SKIP-LIST

(DATA STRUCTURE)

To

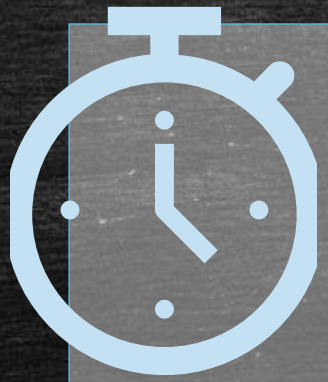
Smt. A. Sireesha

Professor at GITAM University
Visakhapatnam, Andhra Pradesh.

What is Skip lists ?

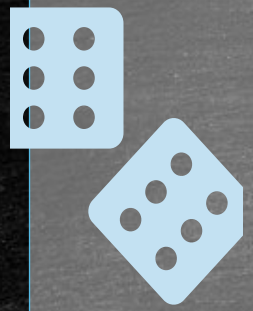
- Introduced by William Pugh 1989
- Based on Linked lists concepts.
- Few lines of code.
- Map Abstract Data Type

Feature's



Efficient

- Time Compexity
- Linked List
- Index

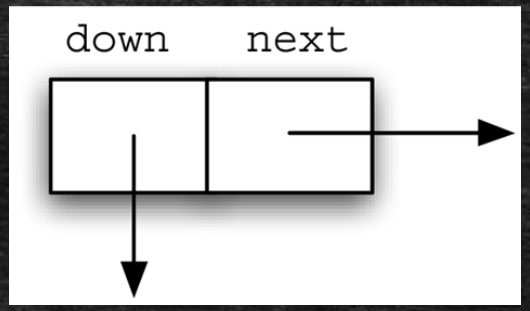


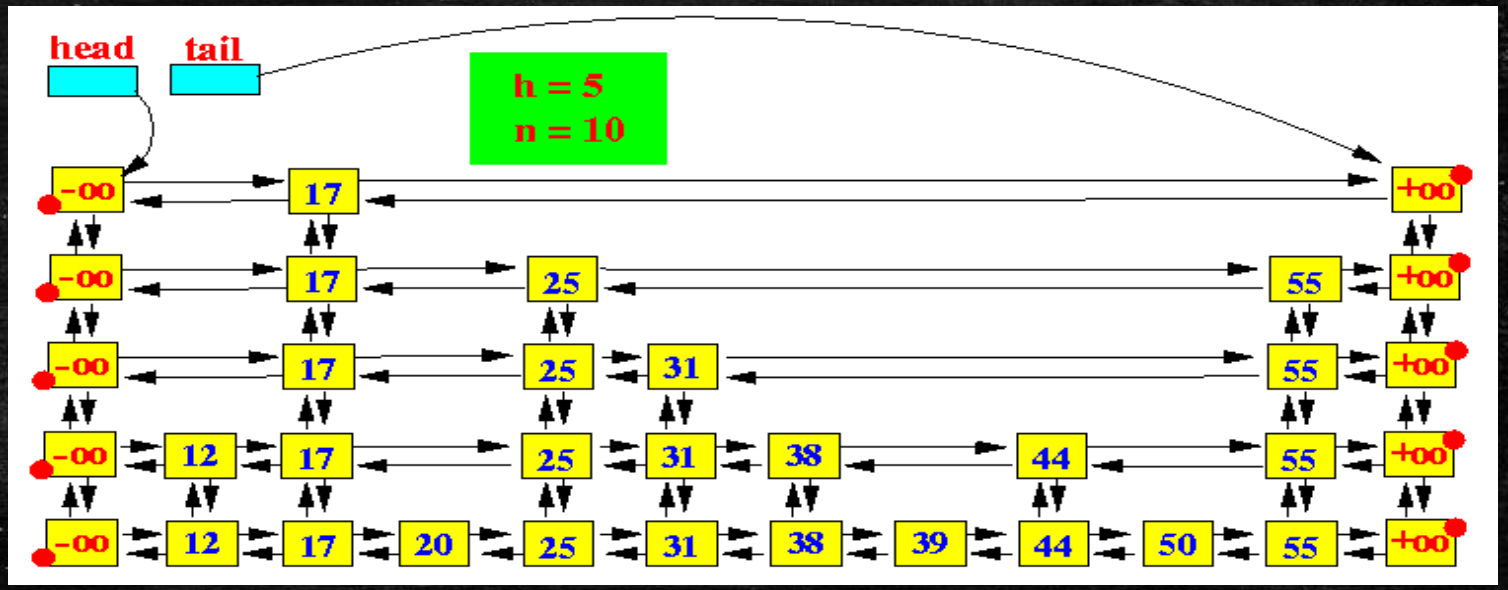
Probabilisitc

- Skip list uses probability to build subsequent layers of linked lists upon an original linked list.

Implementation

- Down
- Next
- Head
- Tail
- Sentinel





Implementation

- Define a skip-list node
- Height where the node appears by a list of
- Sentinel skip node (height =0)
- list of pointers to the next nodes.

```
class SkipNode:
    def __init__(self, height = 0,
elem = None):
        self.elem = elem
        self.next = [None]*height
class SkipList:
    def __init__(self):
        self.head = SkipNode()
```

Insertion

The insertion consists in deciding the height of the new node, using randomHeight() and for each of the levels up to this height, insert this new node after the node specified in update.

```
def insert(self, elem):
    node = SkipNode(self.randomHeight(), elem)
    while len(self.head.next) < len(node.next):
        self.head.next.append(None)
    update = self.updateList(elem)
    if self.find(elem, update) == None:
        for i in range(len(node.next)):
            node.next[i] = update[i].next[i]
            update[i].next[i] = node
```

Deletion

The deletion is pretty much like the insertion, but now we delete the node found using find() from all levels in which it appears

```
def insert(self, elem):
    node = SkipNode(self.randomHeight(), elem)
    while len(self.head.next) < len(node.next):
        self.head.next.append(None)
    update = self.updateList(elem)
    if self.find(elem, update) == None:
        for i in range(len(node.next)):
            node.next[i] = update[i].next[i]
            update[i].next[i] = node
```

Search

If search element is E and Y is an elements appear during searching

- E = Y : We return
- E > Y : We "scan forward"
- E < Y : We "drop down"

Search

Topmost level of the header to list in this level until we find node with the largest element that is smaller than the element we are searching.

Repeat this process but this time start from near to largest element from search element

```
def updateList(self, elem):
    update = [None]*len(self.head.next)
    x = self.head
    for i in
reversed(range(len(self.head.next))):
        while x.next[i] != None and \
            x.next[i].elem < elem:
            x = x.next[i]
        update[i] = x
    return update
```

Comparison

- Can search an element in logarithmic time
- About 10000 insertions in each of these structures with a random sequence, an increasing sequence and a decreasing sequence. We measure the CPU time in seconds

	Skip-list	Red-Black tree	Group B
Random	0.55	0.57	29.25
Increasing	0.36	0.55	78.50
Decreasing	0.38	0.56	0.04

Thank You

Mr. Niteen Balpande
Master of Technology (CSE)
GITAM Deemed to be University
,Visakhapatnam, Andhra Pradesh