Design and Analysis of Algorithm

Advanced Data Structure
(Skip List and Tries)

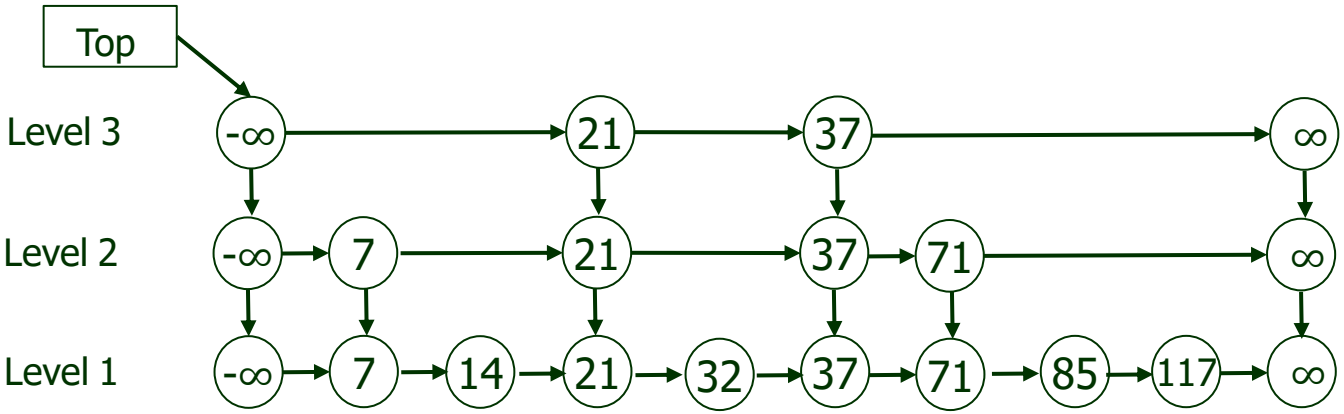
Lecture - 44

Overview

- This section present two advance data structures known as :
  - skip list and
  - Trie.

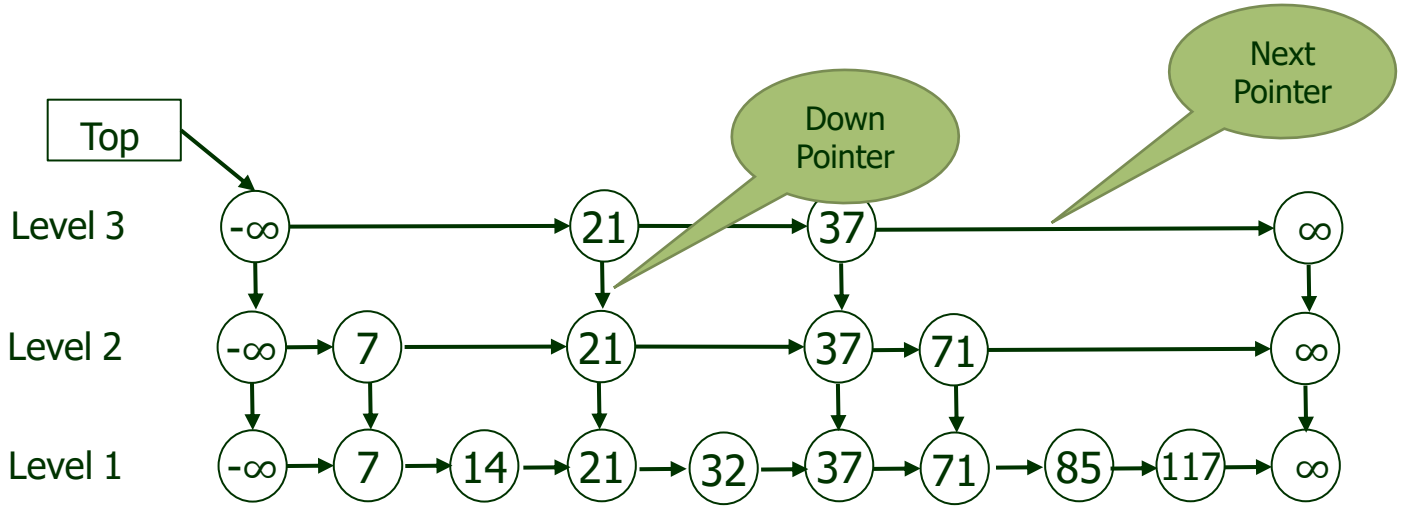
Skip List

- Skip list is a data structure used for maintaining a set of keys in sorted order.
- Rules of Skip List
  - It consists of several levels.
  - In skip list all keys are appear in level 1.
  - Each level of the skip list is a sorted list.
  - In skip list if a key x appears in level i, then it also appears in all levels below i.



Skip List

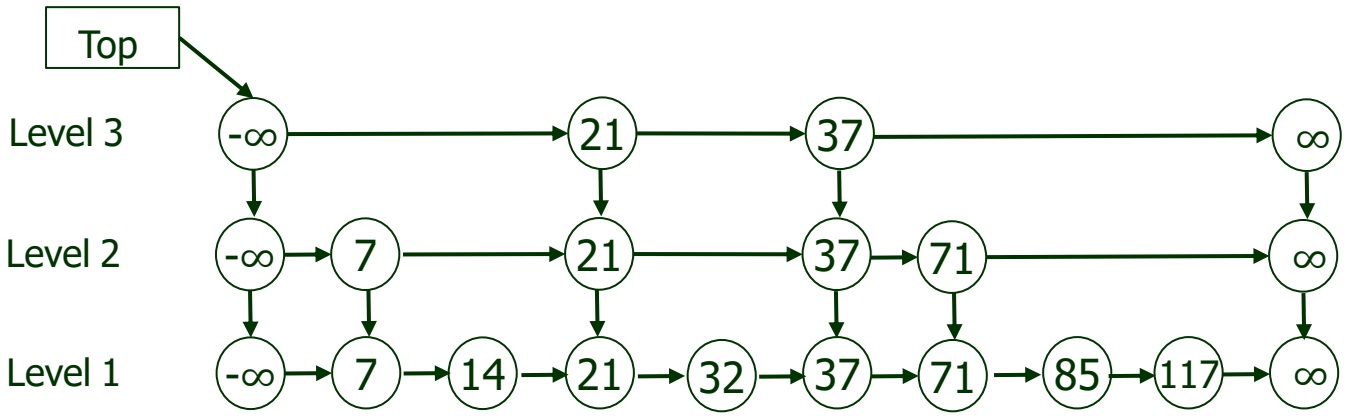
- More Rules
  - An element in level i points (via down pointer) to the element with the same key in the level below.
  - In each level the keys -∞ and ∞ appear.
  - Top points to the smallest element in the highest level.



Skip List

- Finding an element with key x

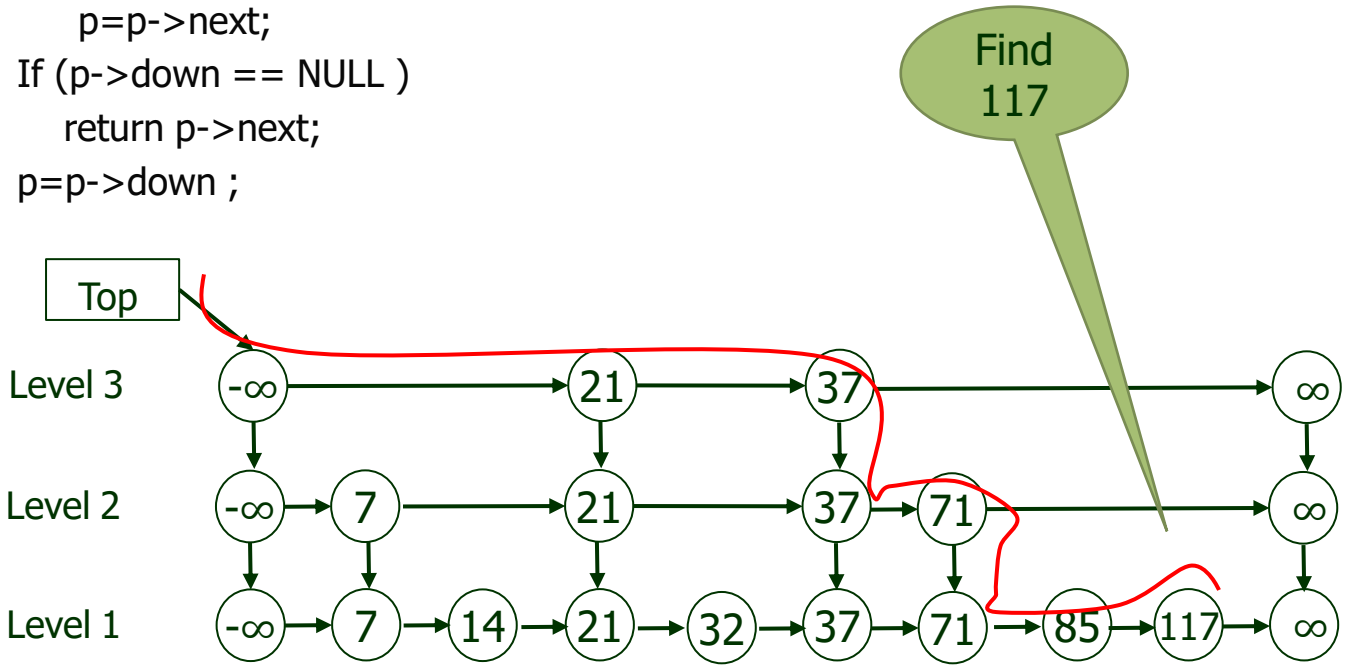
```
p=top
While(1){
while (p->next->key < x )
  p=p->next;
If (p->down == NULL )
  return p->next;
p=p->down ;
}
```



Skip List

- Finding an element with key x

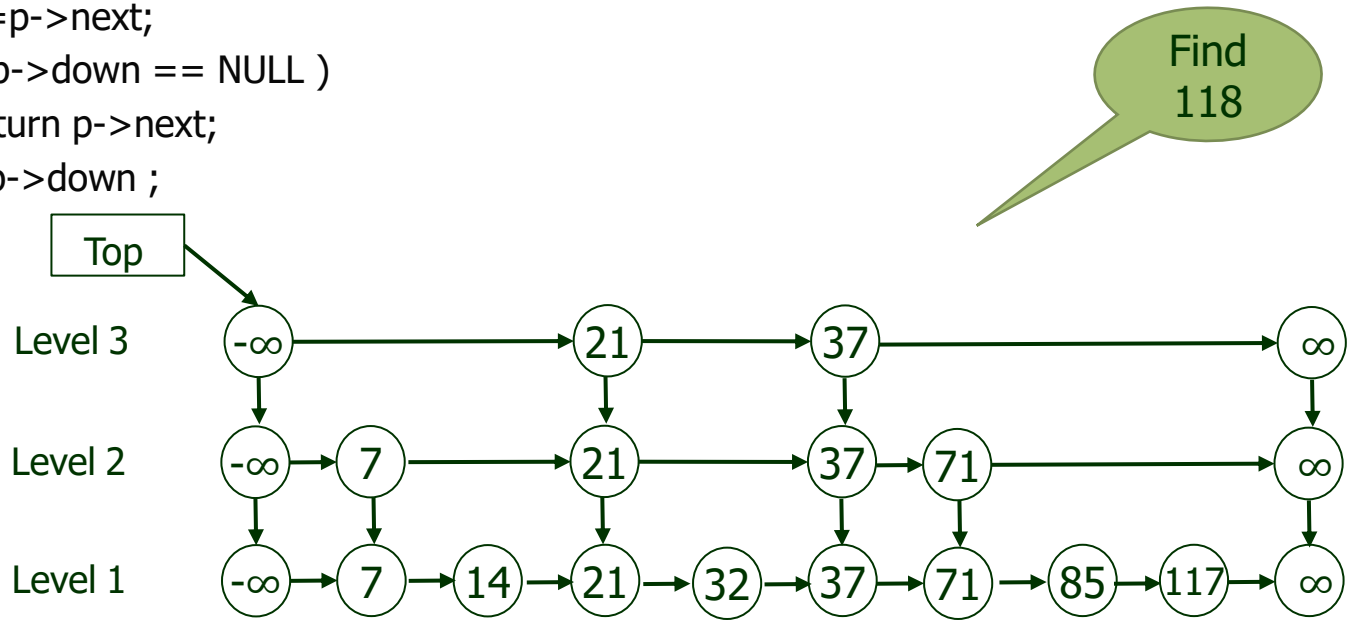
```
p=top
While(1){
    while (p->next->key < x )
        p=p->next;
    If (p->down == NULL )
        return p->next;
    p=p->down ;
}
```



Skip List

- Finding an element with key x

```
p=top
While(1){
while (p->next->key < x )
  p=p->next;
If (p->down == NULL )
  return p->next;
p=p->down ;
}
```

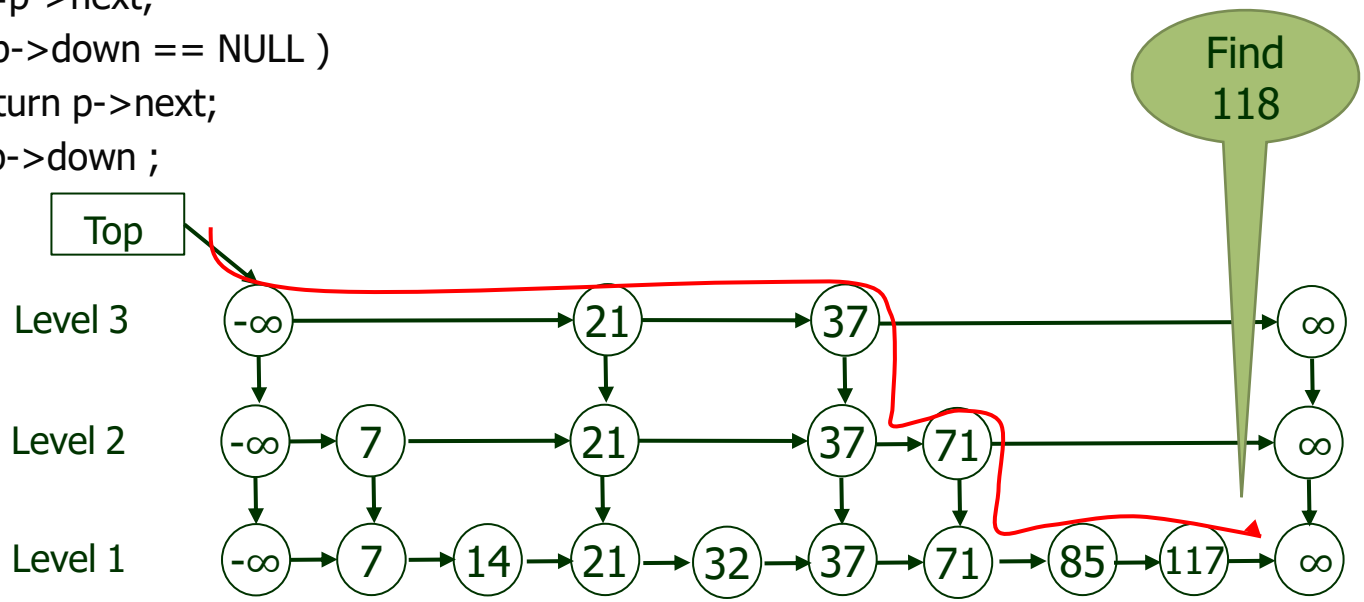


(Note: Observe that we return x, if exists, or succ(x) if x is not in the SkipList)

Skip List

- Finding an element with key x

```
p=top
While(1){
while (p->next->key < x )
  p=p->next;
If (p->down == NULL )
  return p->next;
p=p->down ;
}
```



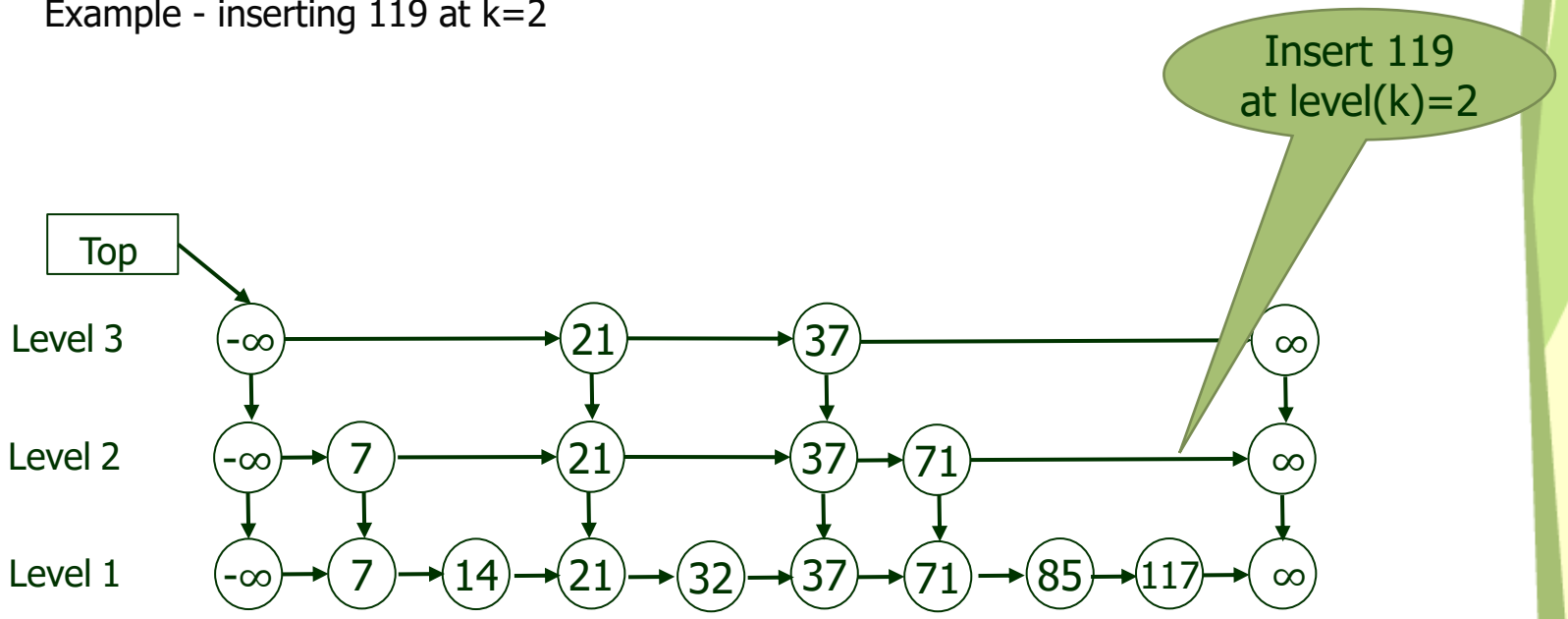
(Note: Observe that we return x, if exists, or succ(x) if x is not in the Skip List)

Skip List

- Inserting new element X

  Do find(x), and insert x to the appropriate places in the k^{th} level

  Example - inserting 119 at k=2

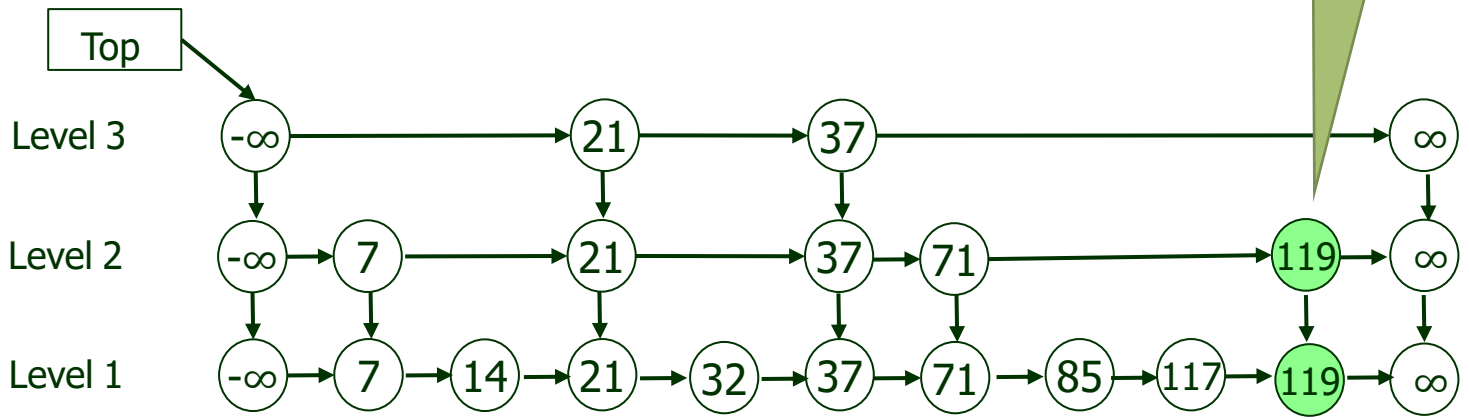


Skip List

- Inserting new element X

  Do find(x), and insert x to the appropriate places in the k^{th} level

  Example - inserting 119 at k=2

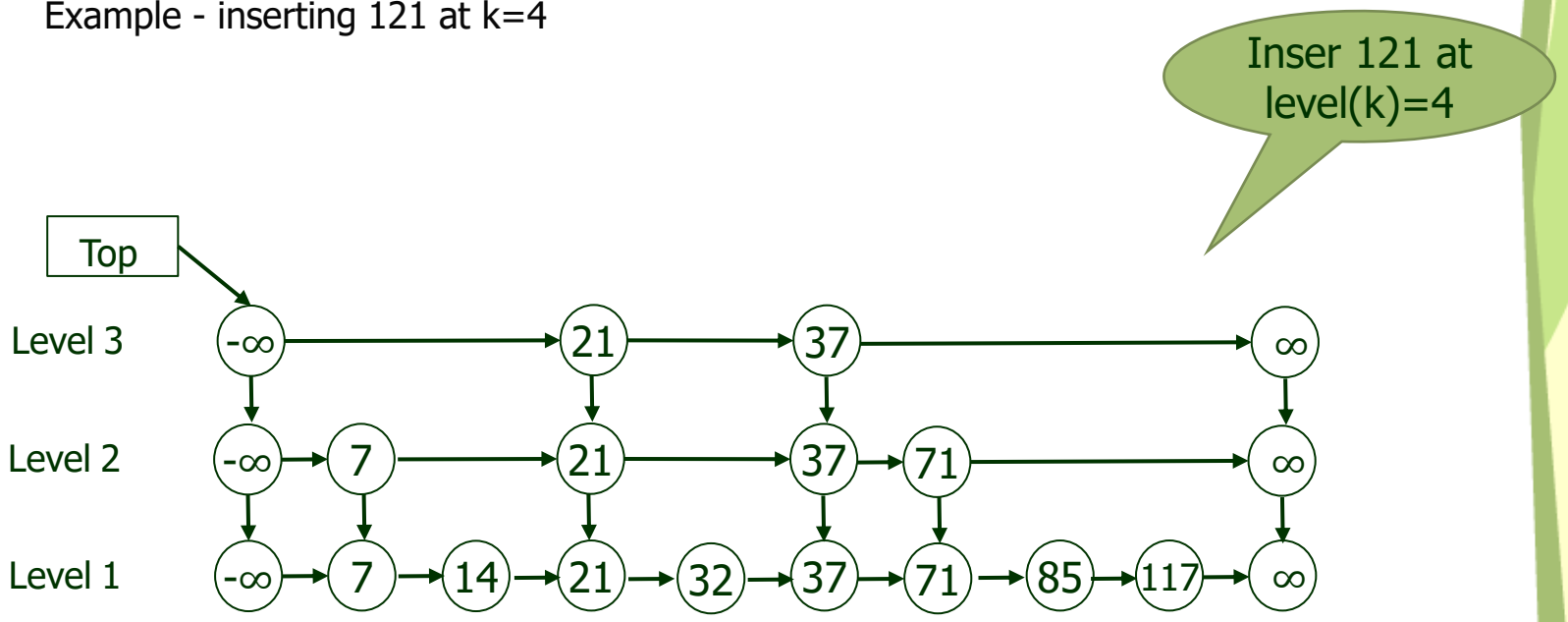


Skip List

- Inserting new element X

  Do find(x), and insert x to the appropriate places in the k^{th} level

  Example - inserting 121 at k=4

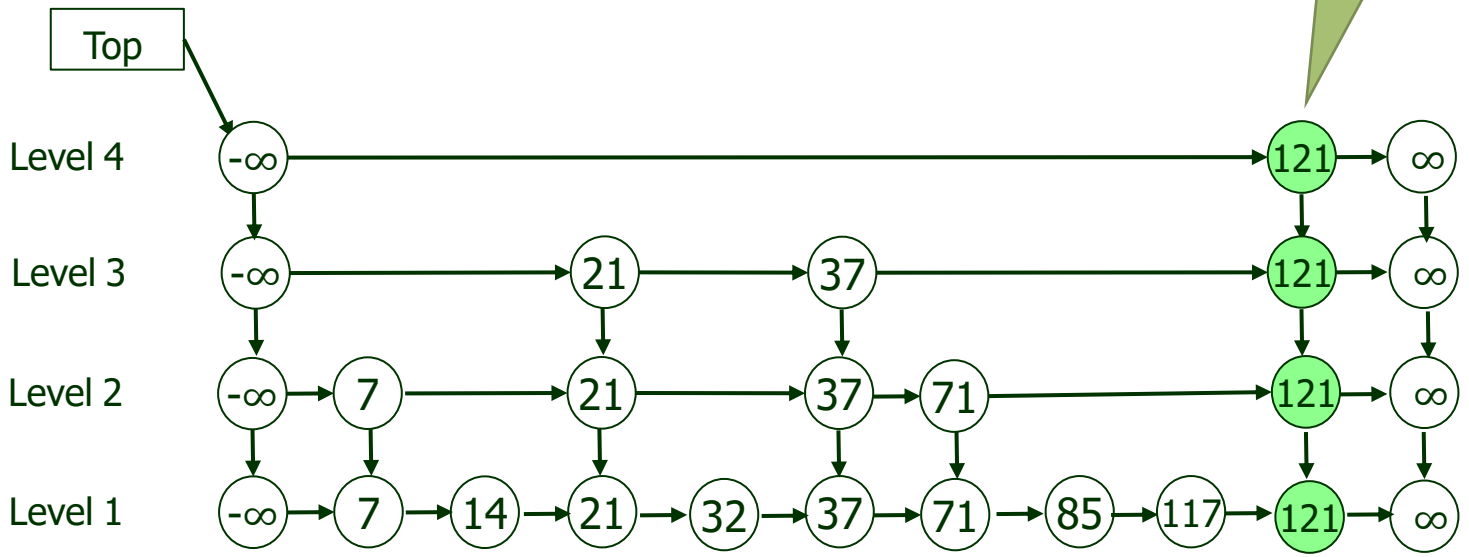


Skip List

- Inserting new element X

  Do find(x), and insert x to the appropriate places in the k^{th} level

  Example - inserting 121 at k=4

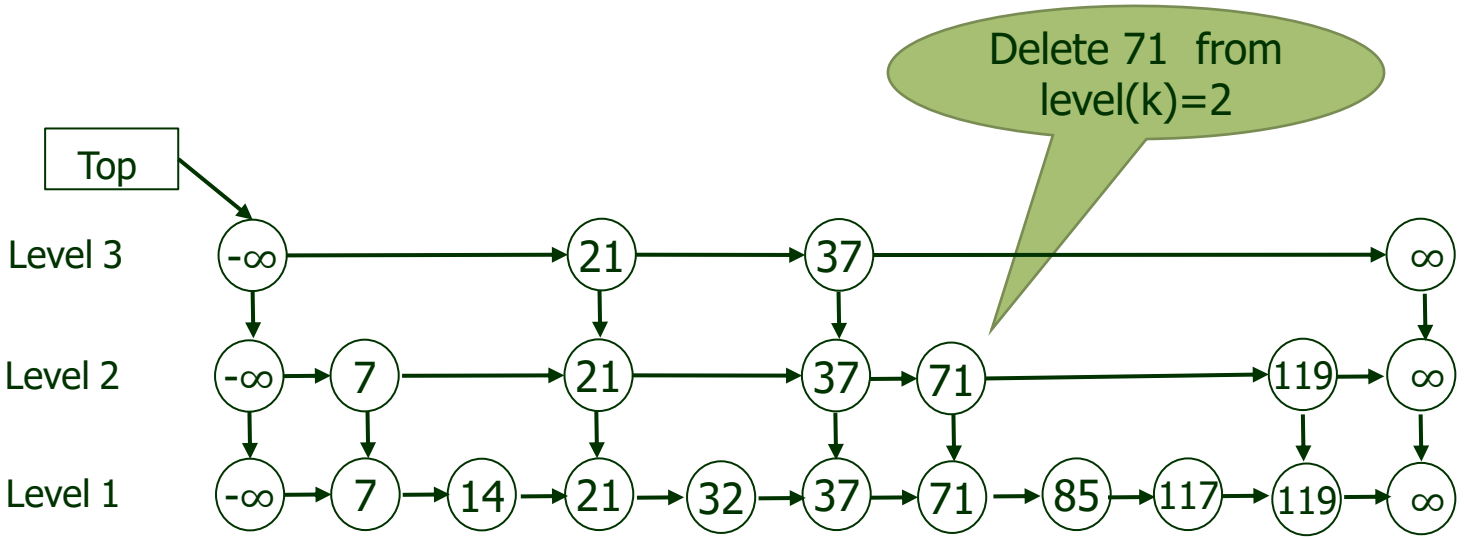


[Note: If k is larger than the current number of levels, add new levels and update the top pointer]

Skip List

- Deleting a key X
  - Apply Find x in all the levels, and delete the key X by using the standard 'delete from a linked list' method.
  - If one or more of the upper levels are empty, remove them and update the top pointer.

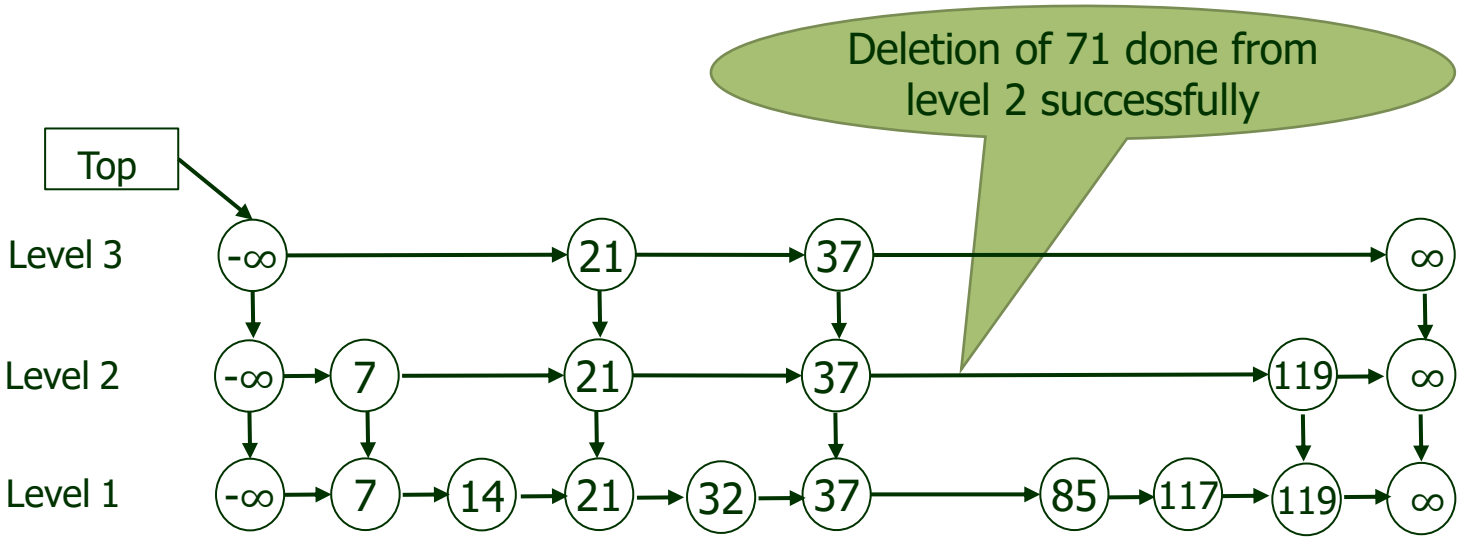
Example : Delete 71 from level 2



Skip List

- Deleting a key X
  - Apply Find x in all the levels, and delete the key X by using the standard 'delete from a linked list' method.
  - If one or more of the upper levels are empty, remove them and update the top pointer.

Example : Delete 71 from level 2



Trie

Trie

Definition:

- A data structure for representing a collection of strings.
- In computer science, a trie, also called digital tree and sometimes radix tree or prefix tree.
- The term trie comes from retrieval.
- This term was coined by Edward Fredkin, who pronounce it tri as in the word retrieval.

Trie

Properties:

- A multi-way tree.
- Each node has from 1 to n children.
- Each edge of the tree is labeled with a character.
- Each leaf nodes corresponds to the stored string, which is a concatenation of characters on a path from the root to this node.

Trie

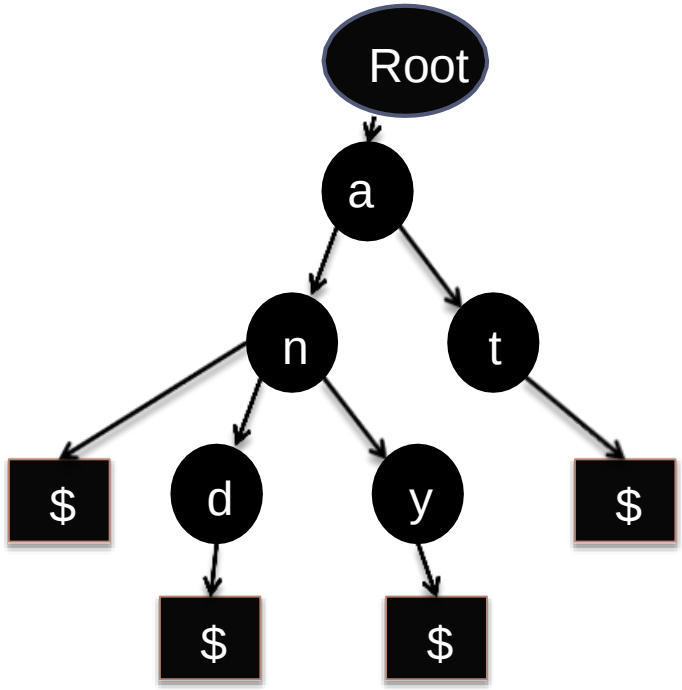
Types:

- Standard Tries
- Compressed/Compact Tries
- Suffix Tries

Trie

Standard Trie:

- The standard trie for a set of strings S is an ordered tree such that:
  - Each node but the root is labeled with a character.
  - The children of a node are alphabetically ordered.
  - The paths from the external nodes to the root yield the strings of S.
  - Example :Strings ={an, and, any, at}
  - append a special termination symbol "$"

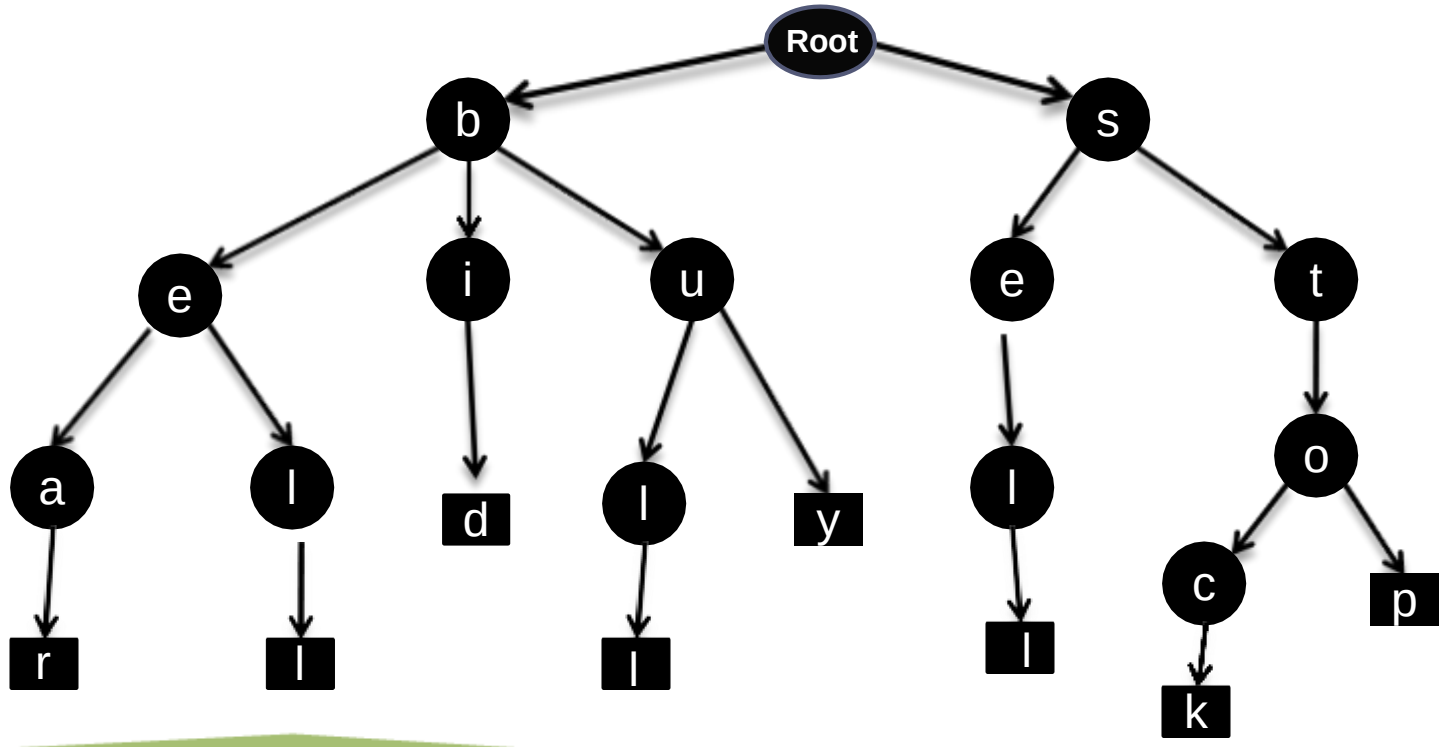


Trie

Standard Trie:

- Example: Standard trie for the set of strings

  S = { bear, bell, bid, bull, buy, sell, stock, stop }

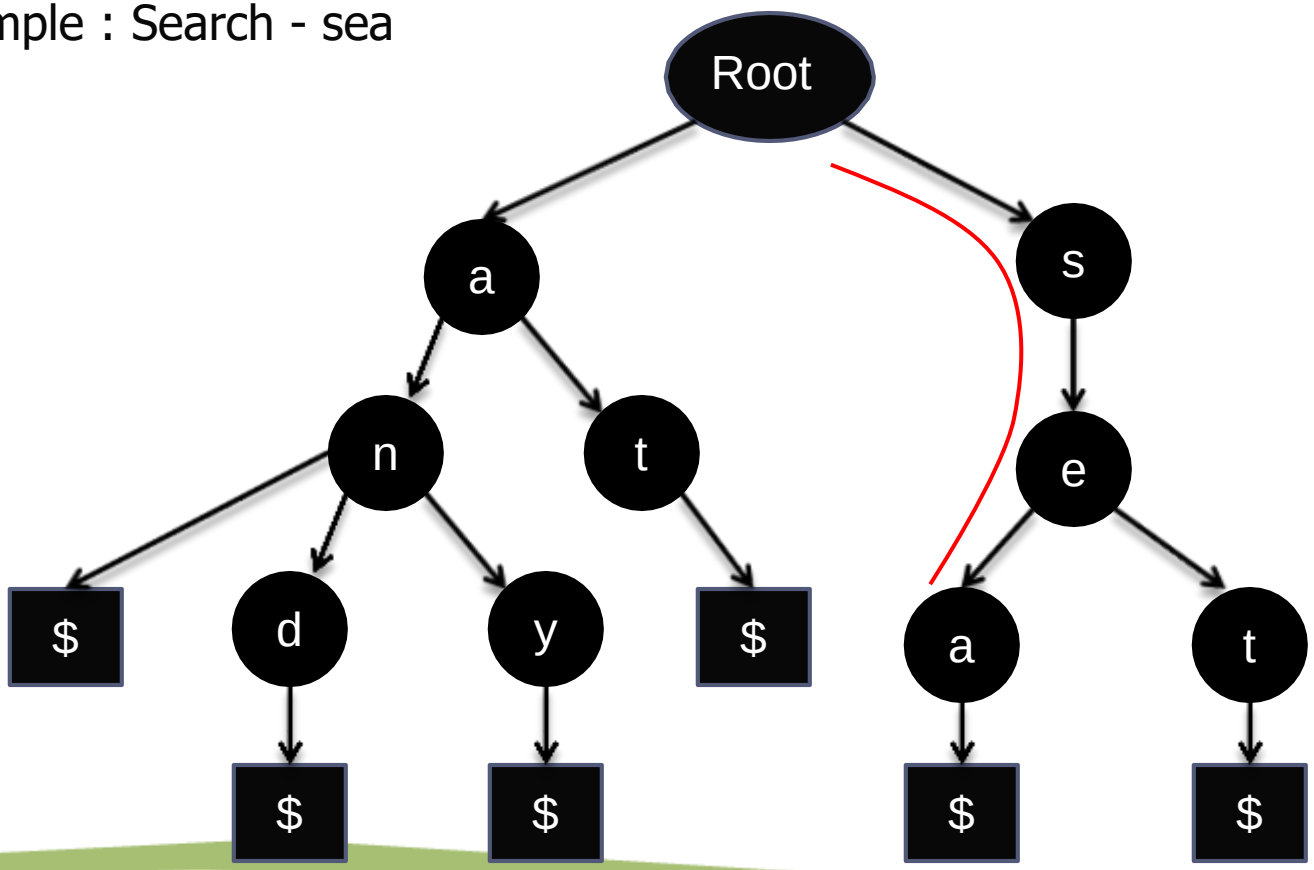


Trie

Standard Trie Searching

Search hit: Node where search ends has a $ symbol

Example : Search - sea

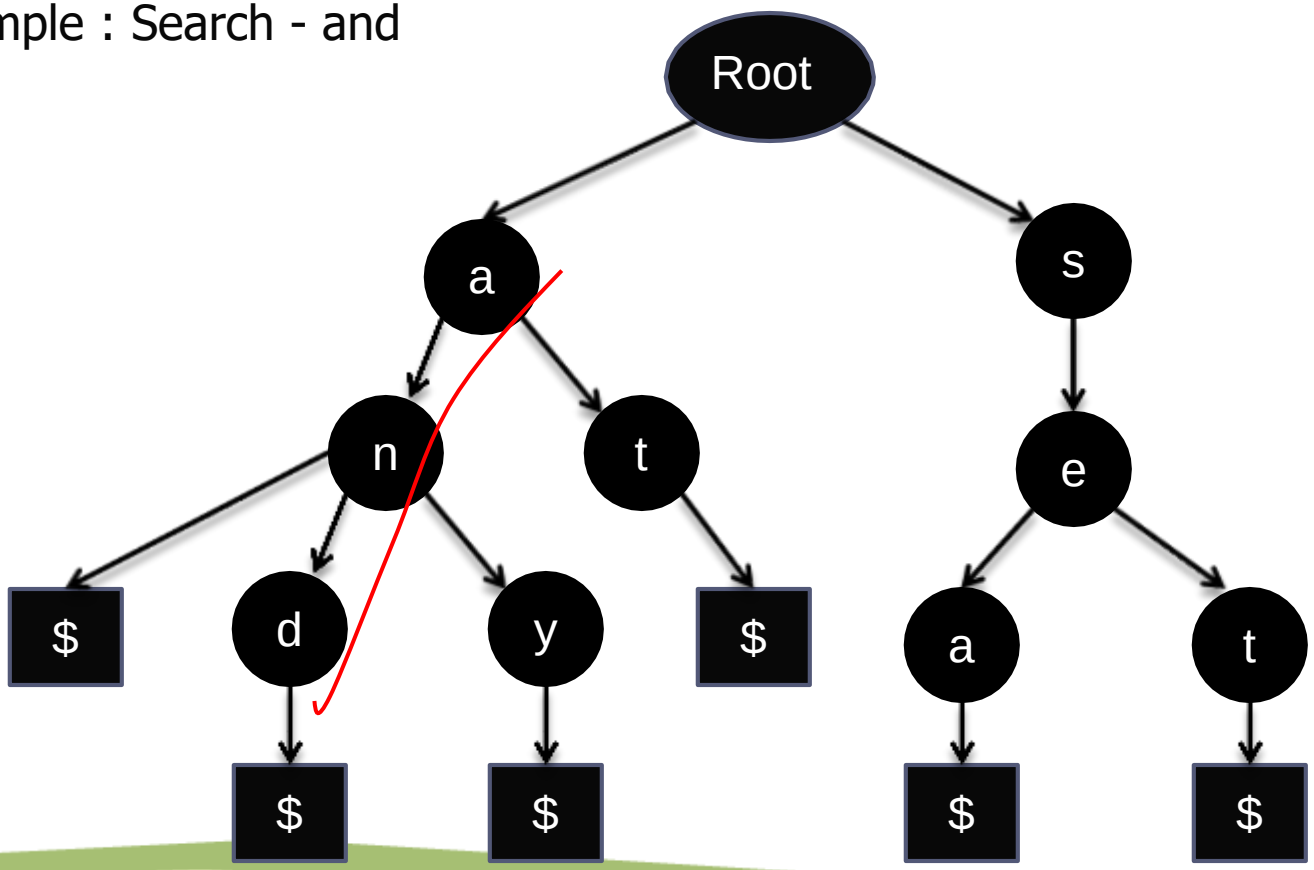


Trie

Standard Trie Searching

Search hit: Node where search ends has a $ symbol

Example : Search - and



Trie

Standard Trie Deletion

- Three cases

  Case 1: Word not found…!
  Case 2: Word exists as a stand alone word.
  Case 3: Word exists as a prefix of another word.

Trie

Standard Trie Deletion

- Three cases

  Case 1: Word not found…!
  then Return False
  Case 2: Word exists as a stand alone word.
  part of any other word
  does not a part of any other word
  Case 3: Word exists as a prefix of another word.

Trie

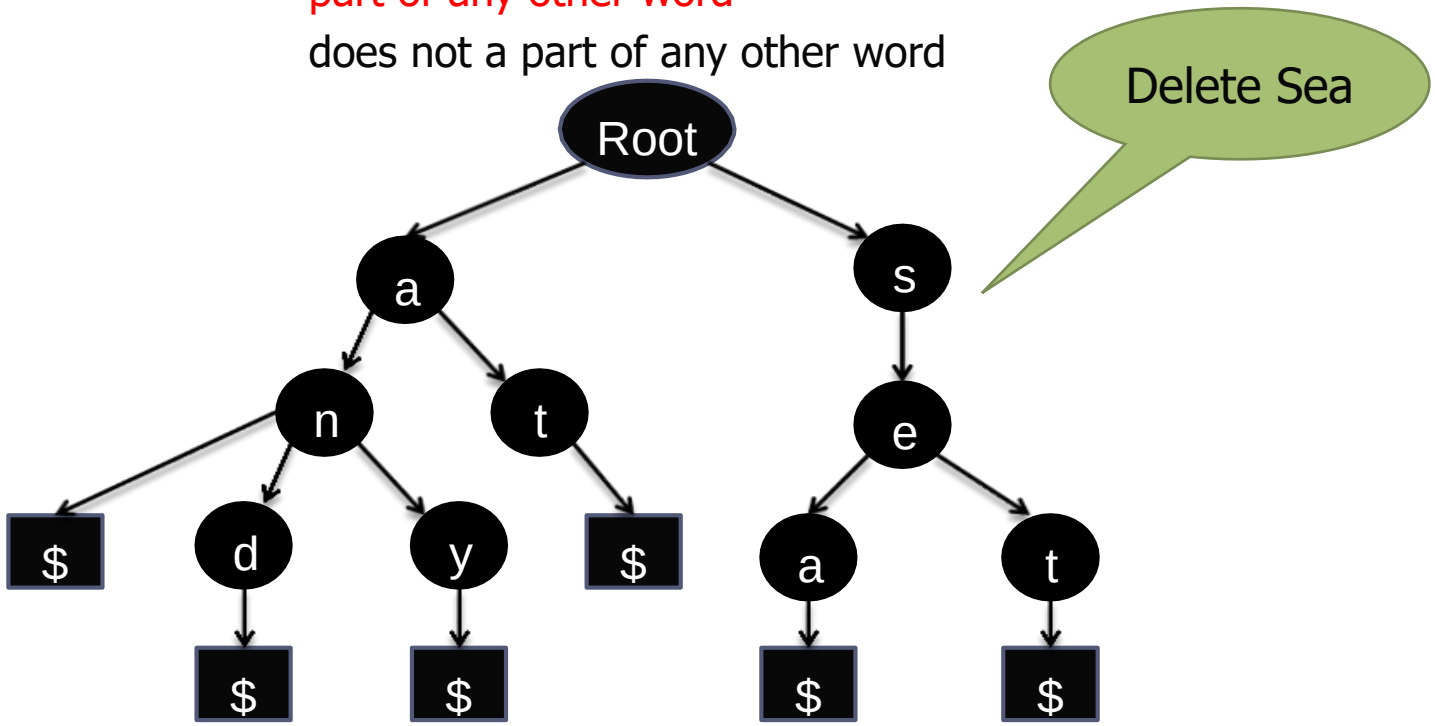
Standard Trie Deletion

- Three cases

  Case 2: Word exists as a stand alone word.

  part of any other word

  does not a part of any other word



Trie

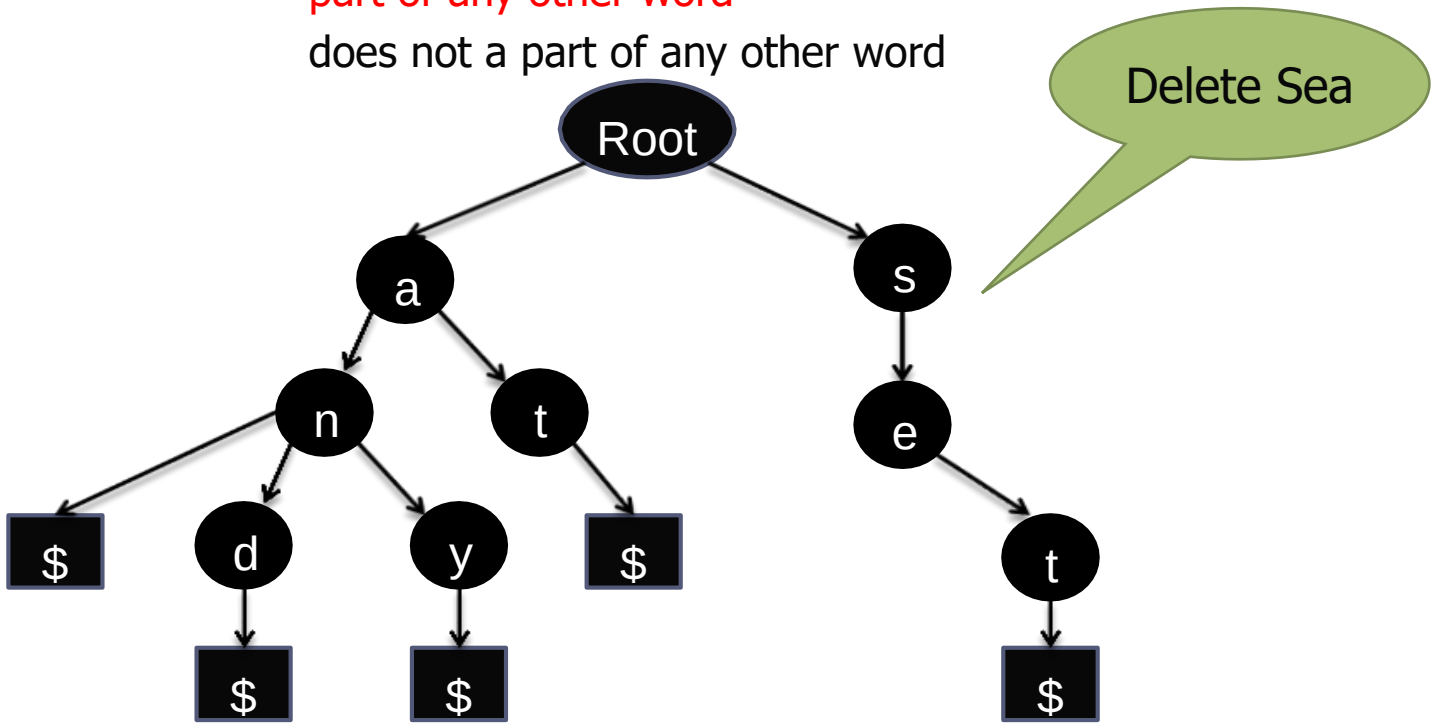
Standard Trie Deletion

- Three cases

  Case 2: Word exists as a stand alone word.

  part of any other word

  does not a part of any other word



Trie

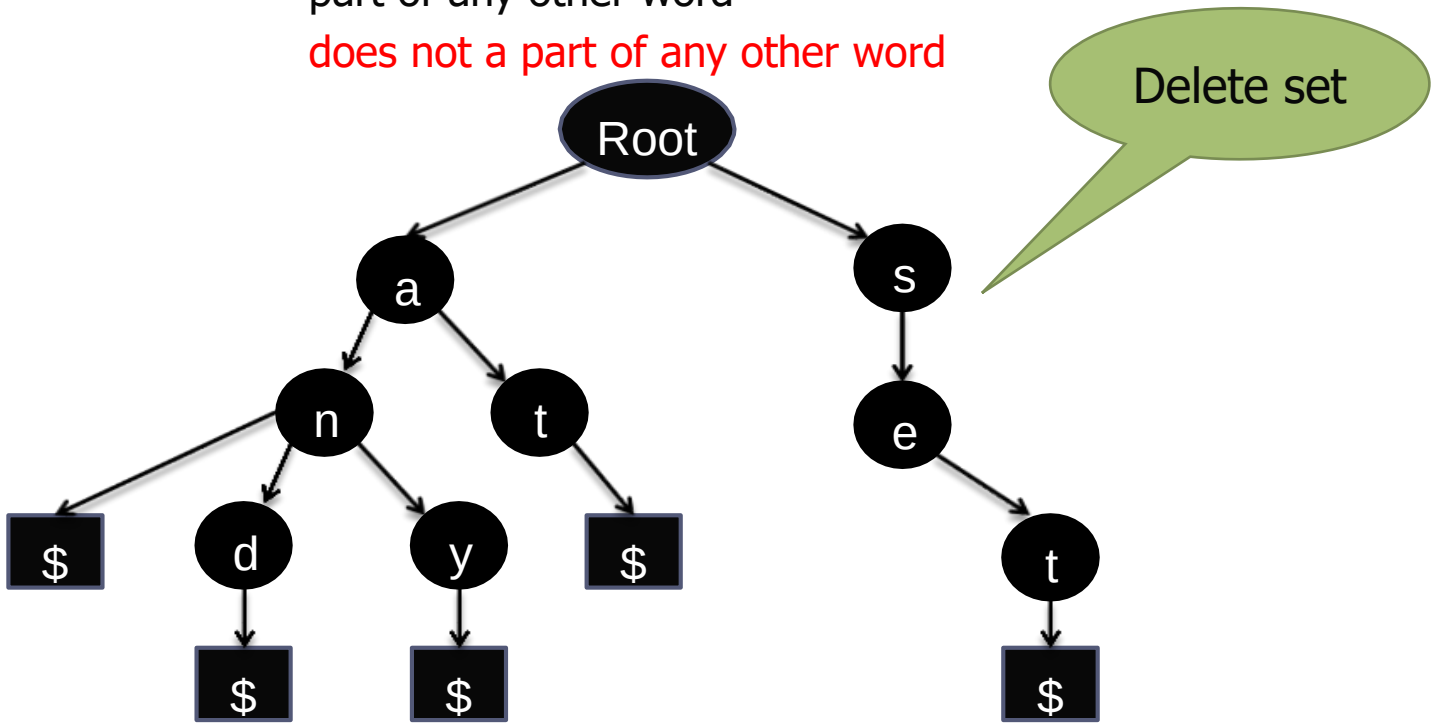
Standard Trie Deletion

- Three cases

  Case 2: Word exists as a stand alone word.

  part of any other word

  does not a part of any other word



Trie

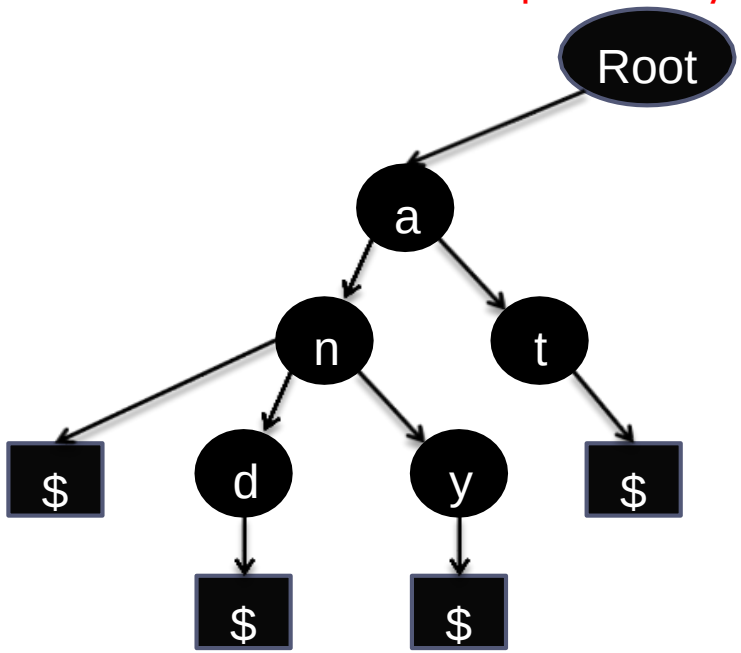
Standard Trie Deletion

- Three cases

  Case 2: Word exists as a stand alone word.

  part of any other word

  does not a part of any other word



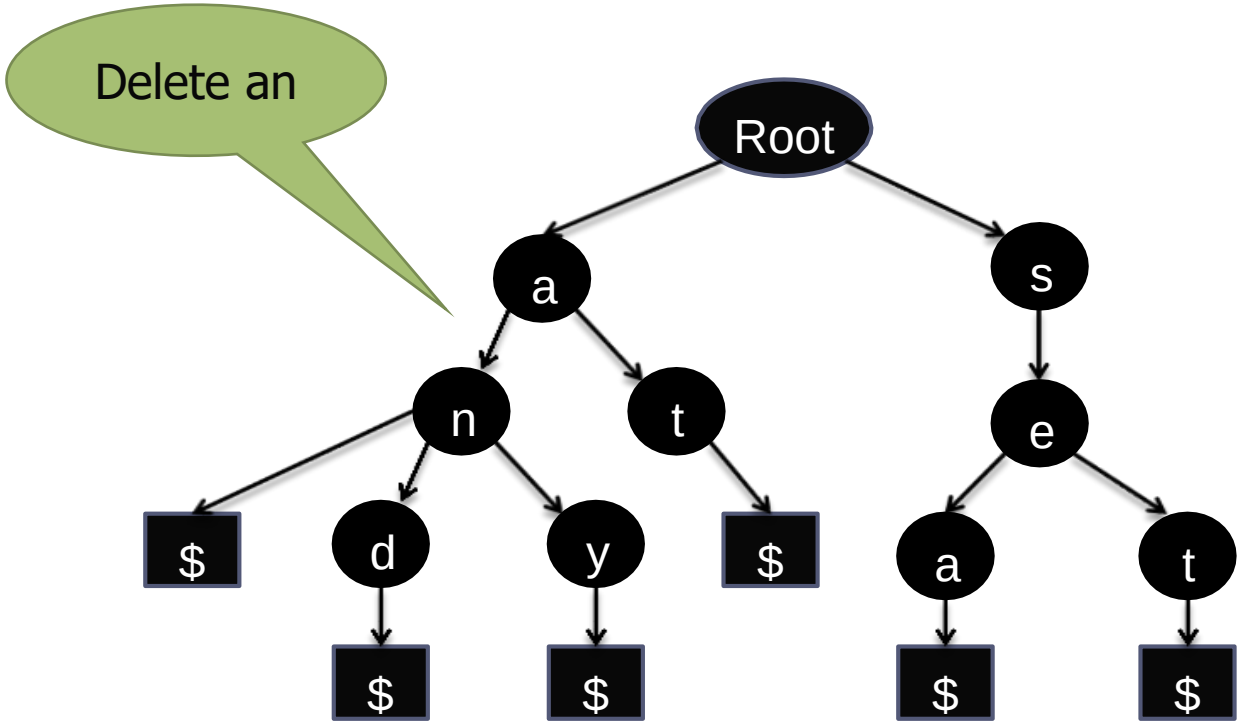
Trie

Standard Trie Deletion

- Three cases

  Case 3: Word exists as a prefix of any other word.

  Delete - an



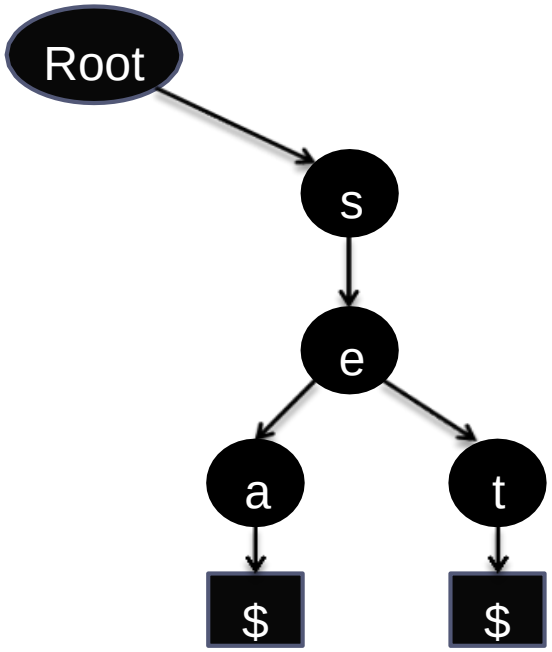
Trie

Standard Trie Deletion

- Three cases

  Case 3: Word exists as a prefix of any other word.

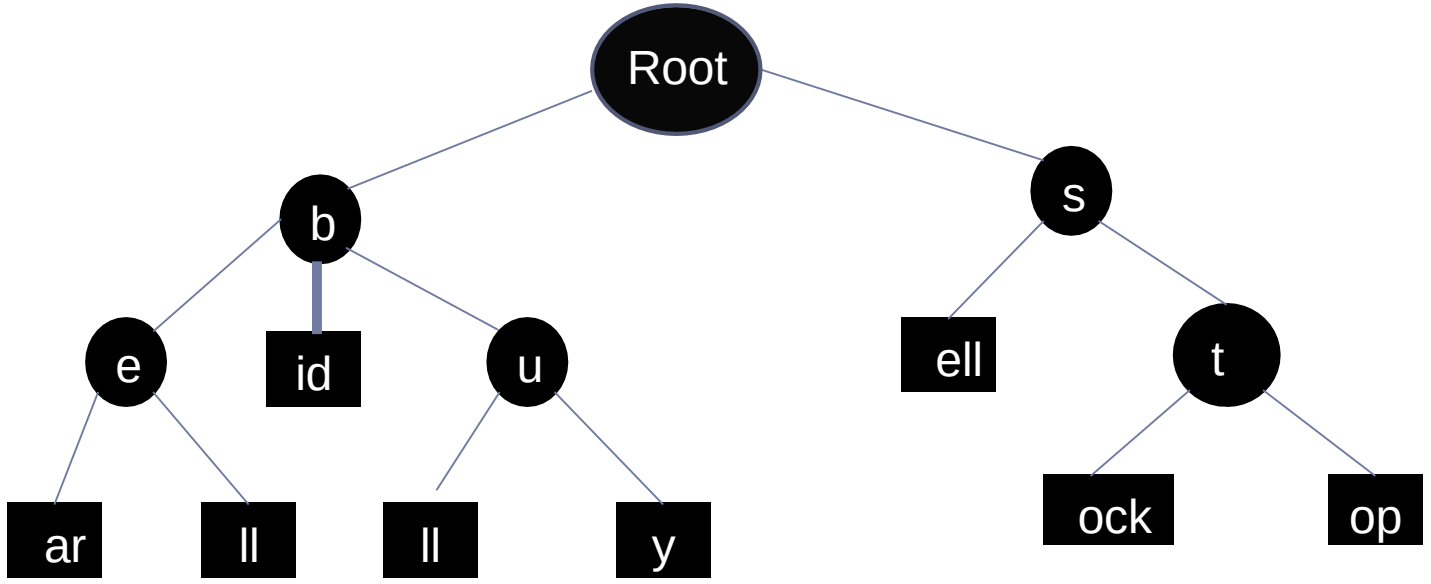
  Delete - an



Trie

Compressed Trie

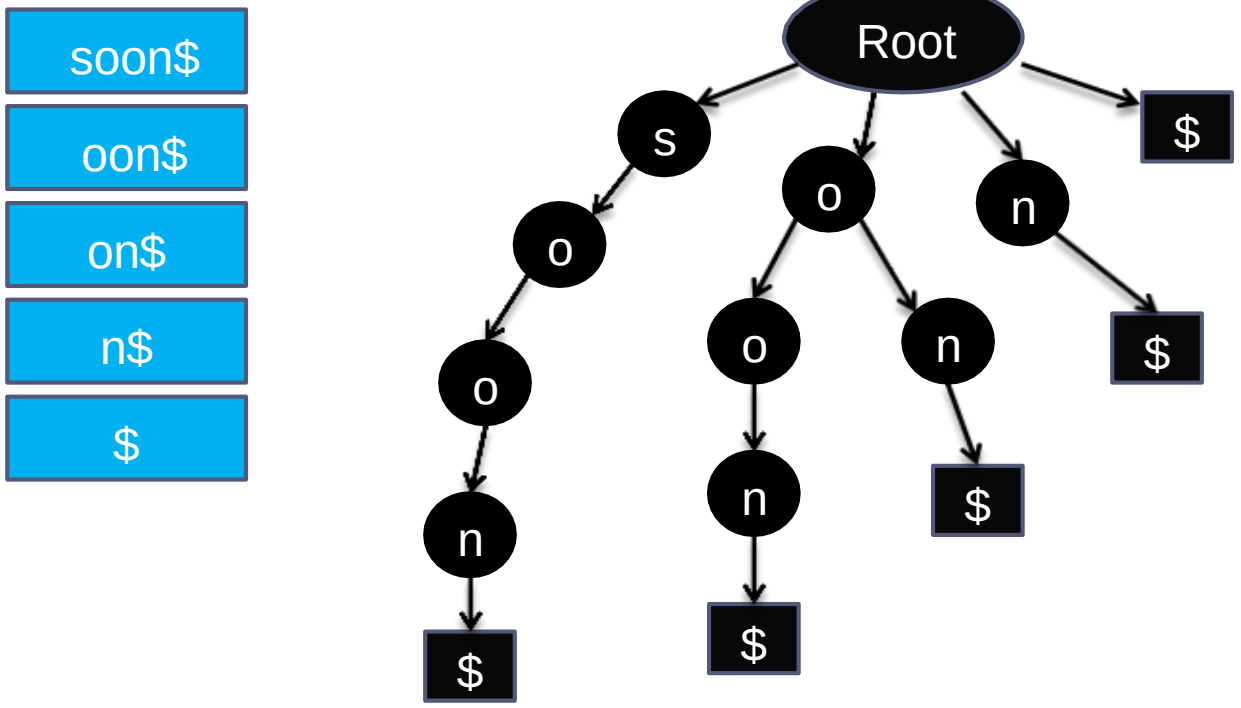
- Tries with nodes of degree at least 2
- Obtained by standard tries by compressing chains of redundant nodes
- Example: S = { bear, bell, bid, bull, buy, sell, stock, stop }



Trie

Suffix Trie

- A suffix trie is a compressed trie for all the suffixes of a text.
- Suffix trie are a space-efficient data structure to store a string that allows many kinds of queries to be answered quickly.




Thank U