Skip List

A skip list is a probabilistic data structure where elements are kept sorted by key.

It allows quick search, insertion & deletions of elements with simple algorithms.

It is a basically a linked list with additional pointers such that intermediate node can be skipped.

A data structure that maintain different levels of linked list.

Search : $O(\log n)$
Insert : $O(\log n)$

Note:→ Unordered Array
↳ Searching and removing takes $O(n)$ time.
↳ Inserting takes $O(1)$ time

Ordered Array
↳ Searching takes $O(\log n)$ time
↳ Inserting and removing takes $O(n)$ time.

Balanced BST
↳ Guaranteed $O(\log n)$ operations.
↳ But complex to implement and keep balanced.
↳ may require additional memory

Date:

So, we want a
- compact and efficient search structure.
- which is easy to implement.
- using a little memory as possible.

What are Skip lists?

- A simple, efficient, dynamic, randomized search structure.
- Invented by Bill Pugh in 1989.
- If a skip list contains n items:
  - The expected time for searching, insertion and deletion is $O(\log n)$
  - The expected space usage is $O(n)$.

<u>Definition:</u>

A skip list for a set S of distinct items is a series of lists $S_0, S_1, \ldots S_h$ such that:

- Each list S_i contains the special items $+\infty$ and $-\infty$.
- S_0 contains the items of S in non-decreasing order.
- Each list is a subsequence of the previous one

$$S_0 \supseteq S_1 \supseteq S_2 \supseteq S_3 \supseteq \ldots \supseteq S_h$$

- List S_h contains only the two special items.

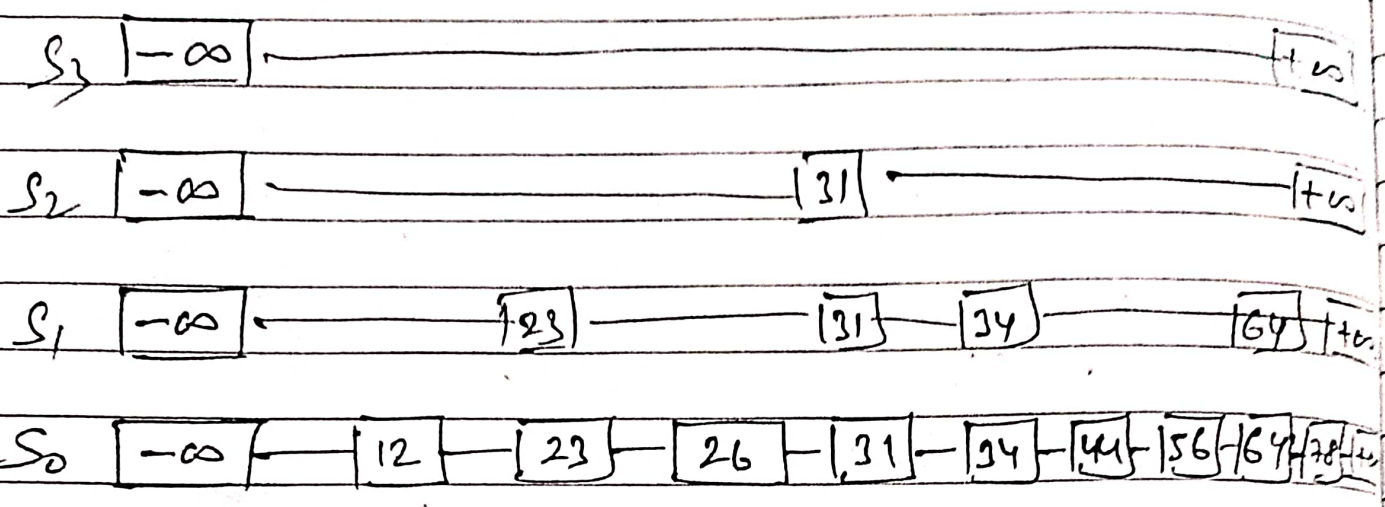


fig. Skip list

two types ① Perfect skip list & Deterministic skip list
② Randomized skip list

Every alternate element of a level is promoted to upper level. i.e S_1 will contain every alternate element of S_0.
S_2 will contain every alternate element of S_1 and so on.

Maintaining this order is quite difficult in Deterministic skip list.
So, we move on to Randomized skip list in which promotion of a node from one level to other is dependent on coin toss.

How do we implement this data structure?

Using Quad nodes

A quad node stores:

(1) Item value
(2) link to the previous node
(3) link to the next node
(4) link to the node above.
(5) link to the node below.

We also define two special nodes PLUS_INF and MINUS_INF.

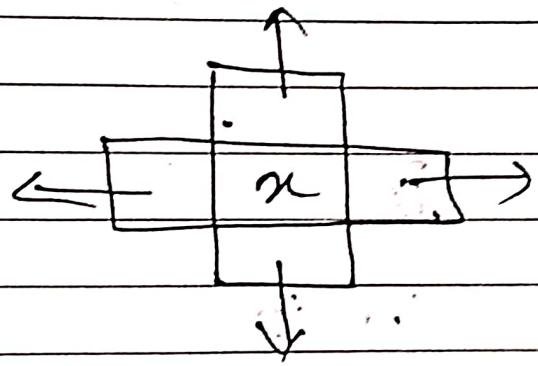


fig. Quad-node example.

How do we search for an item x in a skip list?

✓ (a) We start at the first position of the top most list.

✓ At the current position P, we compare x with y = p.next.Key

- If x==y, we return p.next.element
- If x>y, we move to the ~~xx~~ next item in our current list.
- If x<y, we drop down to the list al-

Search 78

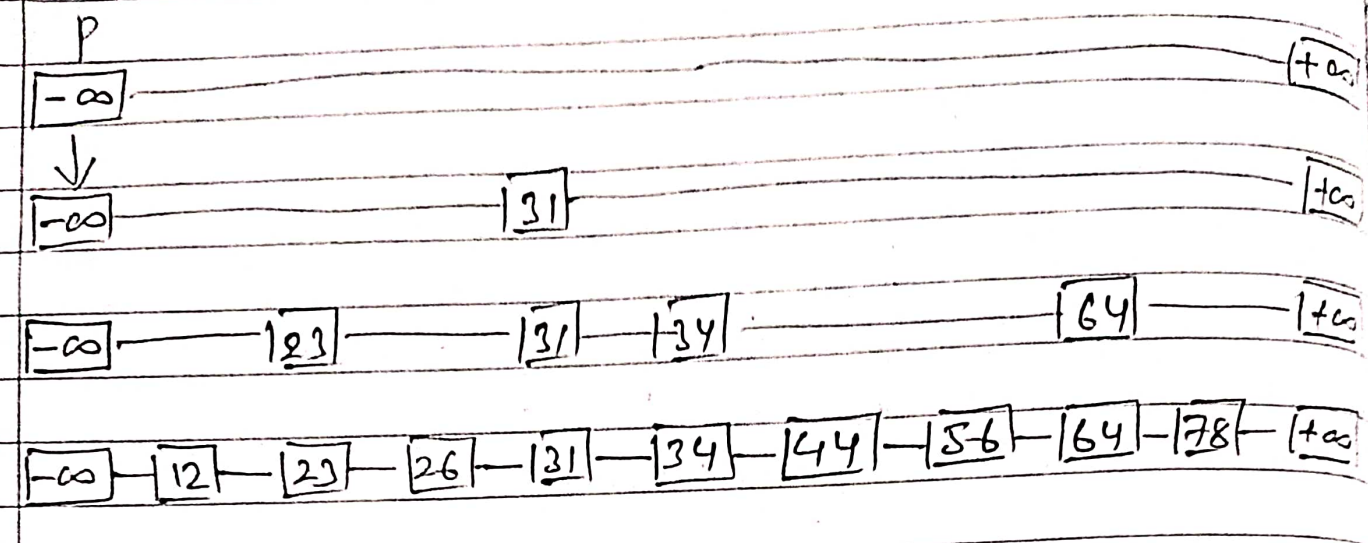


fig. $x < p.next.key$: "drop down"

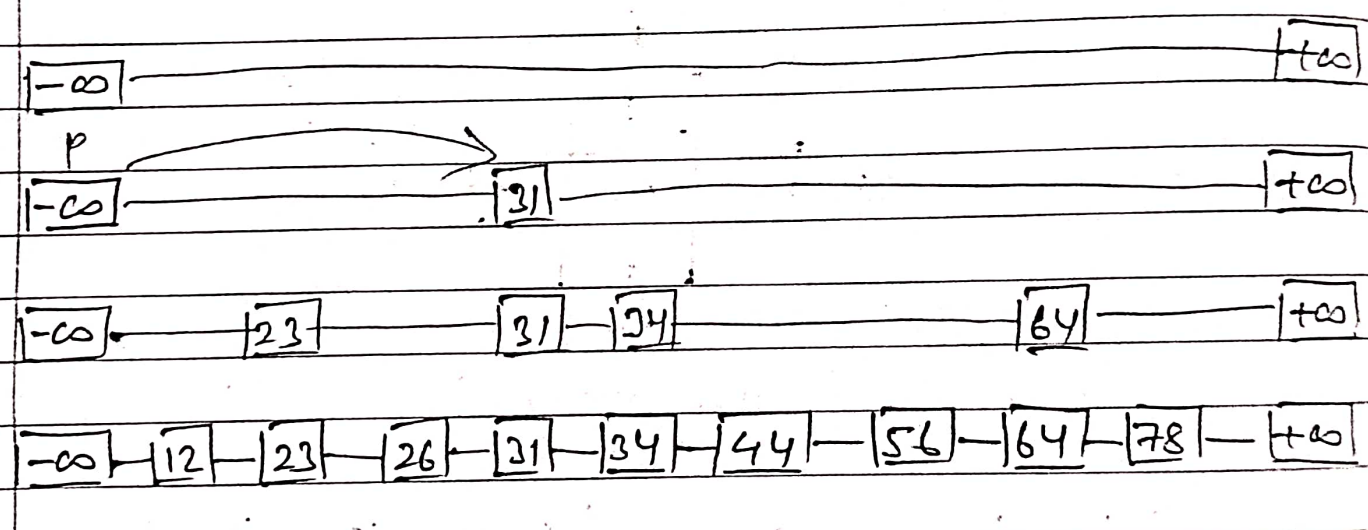
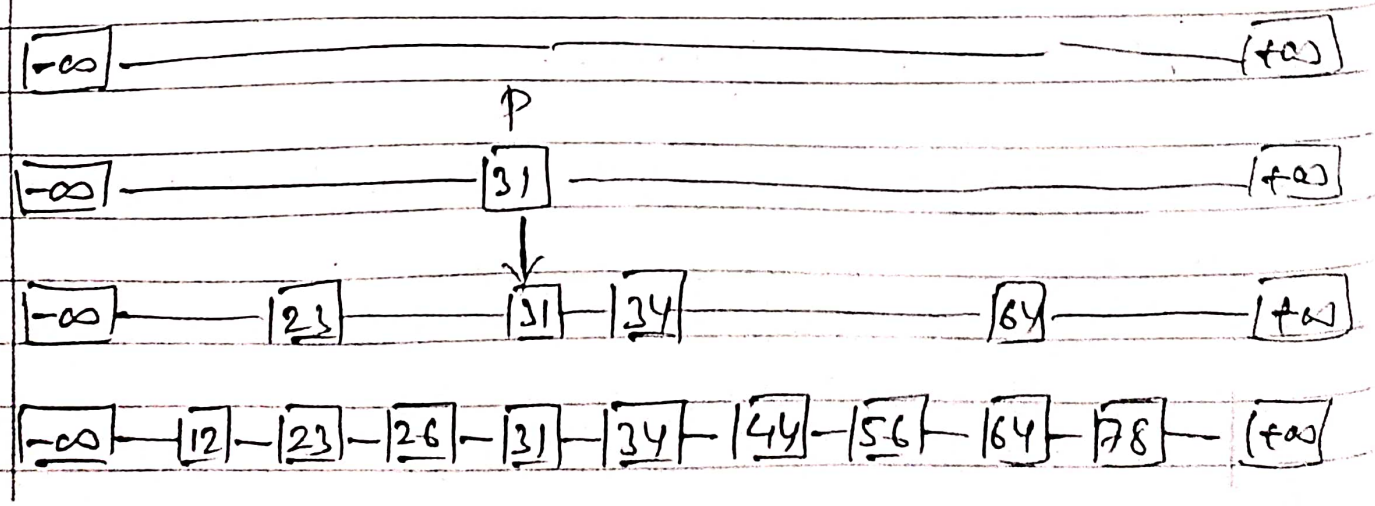


fig. $x > p.next.key$: "scan forward"



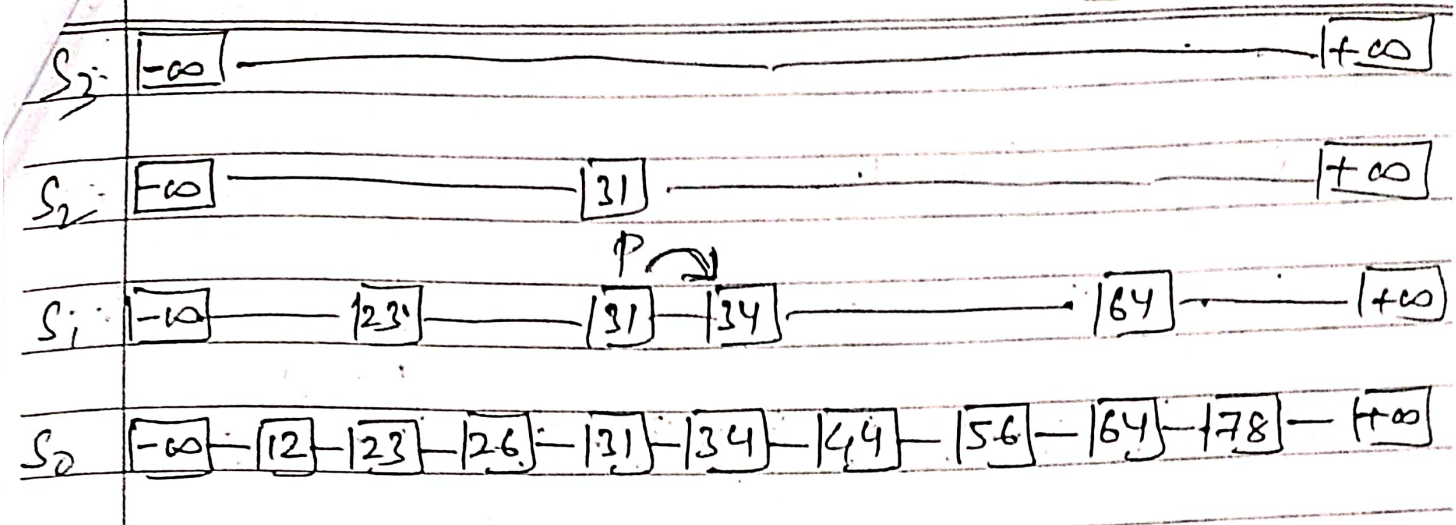


fig. $x > p.next.key$: "scan forward"

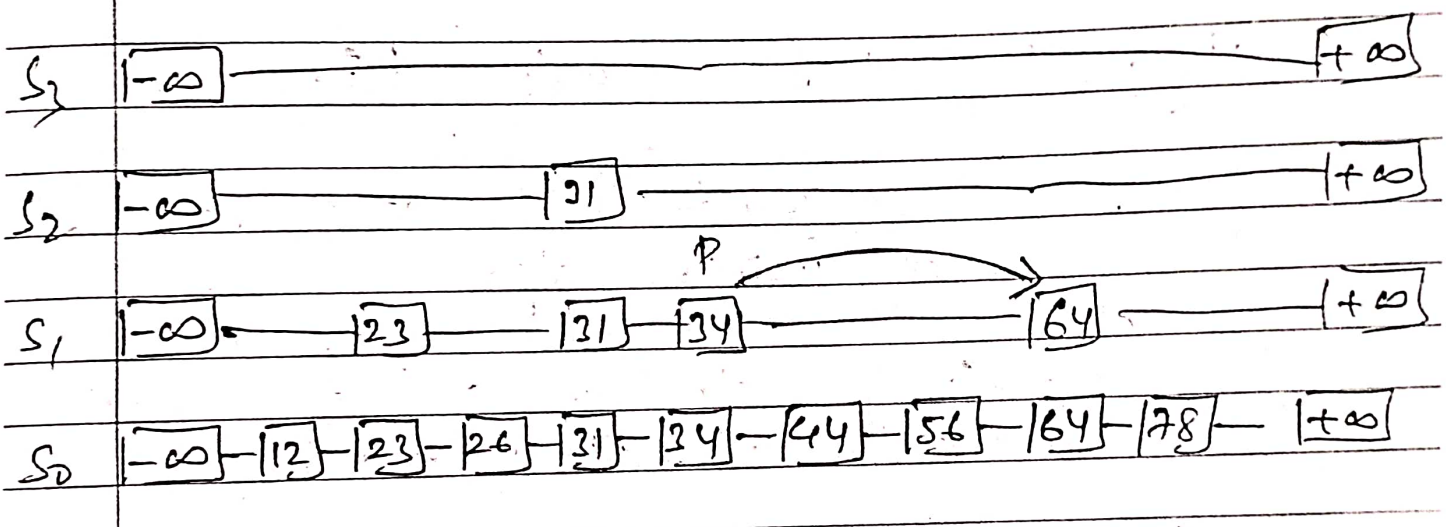


fig. $x > p.next.key$: "scan forward"

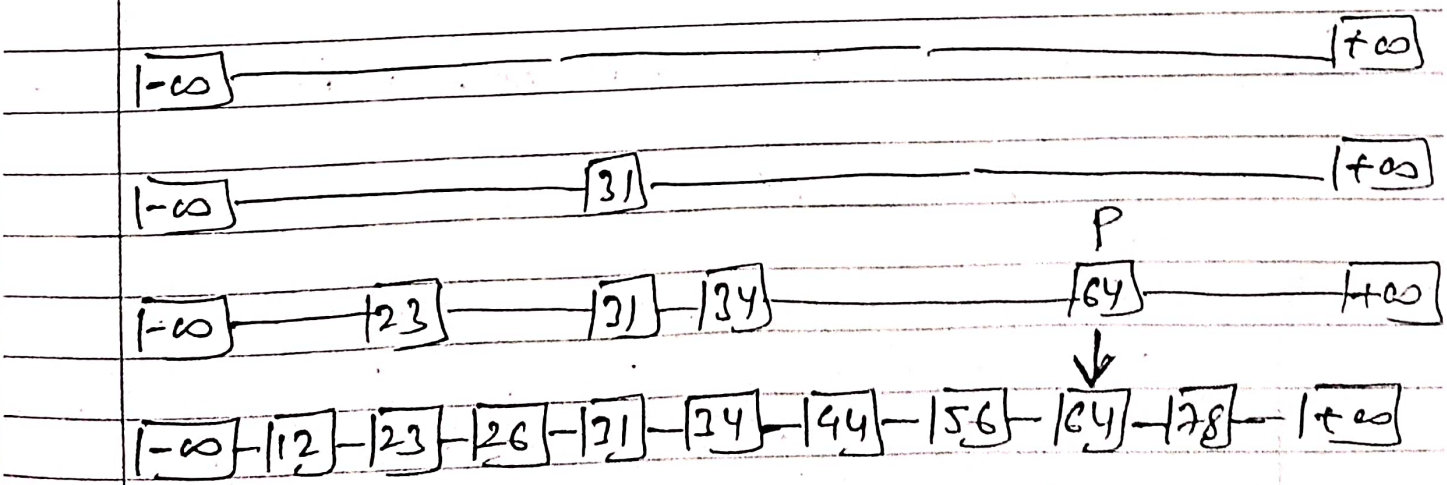


fig. $x < p.next.key$: "drop down"

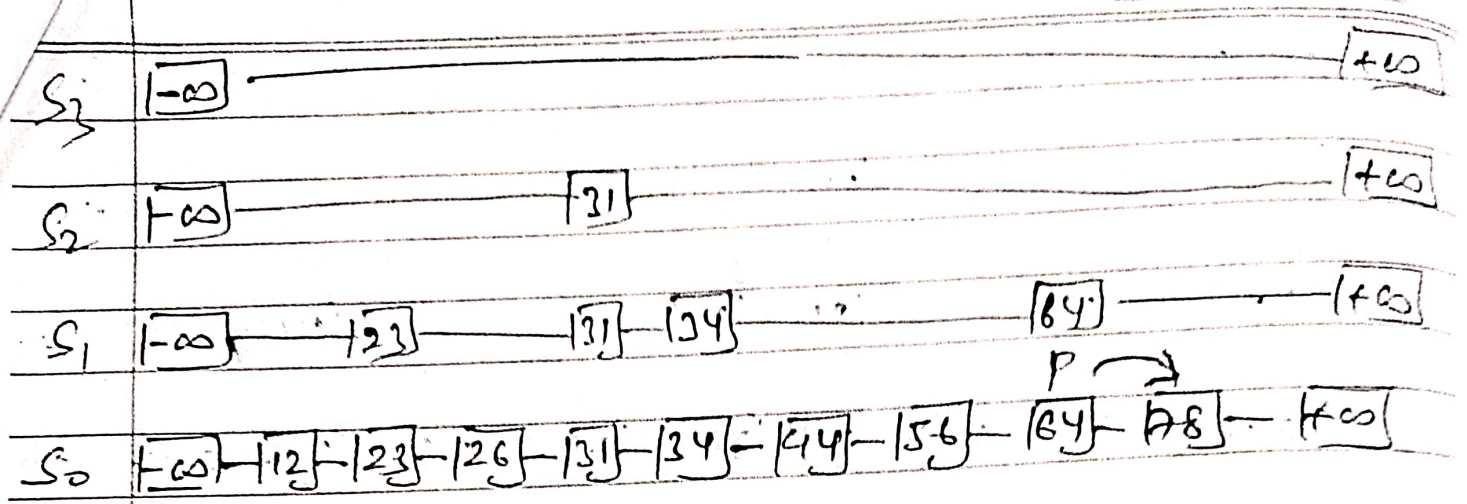


fig. $x == y$: we return p.next.element

Insertion Into Skip List

Skip list is a randomized structure because of its insertion algorithm.

The randomization algorithm and its analysis works under the assumptions that :

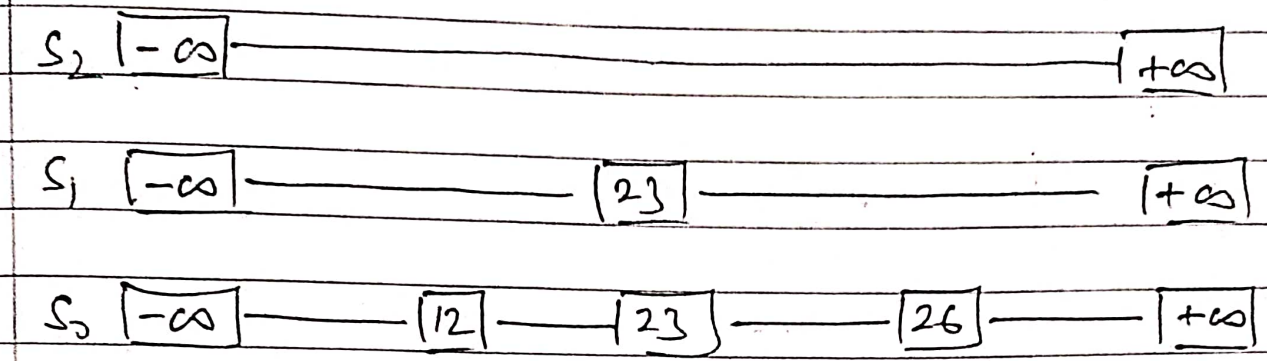
✓ The coins are unbiased.
✓ The coin tosses are independent.

How do we insert a new node x into a skip list?

(1) We start with list S_0.
(2) We toss a coin.
(2.1) If it shows HEADS, we add x in this list (before that item which is just larger than x). We then move on to the next list (S_1 in the case of S_0), toss the coin again and repeat this process.

list and stop

NOTE:→ The maximum number of lists can increase if consecutive heads exceed the height of the skip list.



Insert 15 in this skip list.

We will start from S_0 and we toss a coin. Suppose it shows HEADS 3 times consecutively.

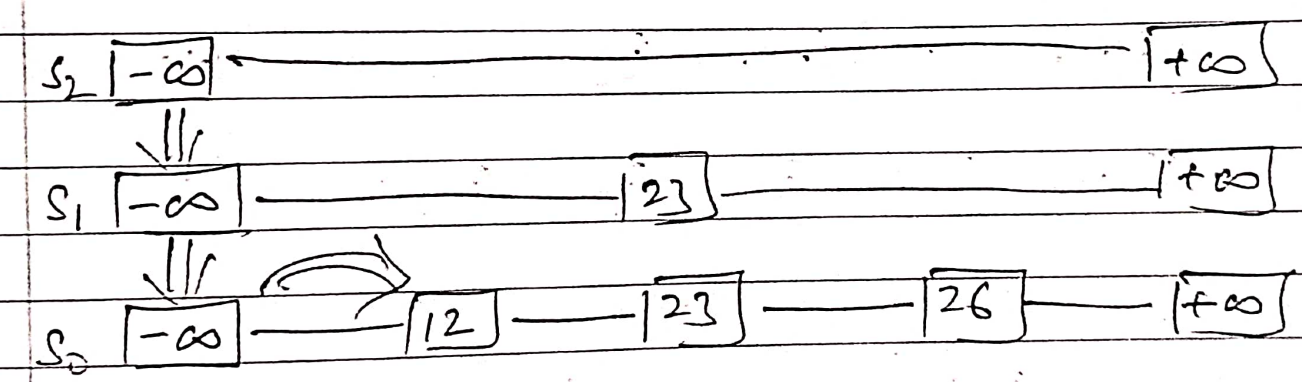
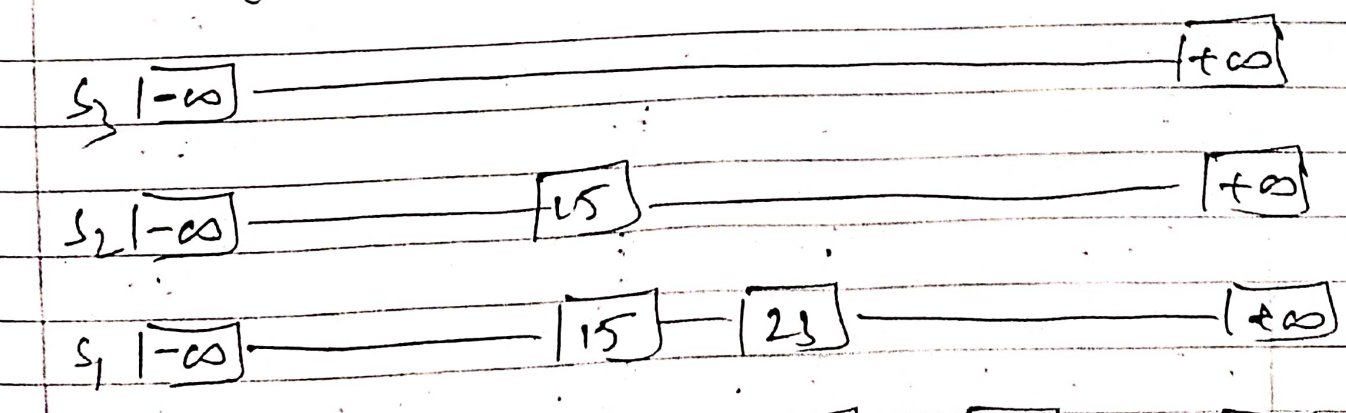


fig. to locate where 15 can be added.



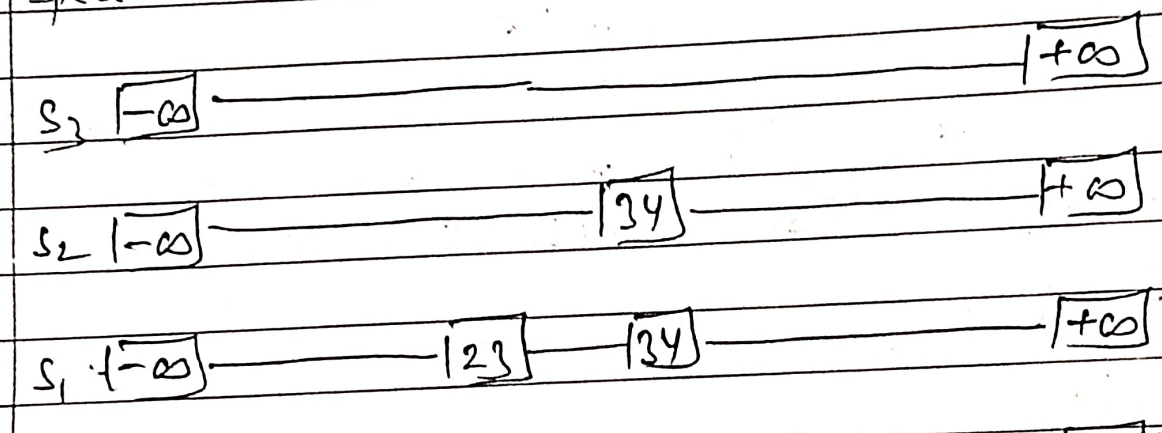
Deletion

How do we delete an item x from a skip list?

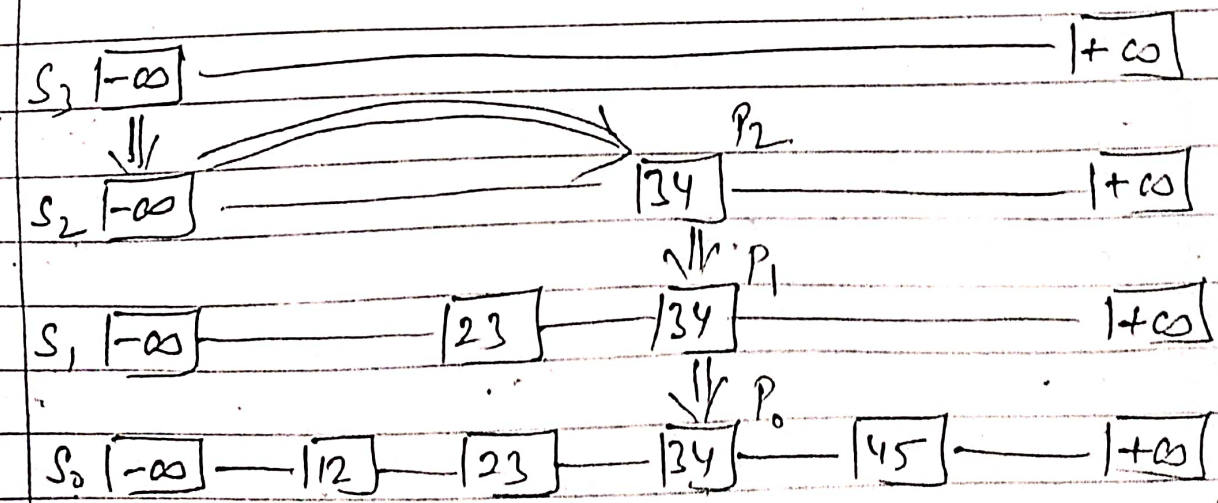
We search for x in the skip list and find the positions of x in all the lists.

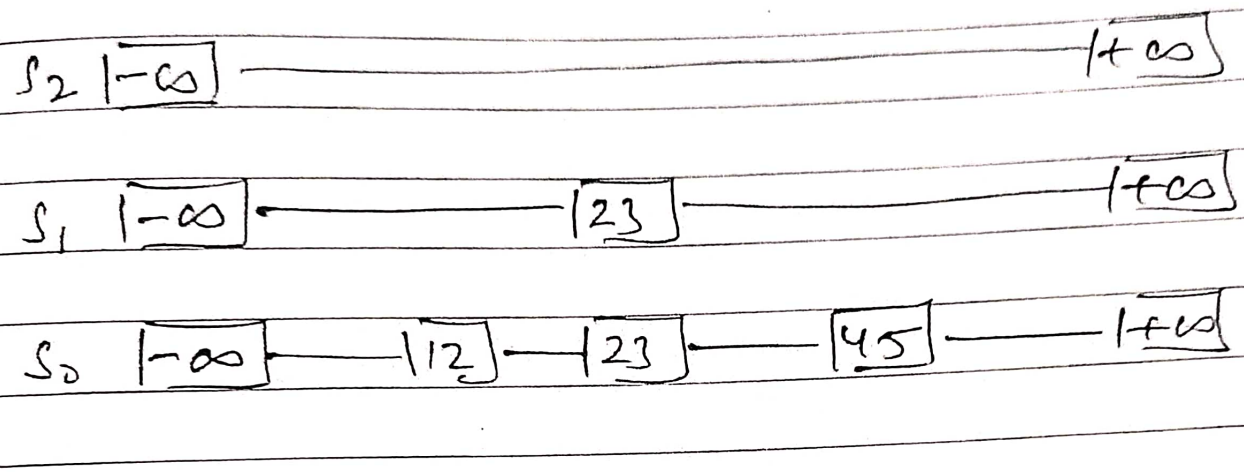
We remove those positions from the corresponding lists.

We remove all but one list which has only 2 special items.



We search for 34 in the skip list and find the positions $P_0, P_1, \ldots, P_2$ of the items with key 34.



Date:

Now delete these quad nodes.