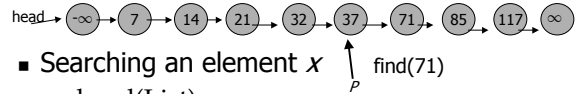


SkipList

Alon Efrat
Computer Science Department
University of Arizona

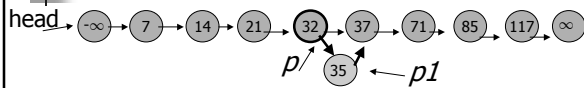
Searching a key in a Sorted linked list



- Searching an element x
- $p = \text{head}(\text{List})$;
- while ($\text{key}[\text{next}[p]] < x$) $p = \text{next}[p]$;
- return p ;
- Note: we actually return the largest element which is smaller than x .

2

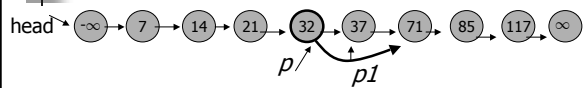
inserting a key into a Sorted linked list



- In order to insert 35 -
- $p = \text{find}(35)$; /*Recall the finds point to the previous element of the one containing x */
- $p1 = \text{new_cell}()$;
- $\text{key}[p1] = 35$;
- $\text{next}[p1] = \text{next}[p]$;
- $\text{next}[p] = p1$;

3

deleting a key from a sorted list

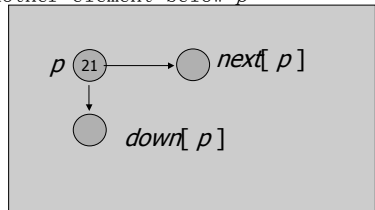


- To delete 37 -
- $p = \text{find}(37)$;
- $p1 = \text{next}[p]$;
- $\text{next}[p] = \text{next}[p1]$;
- $\text{free}[p1]$;

4

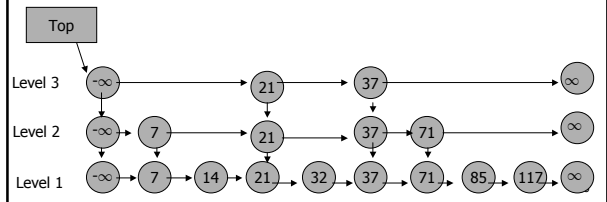
A data structure for maintaining a set of keys in a sorted order --- SKIP LIST

- Introduced by William Pugh (1990).
- Three operations on an element p
 - $key[p]$ - the value of the key that p stores
 - $next[p]$ - similar to its purpose in a linked list
 - $down[p]$ - another element below p



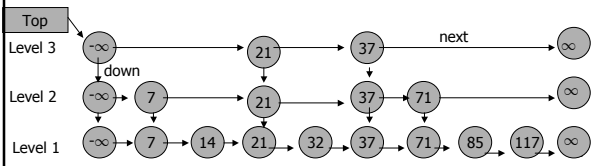
SL - rules

- Each SL Consists of several **levels**.
- All keys appear in level 1 (lowest level)
- Each level is a sorted list.
- If key x appears in level i , then it also appears in all levels below i



More rules

- An element in level i points (via $down[]$) to the element with the same key in the level $i-1$.
- In each level the keys $-\infty$ and ∞ appear.
- Top is the element with the smallest key $-\infty$ in the highest level.

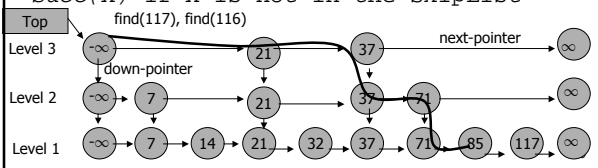


An empty SkipList



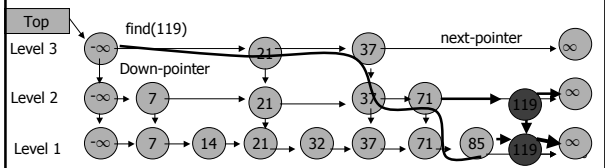
Finding an element with key x

- $p = \text{top}[\text{SL}]$
- While(1){
 - while ($\text{key}[\text{next}[p]] < x$) $p = \text{next}[p]$;
 - if ($\text{down}[p] \neq \text{NULL}$) return $\text{next}[p]$;
 - $p = \text{down}[p]$;
- }
- Observe that we return x , if exists, or $\text{succ}(x)$ if x is not in the SkipList



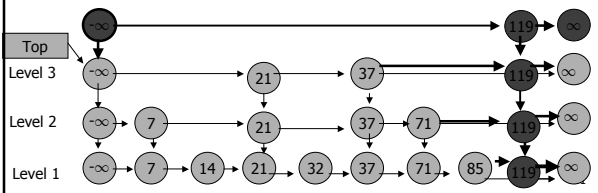
Inserting new element x

- Determine $k(x)$ - the number of levels in which x participates (explained later)
- Do $\text{find}(x)$, and insert x to the appropriate places in the lowest k levels (after the elements at which the search path turns down or terminates)
- Example - inserting 119, $k(x)=2$.



Inserting an element - cont.

- If k is larger than the current number of levels, add new levels (and update top)
- Example - insert(119) when $k=4$

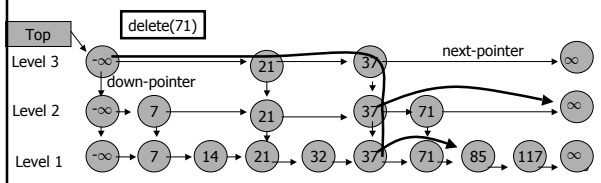


Determining $k(x)$

- $k(x)$ is the number of levels at which an element x participate.
- Use a random function $\text{OurRnd}()$ --- returns 1 or 0 (True/False) with equal probability.
- $k=1$;
- While($\text{OurRnd}()$) $k++$;

Deleteing a key x - overview

- Find x in all the levels it participates, and delete it using the standard 'delete from a linked list' method.
- If one or more of the upper levels are empty, remove them (and update top)

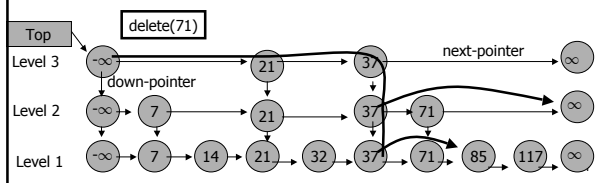


Finding an element with key x

- ```

p=top[SL]
While(1){
 while (key[next[p]] < x) p=next[p];
 if (key[next[p]] == x) delete(p);
 if (down[p]==NULL) return ;
 p = down[p];
}

```



## Facts about SL

- The expected number of levels is  $O(\log n)$
- (here  $n$  is the numer of elements)
- The expected time for insert/delete/find is  $O(\log n)$
- The expected size (number of cells) is  $O(n)$