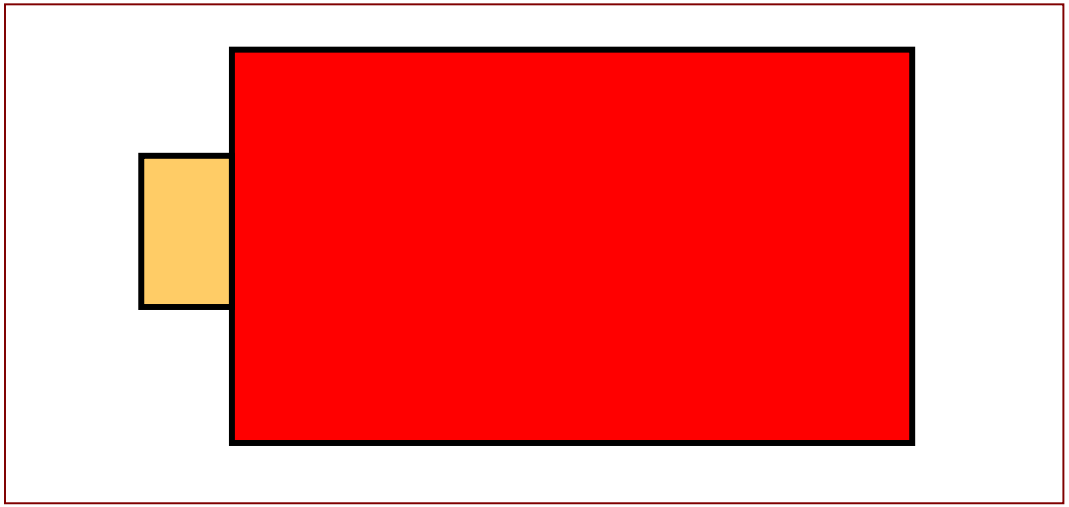
回到起点——一种突破性思维

南京市外国语学校 朱泽园

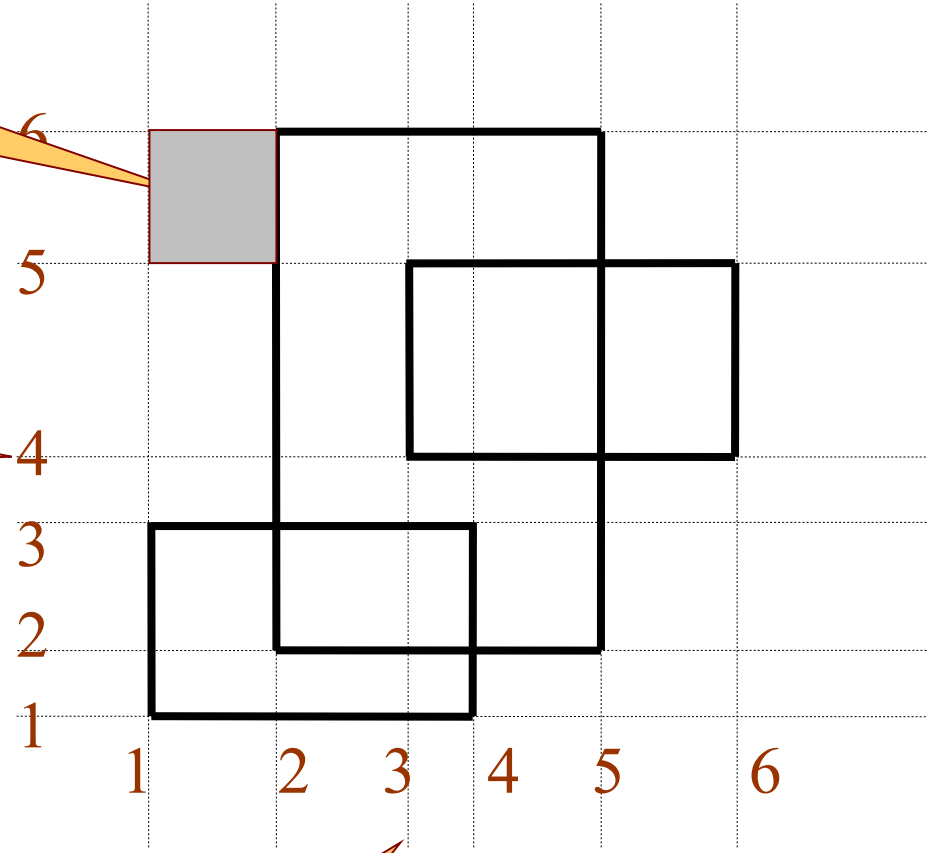
问题一的提出
USACO Shaping Regions 改编

- N 个不同颜色的不透明长方形
  （ 1<=N<=3000 ）
- 放在一张长宽分别为 A 、B 的白纸上
- 边与白纸的边缘平行
- 求俯视时看到的所有颜色的面积

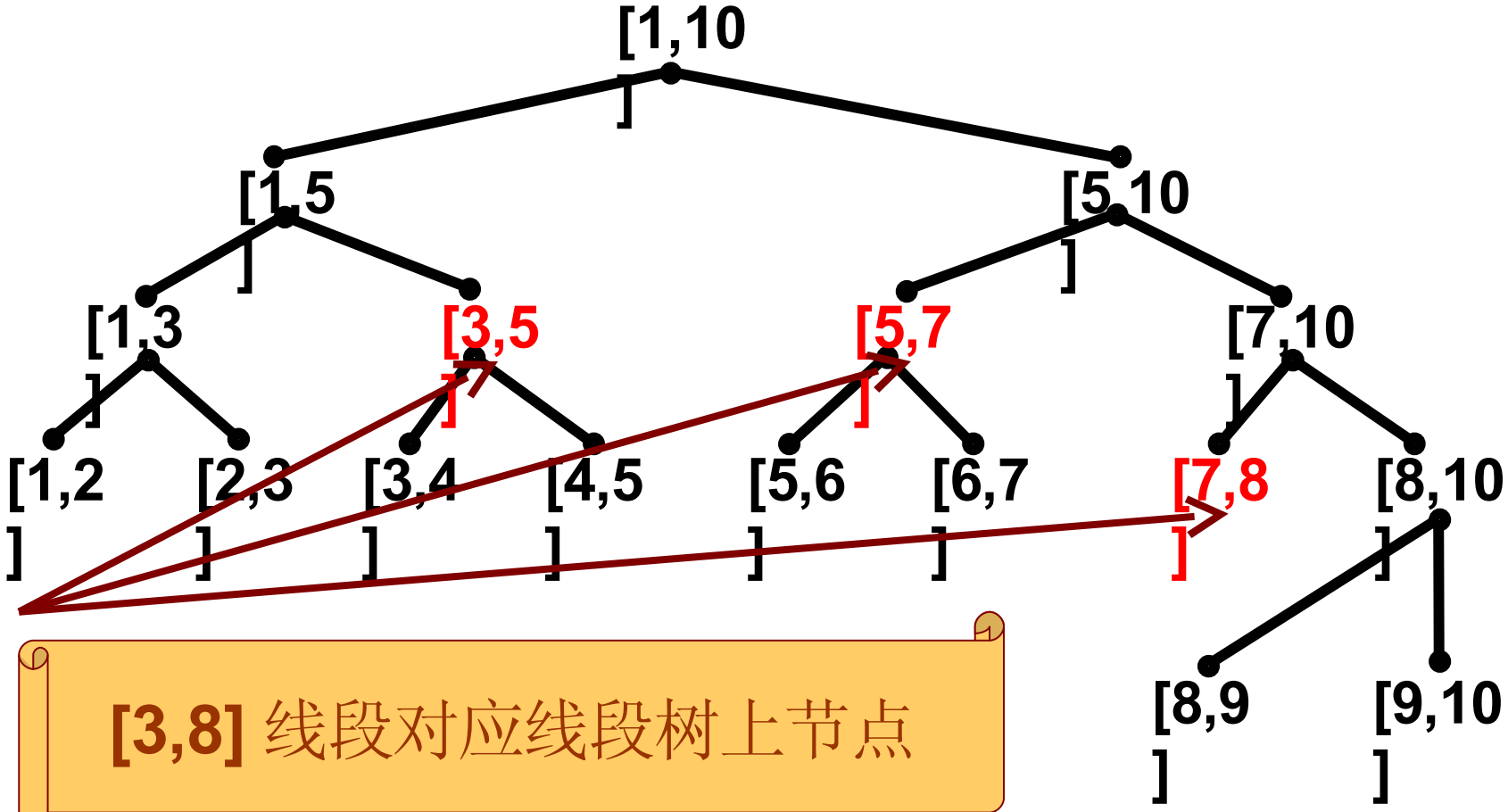


问题一的解决——简单的预处理

- 离散化
- 整数坐标
- 坐标范围在 1~2n 之间。

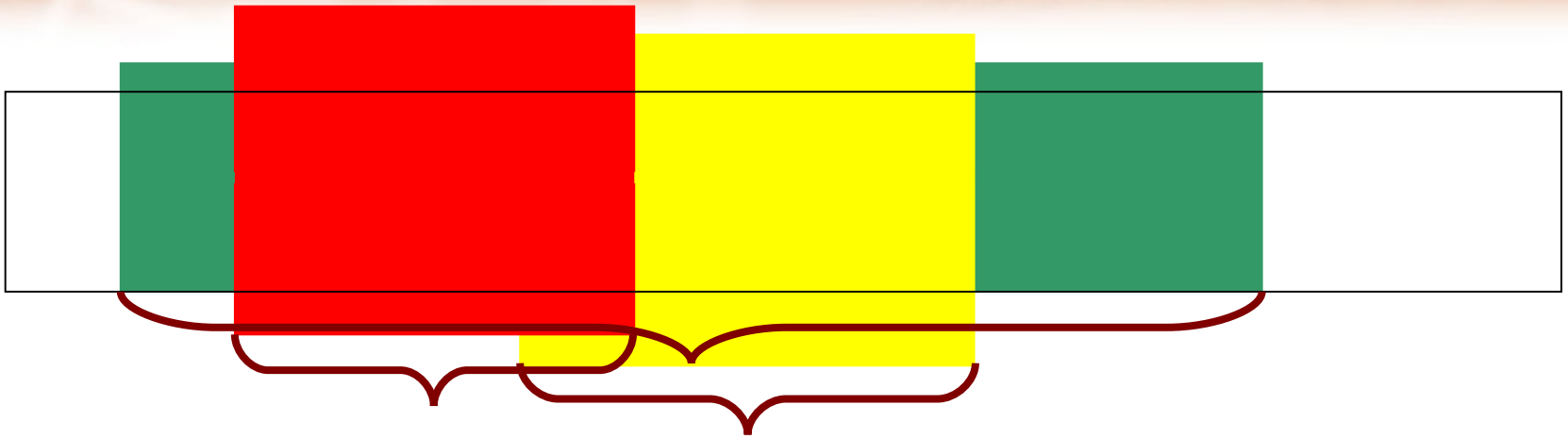


问题一的解决——经典算法



[3,8] 线段对应线段树上节点

问题一的解决——经典算法

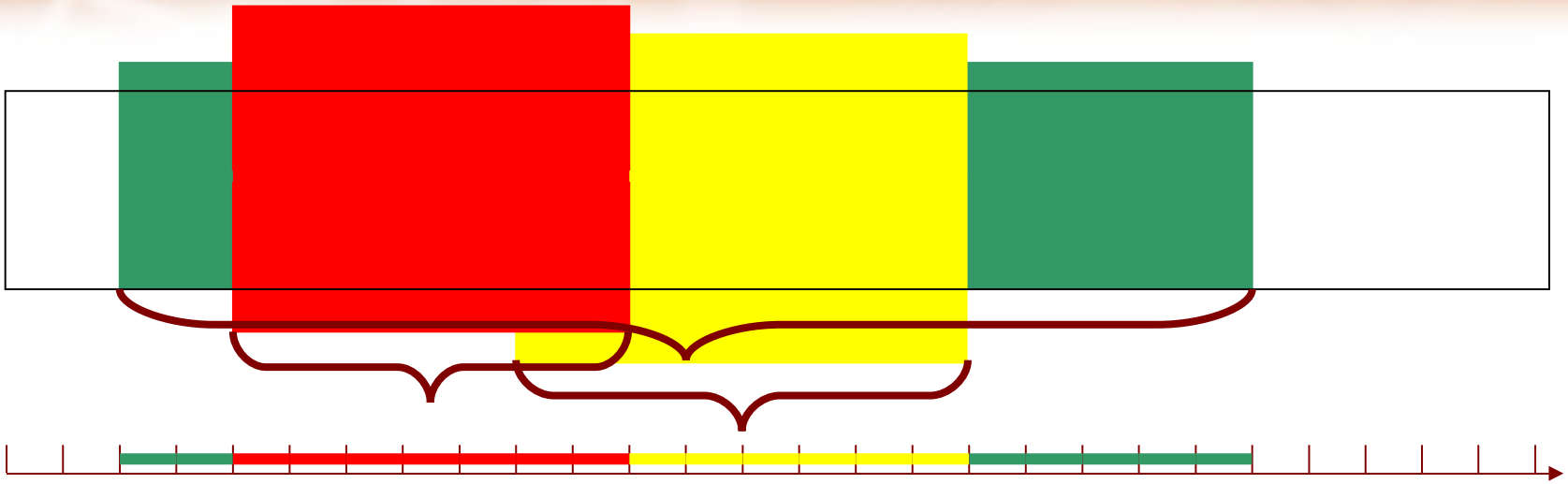


- 自顶至底依次插入颜色为 X 的线段 [l,r]，该区间 [l,r] 上原有颜色不被替换，其余部分染上颜色 X。
  - ✓O(logn)
- 返回所有颜色的覆盖量。
  - ✓O(n)

问题一的解决——经典算法

- $O(n^2 \log n)$
- 优点：
  - ✓广为人知
  - ✓复杂度较低，练习线段树的经典教材

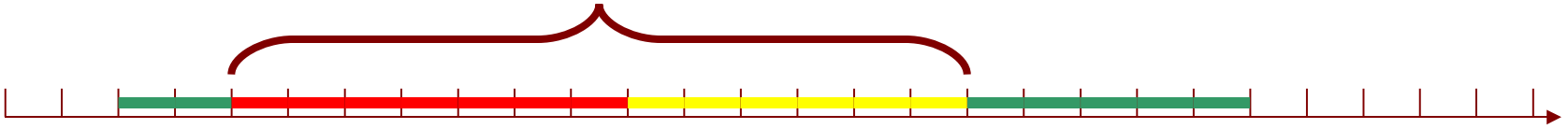
问题一的解决——朴素算法



- $O(n^3)$

问题一的解决——另类算法

- $O(n^3)$
- 优点：
  - ✓极易实现
  - ✓启发性强（有潜力可挖）
- **寻找冗余！**

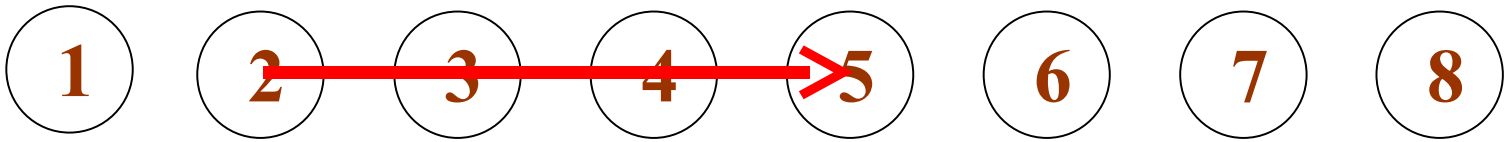


- 这一段的检索有必要吗？

问题一的解决——另类算法

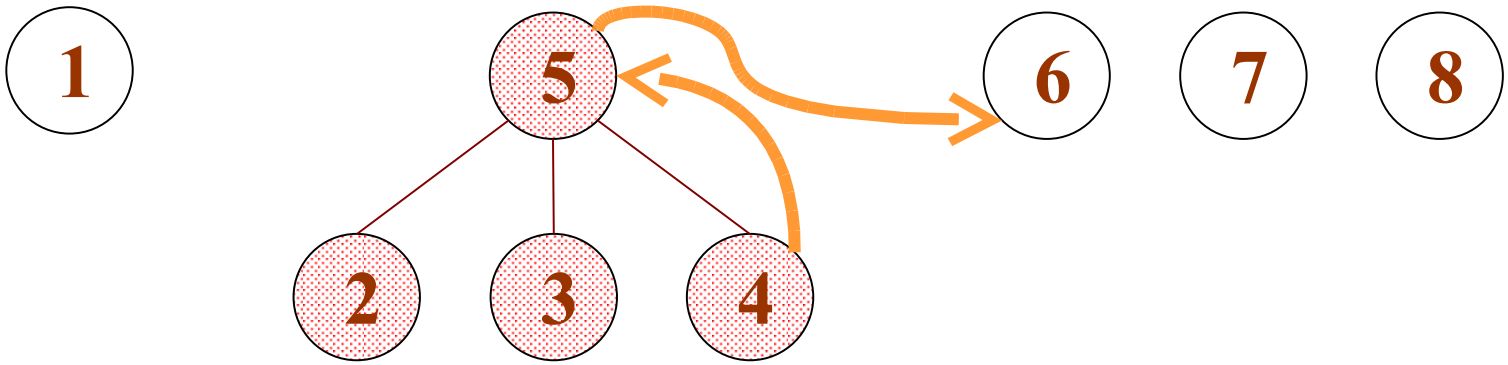
- 对已覆盖的区间，新增后续指针
- 走进已覆盖离散格时，沿指针进入下一个离散格
- 将途径离散格的后续指针设为当前覆盖区间之后的第一格。
  - ✓路径压缩？神似并查集！

问题一的解决——另类算法



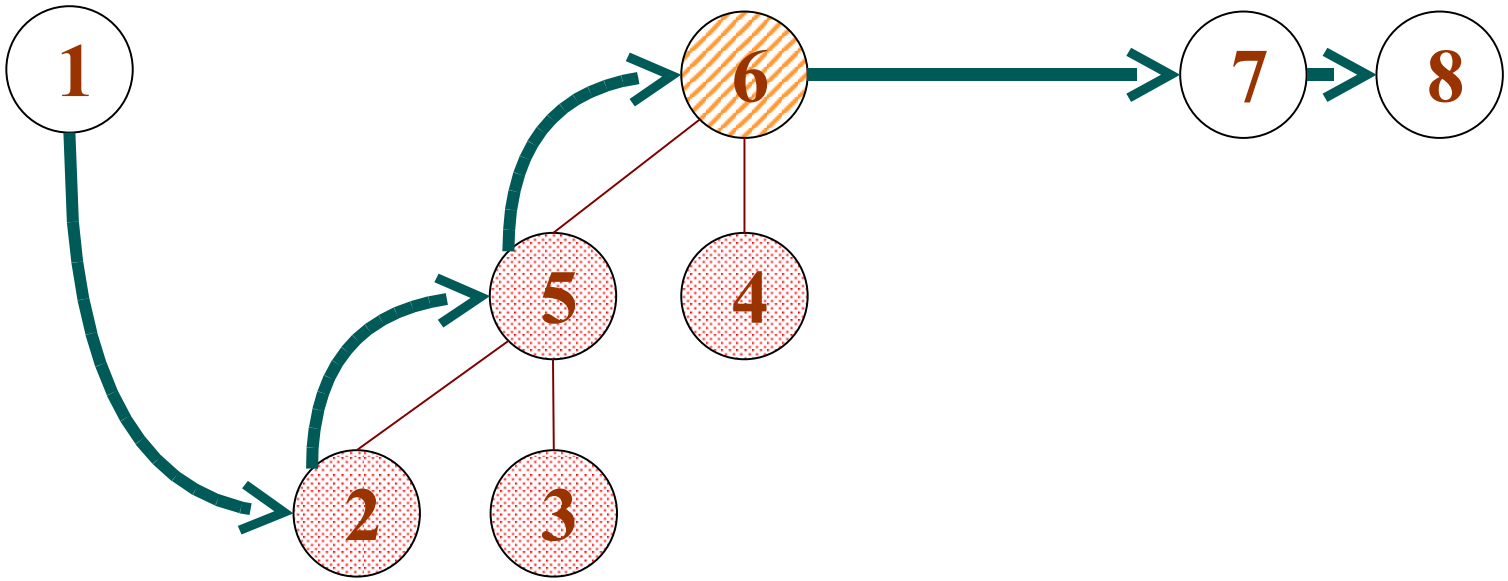
- 将相邻的已染色线段看成一个集合
- 红色 覆盖 [2, 5]

问题一的解决——另类算法



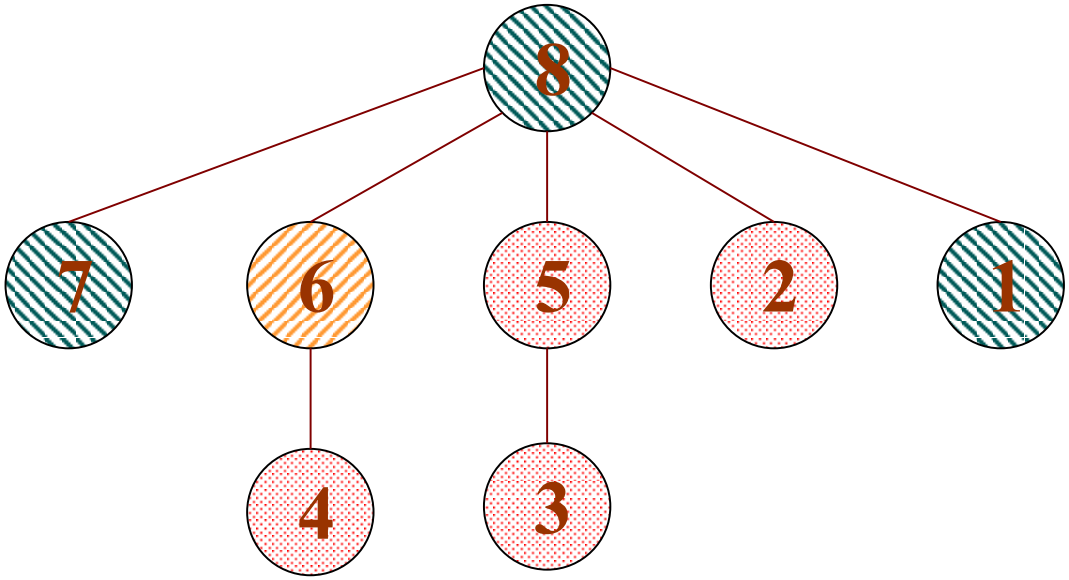
- 黄色 覆盖 [4, 6]

问题一的解决——另类算法



- 绿色 覆盖 [1, 8]

问题一的解决——另类算法

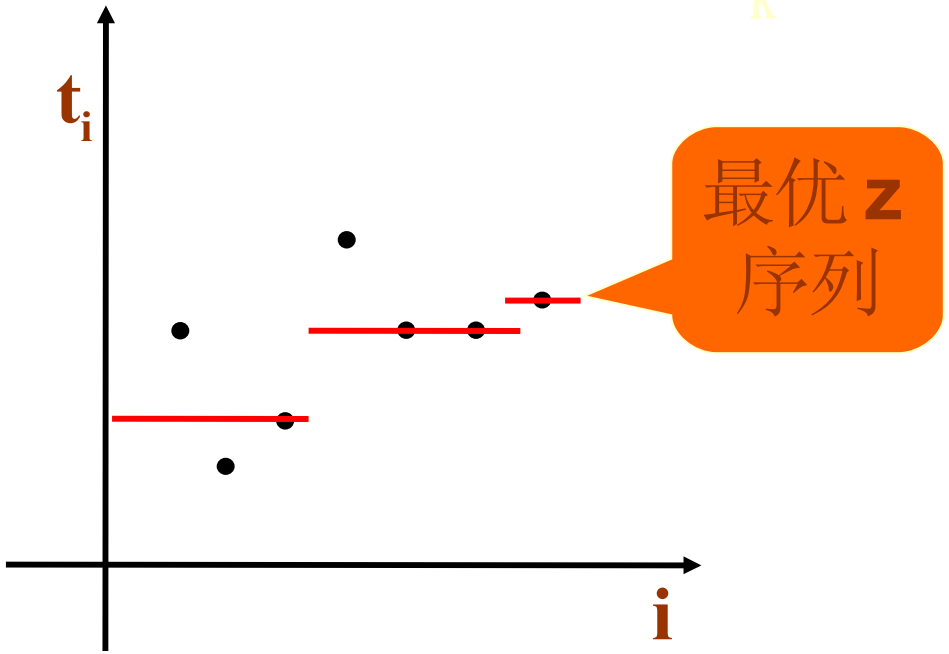


- 完整的路径压缩，再加上按秩合并可以使改进算法的时间复杂度完全降至 $O(n^2)$，具体操作和证明参见我的论文。

问题二的提出

BalticOI2004 1-3 Sequence 改编

- 给定序列 t_1, t_2, …, t_N，要求构建一个递增序列 $z_1 <= z_2 <= \cdots <= z_N$，使得 $|t_1 - z_1| + |t_2 - z_2| + \cdots + |t_N - z_N|$ 尽可能小。其中 $1 \leqslant N \leqslant 1000000$，$0 \leqslant t_K \leqslant 2000000000$。



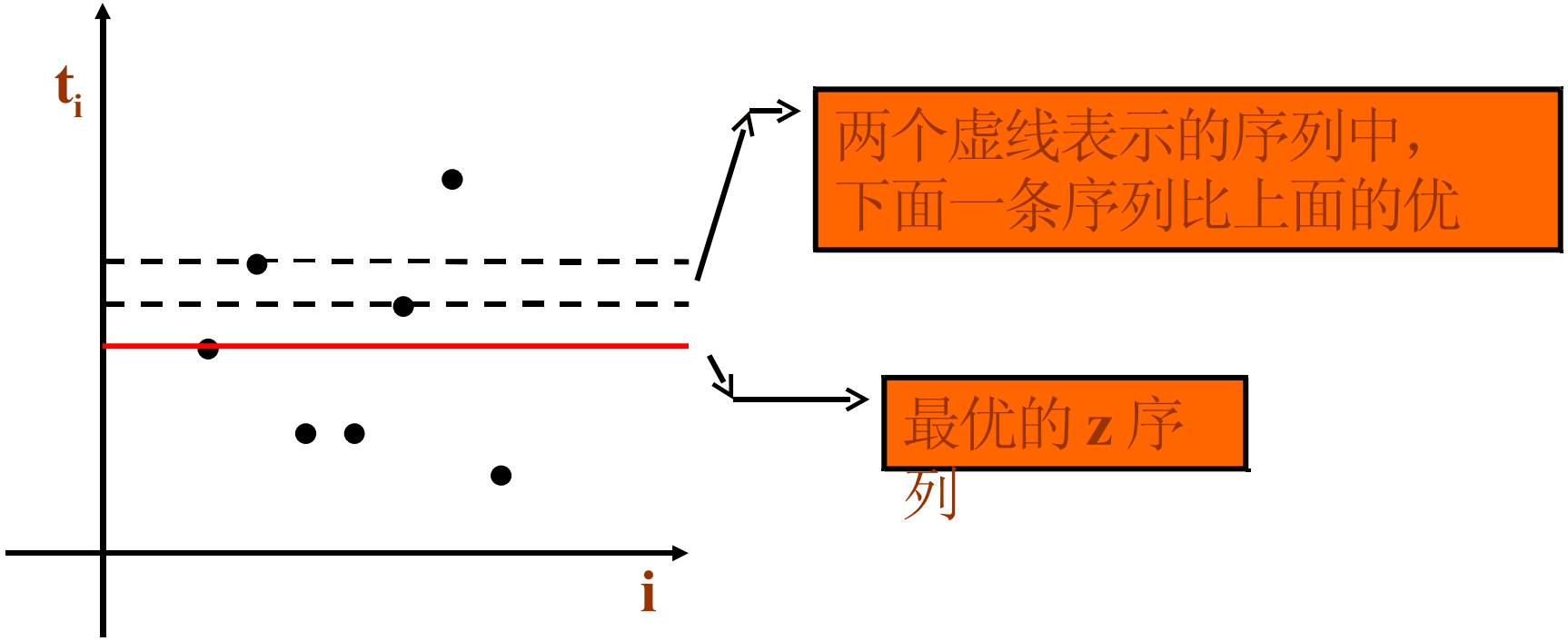
注：为了更清楚地说明诸引理与算法，下文将多次出现类似的图。其中黑点代表**t** 序列，线段代表某一个 **z** 序列的方案。

问题二的解决——定义与说明

- 由于最优方案不为一，下文中描述 X 是一组最优方案的同时，并不表示最优方案一定是 X 。对方案进行微调时，不保证原方案不是最优，但我们可以保证调整后的方案一定不会变差（某种程度上更接近最优）。
- z 序列组成的方案可用（$z_1, z_2, \cdots z_n$）表示。

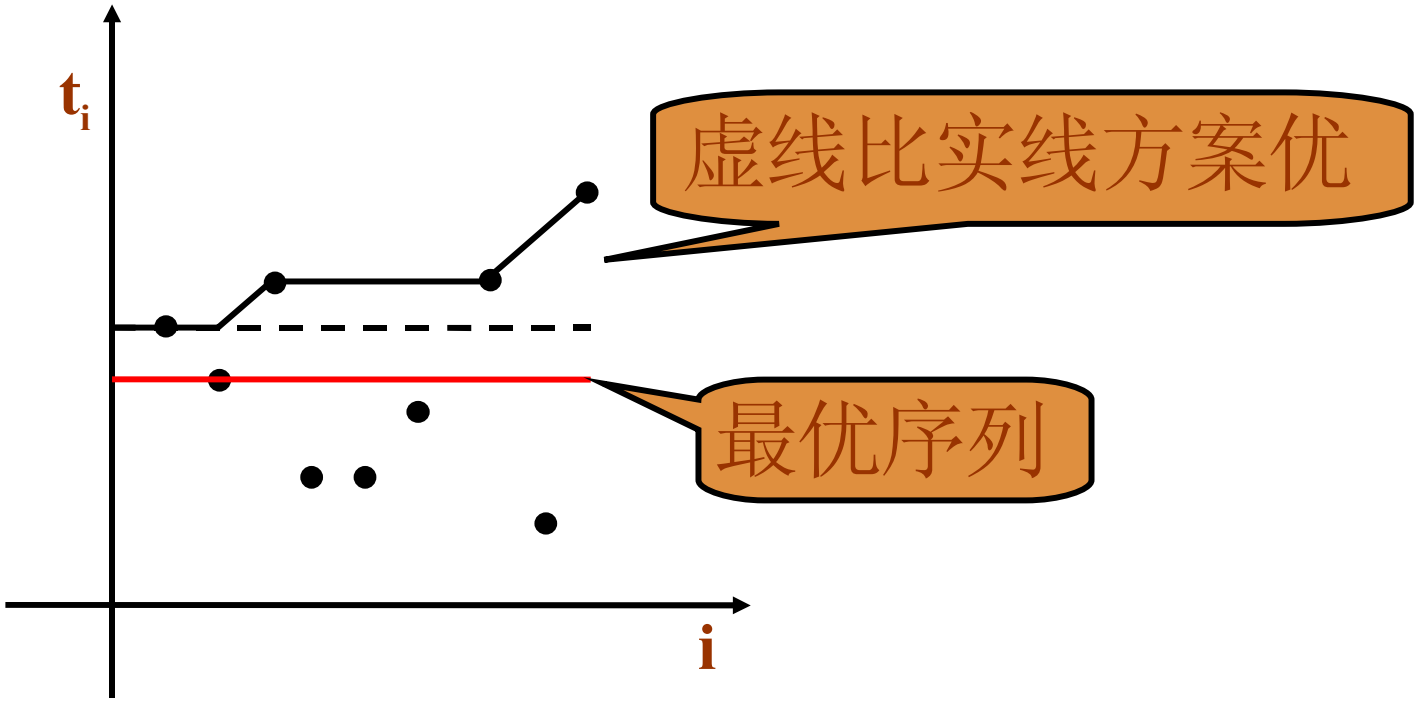
问题二的解决——引理

- 对给定的 t_1，t_2，…，t_n，如果最优方案满足 $z_1=z_2=\ldots=z_n=x$，那么
- x 为 t[1..n] 中位数时，其为一个最优方案。



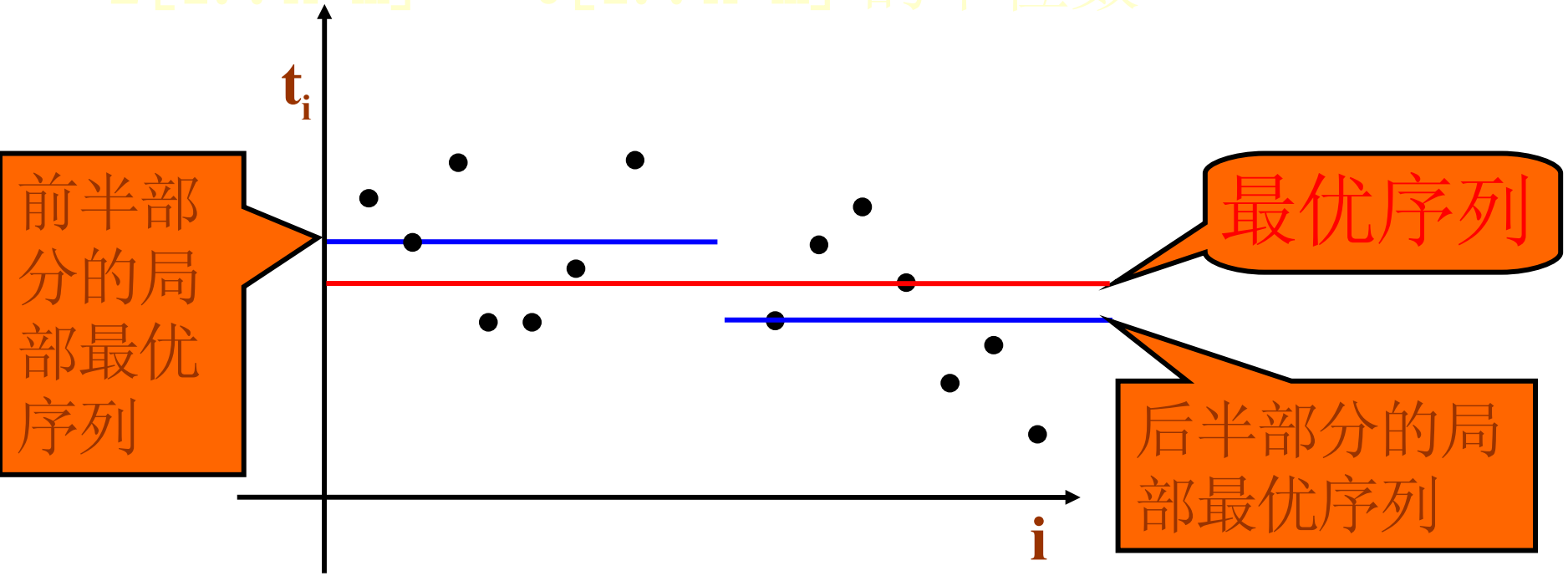
问题二的解决——引理

- 对于给定的 t_1，t_2，…，t_n，如果最优方案是 $z_1 = z_2 = \ldots = z_n = u$，那么



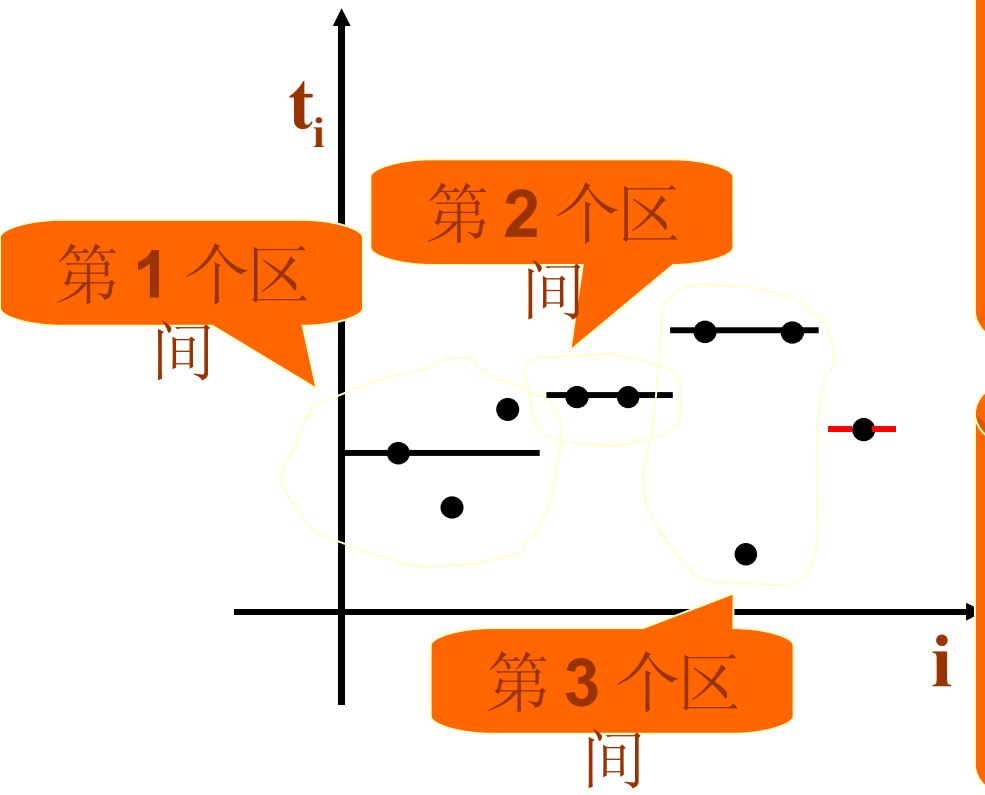
问题二的解决——引理

- 对 t_1，t_2，…t_n，以及 t_{n+1}，t_{n+2}，…t_{n+m}，如果 $(u, u, \ldots u)$ 和 $(v, v, \ldots, v)$ 分别是它们的最优方案，并且 $u \geqslant v$，那么
- $z[1..n+m] = t[1..n+m]$ 的中位数



问题二的解决——第一类算法

- 依次处理每个元素，对先前已经得到的最优方案进行微调

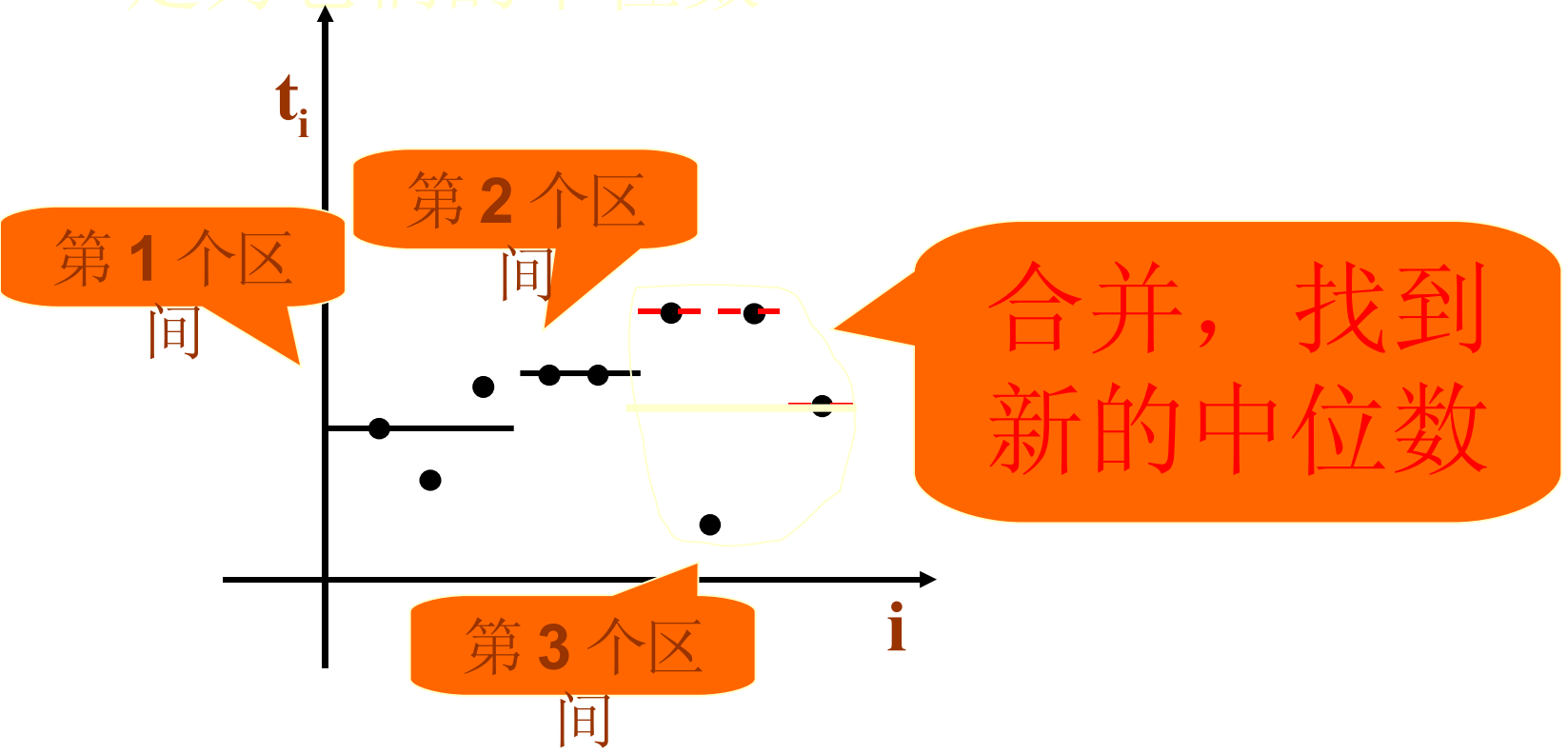


将 z 值相同的子序列看作一个连续的区间

为插入的新元素建立一个独立的临时区间

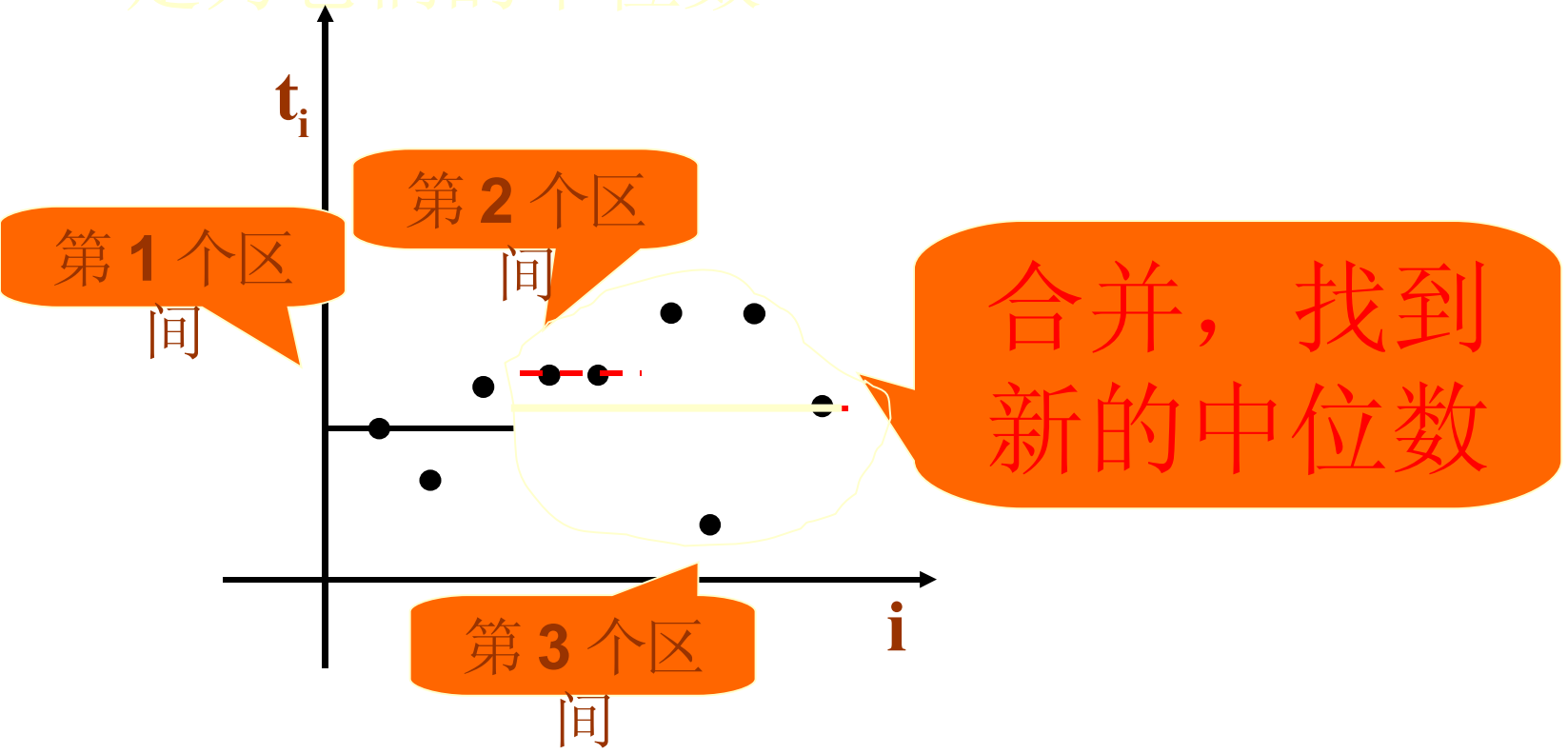
问题二的解决——第一类算法

- 如果当前最后一个区间的 z 值较前一个区间小，根据引理我们合并这两个区间，新的 z 值设定为它们的中位数



问题二的解决——第一类算法

- 如果当前最后一个区间的 z 值较前一个区间小，根据引理我们合并这两个区间，新的 z 值设定为它们的中位数



问题二的解决——第一类算法

- 选取一个优秀的数据结构，它可以高效地完成如下任务：
  - 1、集合合并
  - 2、求出该集合的中位数。
- 注意到集合最多和并 n-1 次，求中位数操作不超过 n-1 次。

问题二的解决——第一类算法

- 方法一：平衡二叉树 $O(n(\log n)^2)$
- 方法二：最大堆 $O(n(\log n)^2)$
  - 可以严密证明，区间合并时相邻两个区间的数，最大一半的并集，恰好是合并后区间最大的一半
- 方法三：在方法二基础上寻找冗余，努力避免集合的合并操作 $O(n\log n)$
- 方法四：左偏树 $O(n\log n)$

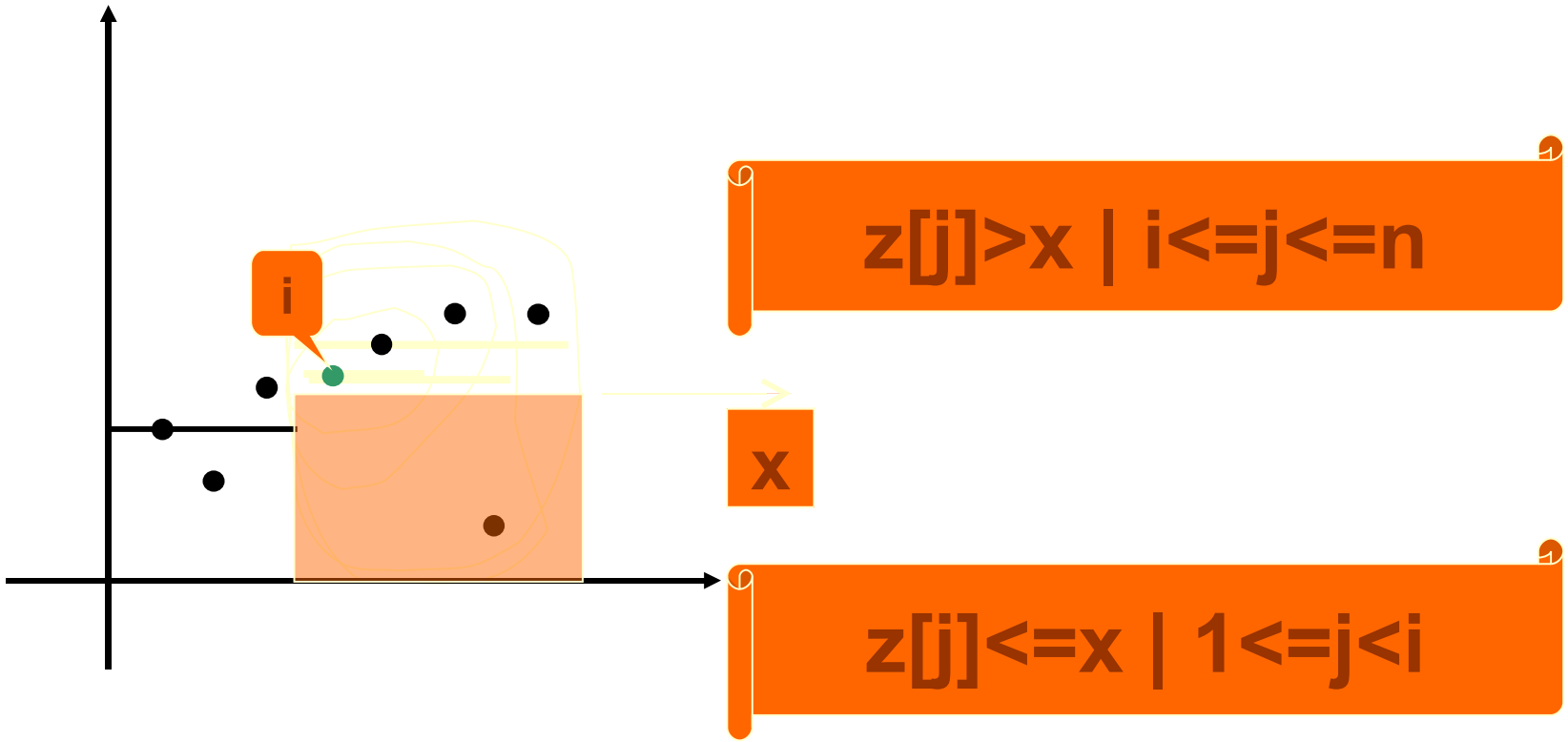
- 实现难！思考深度大！

问题二的解决——引理

- 对给定的 t 序列 t[1..n]，如果 z[1..n] 是一组最优策略，那么我们可以假定：
- 满足 z[i]>x 的最小的 i，恰好是最小的 i 使得对任意 i≤j≤n，t[i..j] 的中位数 >x。
- （如果某组最优方案不满足该条件，我们可以经过调整，使得另一个最优方案满足该条件）

问题二的解决——引理

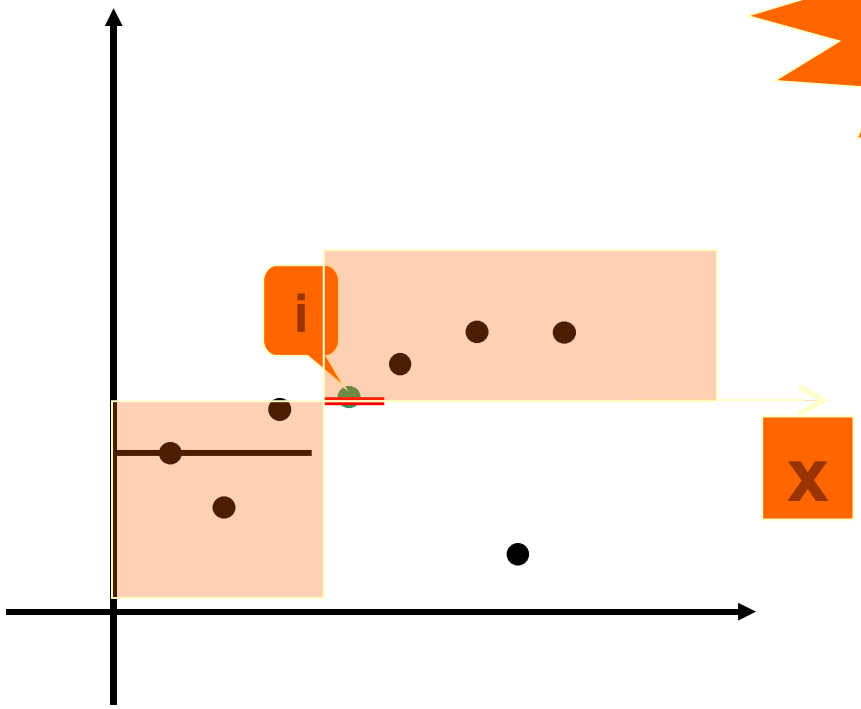
- 满足 z[i]>x 的最小的 i，恰好是最小的 i 使得对任意 i≤j≤n，t[i..j] 的中位数 >x 。



问题二的解决——第二类算法

- 二分！

O(nlogn)



总结

- 问题的表示往往比答案更重要，答案不过乃数学或实验。
- 要提出新的问题、新的可能性、从某个新的角度考虑一个旧问题，都要求创造性的想象力，回到起点对问题重新定义，这才是真正的科学进步之所在。