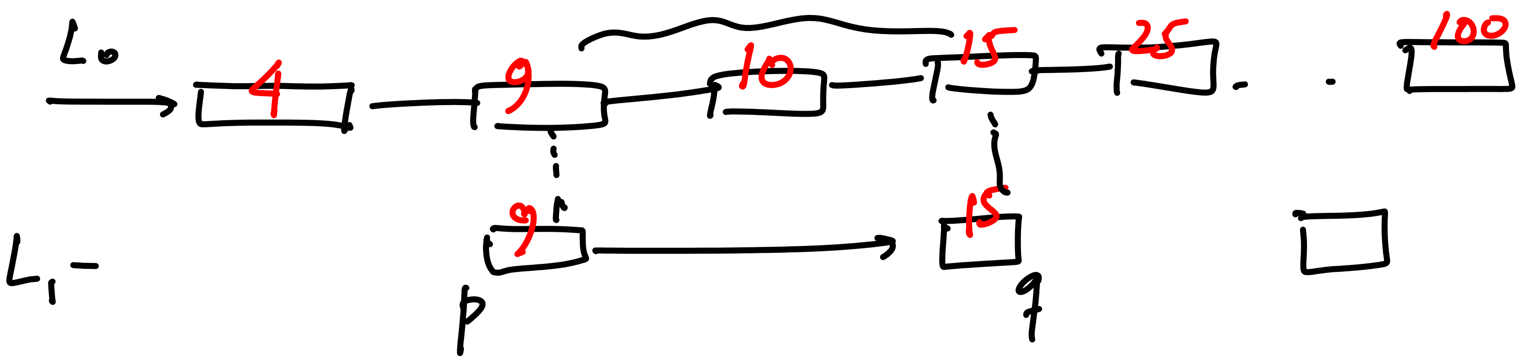
CSL 630 Lecture 20 Oct 13

An alternate scheme for dynamic dictionary.

We have a sorted list



Search the sublist to find two elements, say p, q s.t. the element we are searching for, viz. x

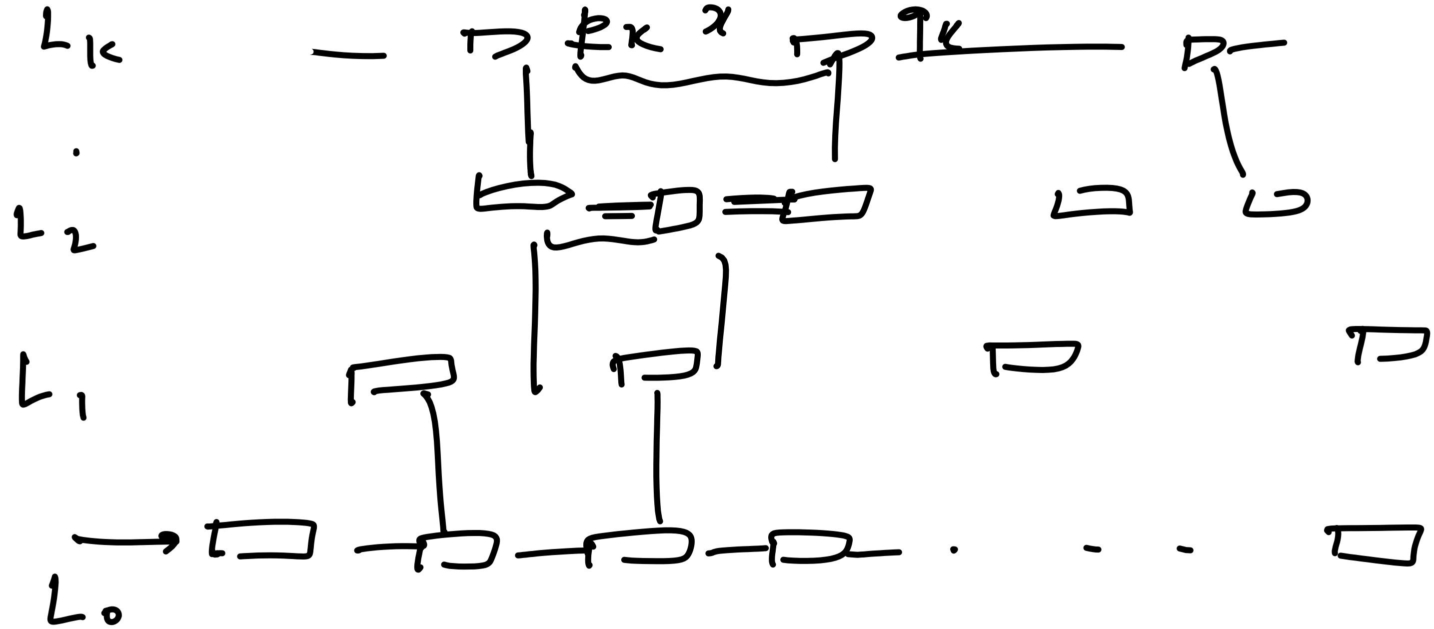
$$p \leq x \leq q$$

The time for refinement is $\left| [p, q] \cap L_0 \right|$

\# elements of L_0 in the interval $[p, q]$

How do we search in L_1?

Recursive search in L_1 using sublist L_2



$L_i \subset L_{i-1}$ L_0 is the original list

We maintain pointers between L_i and L_{i-1}, so that we know which interval in L_{i-1} we must search given the position of x in L_i

We obtain a sequence

$[p_k, q_k]$ $[p_{k-1}, q_{k-1}]$ $\cdots$ $[p_0, q_0]$

$p_i \leq x \leq q_i$

Either $x = p_i$ or $x = q_i$ or these are the closest elements to x in L_i

The time to search is the cost of refinement in each level

$$\sum_i \underbrace{\left| [p_i, q_i] \cap L_{i-1} \right|}_{\text{Suppose this is constant}}$$

Then total time is $O(K)$

We would like K to be $\log n$

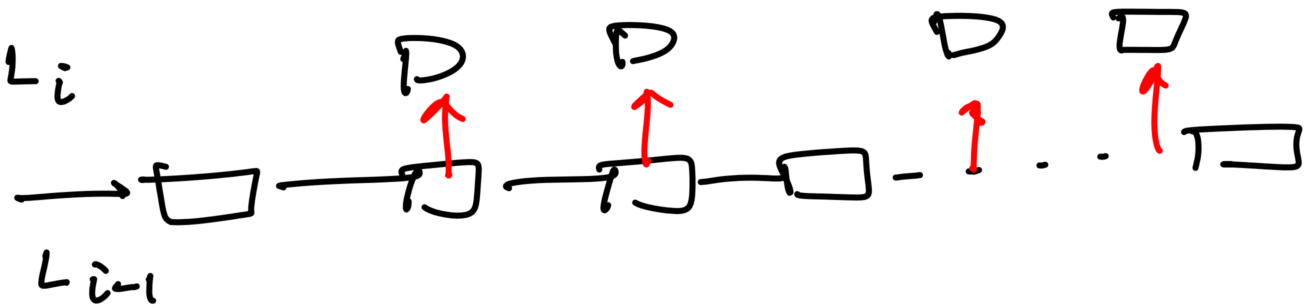
Invariant

If L_i is chosen to be every alternate element of L_{i-1} then cost of refinement is 2 and $K = \log n$

What happen when elements are added / deleted

Picking every 2^{nd} element / 3^{rd} elem. is too rigid and would be expensive to maintain dynamically.

Pick elements from L_{i-1} randomly (with prob $=\frac{1}{2}$) to construct L_i



How many elements in L_i

$$E[|L_i|] = \frac{1}{2}|L_{i-1}|$$

copies of an element (# levels) is the number of consecutive heads we obtain when we toss a fair coin

Say X_i is the r.v. # of copies of element i

$$E[X_i] = 0\cdot\frac{1}{2} + 1\cdot\frac{1}{2}\cdot\frac{1}{2} + 2\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\frac{1}{2} + \cdots i\cdot\left(\frac{1}{2}\right)^{i+1}$$

$$= 2$$

Expected size of the data structure

$$= 2\times n$$

What is the expected size of the "gap" $|[p_i, q_i] \cap L_{i-1}|$