

词典

跳转表：结构

11-XA1

邓俊辉

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。

deng@tsinghua.edu.cn

动机与思路

❖ [William Pugh, 1989]

Skip Lists: A Probabilistic Alternative to Balanced Trees

Skip lists are data structures that use

probabilistic balancing rather than

strictly enforced balancing

Algorithms for insertion and deletion in skip lists

are much simpler and significantly faster than

equivalent algorithms for balanced trees



结构

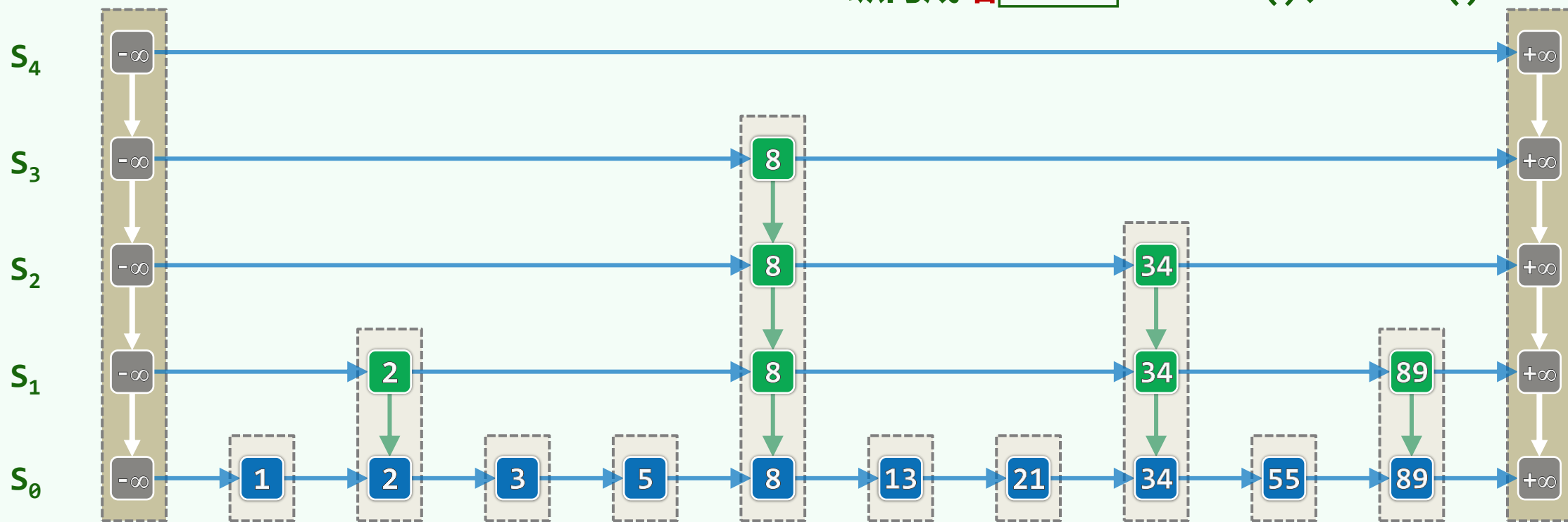
❖ 分层耦合的多个列表： s_0, s_1, \dots, s_h

s_0/s_h 称作底/顶层

❖ 横向为层 `level`：`prev()`、`next()`

设有头、尾哨兵

❖ 纵向成塔 `tower`：`above()`、`below()`



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...

❖ 生长概率逐层减半： 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，这一性质可以通过模拟投掷硬币来保证

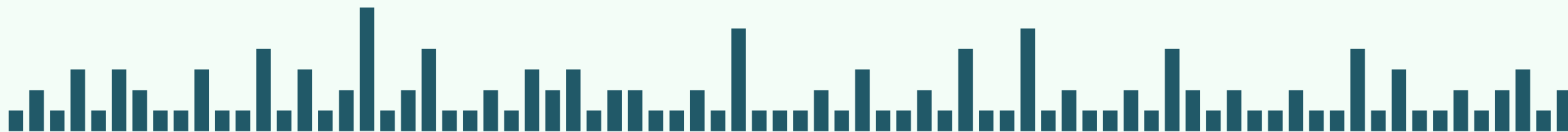
❖ 可见，各塔的高度符合几何分布： $Pr(h = k) = p^{k-1} \cdot (1 - p)$

❖ 于是，期望的塔高： $E(h) = 1/(1 - p) = 2$

❖ 什么，没有学过概率？不要紧，有直观的解释...

空间性能

❖ 既然生长概率逐层减半，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 $E(|S_k|) = n \cdot 2^{-k} = n/2^k$

❖ 于是，所有节点期望的总数（即各层列表所需空间总和）为

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

❖ 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，平均塔高的方差是否足够小？ // 比照稍后对层高的分析

QuadList

```
template <typename T> class Quadlist { //四联表

    private: int _size; QuadlistNodePosi(T) header, trailer; //规模、哨兵

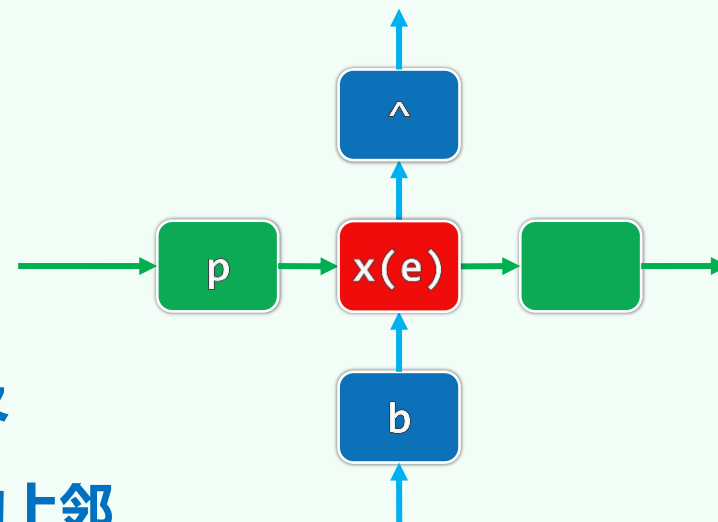
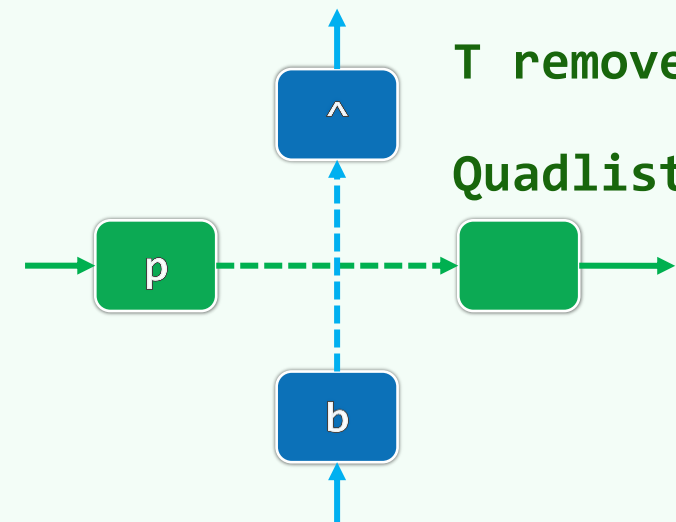
    protected: void init(); int clear(); //初始化、清除所有节点

    public: QuadlistNodePosi(T) first() const { return header->succ; } //首节点

            QuadlistNodePosi(T) last() const { return trailer->pred; } //末节点

            T remove( QuadlistNodePosi(T) p ); //删除p

            QuadlistNodePosi(T) insertAfterAbove( //插入
                T const & e, //数据项e , 使之成为
                QuadlistNodePosi(T) p, //p的后继 , 以及
                QuadlistNodePosi(T) b = NULL ); //b的上邻
```



```
};
```

Skiplist

```
template < typename K, typename V > class Skiplist :  
public Dictionary<K,V>, public List< Quadlist< Entry<K,V> >* > {  
    protected: bool skipSearch( ListNode< Quadlist< Entry<K,V> >* >* & qlist,  
                                   QuadlistNode< Entry<K,V> >* & p, K & k );  
  
    public:  
        int size() { return empty() ? 0 : last()->data->size(); }  
        int level() { return List::size(); } //层高, 即Quadlist总数  
        bool put( K, V ); //插入 ( Skiplist允许词条重复, 故必然成功 )  
        V * get( K ); //读取 ( 基于skipSearch()直接实现 )  
        bool remove( K ); //删除  
};
```