Why SkipList?

- Singly Linked List get(T x) function (search function) is O(n)
- SkipList get(T x) function is O(lgn) mostly

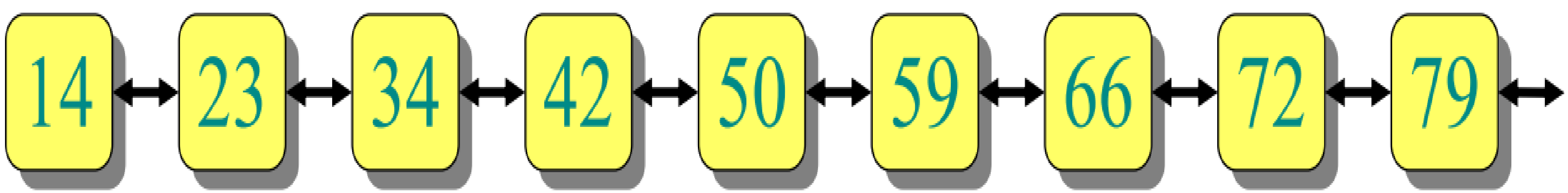
- Almost all of the discussion here is taken from introduction to algorithms cormen slides

One linked list

Start from simplest data structure:

(sorted) linked list

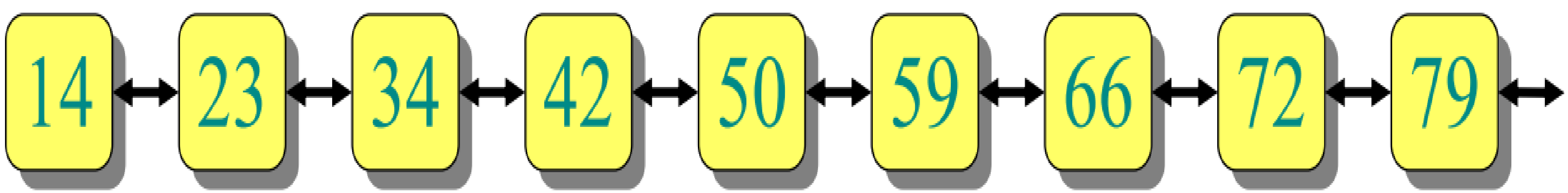
- Searches take $\Theta(n)$time in worst case
- How can we speed up searches?



Two linked lists

Suppose we had two sorted linked lists
(on subsets of the elements)

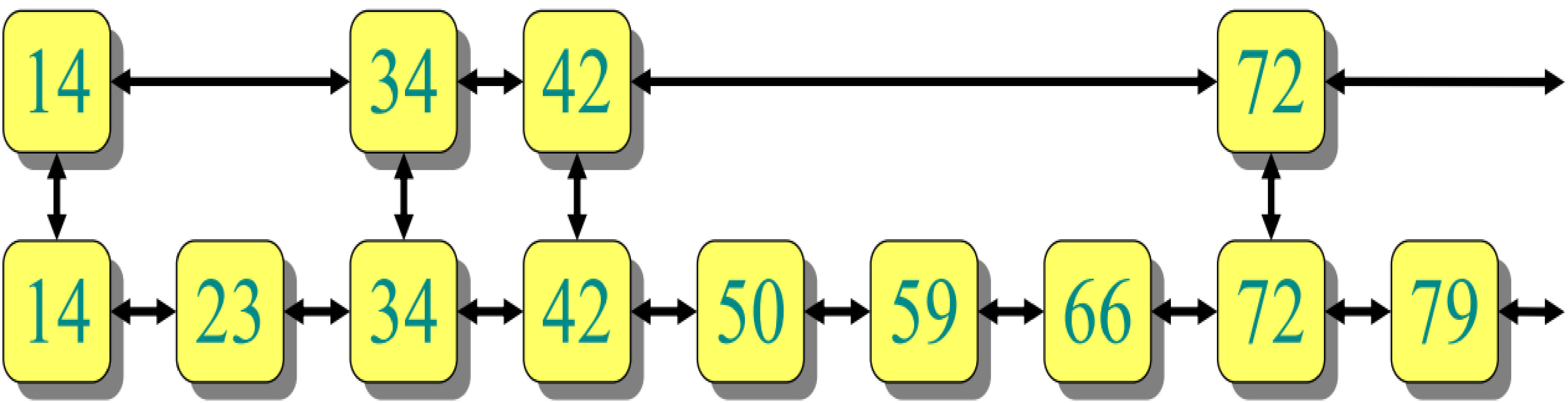
- Each element can appear in one or both lists
- How can we speed up searches?



Two linked lists as a subway

IDEA:Express and local subway lines

- Express line connects a few of the stations
- Local line connects all stations
- Links between lines at common stations



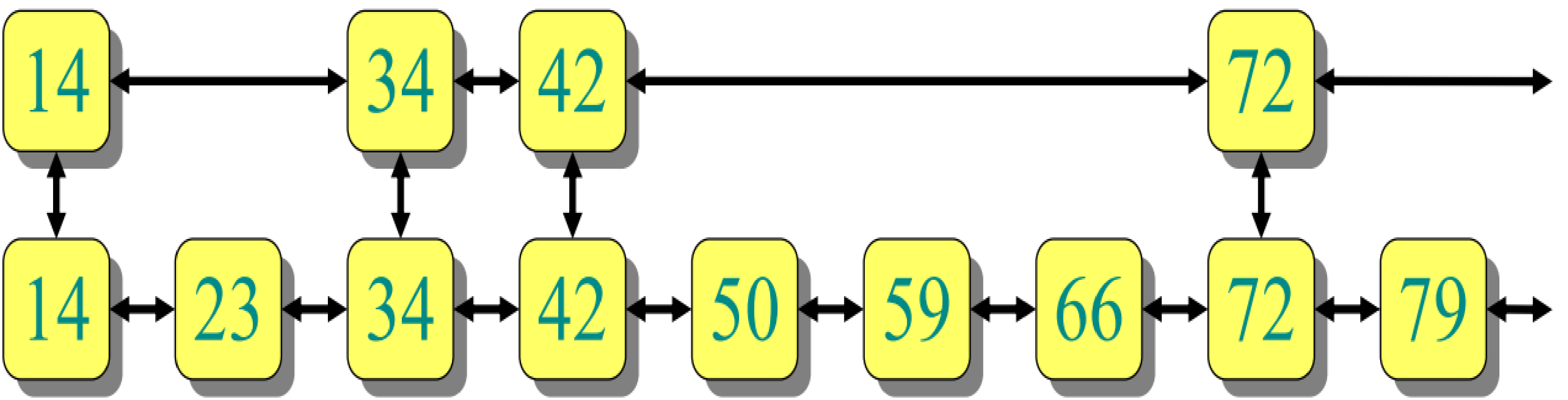
Searching in two linked lists

SEARCH(x):

- Walk right in top linked list (L1)

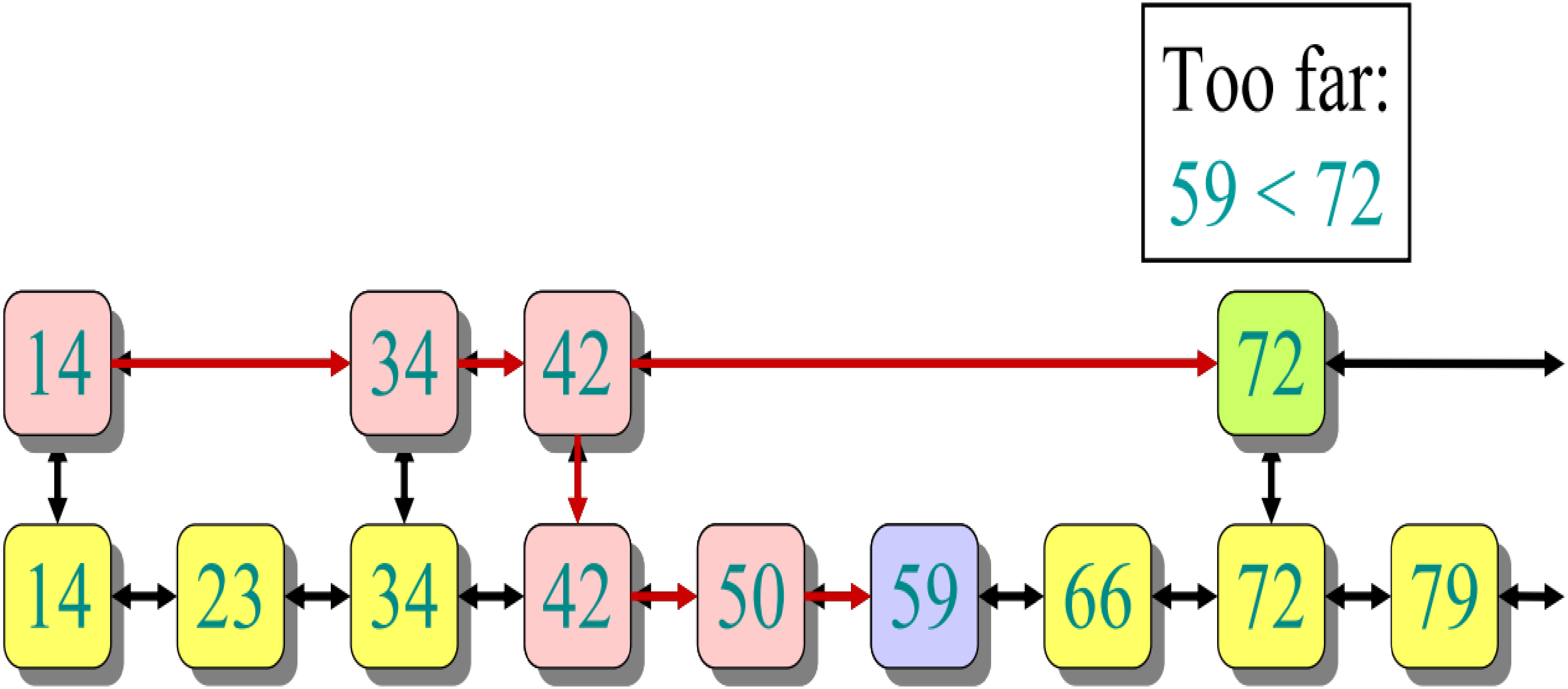
until going right would go too far

- Walk down to bottom linked list (L2)
- Walk right in L2 until element found (or not)



Searching in two linked lists

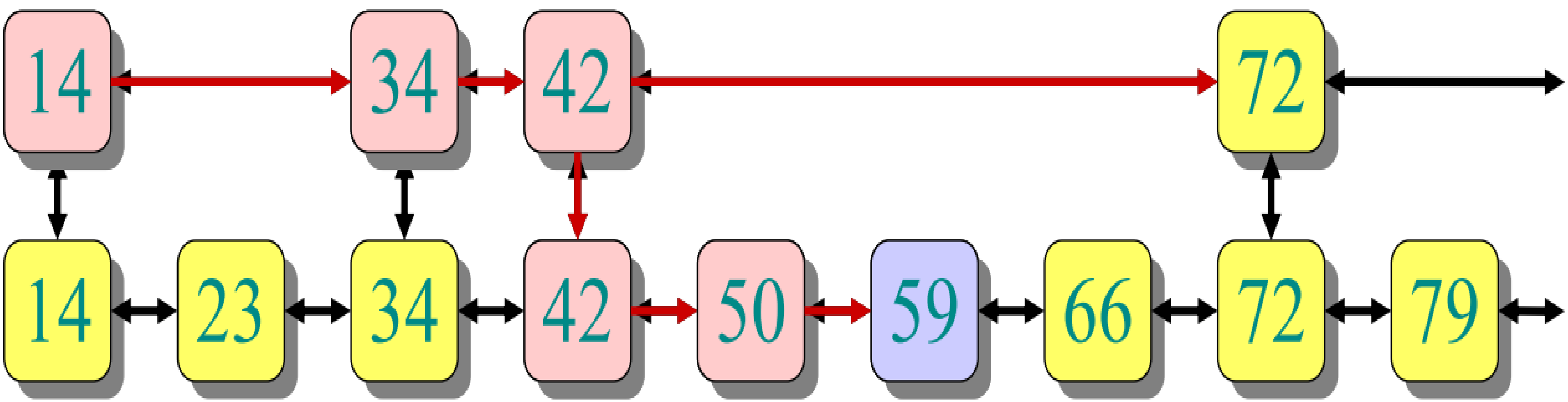
EXAMPLE:SEARCH(**59**)



Design of two linked lists

QUESTION: Which nodes should be in L1?

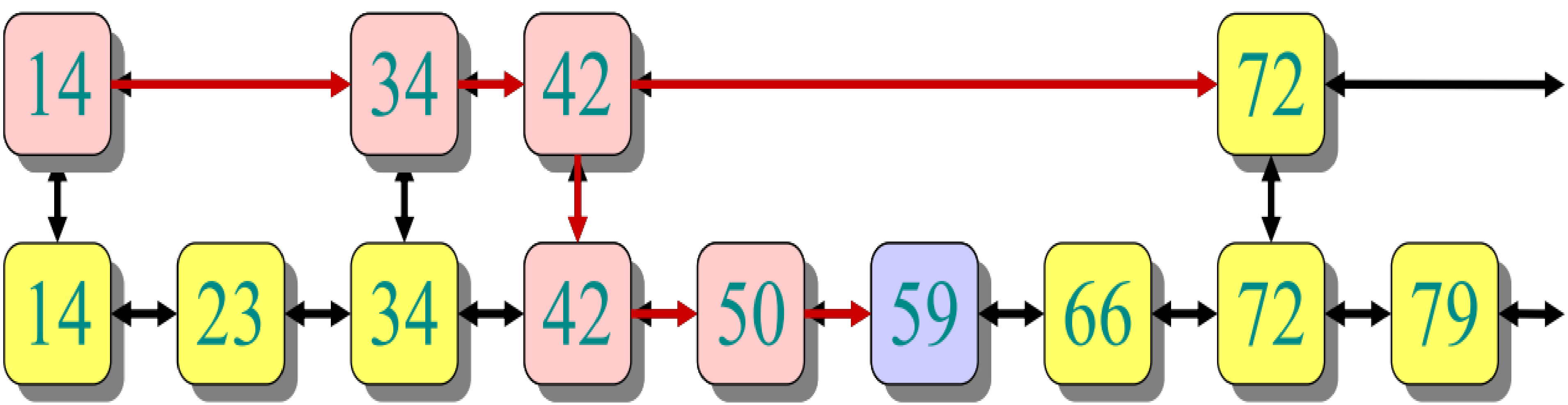
- In a subway, the "popular stations"
- Here we care are about worst-case performance
- Best approach: Evenly space the nodes in L1
- But how many nodes should be in L1?



Analysis of two linked lists

ANALYSIS:

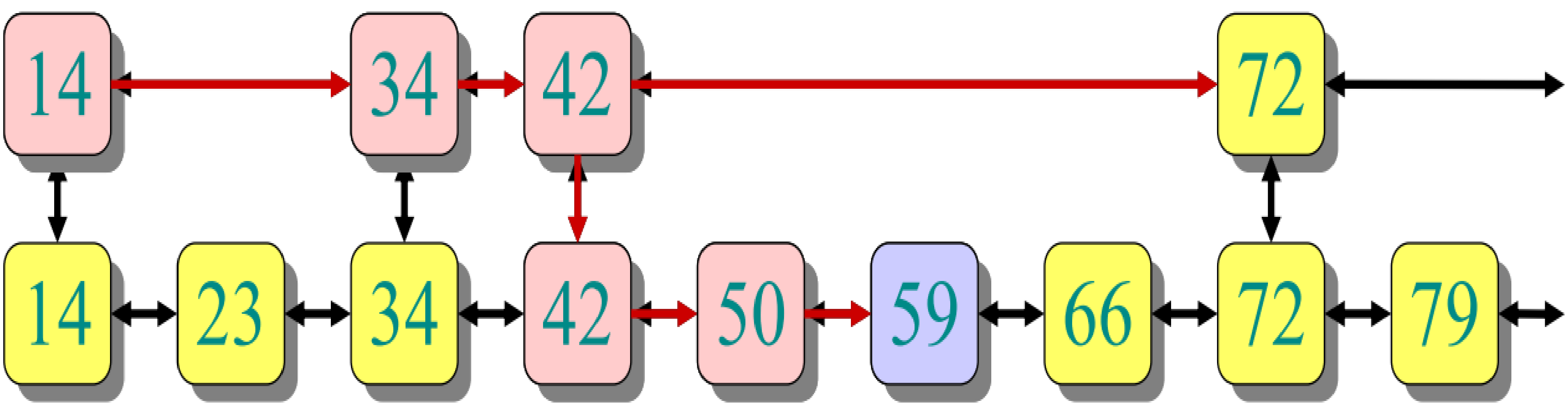
- Search cost is roughly $|L_1| + \frac{|L_2|}{|L_1|}$
- Minimized (up to constant factors) when terms are equal
- $|L_1|^2 = |L_2| = n \Rightarrow |L_1| = \sqrt{n}$



Analysis of two linked lists

ANALYSIS:

- Search cost is roughly $|L_1| + \frac{|L_2|}{|L_1|}$
- Minimized (up to constant factors) when terms are equal
- $|L_1|^2 = |L_2| = n \Rightarrow |L_1| = \sqrt{n}$

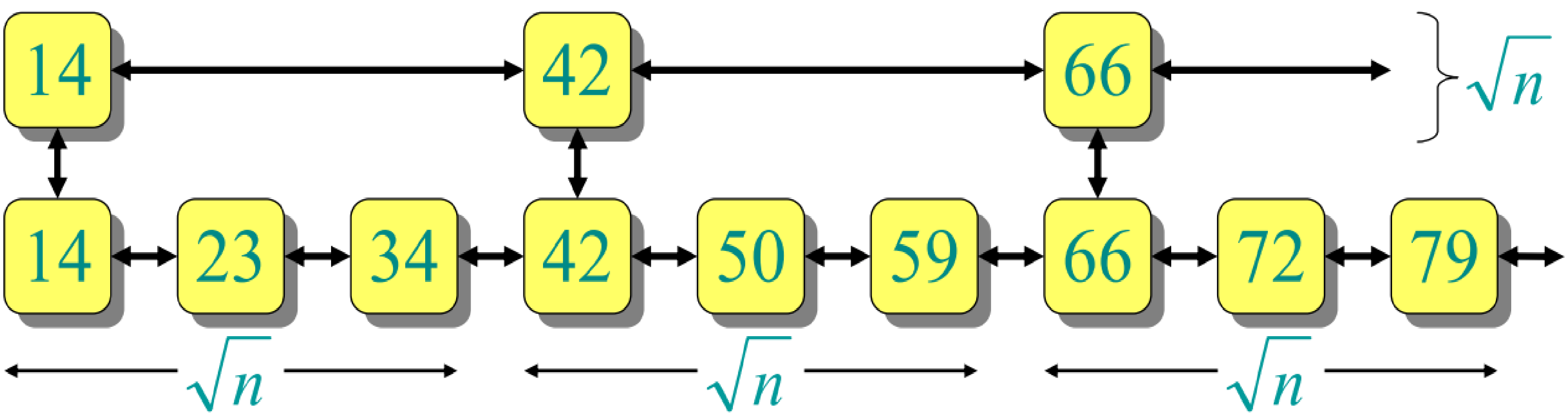


Analysis of two linked lists

ANALYSIS:

- $|L_1| = \sqrt{n}$, $\quad |L_2| = n$
- Search cost is roughly

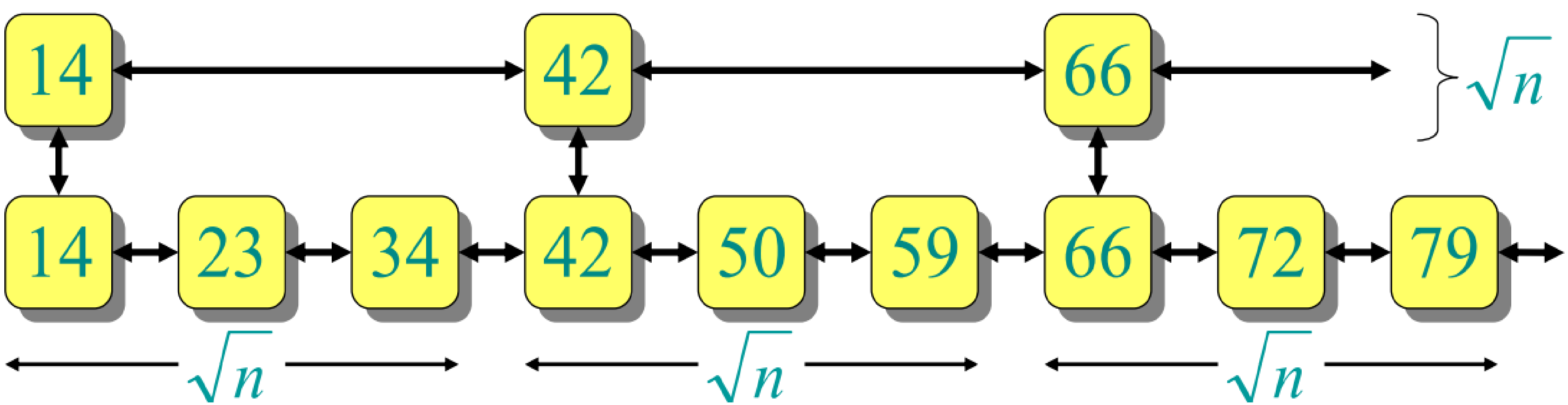
$$|L_1| + \frac{|L_2|}{|L_1|} = \sqrt{n} + \frac{n}{\sqrt{n}} = 2\sqrt{n}$$



More Linked Lists

What if we had more sorted linked lists?

- 2 sorted lists $\Rightarrow 2 \cdot \sqrt{n}$
- 3 sorted lists $\Rightarrow 3 \cdot \sqrt[3]{n}$
- k sorted lists $\Rightarrow k \cdot \sqrt[k]{n}$
- $\lg n$ sorted lists $\Rightarrow \lg n \cdot \sqrt[\lg n]{n} = 2 \lg n$



Simplification

Suppose we have n =16 elements in our bottom list

x = lg n = 4 => there will be 4 lists **L1**, **L2**, **L3**, **L4**

L4 will contain $\mathbf{n^{(x/x)} = n}$ elements => **16** elements

L3 will contain $\mathbf{n^{(x\ -\ 1/x)} = n^{(0.75)}}$ elements => $\mathbf{16^{(3/4)}}$ elements = **8** elements

L2 will contain $\mathbf{n^{(x\ -\ 2/x)} = n^{(0.5)}}$ elements => $\mathbf{16^{(2/4)}}$ elements = **4** elements

L1 will contain $\mathbf{n^{(x\ -\ 3/x)} = n^{(0.25)}}$ elements => $\mathbf{16^{(1/4)}}$ elements = **2** elements

Simply there will be **n elements in L4**, **n/2 in L3**, **n/4 in L2**, **n/8 in L1**

Simplification

Search cost will be

$$|L1| + |L2|/|L1| + |L3|/|L2| + |L4|/|L3|$$

$$= n^{1/4} + n^{2/4} / n^{1/4} + n^{3/4} / n^{2/4} + n^{4/4} / n^{3/4}$$

$$= n^{1/4} + n^{2/4 - 1/4} + n^{3/4 - 2/4} + n^{4/4 - 3/4}$$

$$= n^{1/4} + n^{1/4} + n^{1/4} + n^{1/4}$$

$$= 4\ n^{1/4}$$

Simplification

If number of lists is 5 than we need 5 intervals

{0.2, 0.4, 0.6, 0.8 , 1}

n = 23, lg 23 = 4.52 => number of list can be 5

{|L1|, |L2|, |L3|, |L4|, |L5|} = {$23^{0.2}$,$23^{0.4}$,$23^{0.6}$,$23^{0.8}$,23} = {2, 4 , 7, 12, 23}

If number of lists is 4 than we need 4 intervals

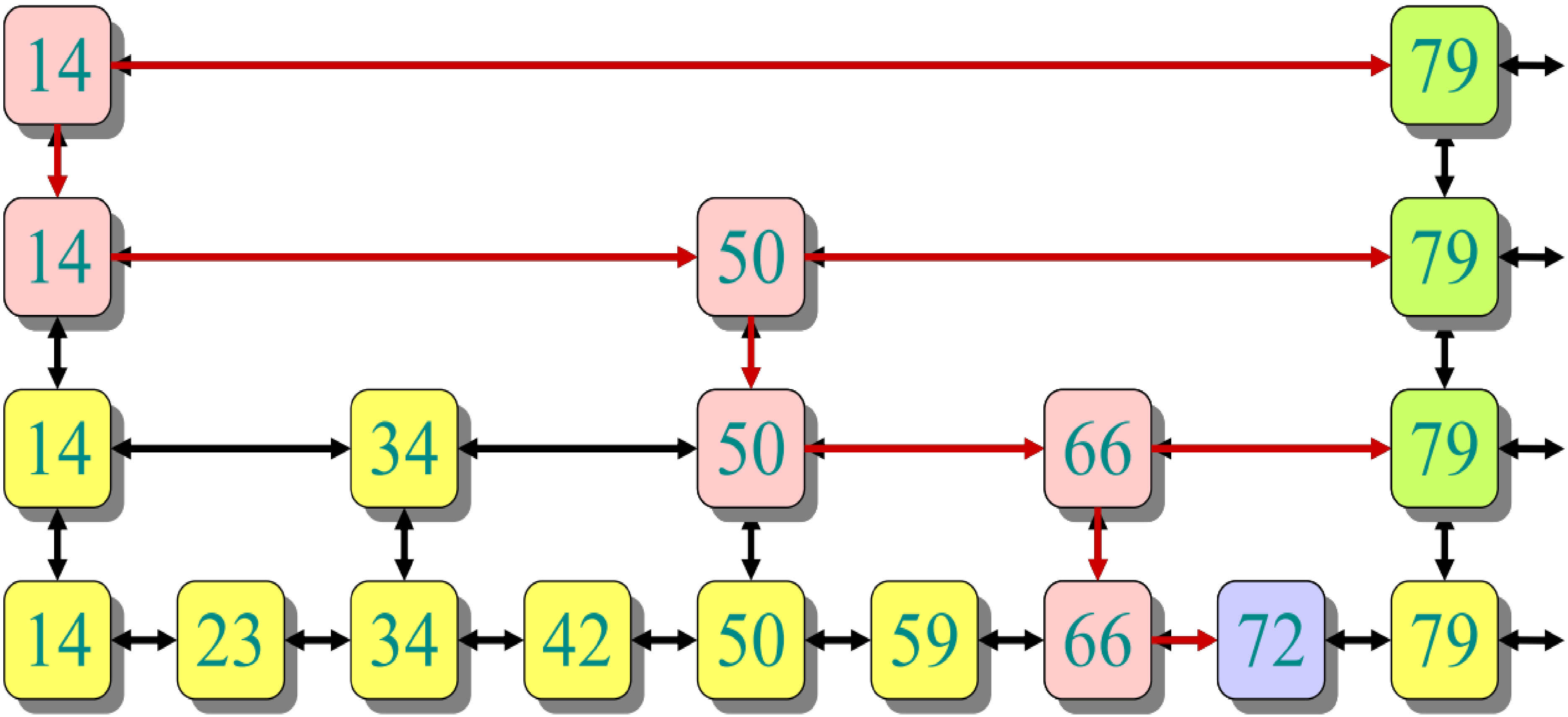
{0.25, 0.5, 0.75, 1}

If number of lists is 3 than we need 3 intervals

{0.33, 0.66, 1}

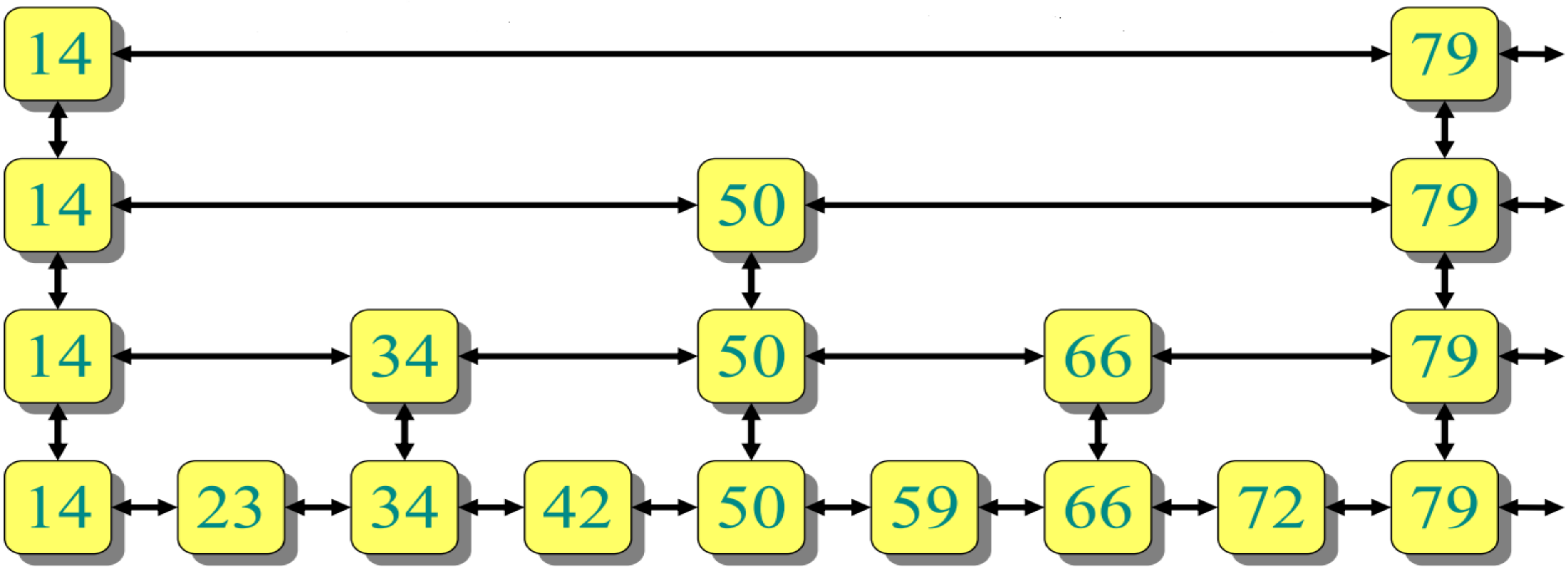
Searching in lg n linked lists

EXAMPLE: SEARCH(72)



Skip lists

Ideal skip list is this lg n linked list structure
Skip list data structure maintains roughly this structure subject to updates (insert/delete)



INSERT(x)

To insert an element xinto a skip list:

- SEARCH(x)to see where xfits in bottom list
- Always insert into bottom list

INVARIANT: Bottom list contains all elements

• Insert into some of the lists above…

QUESTION:To which other lists should we add x

INSERT(x)

QUESTION:To which other lists should we add x?

IDEA: Flip a (fair) coin; if HEADS, promote x to next level up and flip again

- Probability of promotion to next level = 1/2
- On average:

–1/2 of the elements promoted 0 levels

–1/4 of the elements promoted 1 level

–1/8 of the elements promoted 2 levels

–etc.

Example of skip list

EXERCISE:Try building a skip list from scratch by repeated insertion using a real coin

Skip lists

Skip list is the result of insertions (and deletions) from an initially empty structure

- INSERT(x)uses random coin flips to decide promotion level
- DELETE(x) removes x from all lists containing it