

词典

跳转表：查找

11-XA2

只见参仙老怪梁子翁笑嘻嘻的站起身来，向众人拱了拱手，缓步走到庭中，忽地跃起，左足探出，已落在欧阳克插在雪地的筷子之上，拉开架子，...，把一路巧打连绵的“燕青拳”使了出来，脚下纵跳如飞，每一步都落在竖直的筷子之上。

邓俊辉

deng@tsinghua.edu.cn

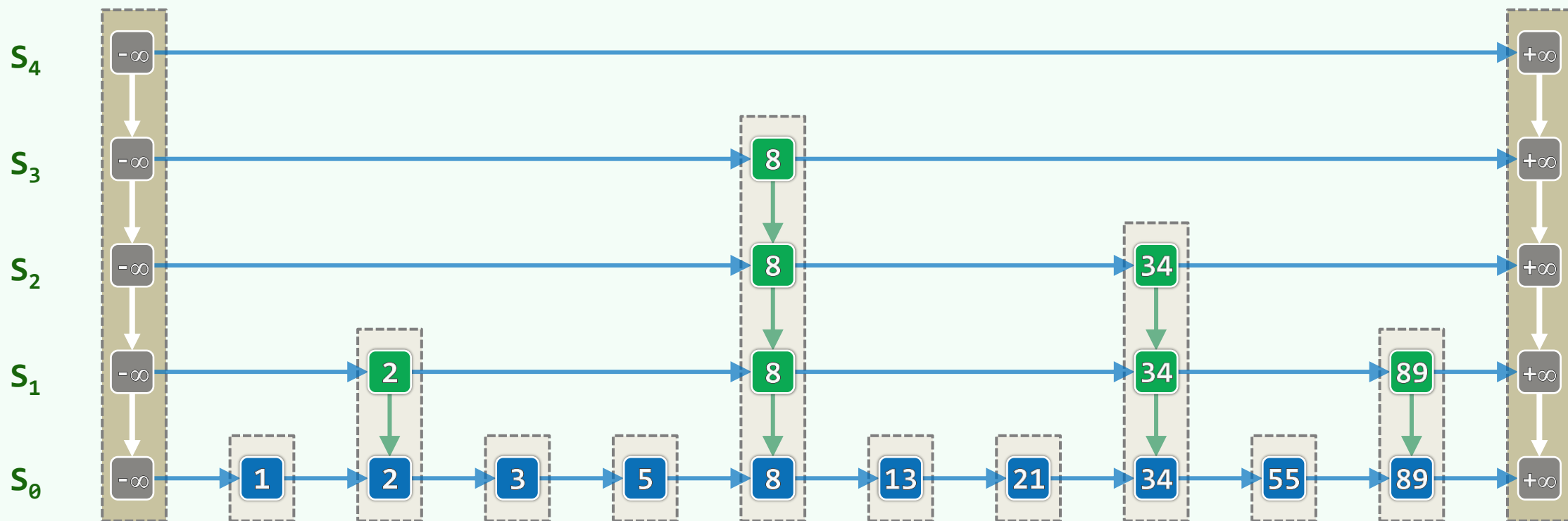
减而治之：由粗到细 = 由高到低

❖ 成功：search(21 | 34 | 1 | 89)

❖ 查找时间取决于**横向**、**纵向**的累计跳转次数

❖ 失败：search(80 | 0 | 99)

❖ 会否因**层次**过多，首先导致**纵向**跳转过多？

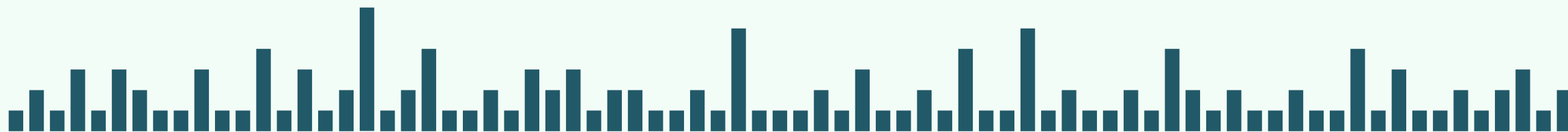


实现

```
template <typename K, typename V> bool Skiplist<K, V>::skipSearch(  
    ListNode< Quadlist< Entry<K,V> >* >* & qlist, //从指定层qlist的  
    QuadlistNode< Entry<K,V> >* & p, K & k ) { //首节点p出发, 向右、向下查找k  
    while ( true ) { //在每一层从前向后查找, 直到出现更大的key, 或溢出至trailer  
        while ( p->succ && ( p->entry.key <= k ) ) p = p->succ; p = p->pred;  
        if ( p->pred && ( k == p->entry.key ) ) return true; //命中则成功返回  
        qlist = qlist->succ; //否则转入下一层  
        if ( ! qlist->succ ) return false; //若已到穿透底层, 则意味着失败; 否则...  
        p = p->pred ? p->below : qlist->data->first(); //转至当前塔的下一节点  
    } //确认: 无论成功或失败, 返回的p均为其中不大于e的最大者?  
}  
//体会: 得益于哨兵的设置, 哪些环节被简化了?
```

纵向跳转 ~ 层高

❖ 观察：一座塔高度不低于 k 的概率 = $p^k \mid 1 - p^k$



❖ 引理：随着 k 的增加, S_k 为空/非空的概率急剧上升/下降

$$Pr(|S_k| = 0) = (1 - p^k)^n \geq 1 - n \cdot p^k$$

$$Pr(|S_k| > 0) \leq n \cdot p^k$$

❖ 推论：跳转表高度 $h = \mathcal{O}(\log n)$ 的概率极大

❖ 比如：若 $p = 1/2$ ，则第 $k = 3 \cdot \log n$ 层非空（当且仅当 $h \geq k$ ）的概率为

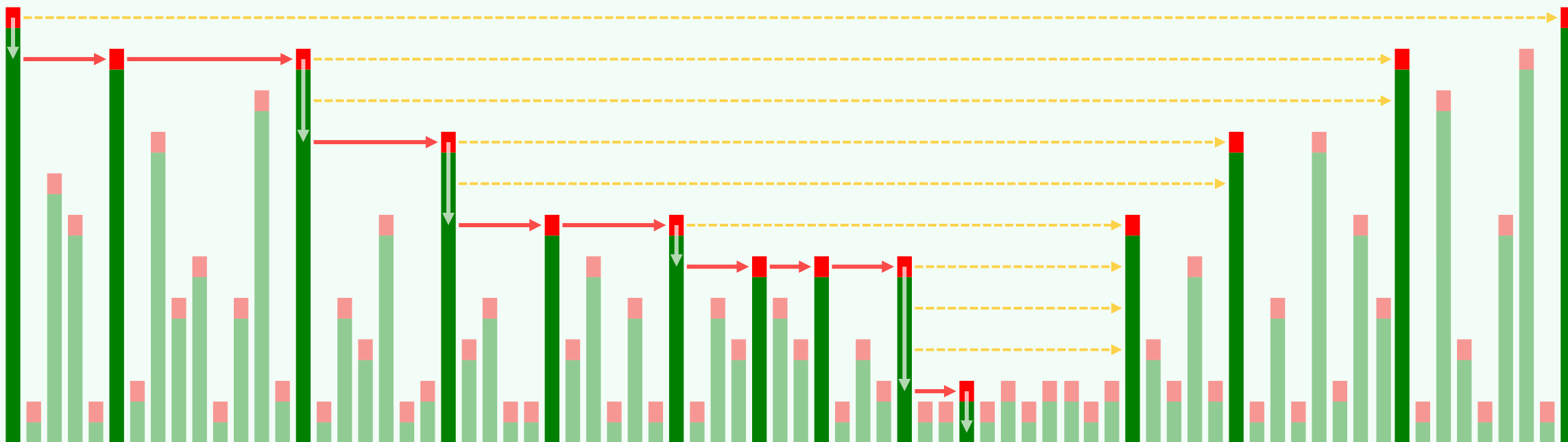
$$Pr(|S_k| > 0) \leq n \cdot p^k = n \cdot n^{-3} = 1/n^2 \mapsto 0$$

❖ 结论：查找过程中，纵向跳转的次数，累计不过 $\text{expected-}\mathcal{O}(\log n)$

横向跳转 ~ 紧邻塔顶

❖ 那么：**横向**跳转是否可能很多次？比如 $\omega(\log n)$ ，甚至 $\Omega(n)$ ？

❖ 观察：在同一水平列表中，横向跳转所经节点必然依次**紧邻**，而且每次抵达都是**塔顶**

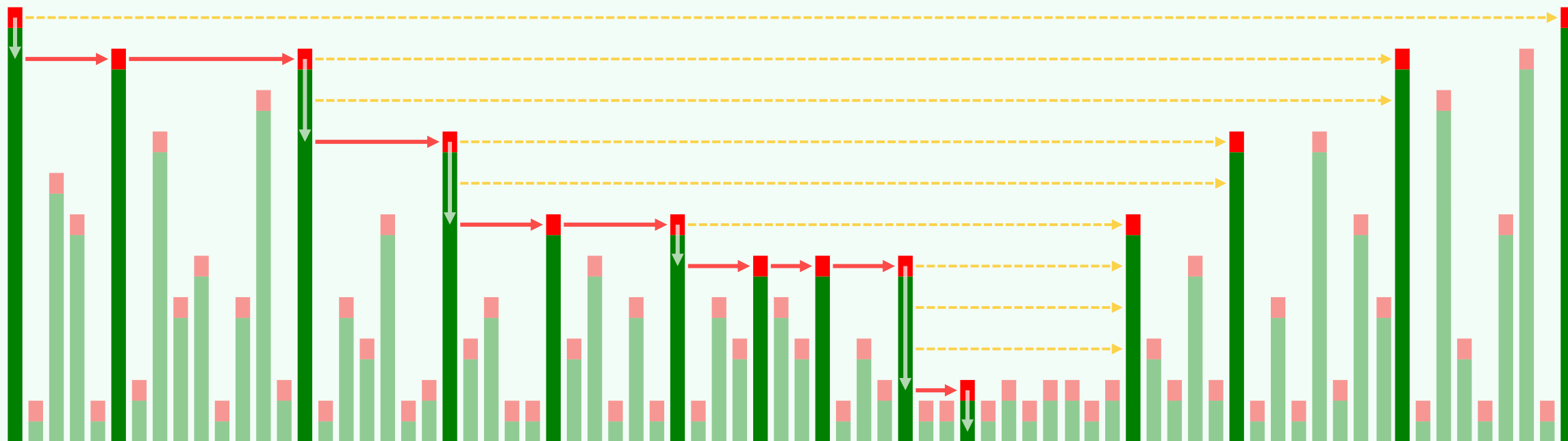


❖ 于是：若将沿同一层列表跳转的次数记作 Y ，则它符合**几何分布** $Pr(Y = k) = (1 - p)^k \cdot p$

横向跳转 ~ 期望时间成本

❖ 定理： $E(Y) = (1 - p)/p = (1 - 0.5)/0.5 = 1$ 次

❖ 因此：在同一层列表中连续跳转的时间成本不过 $1 + 1 = 2$



❖ 结论：跳转表的每次查找，都可在 $\leq \text{expected-}(2h) = \text{expected-}\mathcal{O}(\log n)$ 时间内完成