08-C1

二叉搜索树

平衡：期望树高

邓俊辉
deng@tsinghua.edu.cn

上梁不正下梁歪

树高：最坏情况与平均情况

- 由以上的实现与分析，BST主要接口search()、insert()和remove()的运行时间

  在最坏情况下，均线性正比于其高度$\mathcal{O}(h)$

- 因此，若不能有效地控制树高，则就实际的性能而言

  较之此前的向量和列表等数据结构，BST将无法体现出明显优势

- 比如在最坏情况下，二叉搜索树可能彻底地**退化**为列表

  此时的查找效率甚至会降至$\mathcal{O}(n)$，线性正比于树（列表）的规模

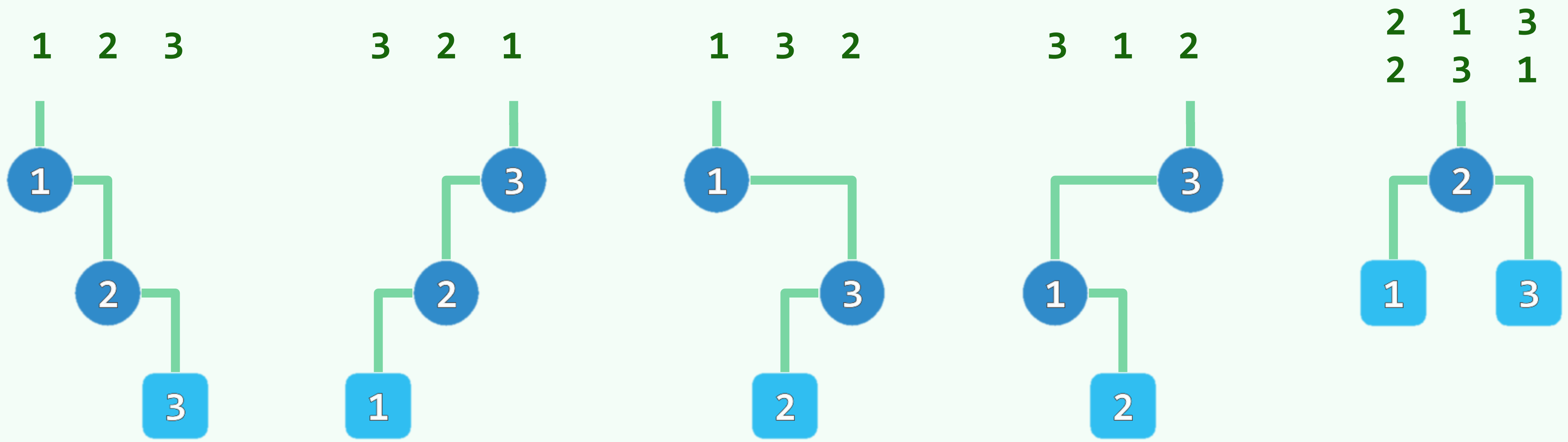
- 那么，出现此类最坏或较坏情况的概率有多大？

  或者，从平均复杂度的角度看，二叉搜索树的性能究竟如何呢？

- 以下按两种常用的随机统计**口径**，就BST的**平均性能**做一分析和对比

随机生成

❖ 考查n个互异词条 $\{\ e_1,\ e_2,\ e_3,\ \ldots\ e_n\ \}$ ，对任一排列 $\sigma\ =\ (\ e_{i_1},\ e_{i_2},\ e_{i_3},\ \ldots\ e_{i_n}\ )$ ...



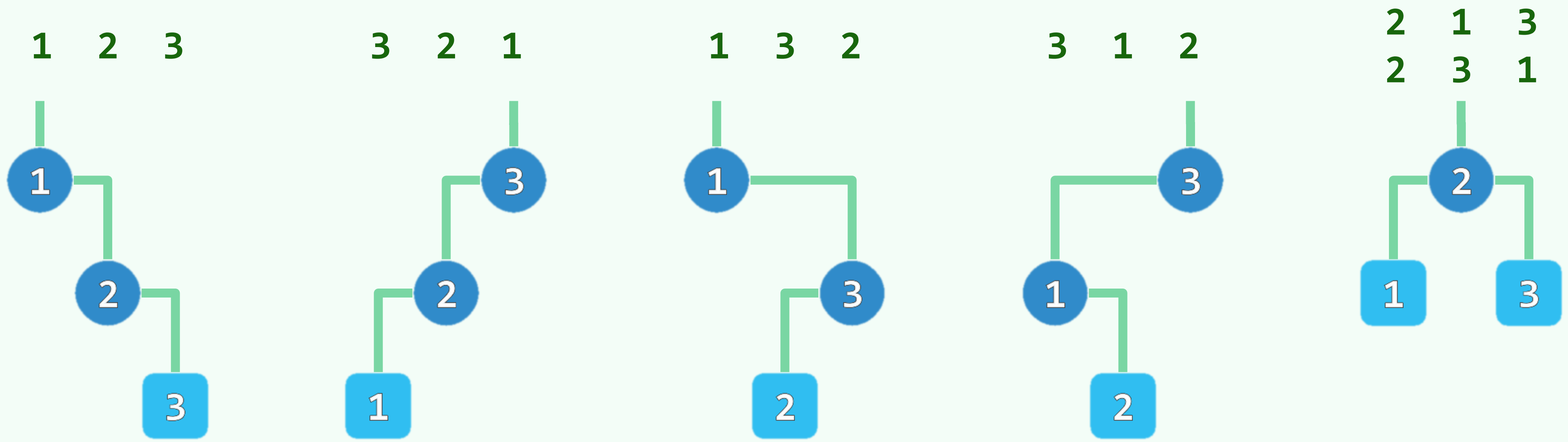
❖ 从空树开始，反复调用insert()接口将各词条**依次插入**，得到T(σ)

❖ 若假定任一排列σ作为输入的概率均等1/n!

则由n个互异词条随机生成的二叉搜索树，平均高度为$\Theta(\log n)$

随机组成

❖ n个互异节点，在遵守顺序性的前提下，可随机确定拓扑联接关系



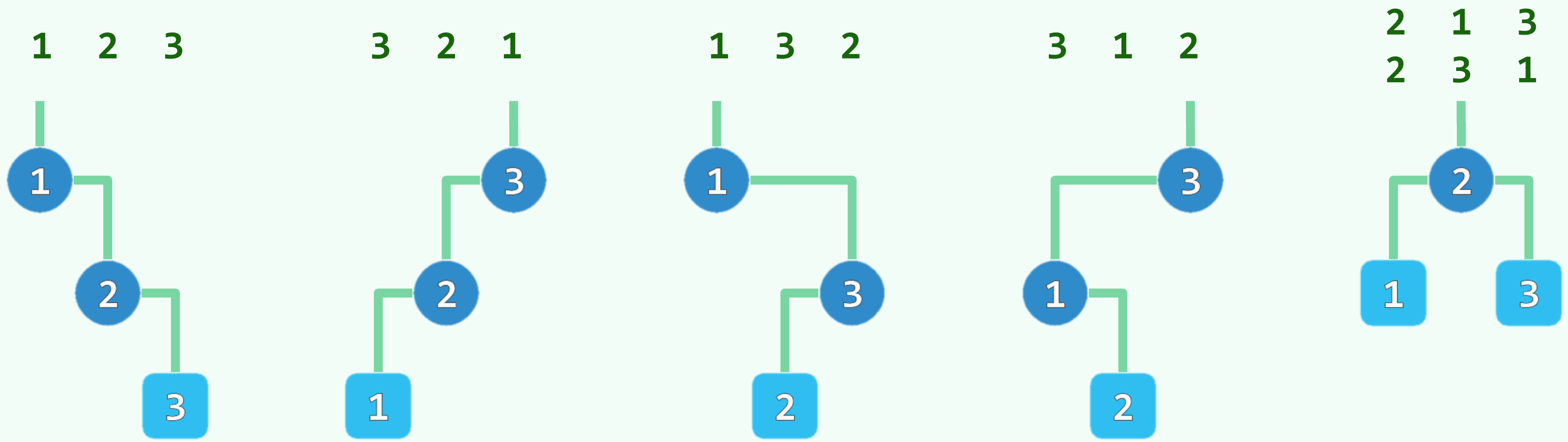
❖ 由n个互异节点随机组成的BST，若共计S(n)棵，则有

$$S(n) \;=\; \sum_{k=1}^{n} S(k-1)\cdot S(n-k) \;=\; catalan(n) \;=\; (2n)!/(n+1)!/n!$$

❖ 假定所有BST等概率出现，则其平均高度为$\Theta(\sqrt{n})$

logn vs. sqrt(n)

- ❖ 按两种口径所估计的平均性能，差异极大——谁更可信？谁更接近于真实情况？



- ❖ 前一口径中，越低的BST被**重复统计**更多次——有过于**乐观**之嫌
- ❖ 若删除算法固定使用succ()，则每棵BST都有越来越**左倾**的趋势
- ❖ 理想随机在实际中并不常见，关键码序列往往有局部性、（分段）单调性、（近似）周期性...
  较高甚至极高的BST频繁出现，不足为怪