COL 702 Sept 6 Lecture 12

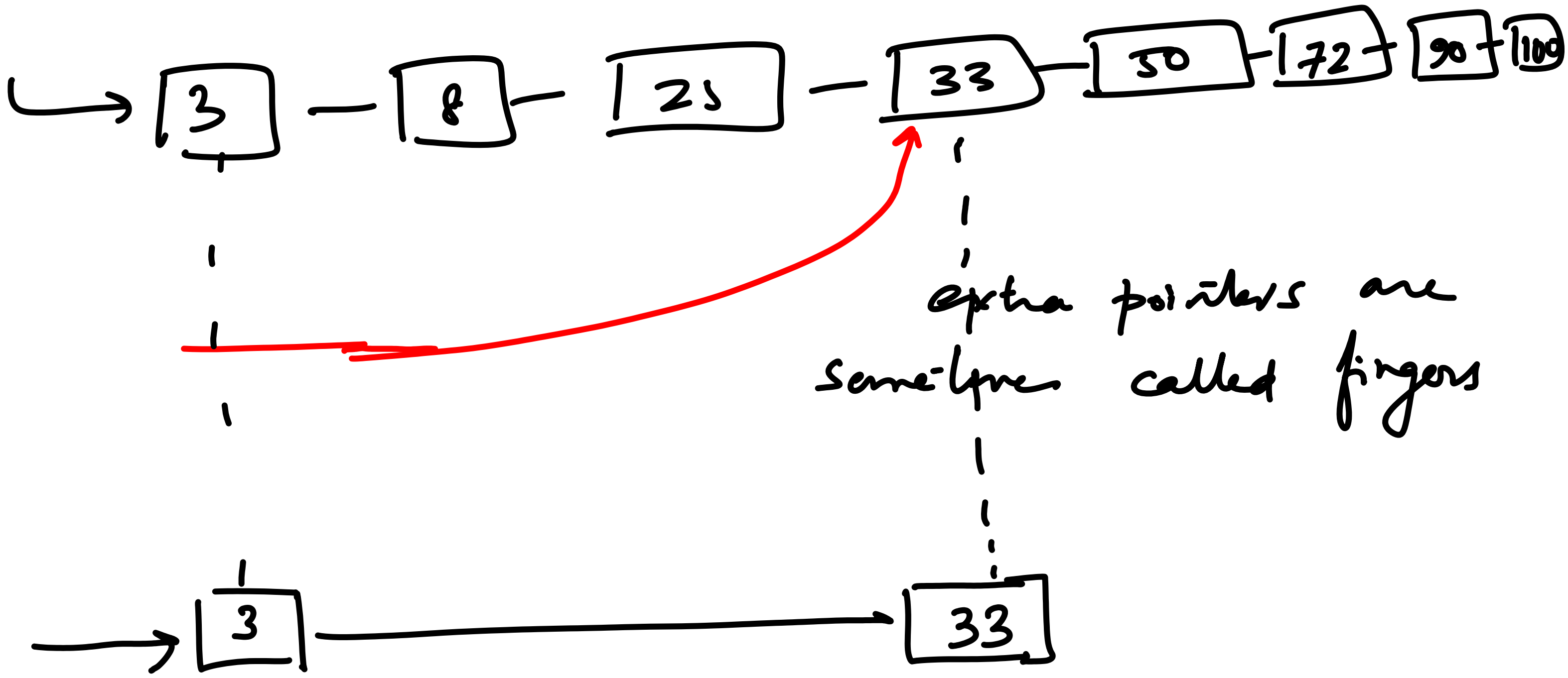
Dictionary data structure : fully dynamic

Search, insert, delete

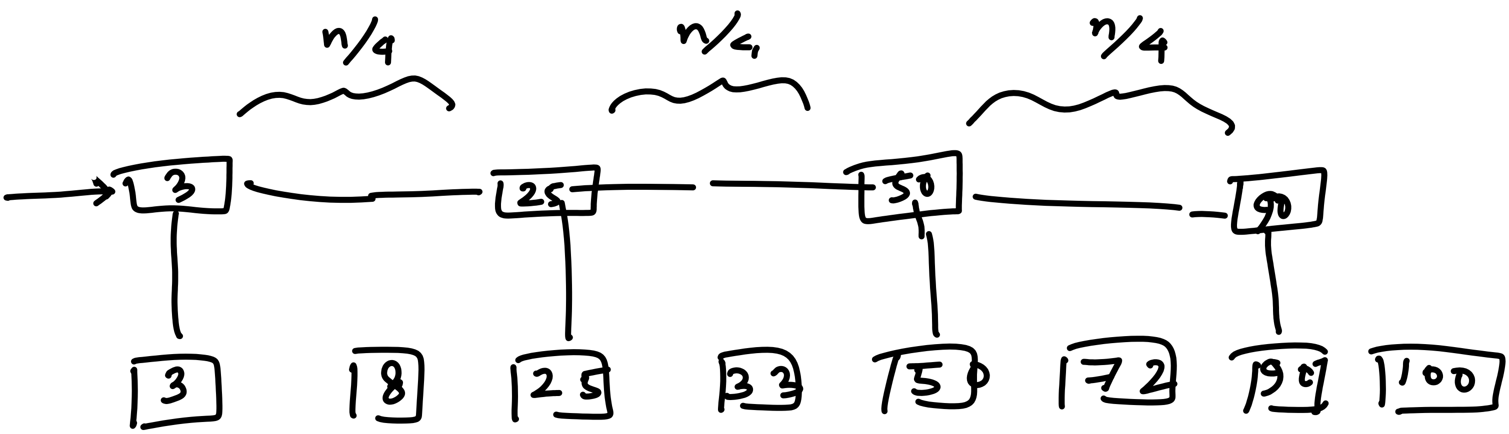
Arrays	Linked Lists	BST
1. for sorted arrays logn time Binary Search (because of random access)	Sequential search Even for sorted list : $O(n)$	$O(\log n)$
2. Insert / Delete are not efficient	Insert / Delete : $O(1)$ given that we know the location	$O(\log n)$ for balanced AVL, red bl. Btrees, a-b trees $BB(\alpha)$ trees Splay

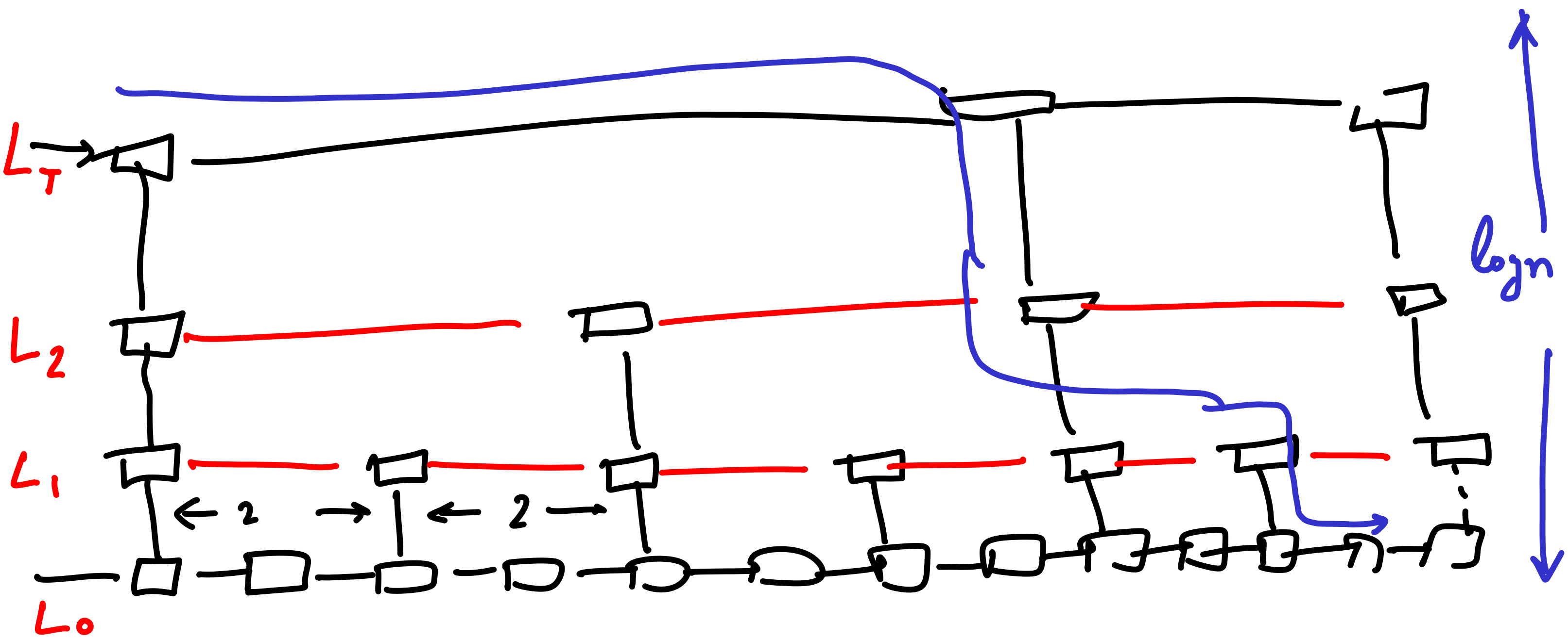
Search 49

find an interval : consecutive elements x_i x_{i+1} s.t. $x_i \le 49 \le x_{i+1}$



extra pointers are sometimes called fingers





$|L_0| = n$

Total size of all lists

$$\therefore \; n + \frac{n}{2} + \frac{n}{4} + \cdots = O(n)$$

Search time : Time to search for L_T +

$$\sum_{\text{all levels } i} \left(\text{Time to refine the search from } L_i \text{ to } L_{i-1} \right)$$

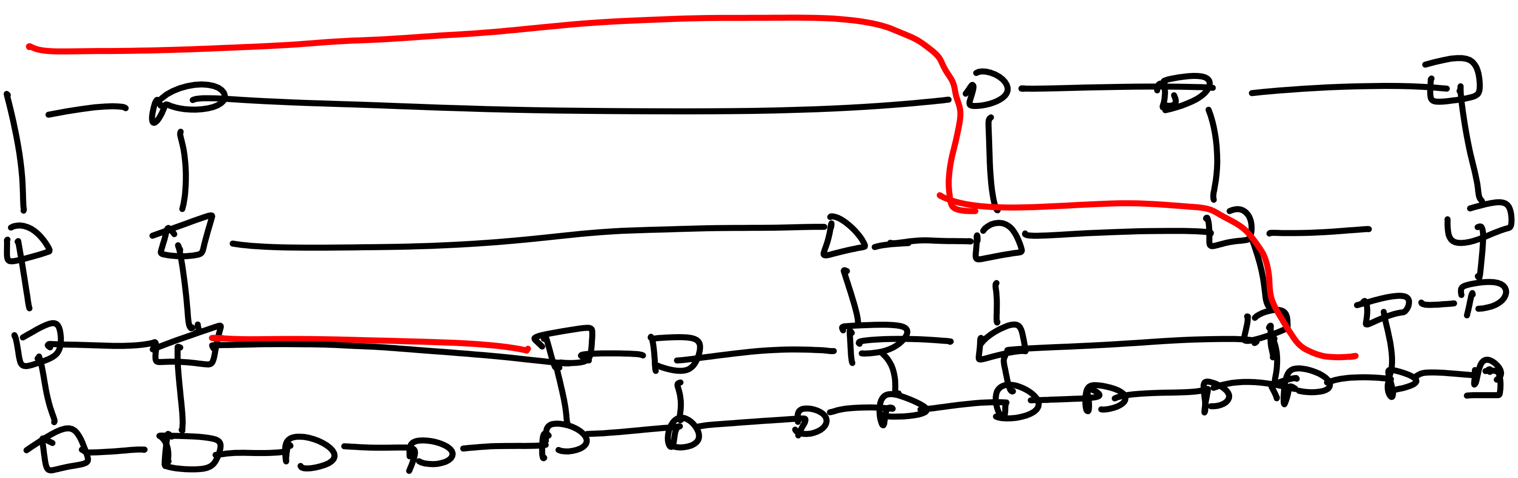
$$\therefore \; O(1) + \#\text{levels} \times O(1)$$

$$= O(\#\text{ levels}) = O(\log n)$$

The efficiency of the data structure is based on the invariant : It takes $O(1)$ time to descend for L_i to L_{i-1} "expected $O(1)$"

This is easy to maintain in the static case (every 2^{nd} element / 3^{rd} element) but very difficult with insertion/deletion

Construct L_i from L_{i-1} using coin tossing : If it is heads then promote else not



SKIP LIST

#elements in a level, # levels depend on outcomes of coin tosses and maybe thought of as random variables

Space of the data

Expected size of the skip list

$$= \sum \text{expected \# copies of each element}$$

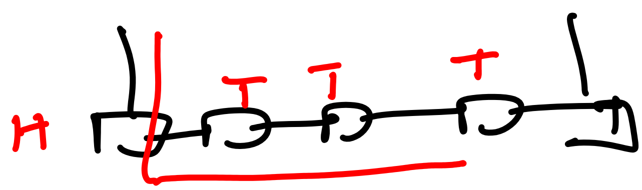
↓

how many time (expected) do we toss a coin to get a "tail"

$= 2$ (follows from properties of geometric r.v)

$= 2n$

Expected length of the path within a level $= 2$



How many levels ?

Two ways to stop the process

① We fix the height & be
$k = 2\log n$: what is expected # elements in level $2\log n$

② What is the expected # levels when the size of the list < 10 ?

: