高级搜索树

红黑树：动机

08-C1

所有的过去

都留下了痕迹

哪怕是

一次最微妙的心动

一声最轻渺的叹息

因为过去要进入未来，所以有了故事；因为在深夜里，你会想不起你是怎么从原来走到的现在，所以有了故事；当记忆被抹去，当你除了故事就再无任何可以去记忆、可以被记住的东西的时候，因为要有永恒，所以有了故事。

邓俊辉

deng@tsinghua.edu.cn

并发性：Concurrent Access To A Database

- 修改之前先**加锁**（lock）；完成后**解锁**（unlock）

  访问延迟主要取决于“lock/unlock”周期

- 对于BST而言，每次修改过程中，唯

  结构有变（reconstruction）处才需加锁

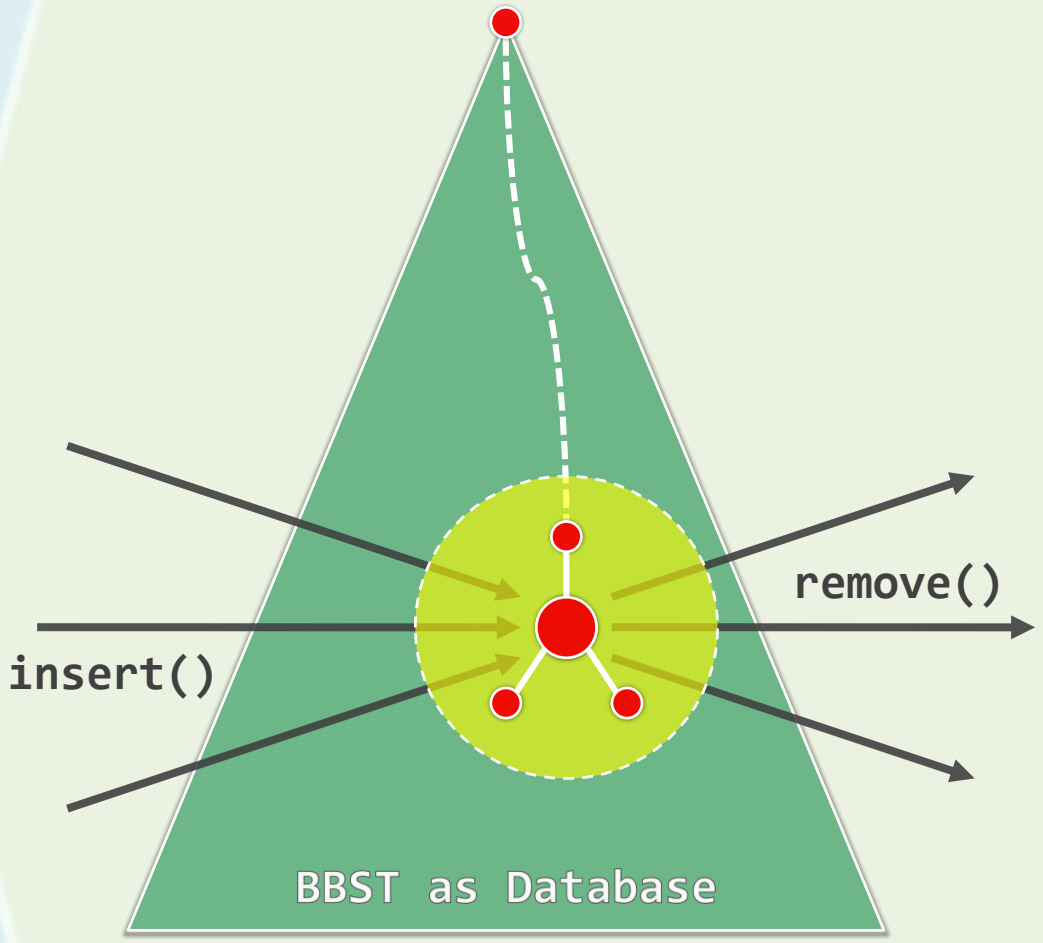
  访问延迟主要取决于这类局部之**数量**...

- Splay：结构变化剧烈，最差可达 $\mathcal{O}(n)$

- AVL：remove()时 $\mathcal{O}(\log n)$ ——尽管

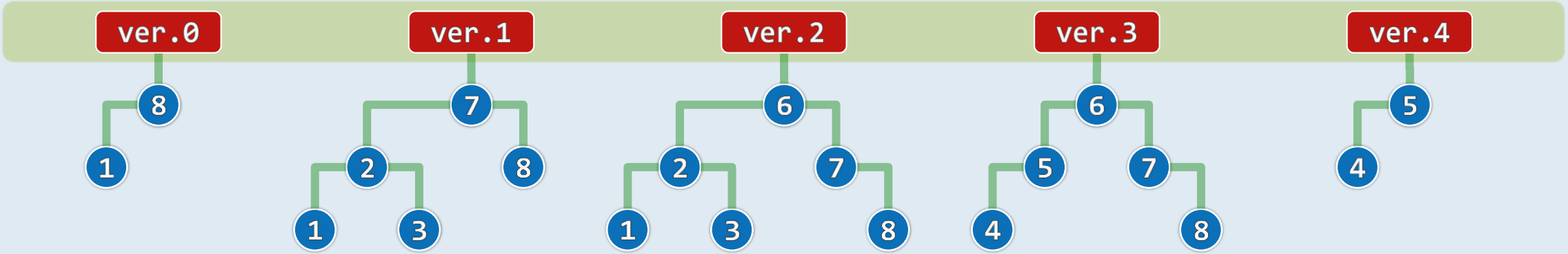
  insert()时可保证 $\mathcal{O}(1)$

- Red-Black：无论insert/remove，**均不超过** $\mathcal{O}(1)$



持久性：Persistent structures：支持对历史版本的访问 //ephemeral

❖ 蛮力实现：每个版本独立保存；各版本自成一个搜索结构



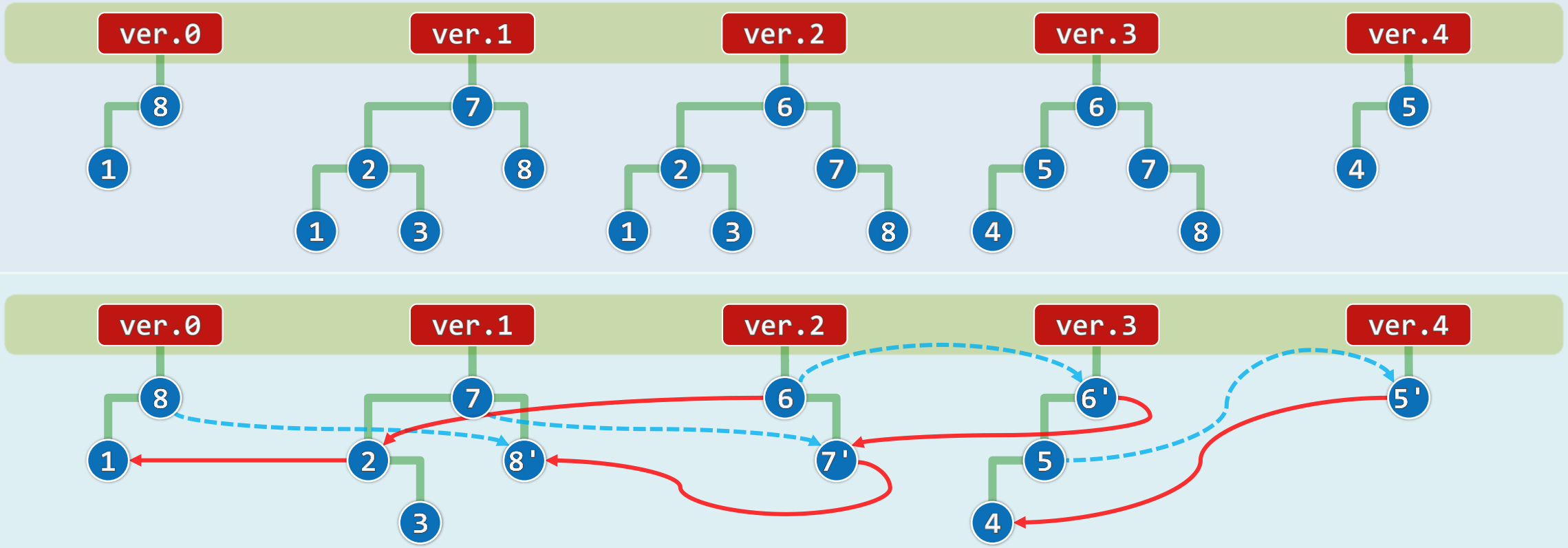
❖ 单次操作 $\mathcal{O}(\log h + \log n)$，累计 $\mathcal{O}(n \cdot h)$ 时间/空间 //h = |history|

❖ 挑战：可否将复杂度控制在 $\mathcal{O}(n + h \cdot \log n)$ 以内?

❖ 可以！为此需利用相邻版本之间的相关性...

压缩存储：大量共享，少量更新：每个版本的新增复杂度，仅为 $\mathcal{O}(\log n)$

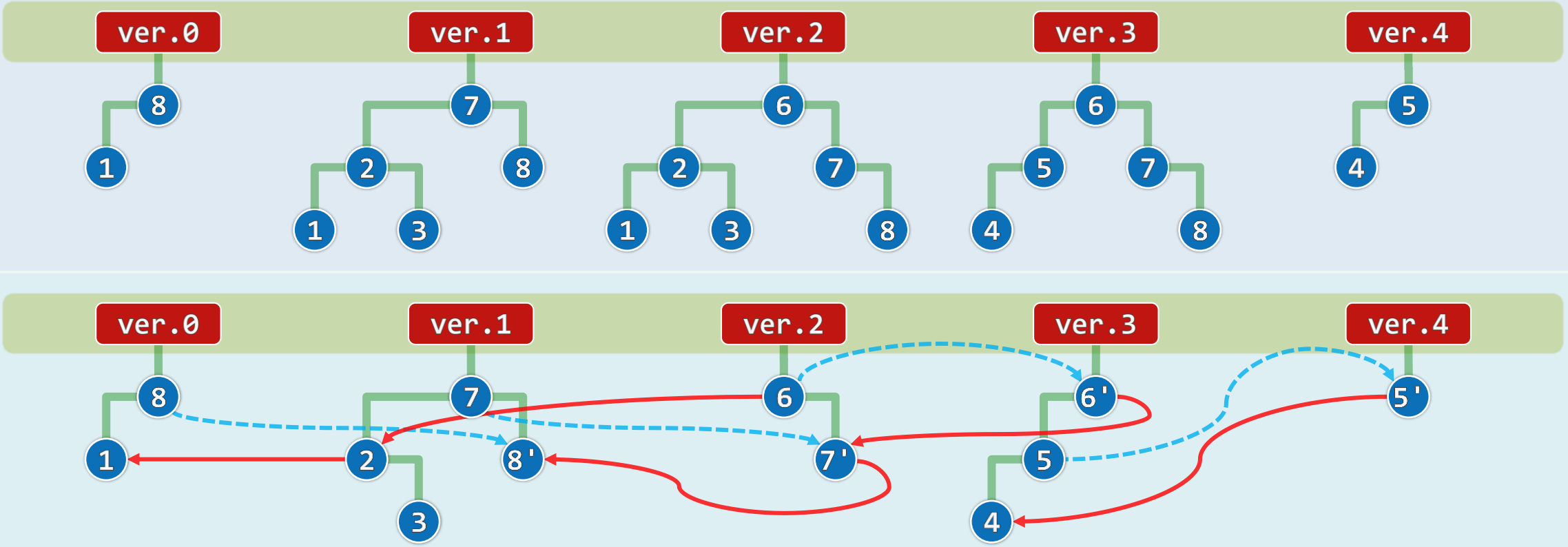
❖ Partial Persistence：仅支持对历史版本的读取　//监控录像、飞行器黑盒子，等等



❖ 这类情况下，还可进一步提高至总体 $\mathcal{O}(n+h)$、单版本 $\mathcal{O}(1)$ ...

$O(1)$重构

❖ 为此，就树形结构的拓扑而言，相邻版本之间的差异不能超过$\mathcal{O}(1)$



❖ 很遗憾，AVL、Splay等BBST均不具备这一性质；须另辟蹊径...

java.util.TreeMap

```
import java.util.*;

public class TestTreeMap {

   public static void main( String[] args ) {

      TreeMap scarborough = new TreeMap();

      scarborough.put("P", "parsley");

      scarborough.put("S", "sage");

      scarborough.put("R", "rosemary");

      scarborough.put("T", "thyme");

      System.out.println( scarborough );

   }

}
```