：“Zipf原则”



# 本讲内容

3.1 数据流中频繁元素

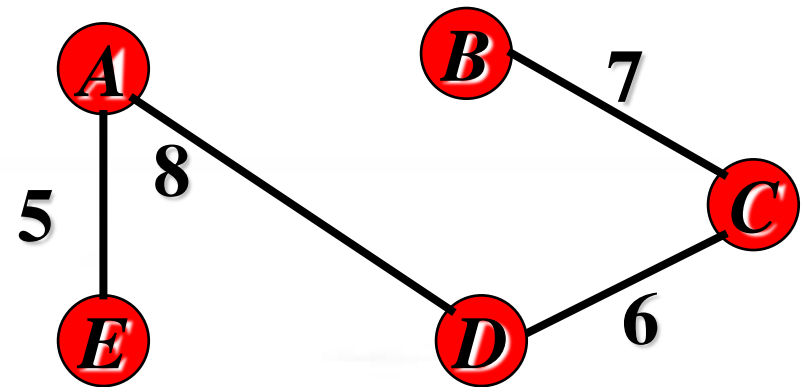
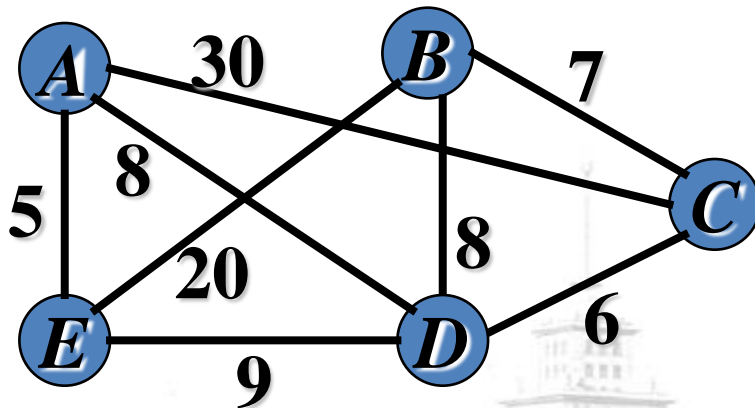
**3.2 最小生成树**

3.3 序列有序的判定



# 问题的定义

- 输入：无向有权联通图 $G=(V, E)$ ，其顶点的度最大为 $D$ ，边上的权来自整数集合 $\{1, \dots, W\}$
- 输出：图 $G$ 的最小生成树的权重



# 精确解

- 贪心法
  - Prime算法
  - Kruskal算法
- 时间复杂性:  $O(m \log n)$
- 超过线性



# 亚线性算法的假设

- 图组织成邻接表的形式
  - 可以直接访问每个结点的邻居
- 可以随机均匀地选择结点





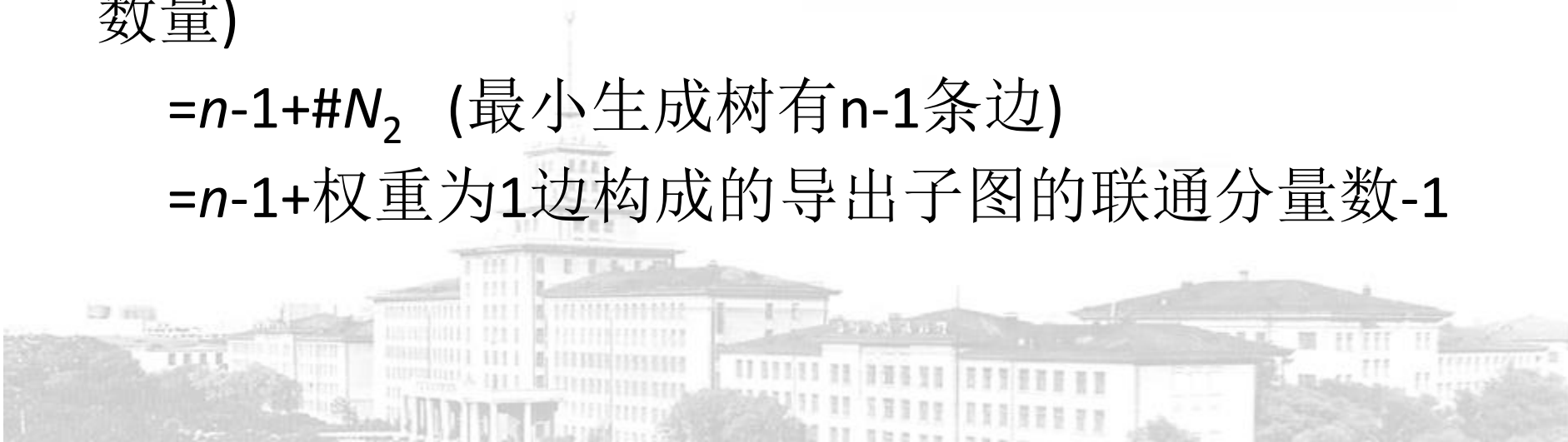
# 时间亚线性算法的思想

- 利用特定子图联通分量的数量估计最小生成树的权重
- 假设所有边的权重都是1或者2，最小生成树的权重

$= \#N_1 + \#N_2$  ( $\#N_i$ : 最小生成树中权重至少为 $i$ 的边的数量)

$= n-1 + \#N_2$  (最小生成树有 $n-1$ 条边)

$= n-1 + \text{权重为1边构成的导出子图的联通分量数} - 1$



# 最小生成树和连通分量的关系

- 一般的情况
  - $G_i$ :  $G$ 中包含所有权重小于 $i$ 的边的子图
  - $C_i$ :  $G_i$ 中的连通分量数
  - 最小生成树权重大于 $i$ 的边数为 $C_i-1$
- $W_{MST}(G) = n - w + \sum_{i=1}^{w-1} C_i$

证明:

令 $\beta_i$ 为最小生成树中权重大于 $i$ 的边的个数

每一条MST边对WMST基础贡献为1, 每个权重大于1的边额外贡献了1, 每条权重大于2的边贡献的更多, 因此

$$W_{MST}(G) = \sum_{i=0}^{w-1} \beta_i = \sum_{i=0}^{w-1} (C_i - 1) = -w + \sum_{i=0}^{w-1} C_i = n - w + \sum_{i=1}^{w-1} C_i$$

# 基础算法：连通分量个数的估计

- 输入：图  $G=(V, G)$ ，有  $n$  个顶点，表示为邻接矩阵，结点最大度为  $d$
- 输出：连通分量的个数
- 精确解时间复杂性：  $\Omega(dn)$
- 利用随机化方法
- 估计连通分量个数  $\#CC$ 
  - $\#CC \pm \varepsilon n$  的概率  $\geq 2/3$
  - 运行时间和  $n$  无关

# 估计连通分量的方法：核心思想

- $C$ : 连通分量的个数
- 对于每个结点  $u$ ,  $n_u$ :  $u$  所在连通分量的结点数
- 对于每个连通分量:  $\sum_{u \in A} \frac{1}{n_u} = 1$ ,
- 故:  $\sum_{u \in V} \frac{1}{n_u} = C$
- 通过估计抽样顶点的  $n_u$  来估计这个和
  - 如果  $u$  所在的分量很小, 其规模可以通过 BFS 估计
  - 如果  $u$  所在连通分量很大,  $1/n_u$  很小, 对和的贡献很小
  - 可以在几步以内完成 BFS



# 每个u所在连通分量结点数的估计

- 令  $\hat{n}_u = \min\{n_u, 2/\varepsilon\}$ 
  - 当结点数小于  $2/\varepsilon$  时,  $\hat{n}_u = n_u$
  - 否则,  $\hat{n}_u = 2/\varepsilon$ , 因此  $0 < \frac{1}{\hat{n}_u} - \frac{1}{n_u} < \frac{1}{\hat{n}_u} = \frac{\varepsilon}{2}$
- 在这种情况下, 对C的估计

$$\hat{C} = \sum \frac{1}{\hat{n}_u}$$

- 则  $|\hat{C} - C| = \left| \sum \left( \frac{1}{\hat{n}_u} - \frac{1}{n_u} \right) \right| \leq \frac{\varepsilon n}{2}$

# 连通分量数估计算法

$CC(G, d, \varepsilon)$

1. for  $i=1$  to  $s=\theta(\frac{1}{\varepsilon^2})$  do
2.     随机选择点  $u$
3.     从  $u$  开始BFS，将访问到的顶点存到排序序列  $L$  中，访问完连通分量或  $L=2/\varepsilon$  时停止， $\hat{n}_u = |L|$
4.      $N=N+\hat{n}_u$
5. 返回  $\tilde{C} = s/N \times n$

运行时间:  $O(\frac{d}{\varepsilon^3} \log \frac{1}{\varepsilon})$

# 联通分量近似计数的分析

- 分析的目的:  $\Pr[|\tilde{C} - \hat{C}| > \frac{\varepsilon n}{2}] \leq \frac{1}{3}$ 
  - 估计值和真实值相差过大的概率很小
- 对于采样中的第 $i$ 个结点 $u$ , 令 $Y_i = 1/\hat{n}_u$
- $Y = \sum_{i=1}^s Y_i = \frac{s\tilde{C}}{n}$
- $E[Y] = \sum_{i=1}^s E[Y_i] = sE[Y_1] = s \cdot \frac{1}{n} \sum_{u \in V} \frac{1}{\hat{n}_u} = \frac{s\hat{C}}{n}$

$$\begin{aligned} \Pr\left[|\tilde{C} - \hat{C}| > \frac{\varepsilon n}{2}\right] &= \Pr\left[\left|\frac{n}{s}Y - \frac{n}{s}E[Y]\right| > \frac{\varepsilon n}{2}\right] \\ &= \Pr\left[|Y - E[Y]| > \frac{\varepsilon s}{2}\right] \end{aligned}$$

# 联通分量近似计数的分析(续)

Hoeffding界

$Y_1, \dots, Y_s$  为  $[0, 1]$  区间内独立同分布的随机变量, 令  $Y = \sum_{i=1}^s Y_i$ , 则  $\Pr[|Y - E[Y]| \geq \delta] \leq 2e^{-2\delta^2/s}$

$$\Pr\left[|\tilde{C} - \hat{C}| > \frac{\varepsilon n}{2}\right] = \Pr[|Y - E[Y]| > \frac{\varepsilon S}{2}] \\ \leq 2e^{-\frac{\varepsilon^2 s}{2}}$$

$$s = \theta\left(\frac{1}{\varepsilon^2}\right) \Rightarrow \Pr\left[|\tilde{C} - \hat{C}| > \frac{\varepsilon n}{2}\right] \leq \frac{1}{3}$$



# 连通分量近似技术的分析(续)

$$\Pr[|\tilde{C} - \hat{C}| > \frac{\varepsilon n}{2}] \leq \frac{1}{3}$$

$$|\hat{C} - C| \leq \frac{\varepsilon n}{2}$$

因此，下列事件发生的概率大于2/3:

$$|\tilde{C} - C| \leq |\tilde{C} - \hat{C}| + |\hat{C} - C| \leq \frac{\varepsilon n}{2} + \frac{\varepsilon n}{2} = \varepsilon n$$

综上所述，有n个顶点的图中，若其顶点的度至多为d，则其连通分量的数量估计误差最多为 $\pm \varepsilon n$

# 最小生成树近似算法

1. for  $i=1$  to  $w-1$  do

2.  $\tilde{C}_i = CC\left(G_i, d, \frac{\varepsilon}{w}\right)$

3. return  $\tilde{w}_{MST} = n - w + \sum_{i=1}^{w-1} \tilde{C}_i$

分析:

- 假设  $C_i$  的所有估计都是正确的,  $|\tilde{C}_i - C_i| \leq \frac{\varepsilon}{w}n$ , 则
$$|\tilde{w}_{MST} - w_{MST}| = |\sum_{i=1}^{w-1} (\tilde{C}_i - C_i)| \leq \sum_{i=1}^{w-1} |\tilde{C}_i - C_i| \leq w \cdot \frac{\varepsilon}{w}n = \varepsilon n$$
- $\Pr[\text{所有 } w-1 \text{ 次估计都正确}] \geq (2/3)^{w-1}$
- 这并不够好! 可以通过CC算法中, 取合适的  $s$  值, 使得每一轮的错误概率  $\leq \frac{1}{3w}$ , 应当取多少呢?
- 利用并集界限,  $\Pr[\text{error}] \leq w \cdot \frac{1}{3w} = \frac{1}{3}$

并集界限

对于事件  $A_1, \dots, A_n$

$$P\left(\bigcup_i A_i\right) \leq \sum_i P(A_i)$$

# 乘近似

- 对于MST的代价，可以从加的近似导出乘的近似

- $w_{MST} \geq n-1 \Rightarrow w_{MST} \geq n/2 \ (n \geq 2)$

- $\varepsilon n$ 加近似

- $w_{MST} - \varepsilon n \leq \hat{w}_{MST} \leq w_{MST} + \varepsilon n$

- $(1 \pm 2\varepsilon)$ 乘近似

- $w_{MST}(1 - 2\varepsilon) \leq w_{MST} - \varepsilon n \leq \hat{w}_{MST} \leq w_{MST} + \varepsilon n \leq w_{MST}(1 + 2\varepsilon)$



# 本讲内容

3.1 数据流中频繁元素

3.2 最小生成树

**3.3 序列有序的判定**



# 数组有序性判定

- 输入：  $n$  个数的数组,  $x_1, x_2, \dots, x_n$
- 输出： 这个数组是否有序？
  - 需要访问这  $n$  个数，时间是  $\Omega(n)$
- 近似版本
  - 这个数组是有序的还是  $\varepsilon$  远离有序的？
- $\varepsilon$  远离
  - 我们必须删除大于  $\varepsilon n$  个元素才能保证剩下的元素有序



# 亚线性算法

**for**  $k=1$  to  $2/\varepsilon$  **do**

选择数组中第 $i$ 个元素 $x_i$

用 $x_i$ 在数组中做二分查找

**if** 发现 $i < j$  但是 $x_i > x_j$  **then** //碰到了“坏”索引

**return** false

**return** true

算法的时间复杂性:  $O(\frac{1}{\varepsilon} \log n)$

# 算法精确性

- 输入数列有序，则总返回True
- 下面证明：当输入数列 $\epsilon$ 远离有序时，算法返回false的概率大于2/3
- **证据引理：**如果一次测试以大于等于 $p$ 的概率获得一个证据，那么 $s=2/p$ 轮测试得到证据的概率大于等于2/3

首先证明：如果输入 $\epsilon$ 远离有序，则存在大于 $\epsilon n$ 个“坏”索引



# 性质的证明

- 如果输入 $\varepsilon$ 远离“有序”，则存在大于 $\varepsilon n$ 个“坏”索引

证明：我们证明其逆否命题，即如果“坏”索引的个数小于 $\varepsilon n$ ，则其存在一个长度大于 $\varepsilon n$ 的单调递增子序列。

对于任意“好”索引 $i$ 和 $j$ ， $x_i < x_j$

令 $k$ 是在二分搜索中将 $x_i$ 和 $x_j$ 分开的最近顶点，也就是对于整个数组建一个二分搜索树，在二分搜索树中 $x_i$ 和 $x_j$ 的最近公共祖先，则 $i < k < j$ ，因为 $i$ 和 $j$ 都是“好”索引，那么 $x_i < x_k < x_j$



# 性质的证明(续)

- 当输入数列 $\varepsilon$ 远离有序时，算法返回false的概率大于 $2/3$

证明: 求证算法返回true的概率小于 $1/3$

我们已经证明，如果输入 $\varepsilon$ 远离有序，则存在大于 $\varepsilon n$ 个“坏”索引，即数组中“坏”索引的概率大于 $\varepsilon$ 。

当数组中“坏”索引的概率大于 $\varepsilon$ 时，选择的索引都是好的概率小于 $(1 - \varepsilon)^{2/\varepsilon} < e^{-2} < \frac{1}{3}$

# 致谢

---

- 本讲义部分内容来自于Edith Cohen和Sofya Raskhodnikova的讲义

