大数据近似算法

- 研究背景与计算模型
- 随机采样算法
- 基于计数的近似算法
- 基于哈希的近似算法
- 研究成果简介

基于计数的近似算法

- 多数问题(Majority)
- Misra-Gries(MG)算法
- MG合并算法

多数问题(Majority)

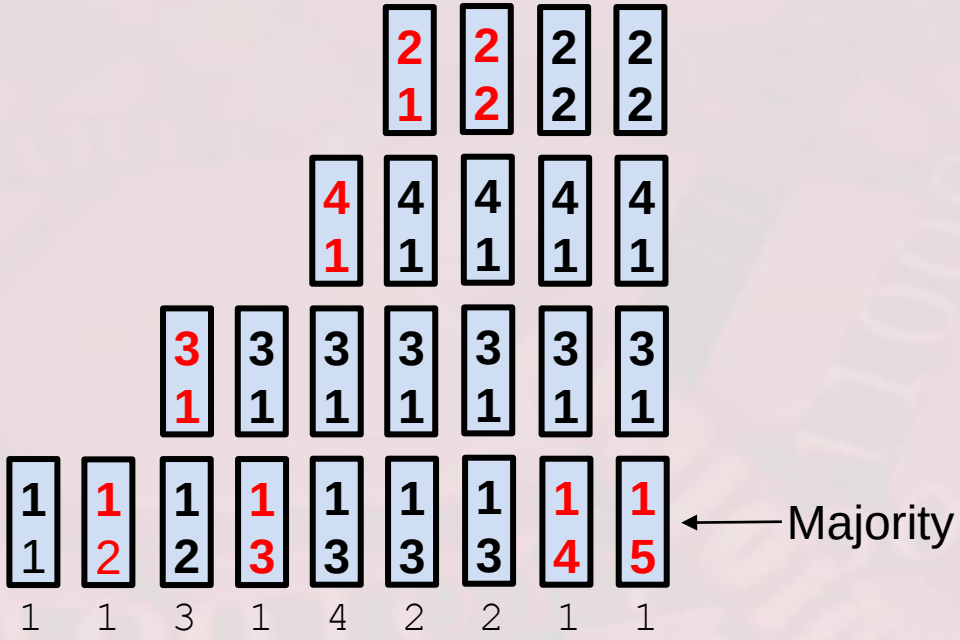
- 输入：N个元素
- 输出：Majority, 即出现次数超过N/2的元素
- 算法1：扫描所有元素，给每个出现过的元素分配一个计数器记录其出现次数，如果某个元素出现次数超过N/2，即为Majority。

Majority =1

1 1 3 1 4 2 2 1 1

多数问题(Majority)

❖ 算法运行过程：



多数问题(Majority)

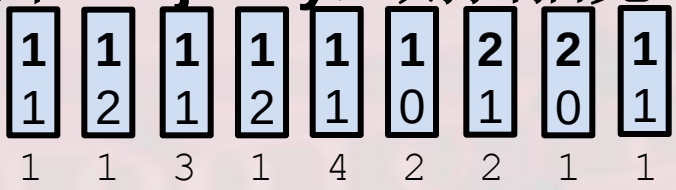
- 输入：N个元素
- 输出：Majority, 即出现次数超过N/2的元素
- 算法1：扫描所有元素，给每个出现过的元素分配一个计数器记录其出现次数，如果某个元素出现次数超过N/2，即为Majority。
- 时间复杂度：O(N)(扫描一遍)
- 空间复杂度：O(N)

多数问题(Majority)

- 输入：N个元素
- 输出：Majority, 即出现次数超过N/2的元素
- 算法2：对每个出现过的元素，扫描一遍N个元素，并记录其出现次数。如果某个元素出现次数超过N/2，即为Majority。
- 时间复杂度：$O(N^2)$(如果N个元素都不相同，需要扫描N遍)
- 空间复杂度：O(1)

多数问题(Majority)

- 输入：N个元素
- 输出：Majority, 即出现次数超过N/2的元素
- 算法3：扫描所有元素，储存一个计数器与对应元素。当扫描到的元素与存储元素相同时，给计数器加1；当扫描到的元素与存储元素不同时，给计数器减一；当计数器归零时，重新开始。
- 性质：如果存在Majority，则扫描完毕时，存储元素一定是Majority。



多数问题(Majority)

- 输入：N个元素
- 输出：Majority, 即出现次数超过N/2的元素
- 算法3：扫描所有元素，储存一个计数器与对应元素。当扫描到的元素与存储元素相同时，给计数器加1；当扫描到的元素与存储元素不同时，给计数器减一；当计数器归零时，重新开始。
- 扫描第二遍，确定候选元素是否为Majority
- 时间复杂度：O(N), 空间复杂度：O(1)

Majority算法分析

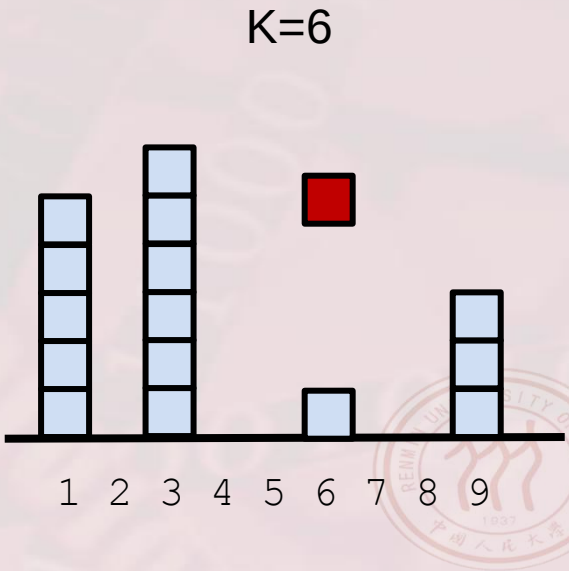
- 定理：如果一个元素出现次数超过N/2，那么它一定会被存储。
  - N = 数据流中的元素个数
  - 每次减一操作可以看成将两个不同元素抵消（当前计数器中的元素以及当前扫描到的元素）
  - 总共有N个元素可以抵消
  - 最多能执行N/2次减一操作
  - 如果一个元素出现次数超过N/2，那么它最多被减N/2次，因此一定会被存储。

Misra-Gries算法

- ❖ 输入：N个元素，整数k
- ❖ 输出：出现次数超过N/k的元素

算法：保存k-1个元素与其对应的计数器。对于一个新元素

- 如果该元素已被记录，则将该元素对应的计数器加一
- 否则，如果未满k-1个元素，则记录新元素，计数器设为一
- 否则，所有计数器减一，移除计数器为0的元素。

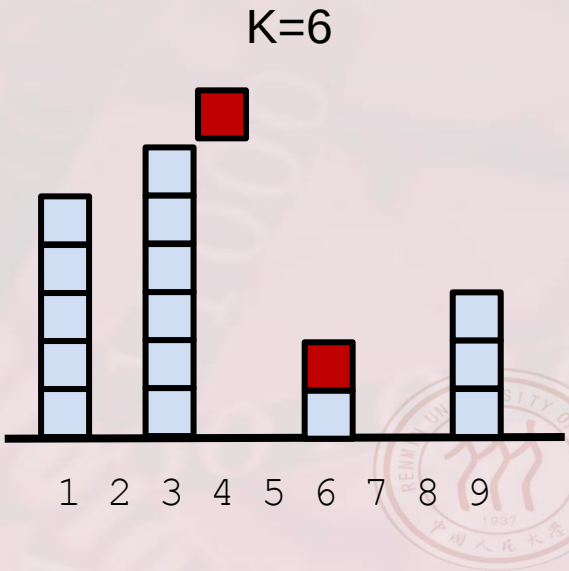


Misra-Gries算法

- ❖ 输入：N个元素，整数k
- ❖ 输出：出现次数超过N/k的元素

算法：保存k-1个元素与其对应的计数器。对于一个新元素

- 如果该元素已被记录，则将该元素对应的计数器加一
- 否则，如果未满k-1个元素，则记录新元素，计数器设为一
- 否则，所有计数器减一，移除计数器为0的元素。

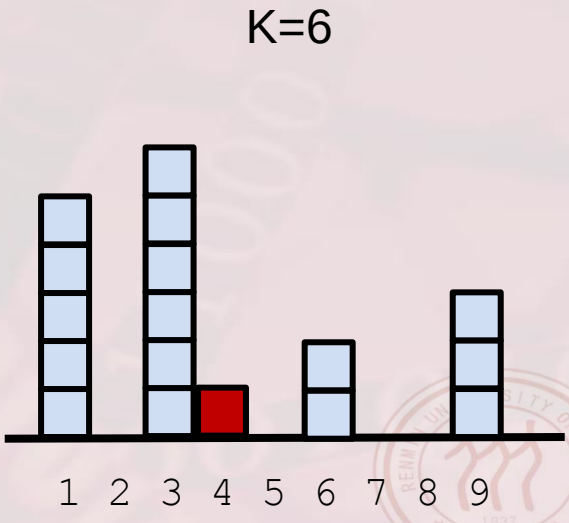


Misra-Gries算法

- ❖ 输入：N个元素，整数k
- ❖ 输出：出现次数超过N/k的元素

算法：保存k-1个元素与其对应的计数器。对于一个新元素

- 如果该元素已被记录，则将该元素对应的计数器加一
- 否则，如果未满k-1个元素，则记录新元素，计数器设为一
- 否则，所有计数器减一，移除计数器为0的元素。

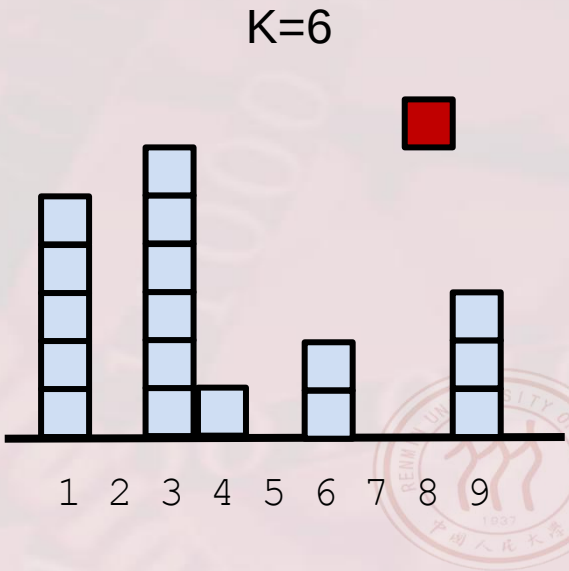


Misra-Gries算法

- ❖ 输入：N个元素，整数k
- ❖ 输出：出现次数超过N/k的元素

算法：保存k-1个元素与其对应的计数器。对于一个新元素

- 如果该元素已被记录，则将该元素对应的计数器加一
- 否则，如果未满k-1个元素，则记录新元素，计数器设为一
- 否则，所有计数器减一，移除计数器为0的元素。

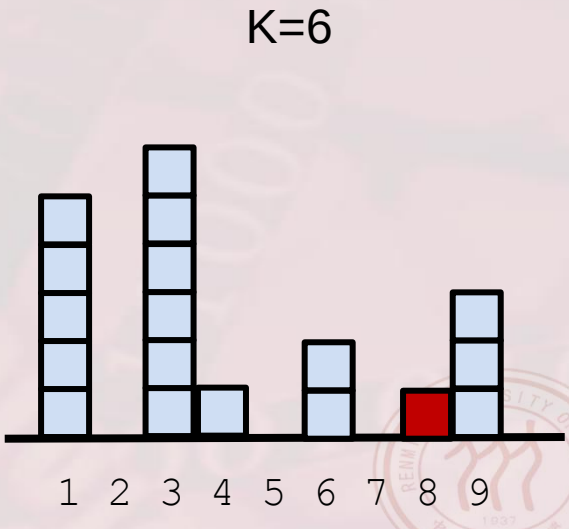


Misra-Gries算法

- ❖ 输入：N个元素，整数k
- ❖ 输出：出现次数超过N/k的元素

算法：保存k-1个元素与其对应的计数器。对于一个新元素

- 如果该元素已被记录，则将该元素对应的计数器加一
- 否则，如果未满k-1个元素，则记录新元素，计数器设为一
- 否则，所有计数器减一，移除计数器为0的元素。

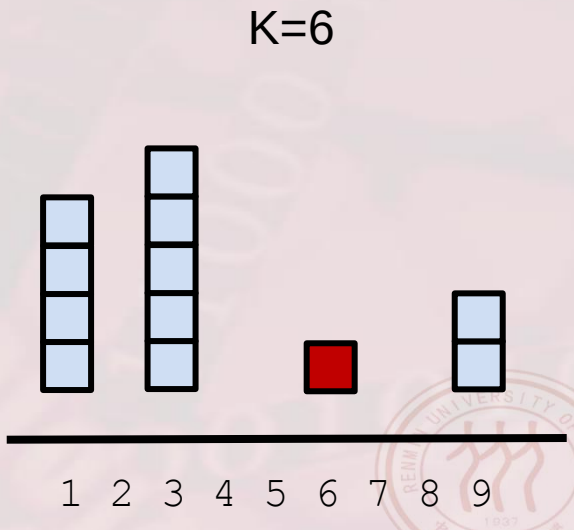


Misra-Gries算法

- ❖ 输入：N个元素，整数k
- ❖ 输出：出现次数超过N/k的元素

算法：保存k-1个元素与其对应的计数器。对于一个新元素

- 如果该元素已被记录，则将该元素对应的计数器加一
- 否则，如果未满k-1个元素，则记录新元素，计数器设为一
- 否则，所有计数器减一，移除计数器为0的元素。

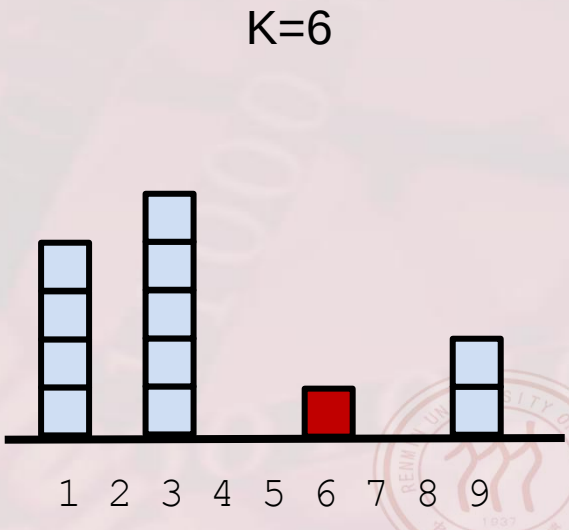


Misra-Gries算法

- 输入：N个元素，整数k
- 输出：出现次数超过N/k的元素

算法：保存k-1个元素与其对应的计数器。对于一个新元素

- 如果该元素已被记录，则将该元素对应的计数器加一
- 否则，如果未满k-1个元素，则记录新元素，计数器设为一
- 否则，所有计数器减一，移除计数器为0的元素。

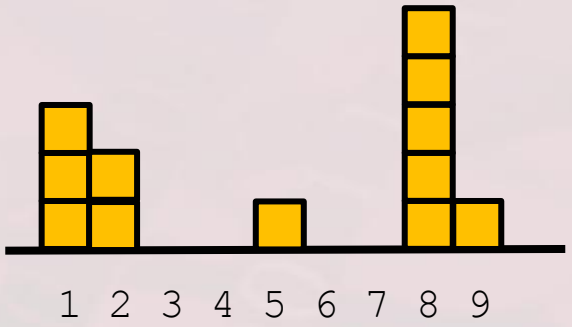


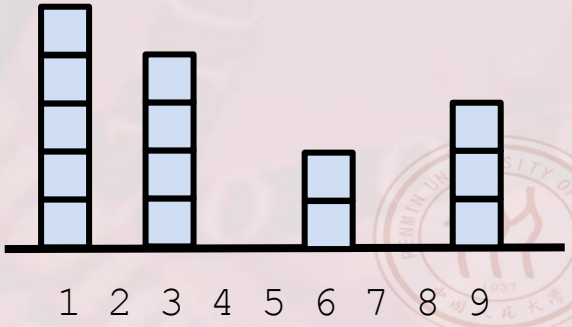
Misra-Gries算法分析

- 定理：如果一个元素未被记录，它的出现次数不会超过N/k
  - N = 数据流中的元素个数
  - 误差 ≤ 所有元素减一出现的次数
  - 每次减一，我们减去了 k 个元素: 1 个新元素和 k-1个老元素
  - 最多能减掉 N 个元素
  - 最多能产生N/k次所有元素减一操作
  - 1% 误差 -> 99 个计数器

Misra-Gries合并算法

- 输入：两个摘要MG_1和MG_2，参数为k，
- 元数据大小为N_1和N_2
- 输出：合集的摘要MG_{12}

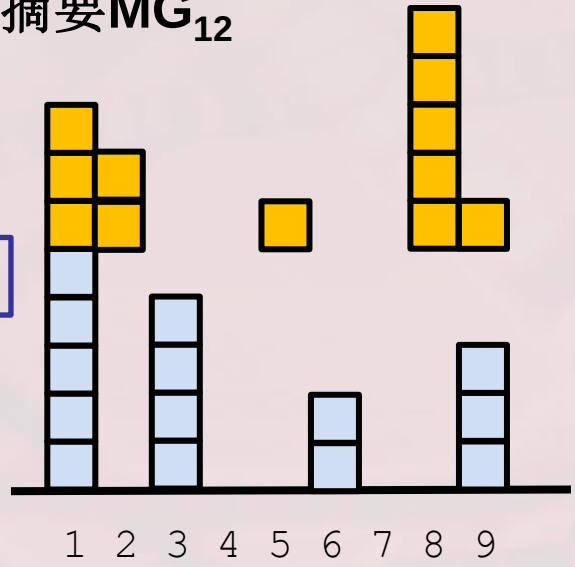


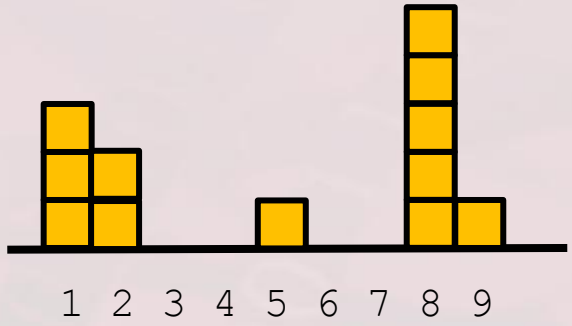


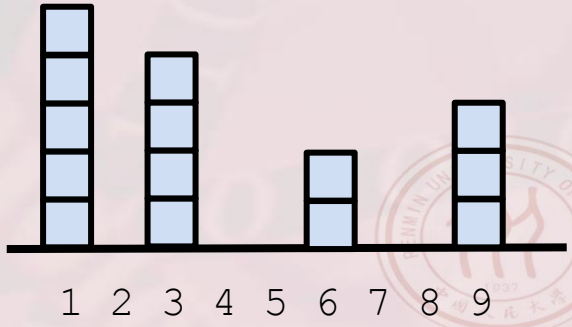
Misra-Gries合并算法

- 输入：两个摘要MG_1和MG_2，参数为k，
- 元数据大小为N_1和N_2
- 输出：合集的摘要MG_{12}

1. 对应计数器相加



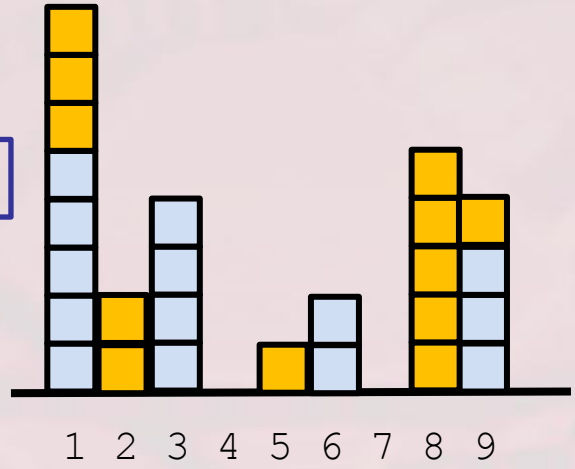


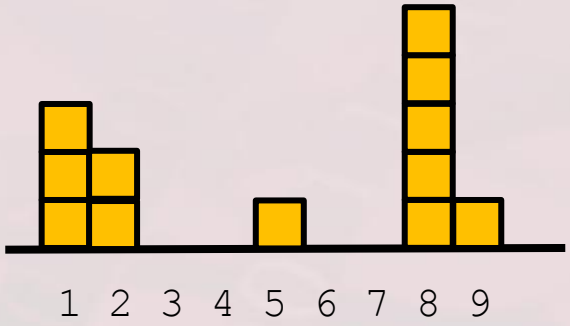


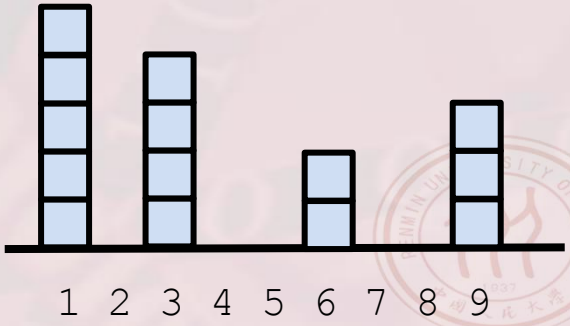
Misra-Gries合并算法

- 输入：两个摘要MG_1和MG_2，参数为k，
- 元数据大小为N_1和N_2
- 输出：合集的摘要MG_{12}

1. 对应计数器相加

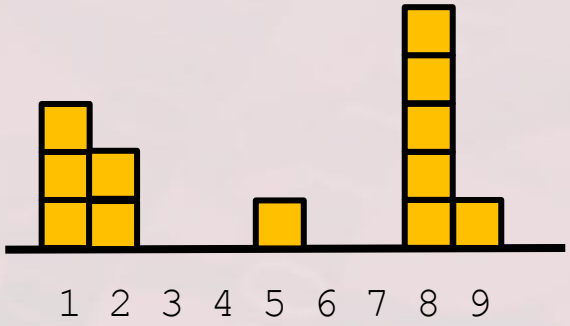




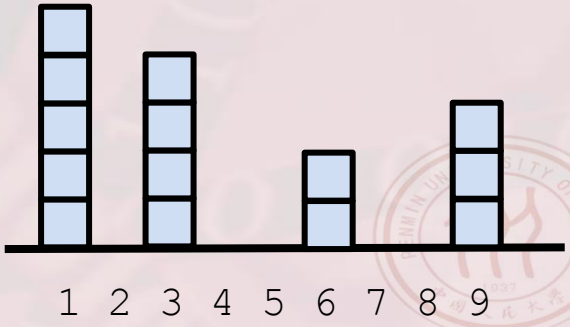


Misra-Gries合并算法

- 输入：两个摘要MG_1和MG_2，参数为k，
- 元数据大小为N_1和N_2
- 输出：合集的摘要MG_{12}



2. 找出第k大计数器C_k，从所有计数器中减去C_k

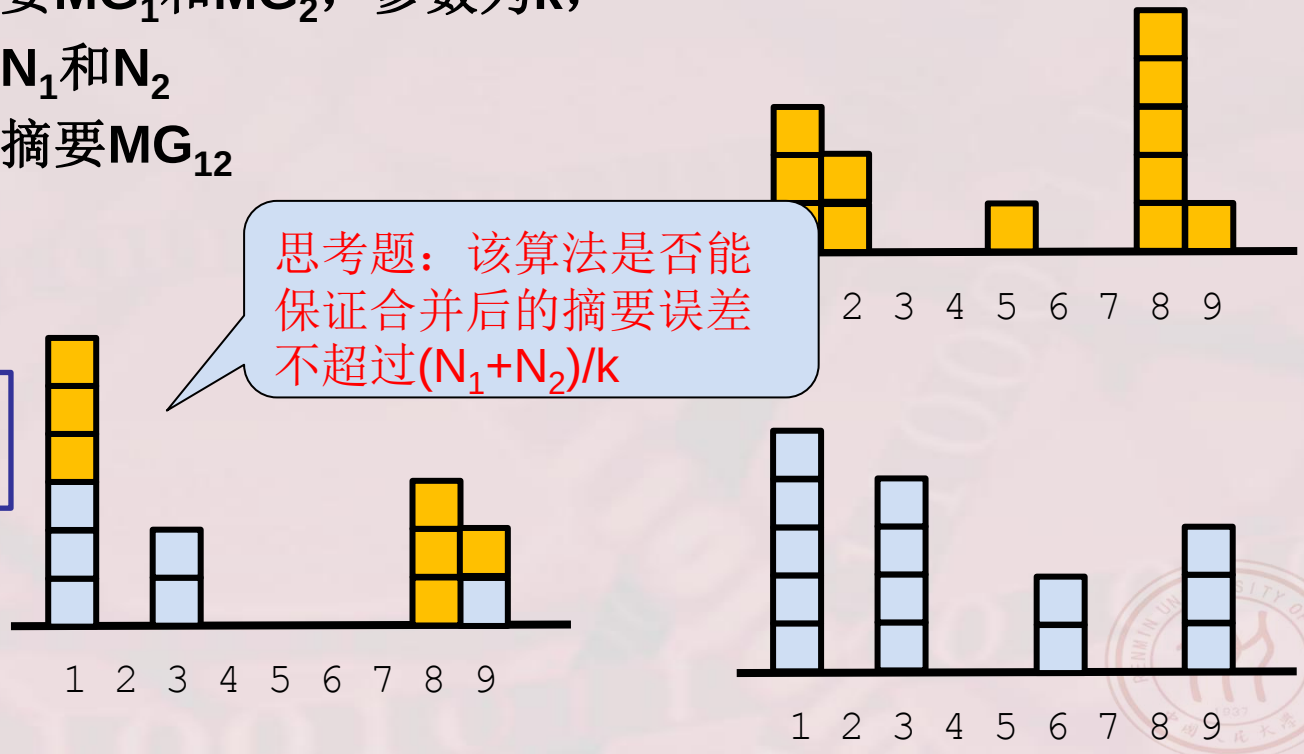


Misra-Gries合并算法

- 输入：两个摘要MG_1和MG_2，参数为k，
- 元数据大小为N_1和N_2
- 输出：合集的摘要MG_{12}

思考题：该算法是否能保证合并后的摘要误差不超过$(N_1+N_2)/k$

3. 移除负数或者零值计数器



Misra-Gries算法分析

- 定理：如果一个元素未被记录，它的出现次数不会超过(N-M)/k
  - M为MG算法中计数器之和
  - N = 数据流中的元素个数
  - 误差 ≤ 所有元素减一出现的次数
  - 每次减一，我们减去了 k 个元素: 1 个新元素和 k-1个老元素
  - 最多能减掉 N -M个元素
  - 最多能产生(N-M)/k次所有元素减一操作
  - 1% 误差 -> 99 个计数器

Misra-Gries合并算法分析

- 定理：如果一个元素未被记录，它的出现次数不会超过(N-M)/k
  - 合并算法从计数器之和中减去了kC_k