Advanced Data Structures

Assignment - 2

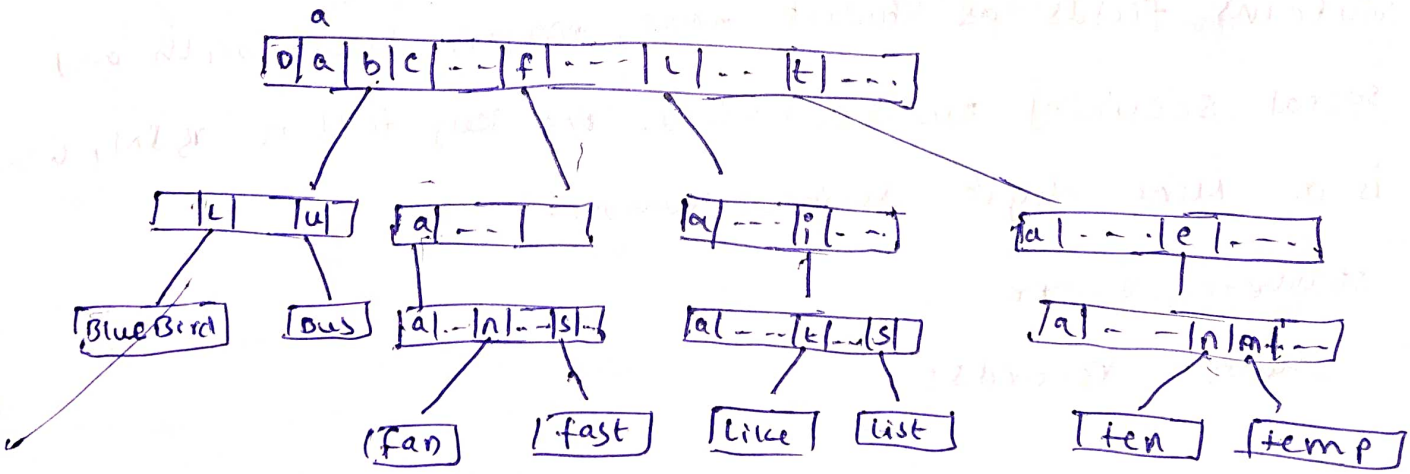
191131A0529
V.V.H. Saketh
CSE-4.

Multiway Tries:

* A Multiway Trie is a structure that is particularly useful when a key values are of varying size.
* Each branch node has more than two pointers where as in binary Trie or others have only two pointers left child and Right child.
* In case of alphabets each branch has 27 pointer fields.
* The extra pointer to a node is blank character i.e, b.

Example:

S = {BlueBird, Bus, Fan, Fast, like, List, ten, Temp}

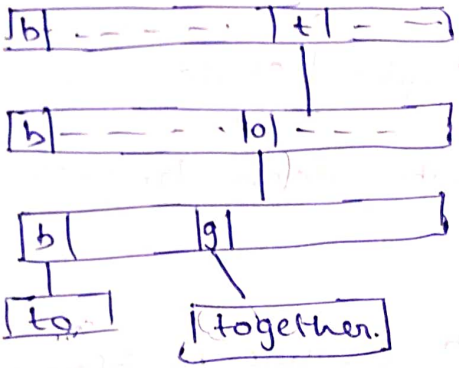


* A tree has two kinds of nodes.

① Branch Nodes. ⟶ pointers to subtrees.

② Elementary Nodes. ⟶ element Node has only data

* No word in s should be the prefix of other.
* At first level all the key values are partitioned into disjoint clases depending on their first character.



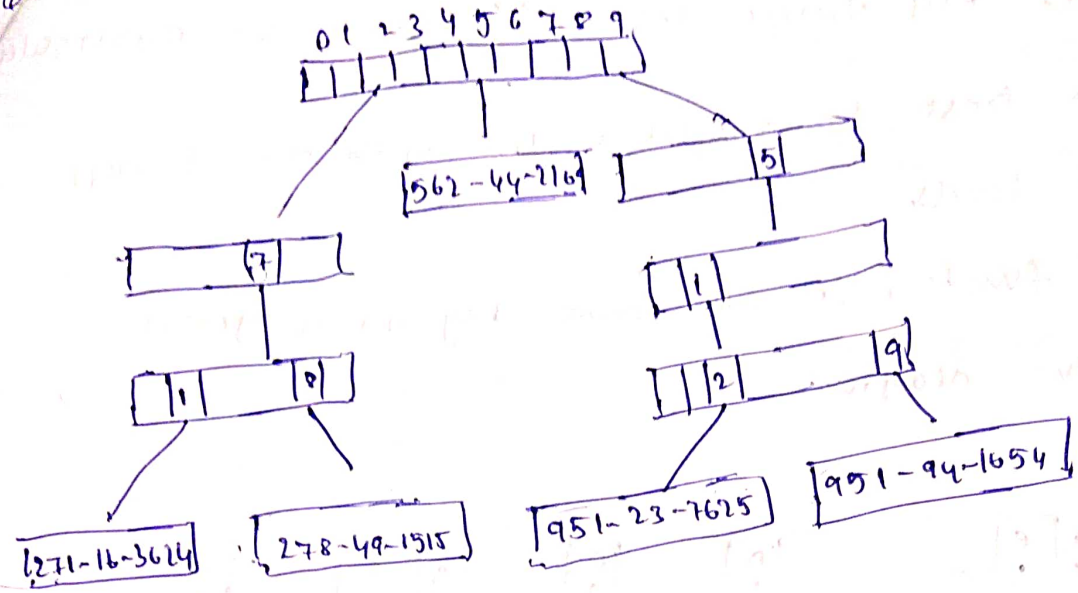
Trie showing need for terminal character.

* Suppose we have a collection of student records that contains fields as student name, major, date of birth and social security number (SSN). The key field is SSN, which is a Nine digit decimal number.

~~Searching a Tr~~

Student records:

Name	SS #
Jack	951-94-1654
Jill	562-44-2169
Bill	271-16-3624
Kathy	278-49-1515
April	951-23-7625



Searching a Trie!

→ An element whose key is k requires breaking up k into its constituent / digits

→ digit(k,i) → returns i^{th} digit of k

→ p=NULL → Not a branch Node.

Worst-case Analysis: O(l) where l is the no. of levels in the trie (including branch and element nodes).

Sampling Strategies:

→ No. of levels depends on the sampling strategies.

→ Sampling function:

sample (x,i) → samples x for branching at i^{th} level

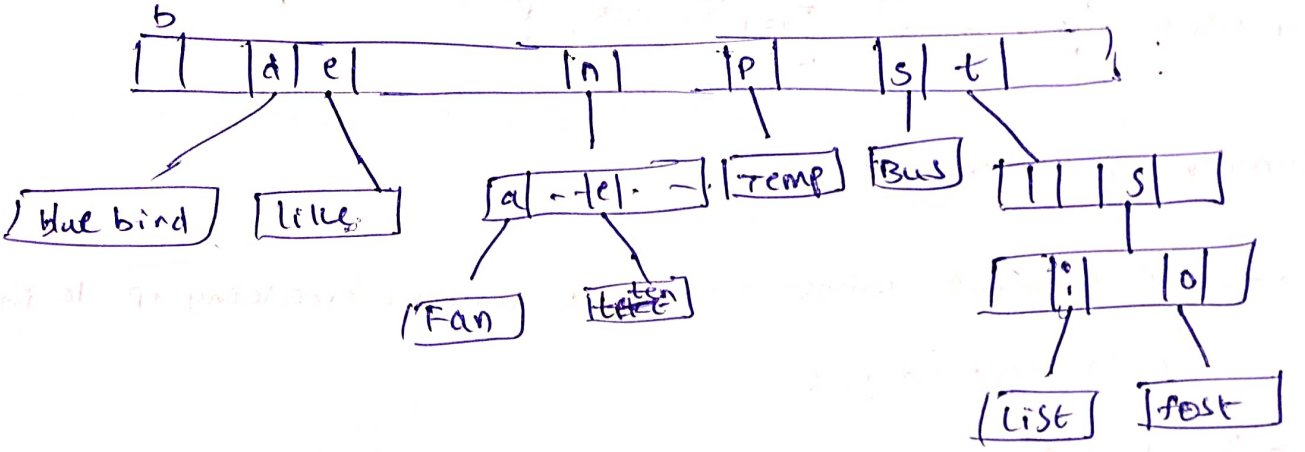
① sample (x,i) = x_{n-i+1}, where $x = x_1, x_2, \cdots, x_n$

② sample (x,i) = $x_{r(x,i)}$ for r(x,i) a randomization function.

③ sample (x,i) = $\begin{cases} x_{i/2}, & \text{if } i \text{ is even} \\ x_{n-(i-1)/2}, & \text{if, } i \text{ is odd.} \end{cases}$

→ can construct key value sets for which the particular function is best i.e, results a Trie with the fewest number of levels.

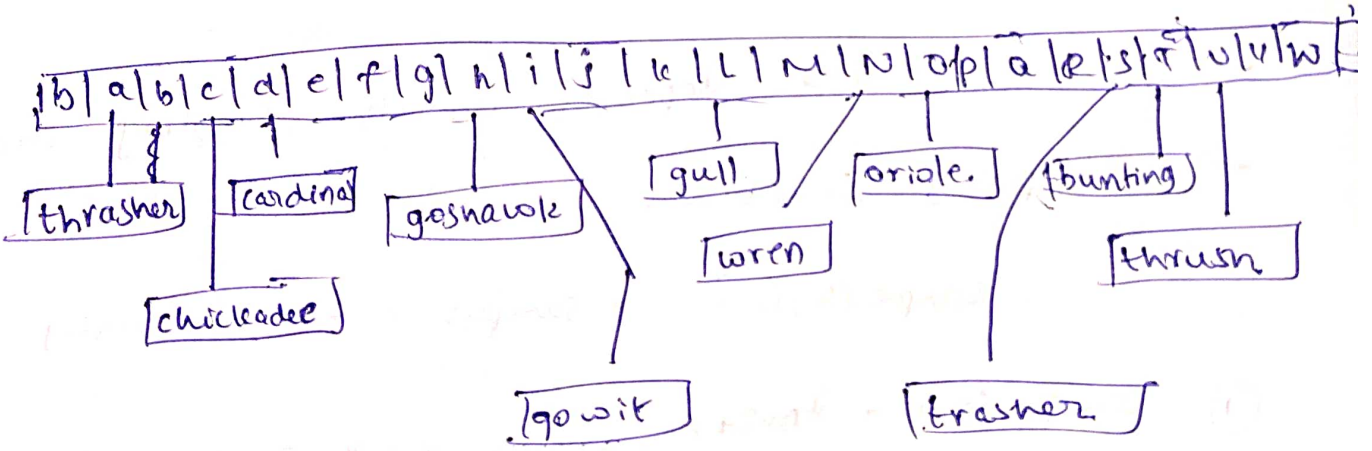
→ Applying function (1) on same key value yields. the below diagram.



→ By using the sampling function the no. of levels are reduced.

S = {bluebird, bunting, chickadee, oriole, thrush, goshawk, cardinal, wren, gull, trasher, gowit}

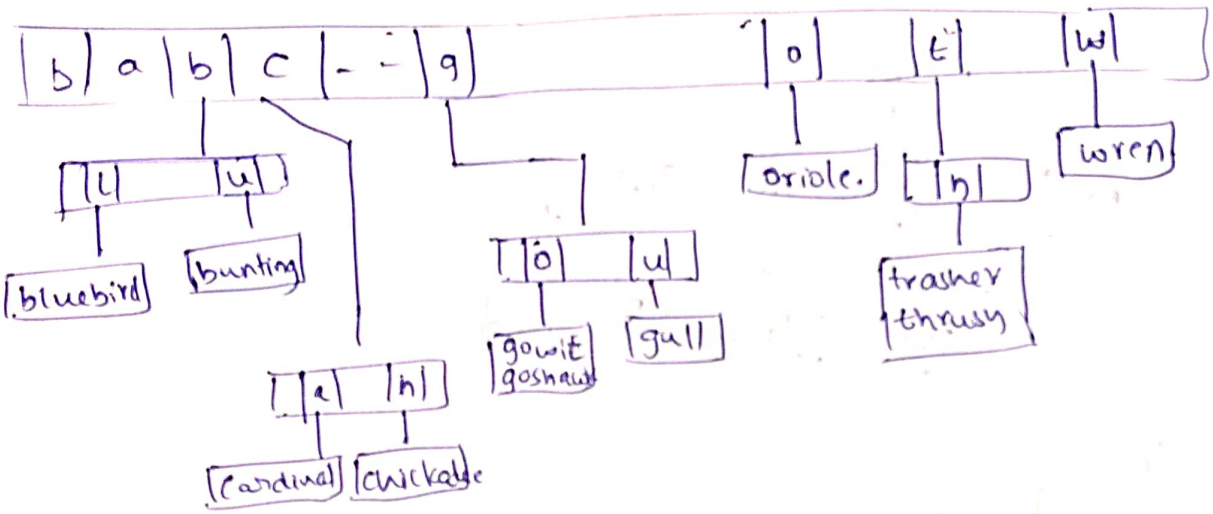
→ By the function (2) if r = 4 then the Trie is.



→ using the sampling function the no. of levels are reduced.

→ if the maximum No. of levels is l, then all key values that are synonyms up to level $l-1$ enter into the same element node.

→ if sampling function is choosen correctly there will be only a few synonyms in each element Node.

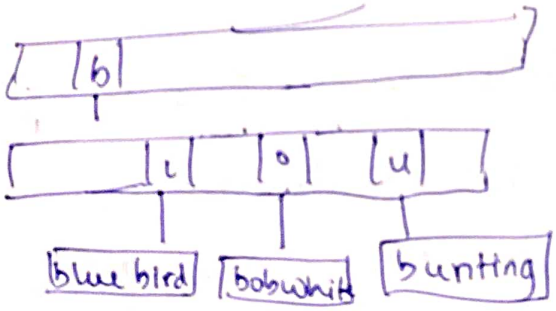


Insertion into a trie:

{blue jay, bobwhite}

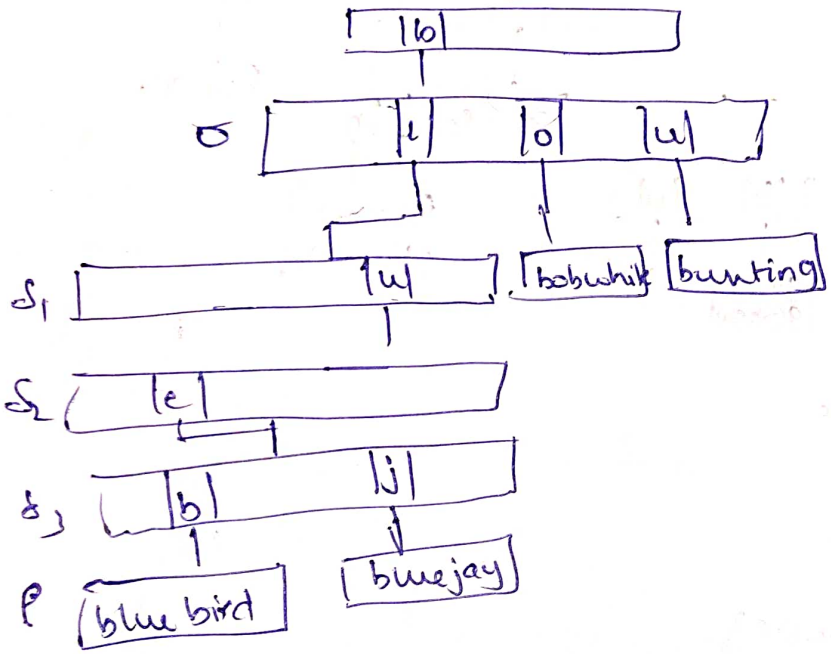
let x = bobwhite.

first search for bobwhite. This leads to Node 5, there we discover that 5.link['o'] = 0. Hence, x is not in the trie and inserted here.



let x = bluejay.

search x in the tree. since, x is not found in the tree. The keys blue bird and blue jay are sampled until the sampling results in two different nodes. This happens at fifth letter.

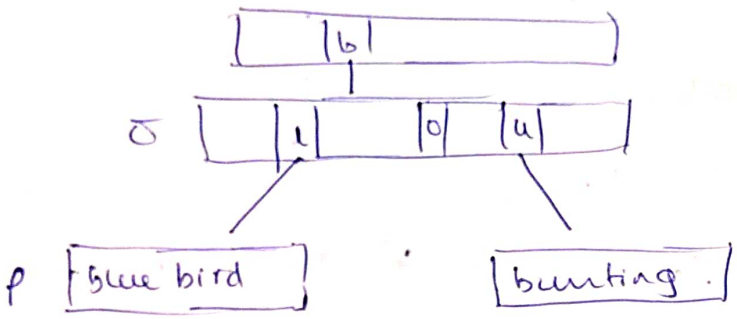


Deletion from a trie:

First let us delete bobwhite from the above example. To do this fist we will set o.link['o'] = 0. and simply remove the data field and no changes need to be made.

Now lets delete bluejay from the trie. This deletion leaves us only one key value in the subtrie, s_3. This means the node s_3 may be deleted and p can be moved up one level. This is same for node s_1 and s_2

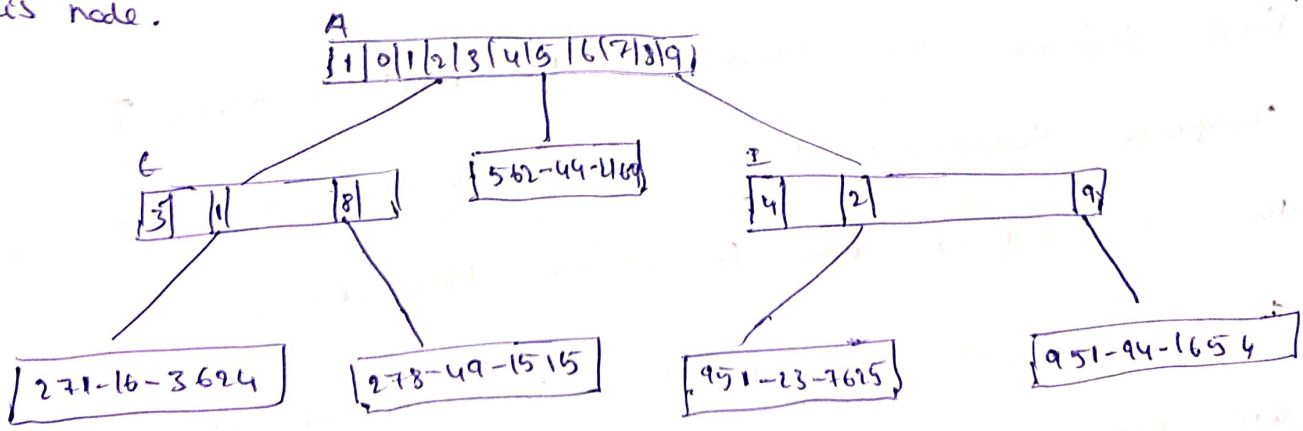
Finally, σ is reached. The σ has more than one key value Therefore p can not be anymore moved up, and we set $\sigma.link['2'] = p$.



After deletion.

Compressed Tries:

* Eleminating all branch nodes that have only one child Then the Resulting trie is called a compressed Trie.
* It will improve both time and space complexity.
* When a branch node with a single child is removed we should store the extra information.
* The Additional information there is stored in a additional field called digit number.
* This tells which digit of the key is used to branch at this node.



Searching a compressed trie with digit numbers:

Searching element ⟶ 951-23-7625

Since, digit No. of A is = 1, we use first digit of the key to determine which subtrie to move to.

A.child[9] = I is made.

So, A move to Node I.

Next, digit No of I = 4, we use 4th digit 2 of the key to determine which subtrie to move to.

I.child[2] = J

Now, we move I to Node J.

We are at element node, so, search key is compared with the element node J.

Since, Search key = J

element is found at J.

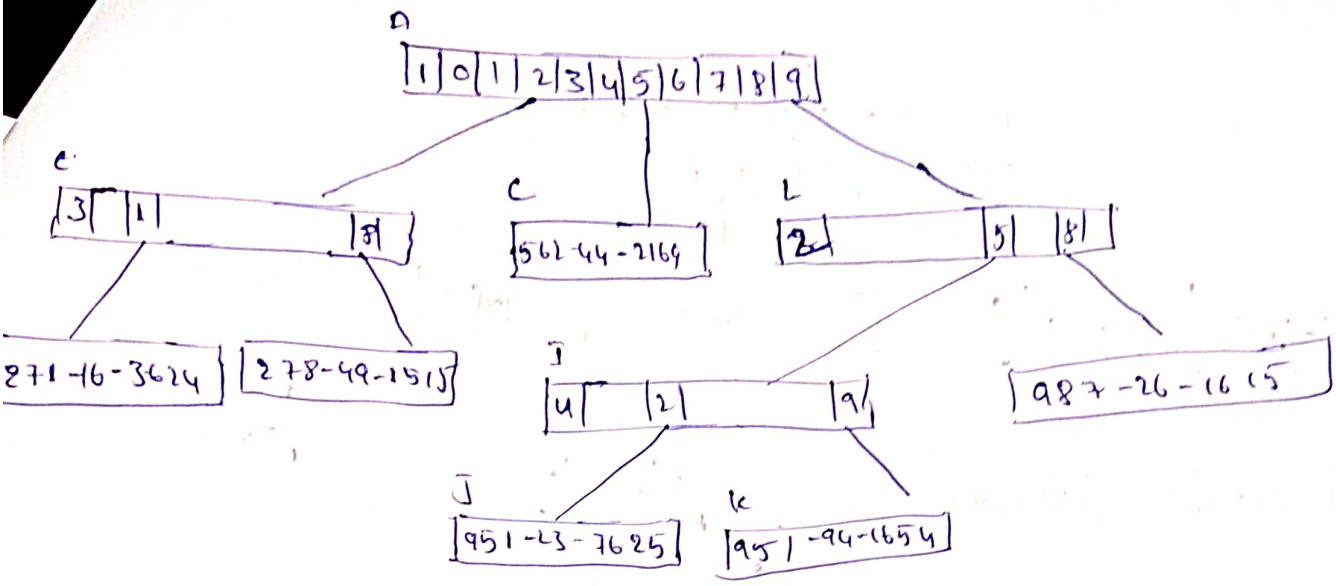
Inserting into a compressed Trie with digit numbers:

987-26-1615 ⟶ inserting.

first search the element it reaches node J

compare search and J it does not matches so, the element is not found in the trie.

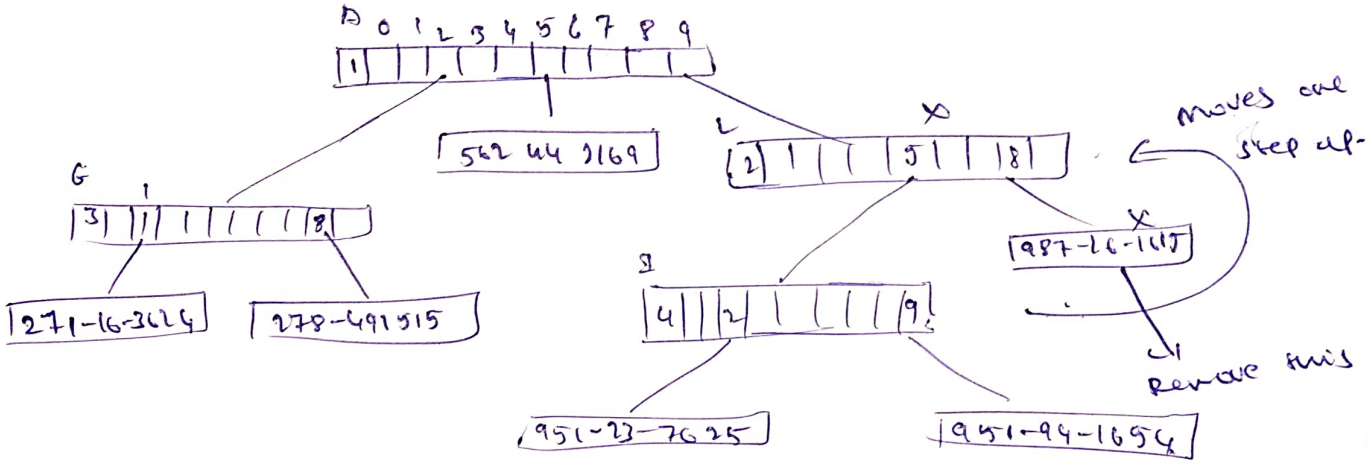
Since, second element of search key and J differ we create a subtrie of digit Number 2



Deleting an element from a compressed Trie with Digit Numbers:

* Search for the key in the compressed Trie.
* if the key value exists remove from the tree.
* if a sub trie is having only one child you can simply remove it and move one level up.

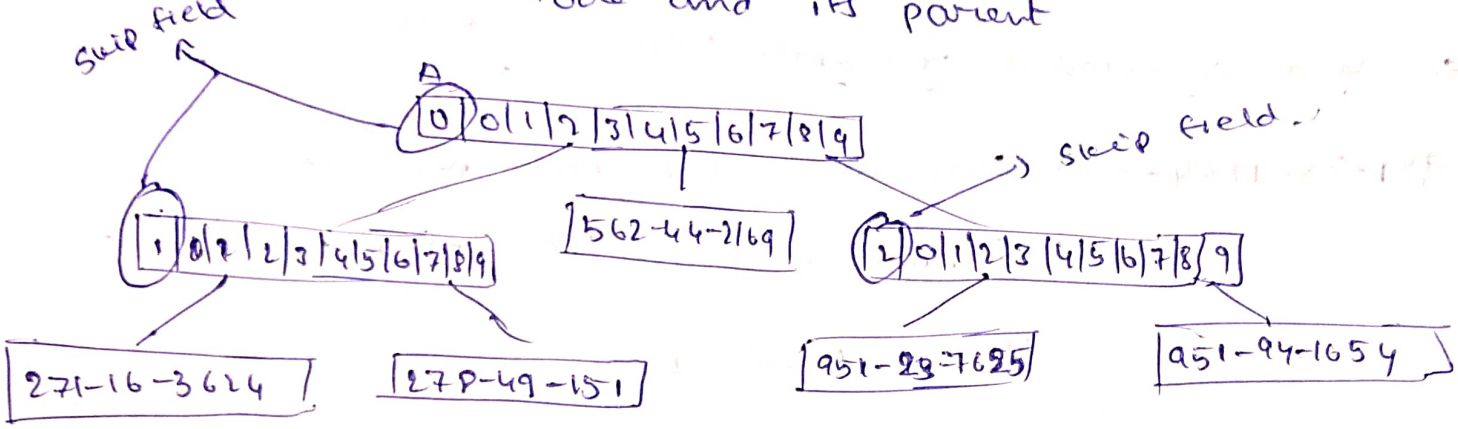
Example: Delete. 987-26-1615



* After deleting the value tree is like this.
* if you observe it L is having a single child so, J moves one level up

67 Compressed Tries with Skip Fields:

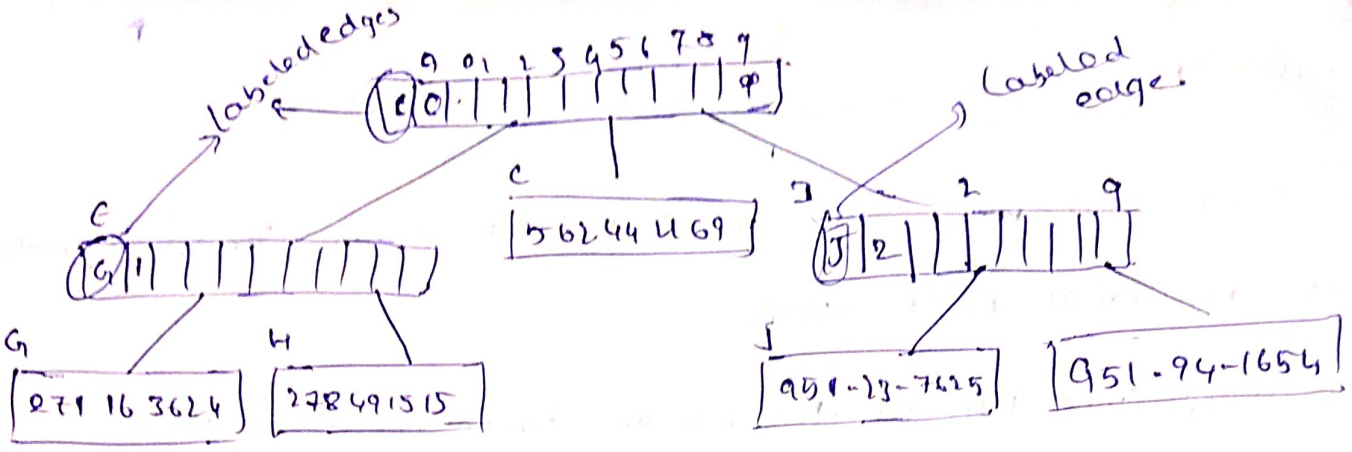
* In compressed Trie with skip fields, each branch node has additional field skip.
* the skip field tells us the no. of branches eliminated b/w its branch node and its parent



* Every left most field of a branch node is skip field
* Insertion, Deletion, and searching are similar to those for a compressed trie with digit numbers.

Compressed Trie with labeled Edges:

* Each branch Node has two additional fields.
  1. pointer / Reference to element node in Trie.
  2. skip field.
* element field stores the label of the element Node.
* It can take any label of the child.
* with the labeled edges you can access the digits which are skipped over.



Search a compressed Trie with labelled Edges:

921-23-1234 ⟶ Skip values for root node is 0.

* So, move to child of 9 ⇒ A.child[9] = I

Move A ⟶ I.

* Skip value for ~~root~~ I node is 2
* The digits 5 and 1 are skipped.
* Since the digits doesn't match Next two digits of search key. It, is terminated.

Inserting into a Compressed Trie with labelled Edges:

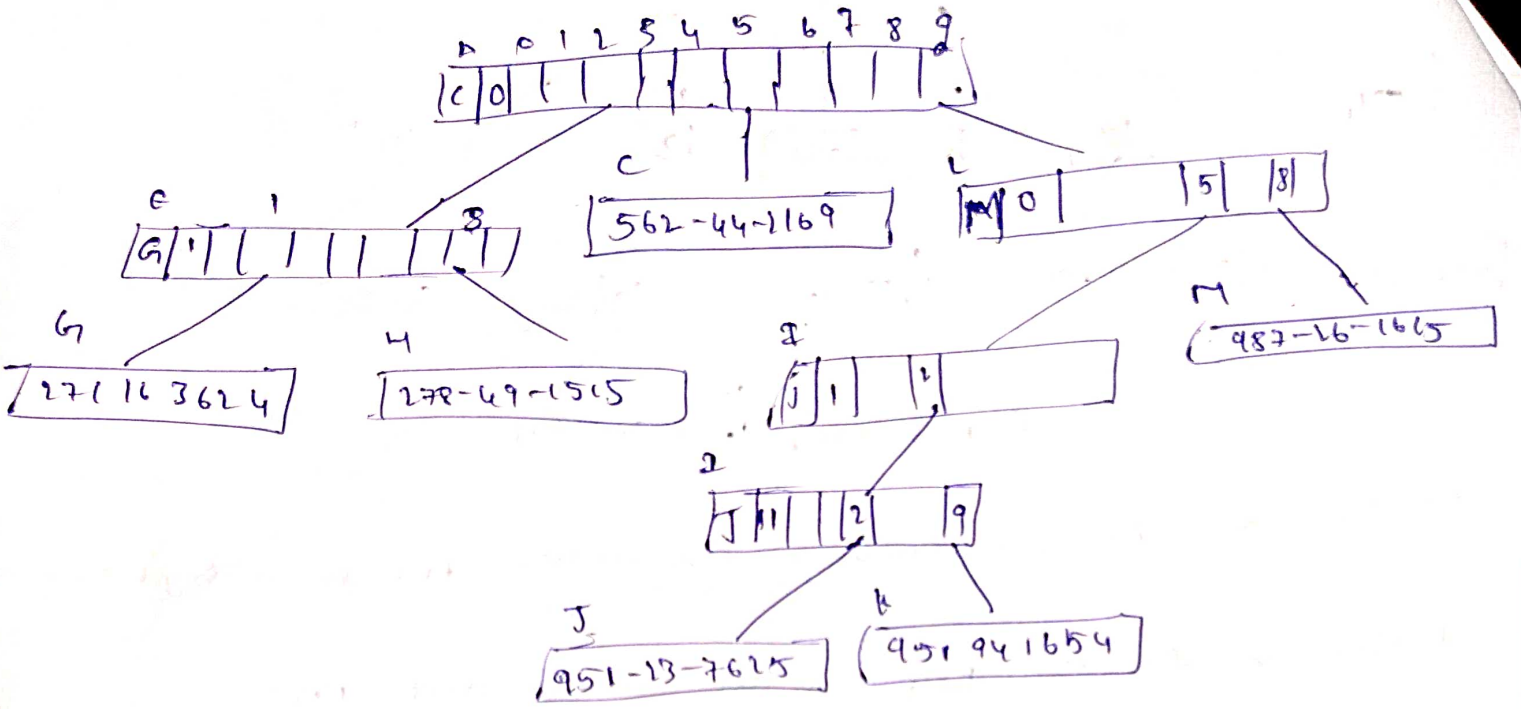
987-26-1615 ⟶ Skip values for root node is 0.

* So, move to child of 9 ⇒ A.child[9] = I

Move A ⟶ I

* Skip value for I node is 2
* The digits 8 and 7 are mismatched with 5 and 1.

* The first mismatch is b/w A and I so, we insert Node b/w them.


A 0 1 2 3 4 5 6 7 8 9
C 0
C
562-44-2169
E 1 8
G 1
L 0 5 8
M
987-16-1665
G
271 16 3624
H
278-69-1515
I
I 1 2
J
J 1 2 9
J
951-13-7625
K
951 94 1654