CSE 100: SKIP LISTS
Announcements

• Please fill out the end of year survey and CAPEs
  https://www.surveymonkey.com/s/QFZPMT2
• Please remove print statements to standard output when submitting your PA4 assignment. It slows down the grading process.
Skip Lists: Motivation

- Which data structure is faster in the worst case for finding elements, assuming the elements are sorted?
  
  A. An array
  B. A linked list
  C. They can both be made equally fast
Skip Lists: Motivation

- Which data structure is faster in the worst case for **inserting** elements, assuming the elements are sorted?
  A. An array \( O(n) \)
  B. A linked list \( O(n) \) → better constants
  C. They can both be equally fast (slow) \( O \)
Toward Skip lists

• Adding forward pointers in a list can vastly reduce search time, from $O(N)$ to $O(\log N)$ in the worst case.

• However, what is the main problem with this approach?
  A. Pointers have to stay appropriately spaced to maintain the worst case bound
  B. The pointers take the space requirements for the list from $O(N)$ to $O(N^2)$
  C. Pointers only point forward in the list, so you can get better running time bounds by including backward pointers as well.
Toward Skip lists

• Adding forward pointers in a list can vastly reduce search time, from $O(N)$ to $O(\log N)$ in the worst case.

• However, what is the main problem with this approach?

A. Pointers have to stay appropriately spaced to maintain the worst case bound

Deterministically adjusting pointer location is costly. Skip lists fix this problem by using randomness to randomly determine where pointers go
Is the following a correct implementation of a set using a SkipList? (As described in your reading?)

A. Yes
B. No, because the values in the nodes are not in sorted order
C. No, because the nodes at different levels are not properly spaced
D. No, because there are no level 4 nodes
E. More than one of these
Our definition of SkipLists: Always sorted

The book is a bit vague about whether SkipLists have to be sorted.

When we talk about a SkipList in this class, the elements in the SkipList must be sorted. The structure above is not a valid SkipList.
Creating a Skip List

Draw the Skip List that results from inserting the following elements with the following levels

Elements: 42, 3, 12, 20, 9, 5
Levels: 3, 1, 1, 2, 1, 2
Creating a Skip List

Draw the Skip List that results from inserting the following elements with the following levels

Elements: 42, 3, 12, 20, 9, 5
Levels: 3, 1, 1, 2, 1, 2
**SkipList find, pseudocode (slight difference from the reading)**

- To find key $k$:
  1. Let $x$ be list header (root). Let $i$ be the highest non-null index pointer in $x$
  2. While pointer $x[i]$ is not null and points to a key smaller than $k$, let $x = x[i]$ (follow the pointer)
  3. If the pointer $x[i]$ points to a key equal to $k$ return true
  4. If the pointer $x[i]$ points to a key larger than $k$, decrement $i$ (drop down a level in $x$)
  5. If $i < 0$ return false. Else go to 2.

Assumes index pointers are 1 less than level
Find in a Skip List

Which of the following pointers are checked and/or followed in a find for the element 35?

A. Red only
B. Red and blue only
C. Red, blue and purple only
D. Red, blue, purple and black
E. Some other combination of pointers
Find in a Skip List

Highlight the pointers that are checked/followed in a find for the element 12. Annotate the order in which they are checked.
Why Skip Lists?

• Why use a skip list?
  • Simple implementation
  • Simple in-order traversal
  • Fast (comparable to Balanced Binary Tree in the average case)
  • Amenable to concurrent modification (changes are quite local)

Insert 22 with level 2: highlight everything that needs to change in the list. Try to figure out how you would keep track of all this information.
**SkipList** insert: slow motion

- prev[i] should point to the node whose level i pointer needs to point to the new node.
- curr[i] should point to the same node that new node's level i pointer must point to after its inserted.

```
root

prev

lvl_ind:  

curr

3 5 6 21 25

```
SkipList insert: slow motion
Because prev[3] is null start at root again

lvl_ind:

2
SkipList insert: slow motion
SkipList insert: slow motion

prev

root

prev

lvl: 1

26 3 5 6 21 25 26
SkipList insert: slow motion

root

prev

lvl: 0

curr

insertion process:
- Inserting 21:
  - Move to level 0
  - Move to level 1
  - Move to level 2
  - Move to level 3
- Inserting 25:
  - Move to level 0
  - Move to level 1
  - Move to level 2
- Inserting 26:
  - Move to level 0
  - Move to level 1
  - Move to level 2
  - Move to level 3

The SkipList structure is shown with levels and pointers.
**SkipList insert: slow motion**

![Diagram of SkipList insertion](image)
SkipList insert: slow motion
SkipList insert: slow motion
SkipList insert: slow motion
SkipList insert: slow motion
SkipList insert: demo

- http://iamwww.unibe.ch/~wenger/DA/SkipList/