CS 261

Skip Lists
by
Tim Budd
Sinisa Todorovic
Complexity – Lists and Arrays

• Ordinary linked lists and arrays have:
  – fast addition O(1)
  – slow search O(n)

• Sorted Arrays have:
  – fast search O(log n)
  – slow insertion O(n)
Problem with Sorted List

- What’s the use?
- Add, contains, remove $\rightarrow O(n)$
- No better than an ordinary list
- Major problem: sequential access
Sorted Linked List

• We could use two sorted linked lists, with pointers between them
Two Sorted Linked Lists

- Constructed from the same elements
- Establish pointers between equal links
Motivation

- Regular Trains vs. Express Trains
Two Sorted Linked Lists

• List 2: stores all elements

• List 1: stores only a subset of elements
How to Search for an Element?

• We start from the 1st element of List 1
• Stay on the "express line" as long as you can
• Then, take the "local line"
How to Choose Elements for List 1?

- Goal: Maximize fast access to all elements
- List 1 picks uniformly a subset of elements
What is Complexity of Search?
What is Complexity of Search?

num. of elements in List 1 $\rightarrow |L_1| + \frac{|L_2|}{|L_1|} \leftarrow$ num. of elements in a segment of List 2

List 1

List 2
What is Complexity of Search?

\[ |L_1| + \frac{|L_2|}{|L_1|} = n \]

\[ \text{List 1} \]

List 2

<table>
<thead>
<tr>
<th>3</th>
<th>5</th>
<th>7</th>
<th>13</th>
<th>19</th>
<th>23</th>
<th>31</th>
<th>41</th>
</tr>
</thead>
</table>

fs

<table>
<thead>
<tr>
<th>3</th>
<th>5</th>
<th>7</th>
<th>13</th>
<th>19</th>
<th>23</th>
<th>31</th>
<th>41</th>
</tr>
</thead>
</table>

bs

\[ \text{List 1} \]

<table>
<thead>
<tr>
<th>3</th>
<th>5</th>
<th>7</th>
<th>13</th>
<th>19</th>
<th>23</th>
<th>31</th>
<th>41</th>
</tr>
</thead>
</table>

fs

<table>
<thead>
<tr>
<th>3</th>
<th>5</th>
<th>7</th>
<th>13</th>
<th>19</th>
<th>23</th>
<th>31</th>
<th>41</th>
</tr>
</thead>
</table>

bs
What is Complexity of Search?

minimize $x + \frac{n}{x}$
What is Complexity of Search?

minimum of the function 

$$\frac{d}{dx} \left( x + \frac{n}{x} \right) = 0$$
What is Complexity of Search?

\[
d \left( x + \frac{n}{x} \right) dx = 1 - \frac{n}{x^2} = 0 \quad \Rightarrow \quad x = \sqrt{n}
\]
What is Complexity of Search?

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

$\approx \sqrt{n}$
What is Complexity of Search?

$2\sqrt{n}$
What is Complexity of Search?

two sorted lists

$O(2\sqrt{n}) < O(n)$

one sorted list
What is Complexity for 3 Lists?
What is Complexity for 3 Lists?

$3^{\sqrt[3]{n}}$
What is Complexity for $k$ Lists?

for $k$ linked lists: $k \sqrt[3]{n}$
How Many Lists?

minimize \( k \frac{k}{\sqrt{n}} \)
How Many Lists?

\[
\begin{align*}
\text{minimize} \quad & k^n \frac{1}{k} \\
\text{subject to} \quad & \log k + \frac{1}{k} \log n
\end{align*}
\]
How Many Lists?

\[
\frac{d \left( \log k + \frac{1}{k} \log n \right)}{dk} = 0
\]

\[
\Rightarrow
\]

\[
\frac{1}{k} - \frac{1}{k^2} \log n = 0
\]

\[
k = \log n
\]
Complexity for $k=\log n$ Lists?

$$k \ n^{\frac{1}{k}} = (\log n) \ n^{\frac{1}{\log n}}$$
\[ n = 2^{\log n} \]

\[ n \frac{1}{\log n} = (2^{\log n}) \frac{1}{\log n} = 2 \]

\[ (\log n) \ n \frac{1}{\log n} = 2 \log n \]
Complexity for $\log n$ Lists

$O(\log n)$
Skip Lists – Have It All
Pugh 1989

• Fast addition $O(\log n)$
• Fast search $O(\log n)$
• Fast removal $O(\log n)$

• Disadvantage: - *Slightly* more complicated
Contains Skip List

1. Start at topmost sentinel
2. Loop as follows
   1. Slide right, get a link right before
   2. If next element is OK, return true
   3. If no down element, return false
   4. Move down
Complexity of Contains

• Makes zig-zag motion to bottom
• Proportional to height
• $O(\log n)$
Remove Skip List

1. Start at topmost sentinel
2. Loop as follows
   1. Slide right, get a link right before
   2. If next element is OK, remove it
   3. If no down element, reduce size
   4. Move down
Note about Remove

- Only decrement size at bottom level
- Makes zig-zag motion to bottom
- Proportional to height
- $O(\log n)$
How to construct a skip list when we do not know the number of elements in advance?
Add Skip List

- Add the element to the bottom list
  - must increment size
- Move up by flipping a coin, and add the element to the next higher list as long as heads
- At the top list, if the number of lists < log (size)
  - Flip a coin, if heads make a new top list
Add Example

Insert the following: 9 14 7 3 20

Coin toss: T H T H H T H T (move up if heads)
Add Example:
Insert : 9 14 7 20
Coin toss: T HT HHH T (H = move up)
Add Example:
Insert : 9  14  7  20
Coin toss:   T   HT   HHH   T
(H=move up)
Complexity of Add

• Proportional to height, not to the number of nodes in the list
• $O(\log n)$
Useful Properties of Skip Lists

- Insertion is $O(\log n)$
- Keep elements in order
- These can be combined for sorting
Skip List Sorting Algorithm

**Problem:** Sort an array A

Step 1. Copy elements from A into a skip list

Step 2. Copy elements from the skip list to A
Skip List Sorting Algorithm

Complexity:

Step 1. Copy elements from A into a skip list
O(??)

Step 2. Copy elements from the skip list to A
O(??)