09-G1

词典

跳转表：构思与结构

邓俊辉
deng@tsinghua.edu.cn

去沿江上下，或二十里，或三十里，选高阜处置一烽火台，每台用五十军守之

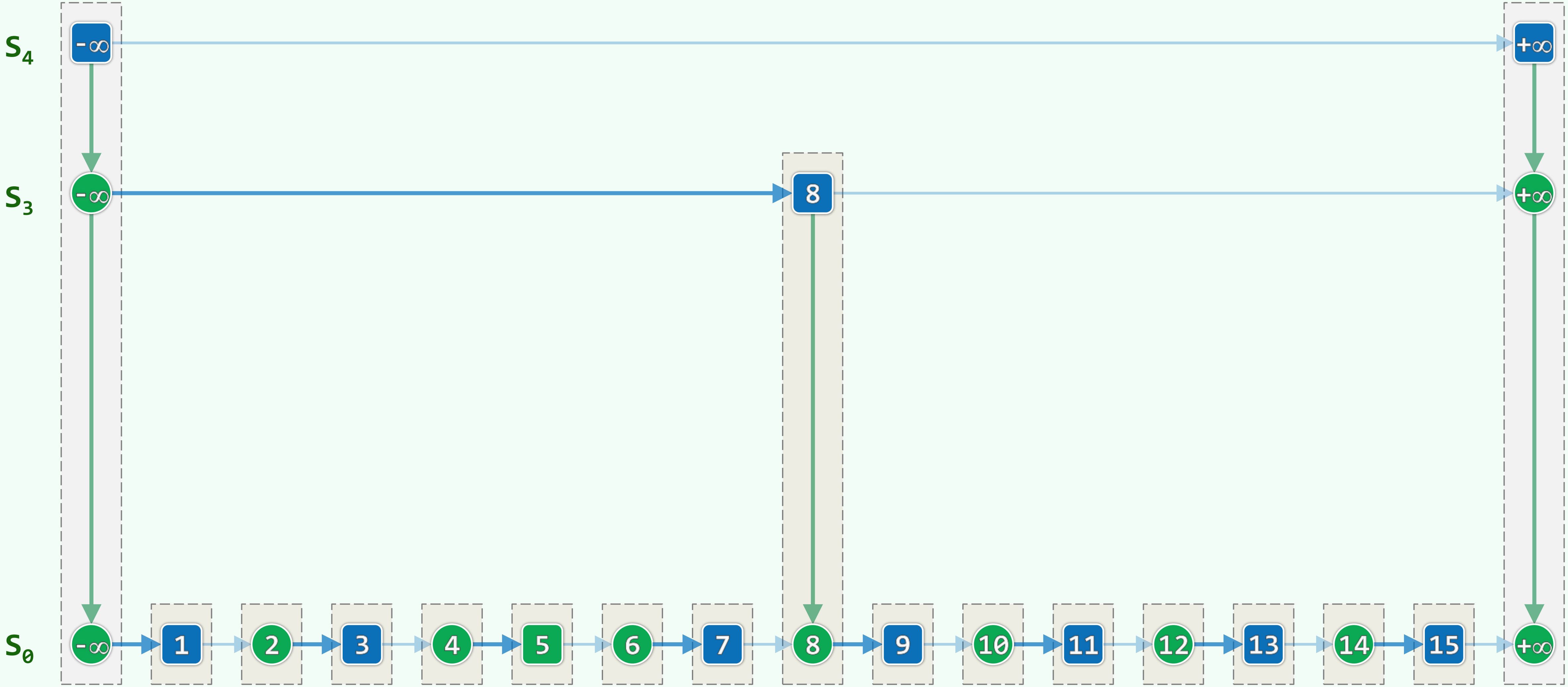
动机与思路

- [William Pugh, 1989] Skip Lists: A Probabilistic **Alternative** to Balanced Trees

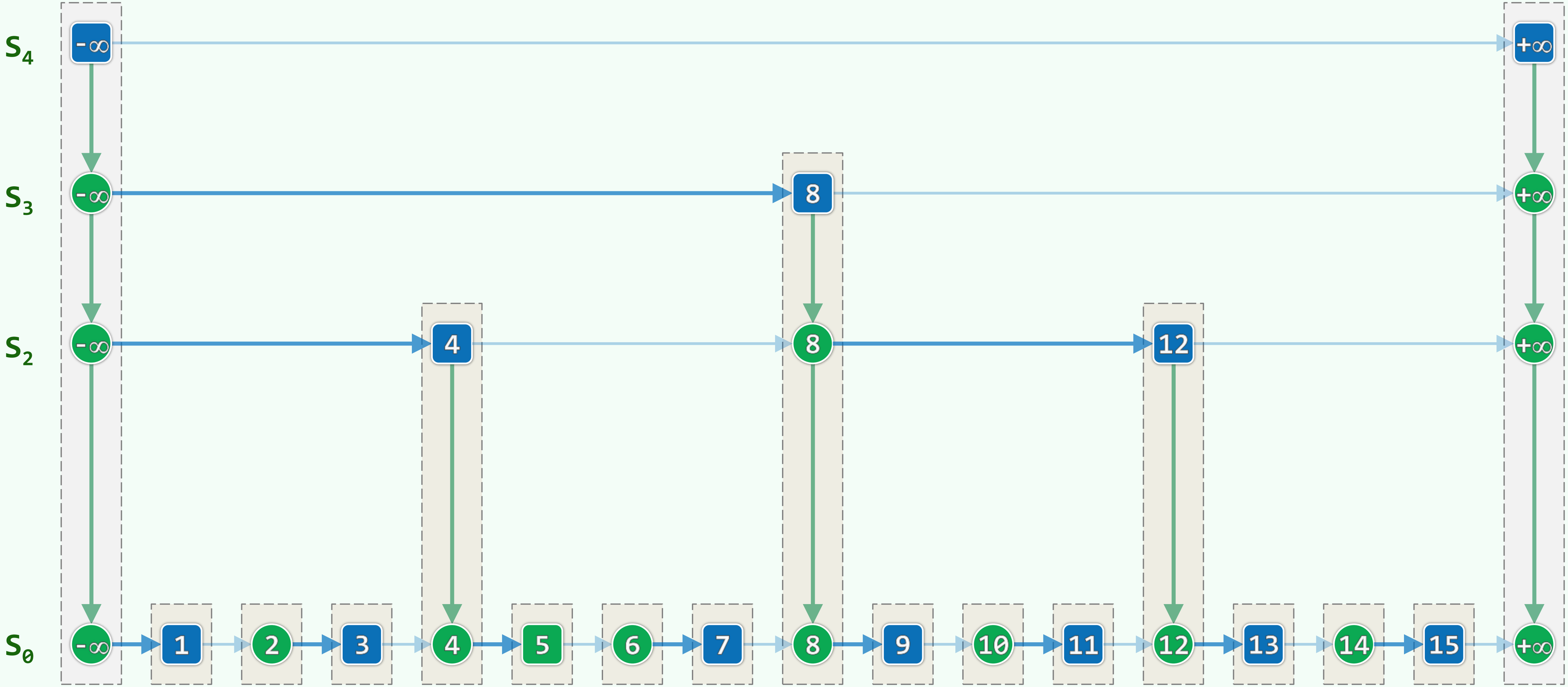
- 基于朴素的List结构，采取随机策略
  - 从**数学期望**的角度控制查找长度
  - 而不再如BBST那样**严格**地控制

- 较之BBST，insert/remove操作
  - **实现**更加简捷
  - **性能**显著提高

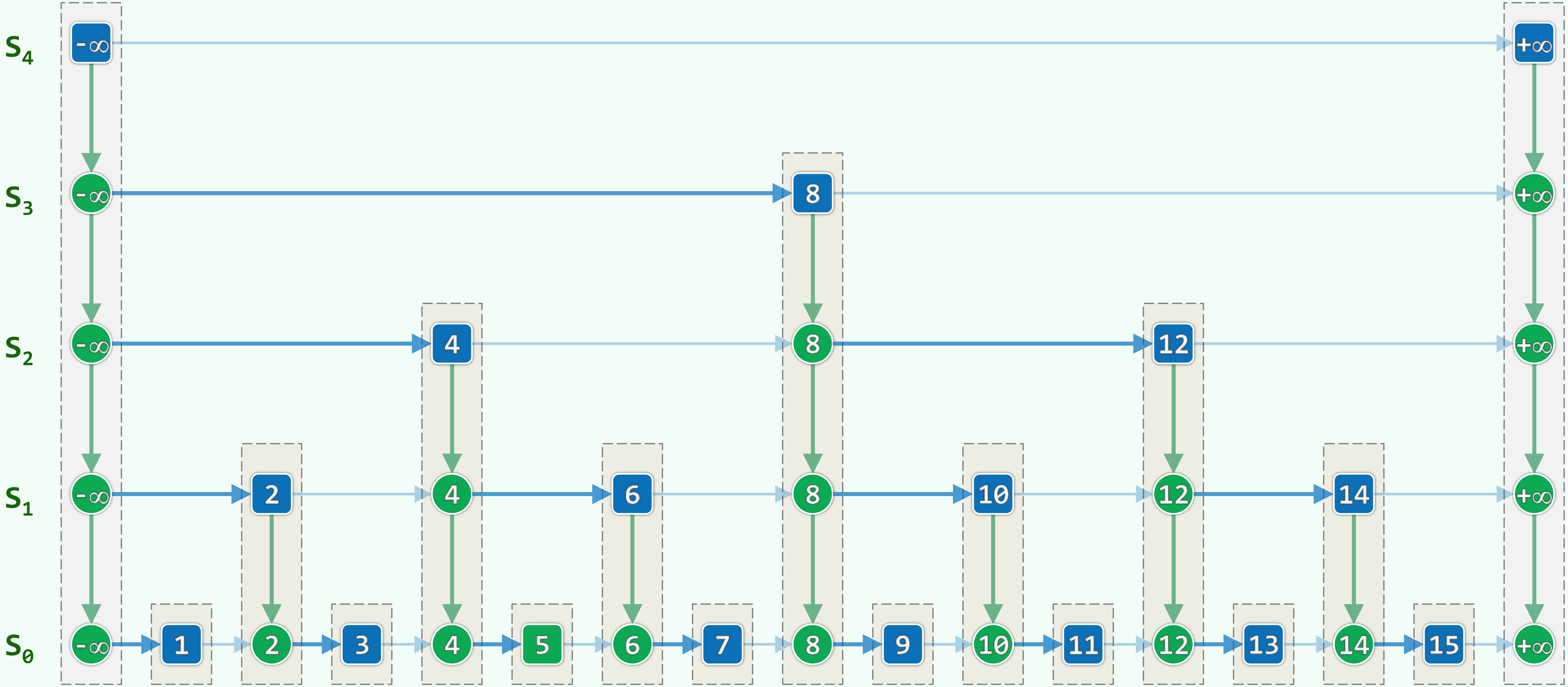
构思：捷径（1/3）



构思：捷径（2/3）



构思：捷径（3/3）



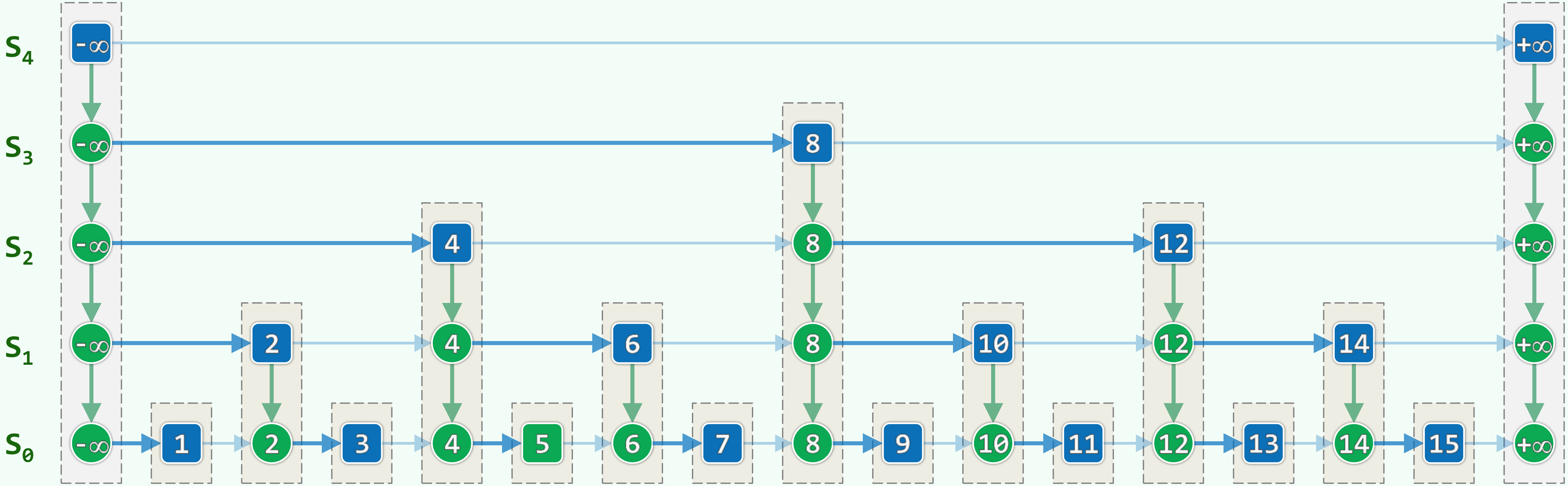
结构：理想 + 确定

❖ 逐层折半**抽稀**的列表：S_0，S_1，...，S_h

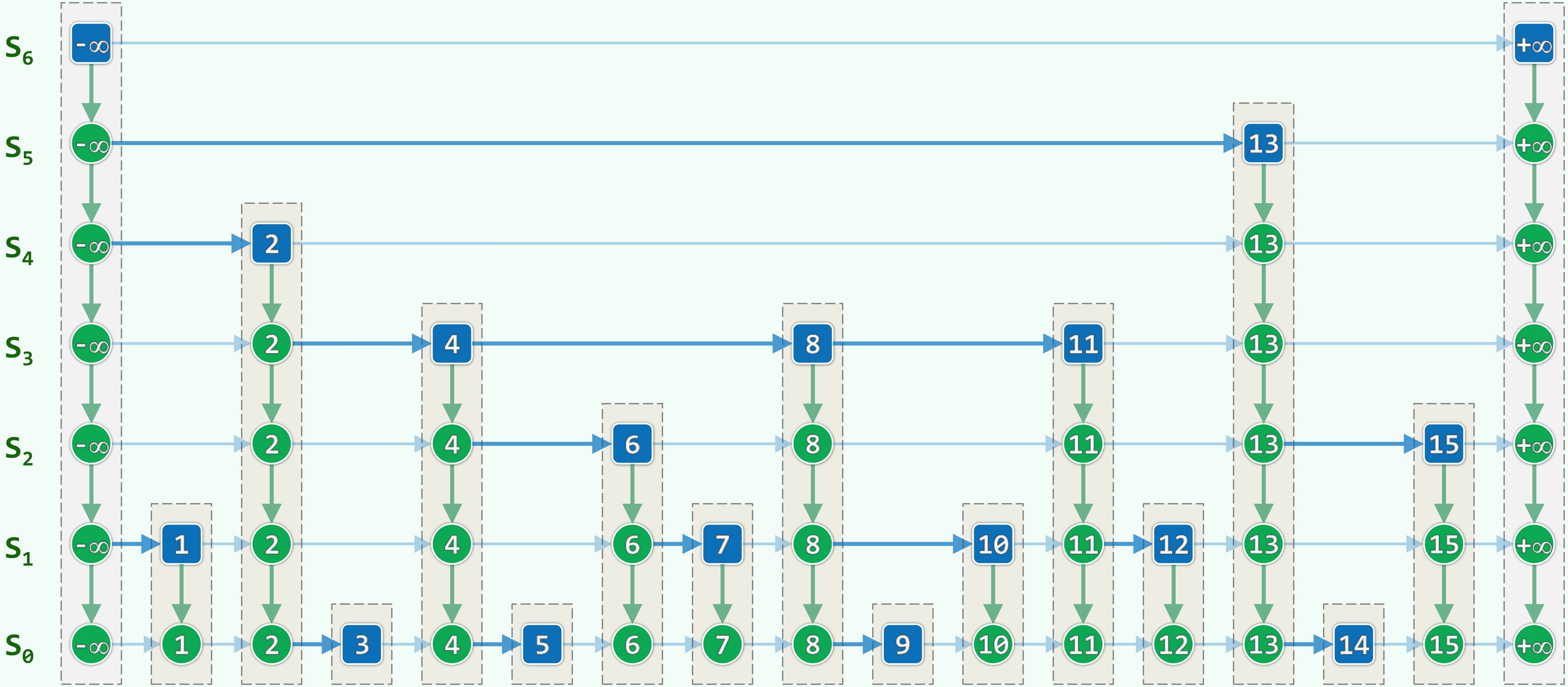
依次叠放**耦合**：S_0/S_h称作**底**/**顶**层

❖ 横向为**层**/level：prev()、next()、哨兵

❖ 纵向成**塔**/tower：above()、below()

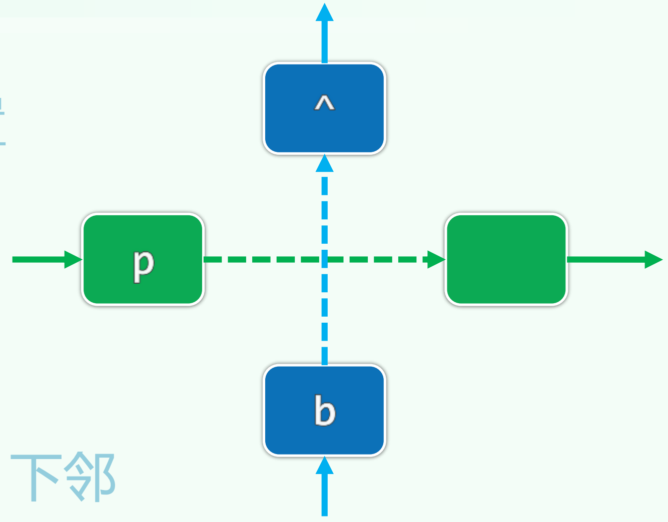


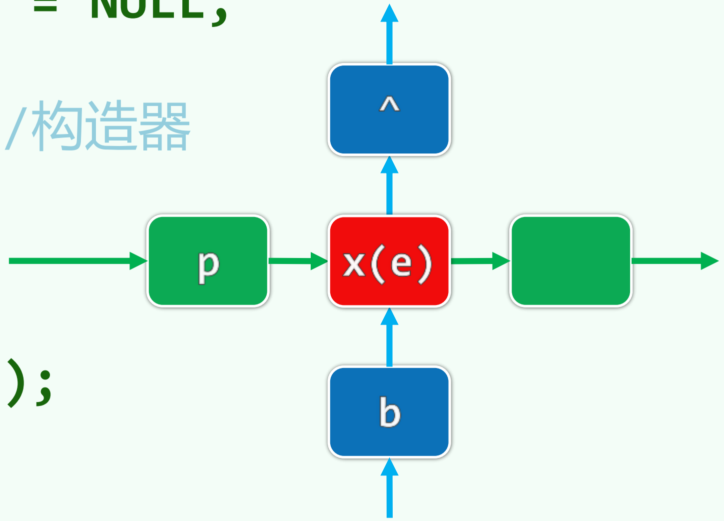
结构：实际 + 随机



QuadNode

```
template <typename T> using QNodePosi = QNode<T>*; //节点位置

template <typename T> struct QNode { //四联节点

   T entry; //所存词条

   QNodePosi<T> pred, succ, above, below; //前驱、后继、上邻、下邻

   QNode( T e = T(), QNodePosi<T> p = NULL, QNodePosi<T> s = NULL,

          QNodePosi<T> a = NULL, QNodePosi<T> b = NULL ) //构造器

      : entry(e), pred(p), succ(s), above(a), below(b) {}

   QNodePosi<T> insert( T const& e, QNodePosi<T> b = NULL );

      //将e作为当前节点的后继、b的上邻插入

};
```





QuadList

```
template <typename T> struct Quadlist { //四联表
   Rank _size; //节点总数
   QNodePosi<T> head, tail; //头、尾哨兵
   void init(); int clear(); //初始化、清除
   Quadlist() { init(); } //构造
   ~Quadlist() { clear(); delete head; delete tail; } //析构
   T remove( QNodePosi<T> p ); //删除p
   QNodePosi<T> insert( T const & e, QNodePosi<T> p, QNodePosi<T> b = NULL );
      //将e作为p的后继、b的上邻插入
};
```

Skiplist

```
template < typename K, typename V > struct Skiplist :
public Dictionary<K, V>, public List< Quadlist< Entry<K, V> >* > {
   Skiplist() { insertFirst( new Quadlist< Entry<K, V> > ); }; //至少有一层空列表
   QNodePosi< Entry<K, V> > search( K ); //由关键码查询词条
   Rank size() { return empty() ? 0 : last()->data->size(); } //词条总数
   Rank height() { return List::size(); } //层高，即Quadlist总数
   bool put( K, V ); //插入（Skiplist允许词条相等，故必然成功）
   V * get( K ); //读取
   bool remove( K ); //删除
};
```

空间性能

❖ 较之常规的单层列表

- 每次操作过程中，需**访问的节点**是否会实质地增多？
- 每个节点都至多可能**重复h份**，空间复杂度是否因此有实质增加？ //先来回答后者...

❖ **逐层随机减半**： S_k 中的每个关键码，在 S_{k+1} 中依然出现的概率，均为 $p = 1/2$

——稍后将会看到，这一性质可以通过**投掷硬币**来**模拟**

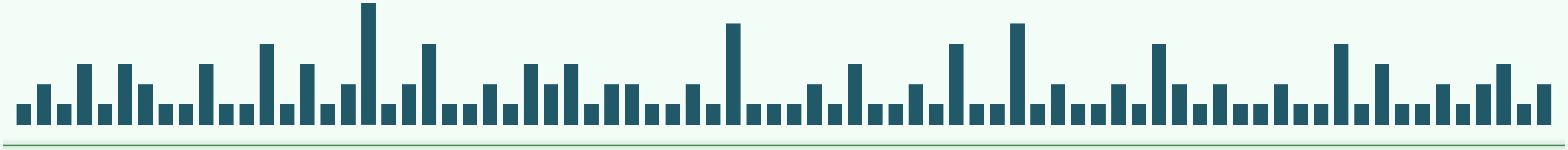
❖ 可见，各塔的高度符合**几何分布**： $Pr(h = k) \ = \ p^{k-1} \cdot (1-p)$

❖ 于是，期望的塔高： $\mathbb{E}(h) \ = \ 1/(1-p) \ = \ 2$

❖ 什么，没有学过概率？不要紧，有直观的解释...

空间性能

❖ 既然逐层随机减半，故 S_0 中任一关键码在 S_k 中依然出现的概率为 2^{-k}



❖ 第 k 层节点数的期望值 $\mathbb{E}(|S_k|) \ = \ n \cdot 2^{-k} \ = \ n/2^k$

❖ 于是，所有节点期望的总数（即各层列表所需空间总和）为

$$\mathbb{E}(\sum_k |S_k|) \ = \ \sum_k \mathbb{E}(|S_k|) \ = \ n \times \sum_k 2^{-k} \ < \ 2n \ = \ \mathcal{O}(n)$$

❖ 结论：跳转表所需空间为 expected-$\mathcal{O}(n)$

❖ 类比：**半衰期为1年**的放射性物质中，各粒子的**平均寿命**不过**2年**

❖ 更为细致地，塔高的**方差**是否足够小？ //比照稍后对层高的分析