07-E

搜索树应用

范围树

邓俊辉
deng@tsinghua.edu.cn

顺藤摸瓜

y-query的相关性 (Coherence)

❖ 以2维范围查询为例，不难观察到：

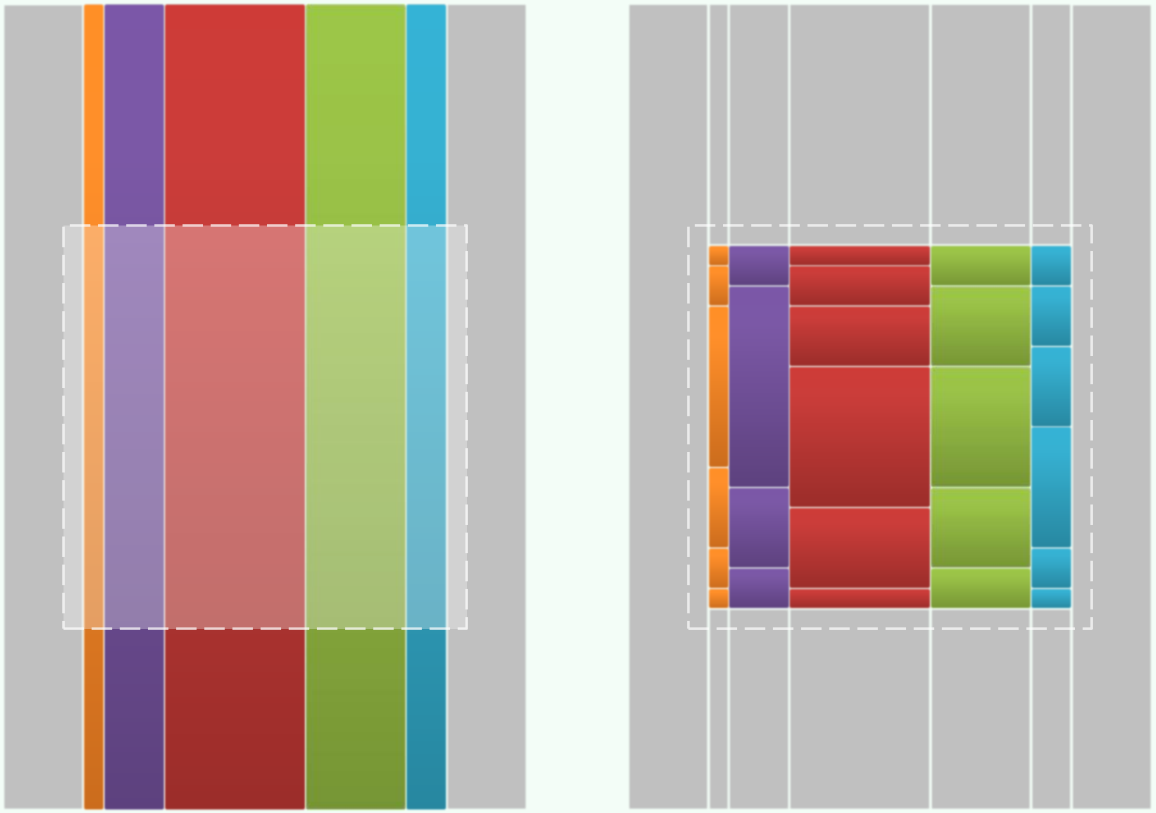
- 虽然会做 $\mathcal{O}(\log n)$ 次的y-query(y_1,y_2)

- 但只是针对的y-树各自不同

- 而对应的范围则完全相同

❖ 既然所有y-query之间有如此紧密的相关性

目前这样令它们各自独立的完成，很不科学...



BBST<BBST<T>> --> BBST<List<T>>

❖ 作为最后一个维度，每次y-query

都是**不需递归**的1维范围查询

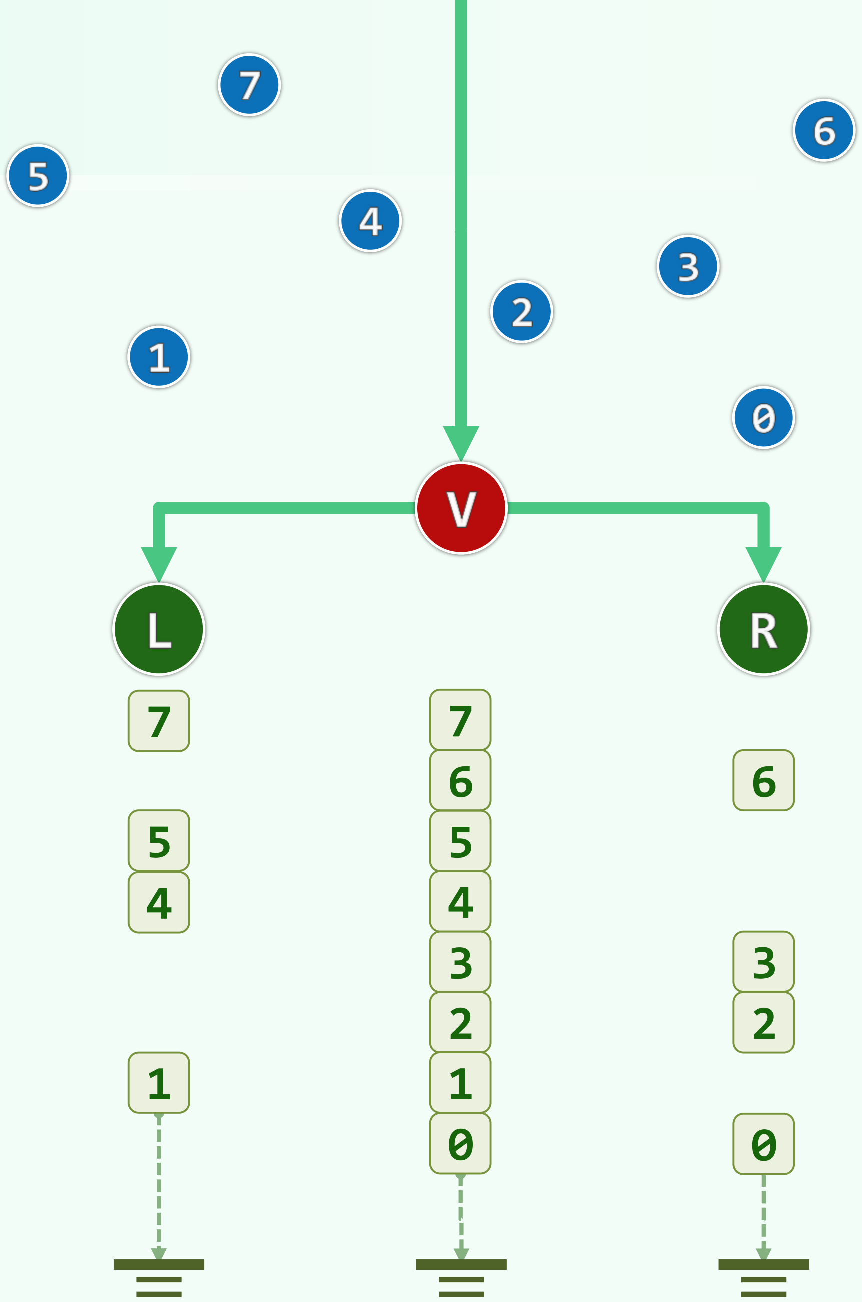
❖ 既如此，完全不必将其组织为一棵BBST

实际上，一个**有序序列**（列表或向量）即足矣

❖ 于是，可以将整个结构理解并改造为

- 一棵x-树，以及

- 与其中每个节点/子树相**关联**的一个有序**序列**



分散层叠：Shortcuts

❖ Fractional Cascading:

进一步地，可在任意节点v的关联序列

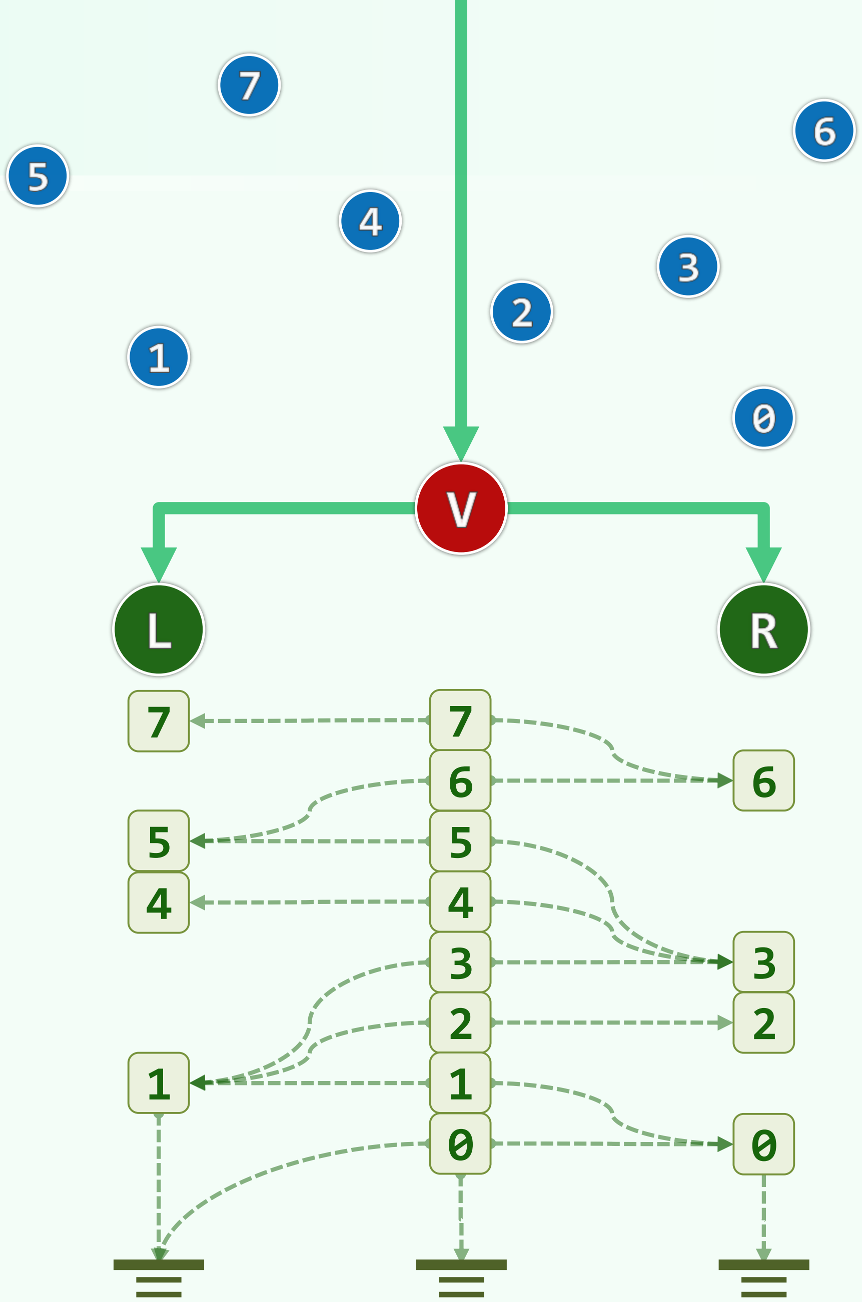
及其孩子的关联序列之间，建立快捷引用

❖ 对于同一y坐标值，由V.search(y)

可以直接得到L.search(y)和R.search(y)

❖ 也就是说，在执行一批y-query()的过程中

- 一旦耗费 $\mathcal{O}(\log n)$ 时间完成了LCA.query()
- 所有后代们的y-query都各自仅需 $\mathcal{O}(1)$ 时间



分散层叠：二路归并

❖ 一方面，V的关联列表，总是

等于L和R的关联列表之并

❖ 另一方面，正如此前所推荐地

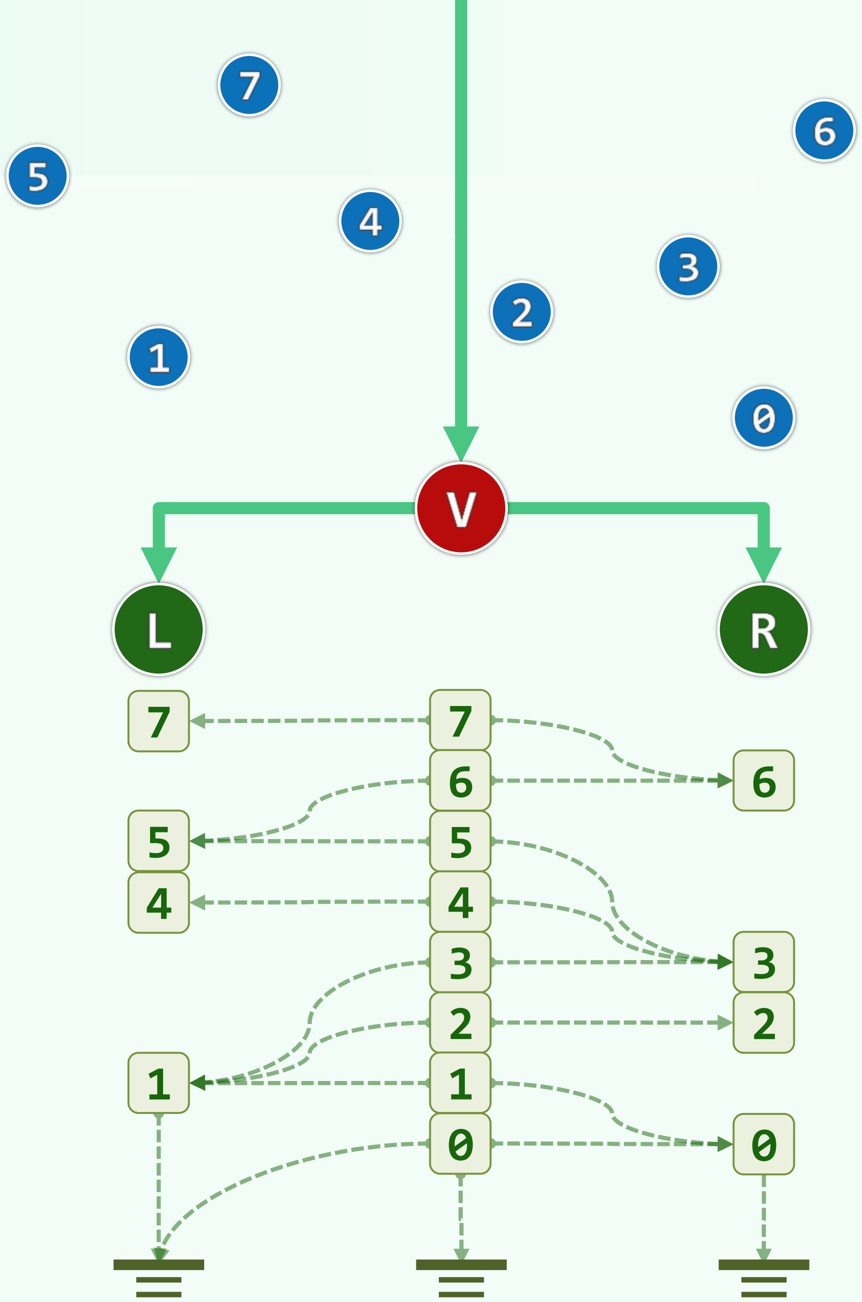
x-树应该自底而上地逐层合并而成

❖ 既如此，x-树中各节点的关联列表

也可以通过二路归并

自底而上地依次生成

❖ 如此，并不会增加预处理的整体时间复杂度



范围树

❖ 引入了分散层叠技术的MLST

称作**范围树**（Range Tree）

❖ 由平面上的任意n个点，都可以

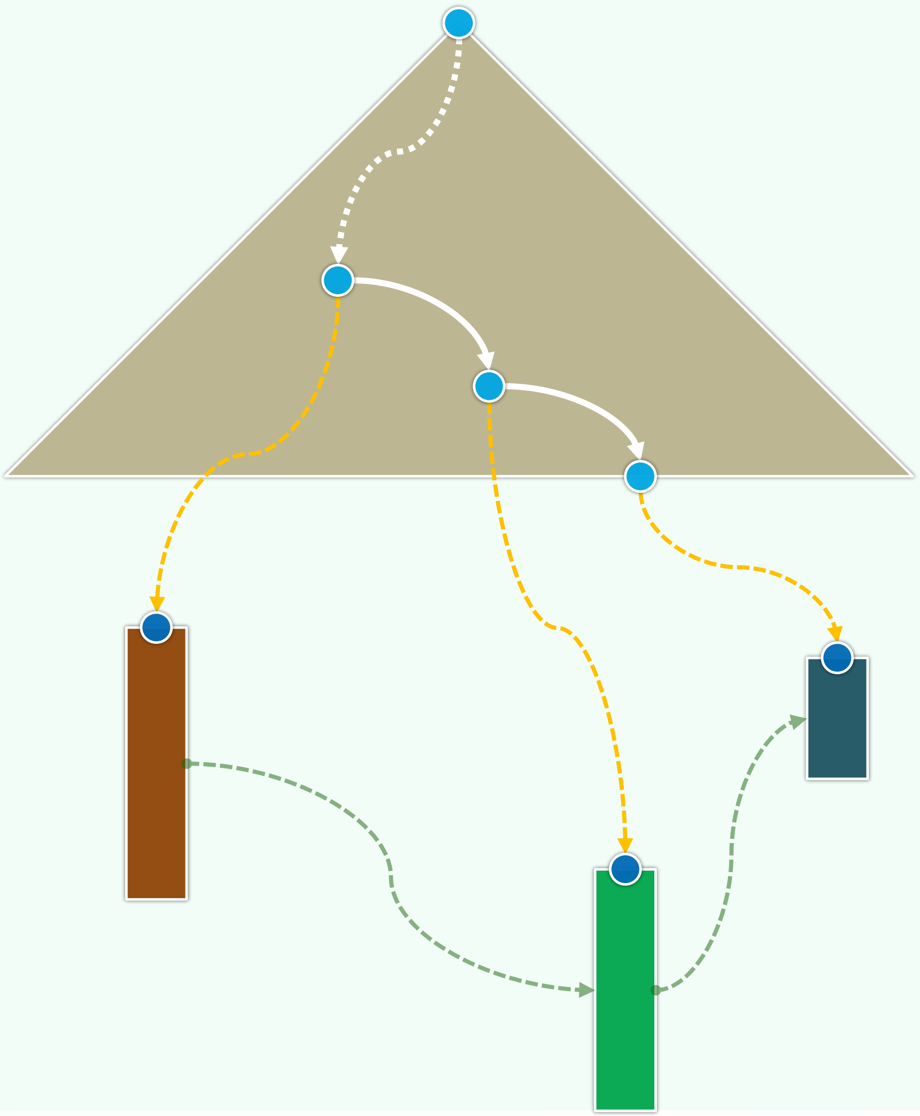
A> 在 $\mathcal{O}(n \cdot \log n)$ 时间内

构造出一棵2层的范围树

B> 该结构只需 $\mathcal{O}(n \cdot \log n)$ 空间

C> 借助该结构，每次二维范围查询

都可以在 $\mathcal{O}(r + \log n)$ 时间内完成



更高维度

- 遗憾的是，对MLST结构而言
  分散层叠的技巧只能运用于**最后一个**维度

- 由欧氏空间$\mathcal{E}^d$（d≥2）中的任意n个点，都可以

  A> 在 $\mathcal{O}(n \cdot \log^{d-1} n)$ 时间内
  构造出一棵d层的MLST

  B> 该结构只需 $\mathcal{O}(n \cdot \log^{d-1} n)$ 空间

  C> 借助该结构，每次d维范围查询
  都可以在 $\mathcal{O}(r + \log^{d-1} n)$ 时间内完成