9.词典

（xa1）跳转表：结构

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。

邓俊辉

deng@tsinghua.edu.cn

动机与思路

❖ 可否综合向量与列表的优势，高效地实现词典接口？

具体地，如何使得各接口的效率均为$O(\log n)$？

❖ [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

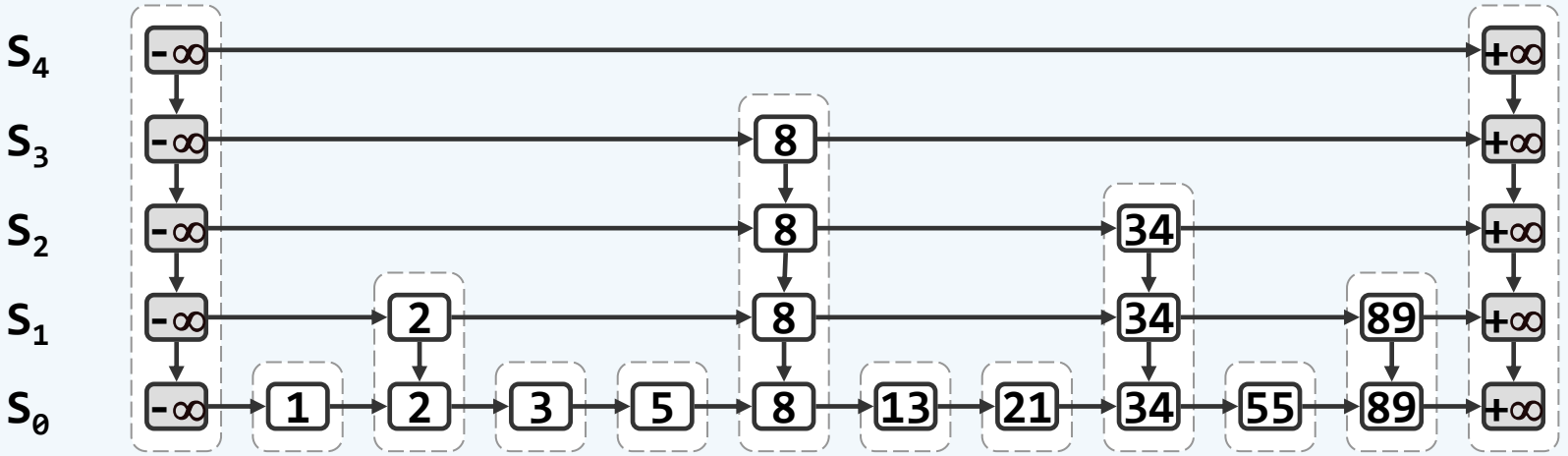
❖ 本节介绍非确定型跳转表，确定型 deterministic 跳转表可自学

结构：设计

❖ 分层次、相互耦合的多个列表：S_0, S_1, S_2, ..., S_h //层高 = h

❖ S_h称作顶层 top

S_0称作底层 bottom



❖ 各节点至多拥有四个引用：

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次，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 $p = 1/2$

——暂且假设成立，稍后说明如何保证

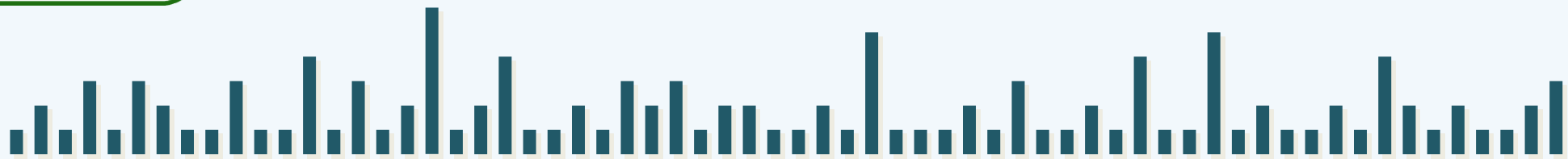
❖ 可见，各塔的高度符合几何分布：

$$Pr(h = k) = p^{k-1} \cdot (1 - p)$$

❖ 于是，期望的塔高 $E(h) = 1 / (1 - p) = 2$

❖ 什么，没有学过概率？不要紧，有初等的解释...

空间性能

❖ 

既然**生长概率逐层减半**：　S_0中任一关键码在S_k中依然出现的概率均为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

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

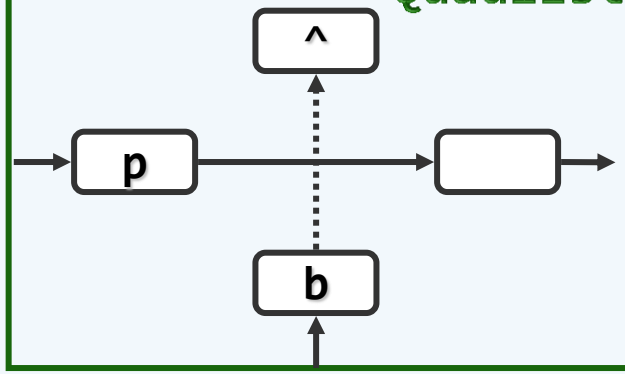
❖ 定理：　跳转表所需空间为expected-$O(n)$

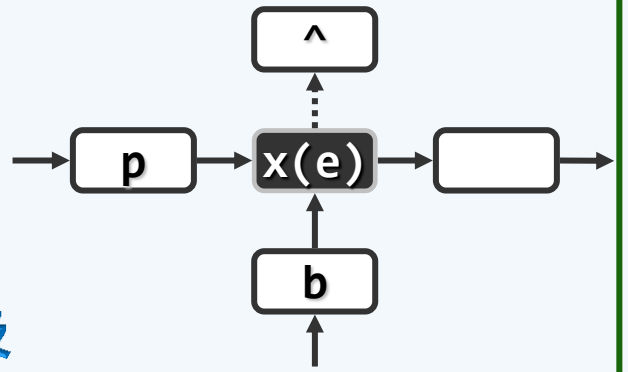
❖ 类比：　**半衰期**为**1年**的放射性物质中，各粒子的**平均寿命**不过**2年**

❖ 更为细致地，平均塔高的**方差**或**标准差**是否足够地小？　　//比照稍后对层高的分析

结构：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
};
```





结构：Skiplist

```
template < typename K, typename V > class Skiplist : //多重继承
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(); } //词条总数
      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
};
```