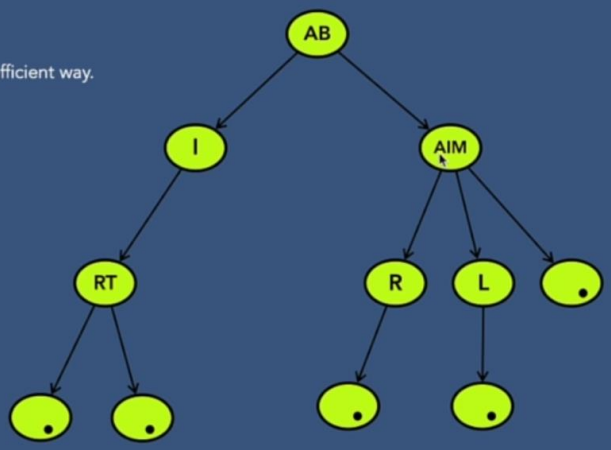
What is a Trie?

A Trie is a tree-based data structure that organizes information in a hierarchy.

Properties:

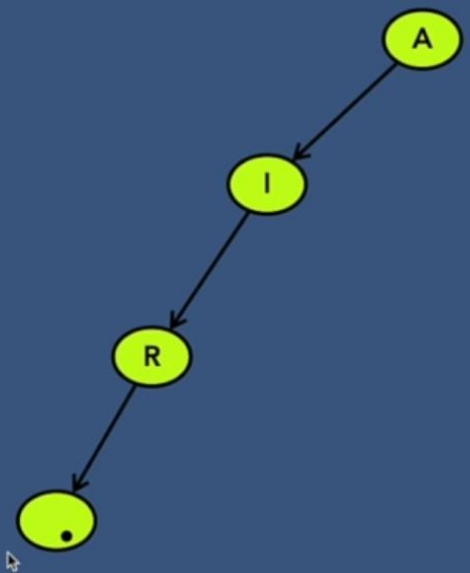
- It is typically used to store or search strings in a space and time efficient way.
- Any node in trie can store non repetitive multiple characters
- Every node stores link of the next character of the string
- Every node keeps track of 'end of string'



What is a Trie?

A Trie is a tree-based data structure that organizes information in a hierarchy.

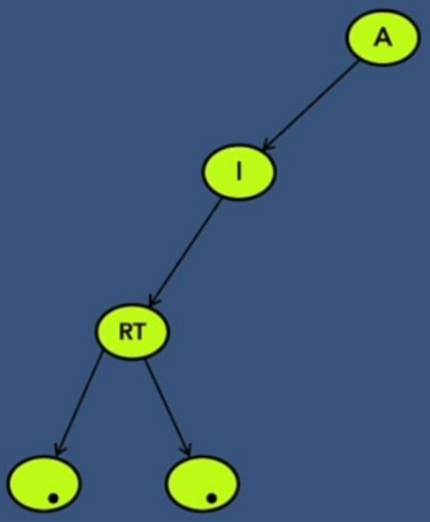
AIR



What is a Trie?

A Trie is a tree-based data structure that organizes information in a hierarchy.

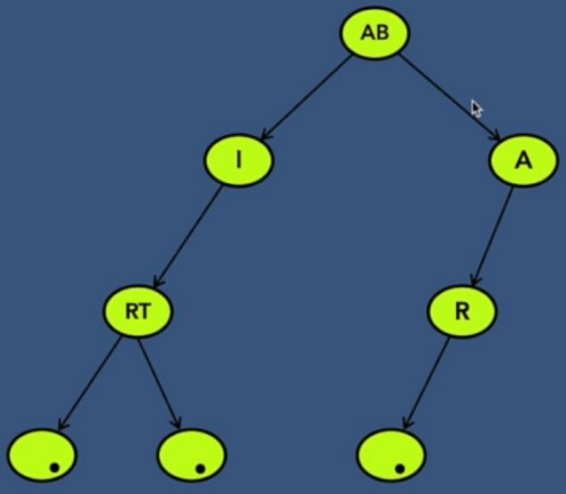
BAR



What is a Trie?

A Trie is a tree-based data structure that organizes information in a hierarchy.

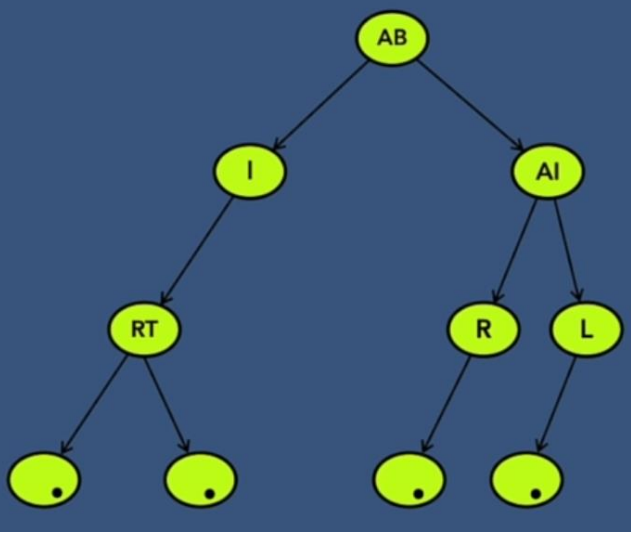
BIL



What is a Trie?

A Trie is a tree-based data structure that organizes information in a hierarchy.

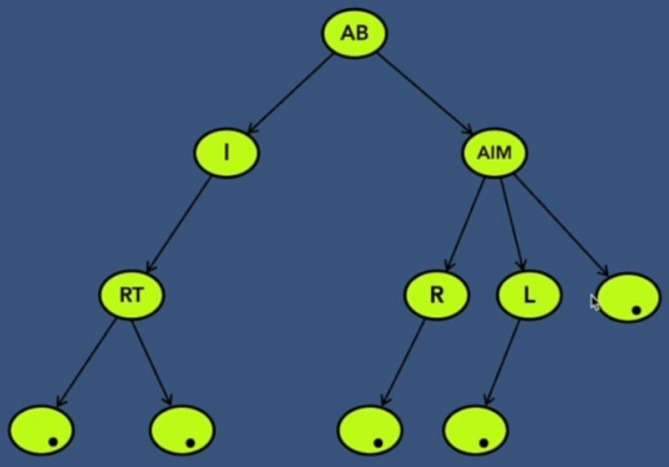
BIL



What is a Trie?

A Trie is a tree-based data structure that organizes information in a hierarchy.

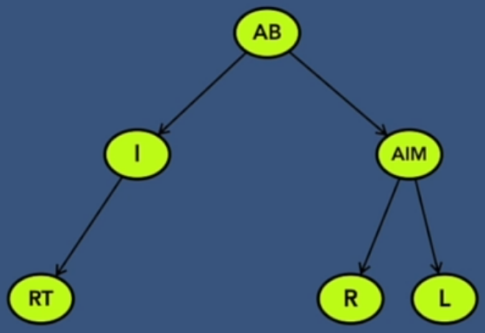
BM



Why we need Trie?

To solve many standard problems in efficient way

- Spelling checker
- Auto completion

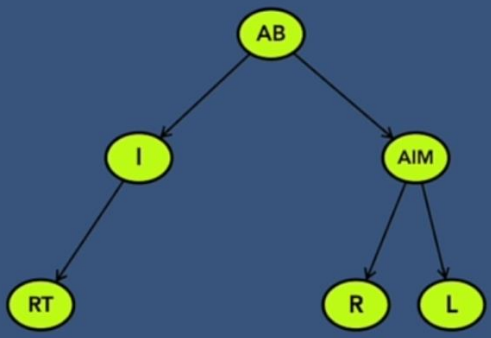


Why we need Trie?

To solve many standard problems in efficient way

- Spelling checker
- Auto completion

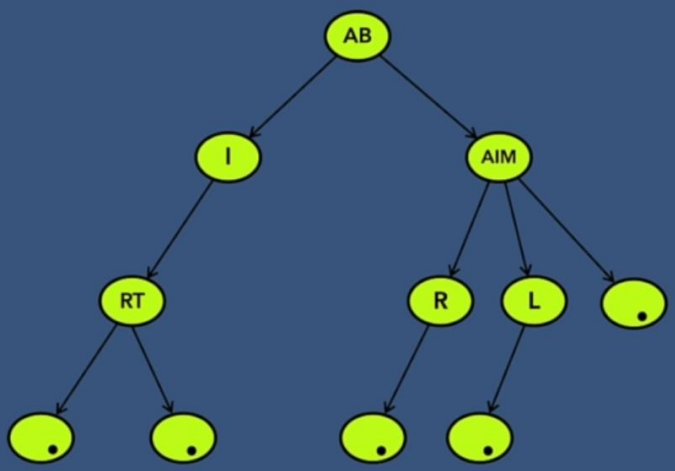
Map	
Characters	Link to Trie Node
End of String	



Also used for prefix check in English

Common Operations on Trie

- Creation of Trie
- Insertion in Trie
- Search for a String in trie
- Deletion from Trie



Common Operations on Trie

- Creation of Trie

Logical

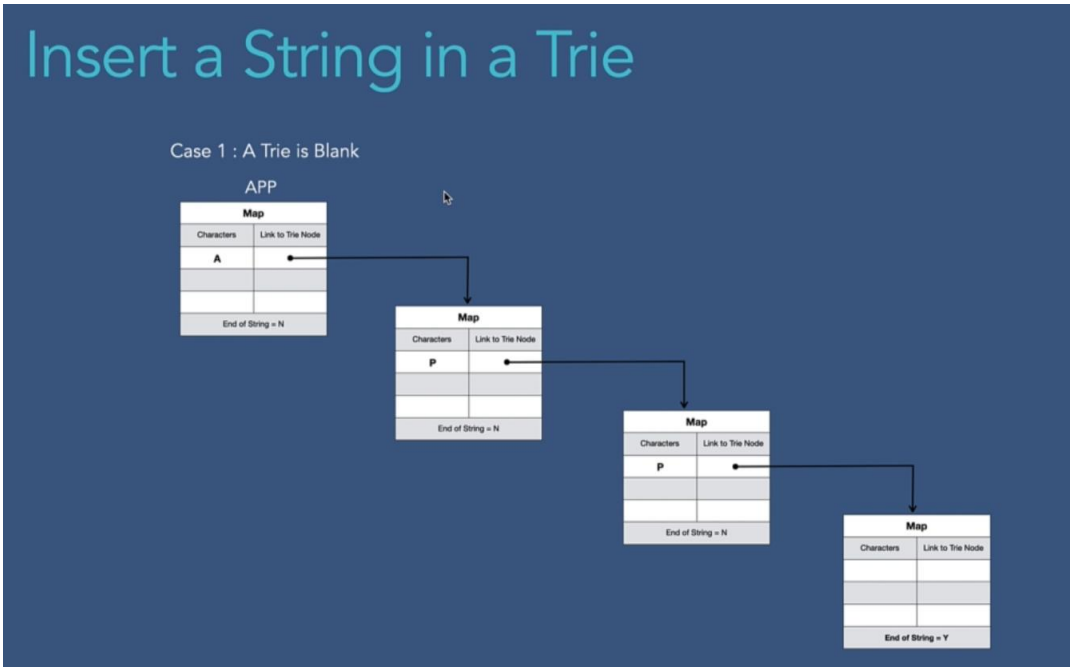
Physically

Map	
Characters	Link to Trie Node
End of String	

Initialize Trie() class

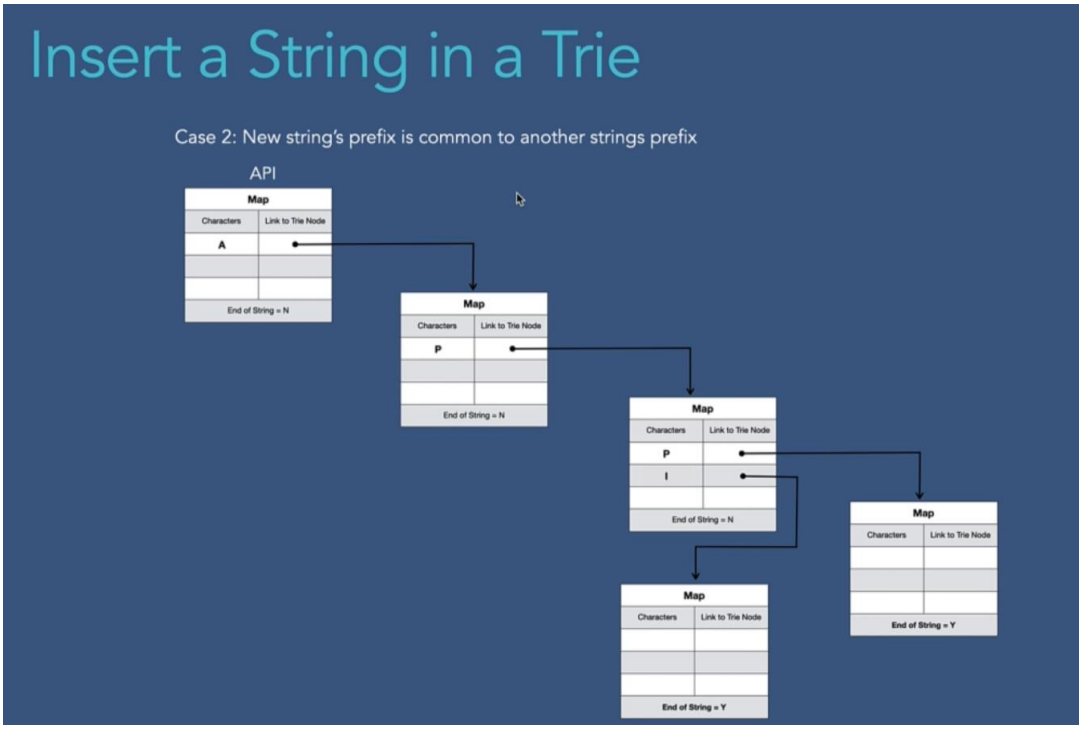
We are using HaspMap for the implementation of Trie Data Structure

Insertion of a String “APP” to Trie Data Structure

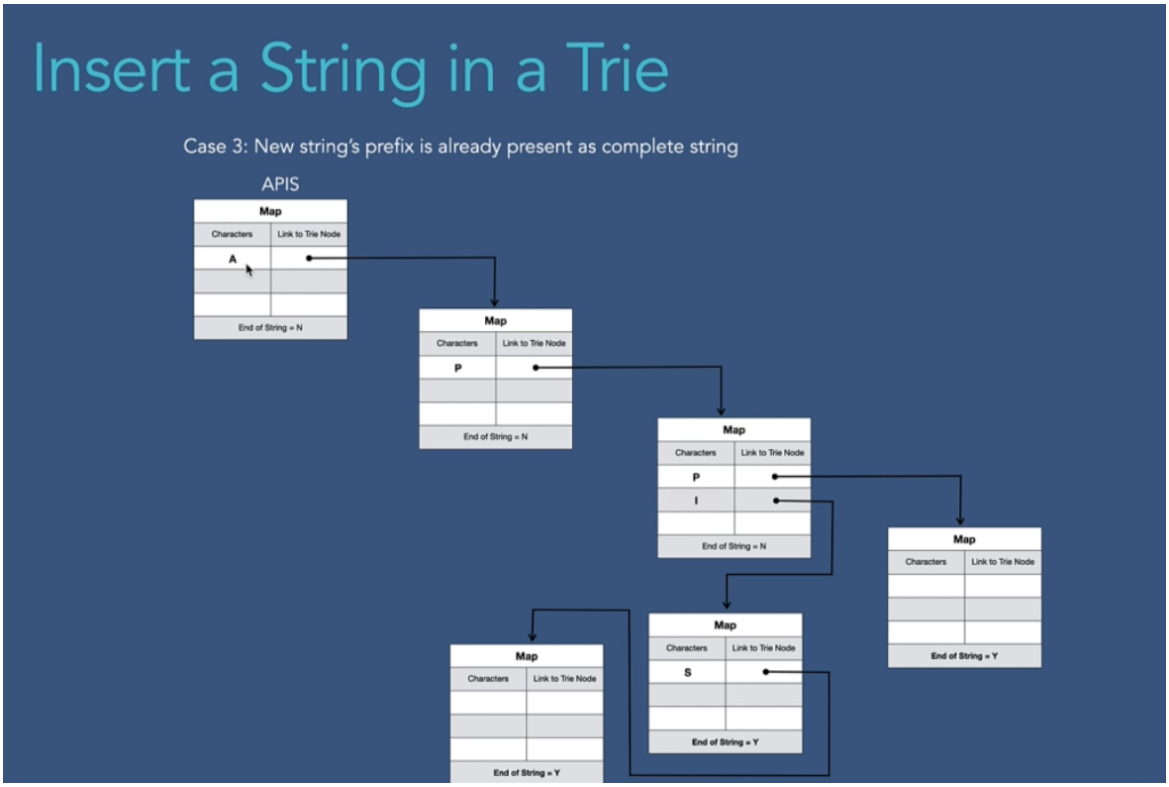


N stands for NO and Y stands for YES in End of String above in the image.

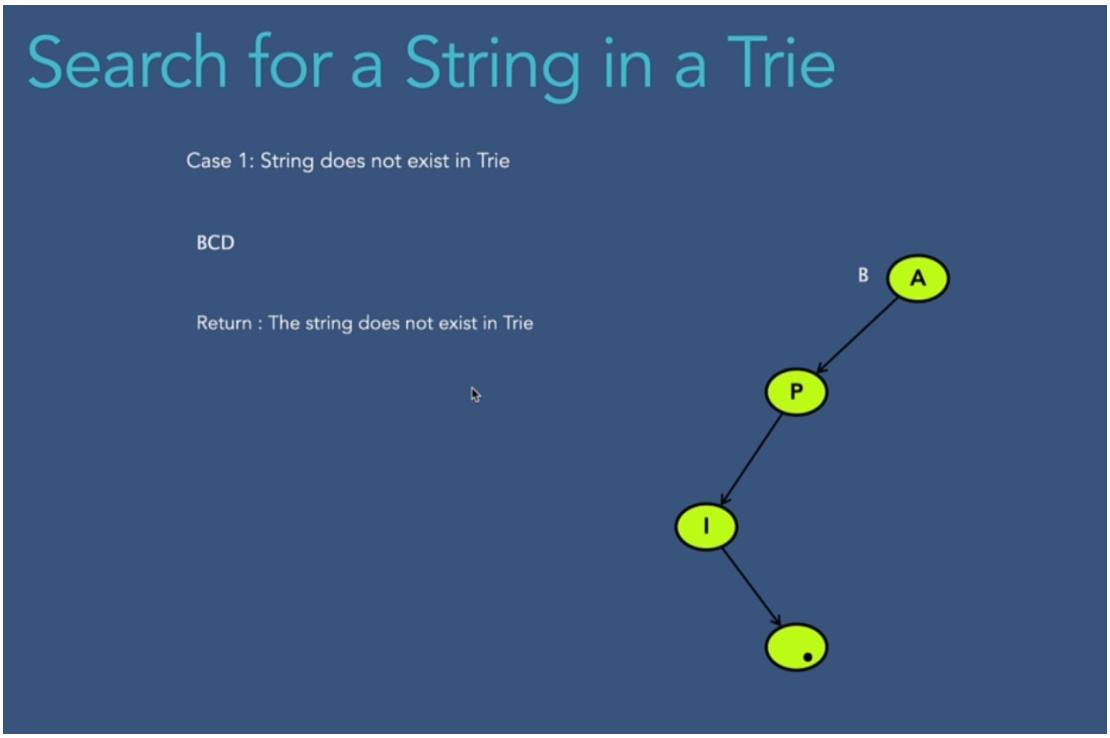
When the End of String is given as Y (which means YES), then that is the end of the string.



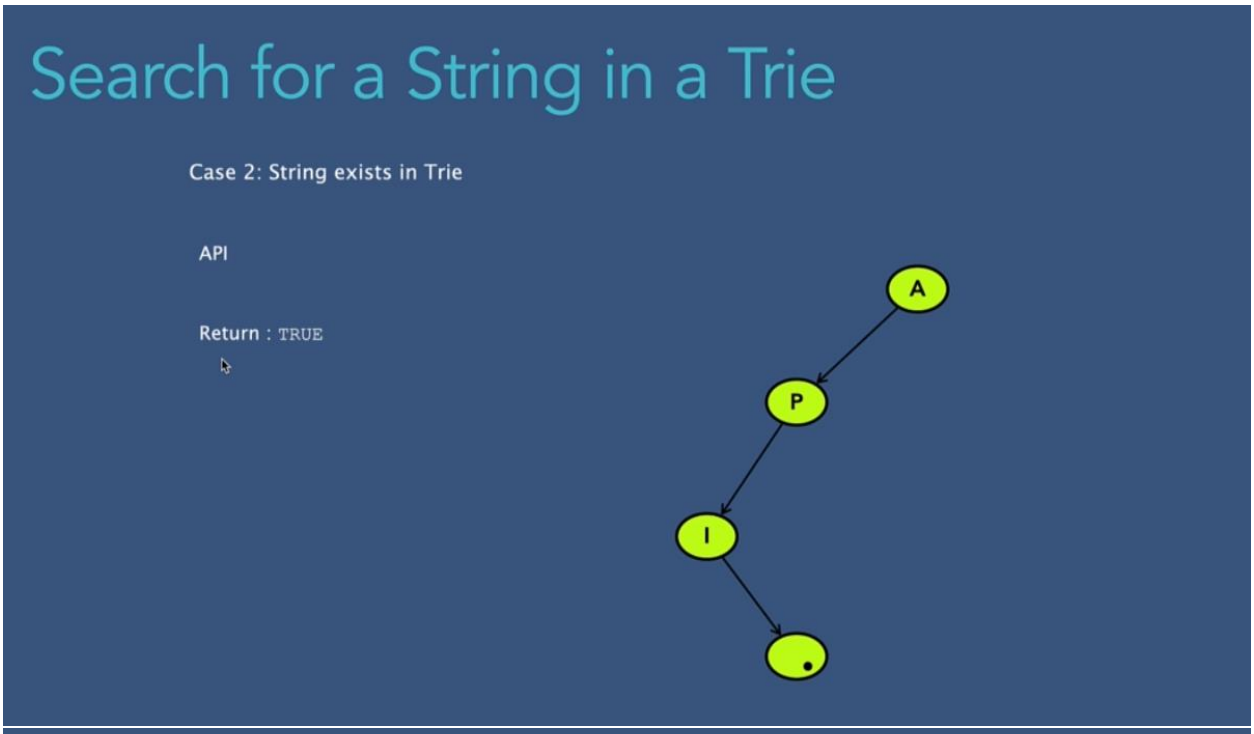
Insertion of the String “API”. Here, “A” and “P” are already present in the Trie and we only insert (append) character “I” to the third node in the Trie

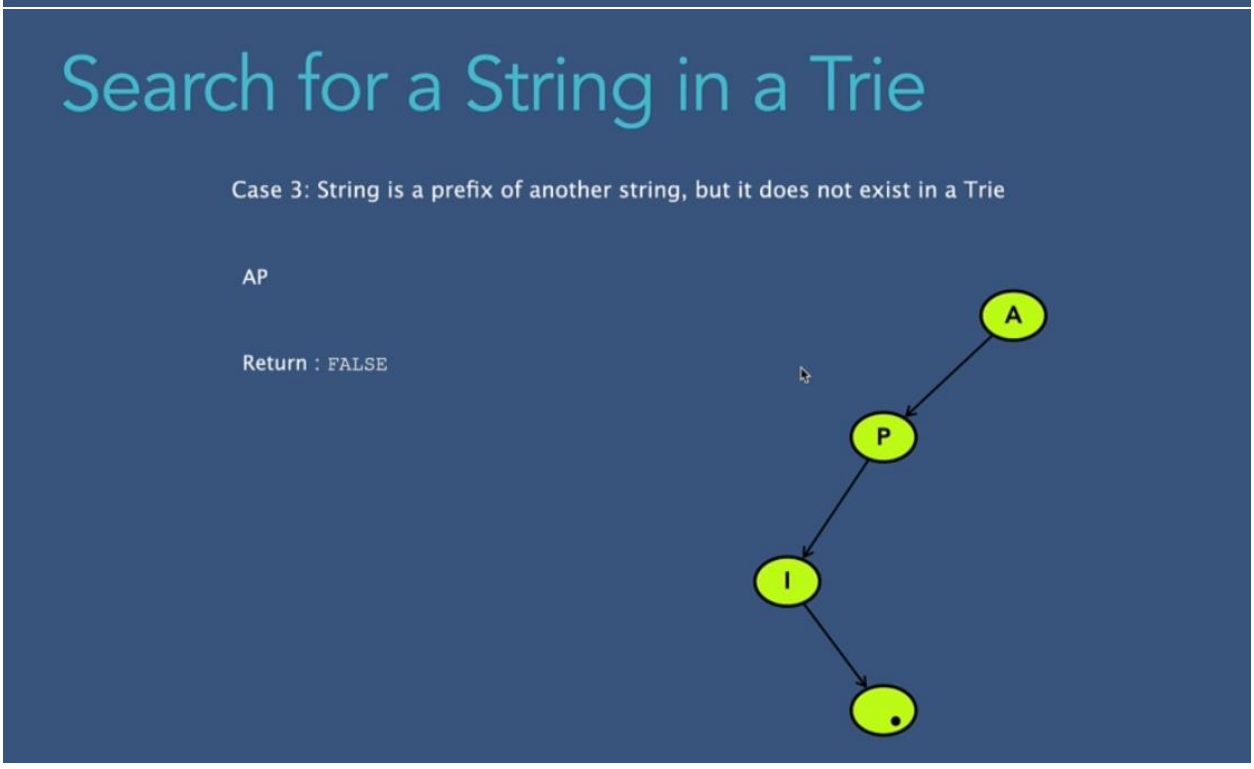


Insertion of the String “APIS”. Here, “A”,“P”,“I” are already present in the string so we only insert(append) “S” to the Trie Data Structure.



Here checking for the first character in the HashMap is enough as B is not present as the root node in the Trie so we return that the string does not exist in Trie.

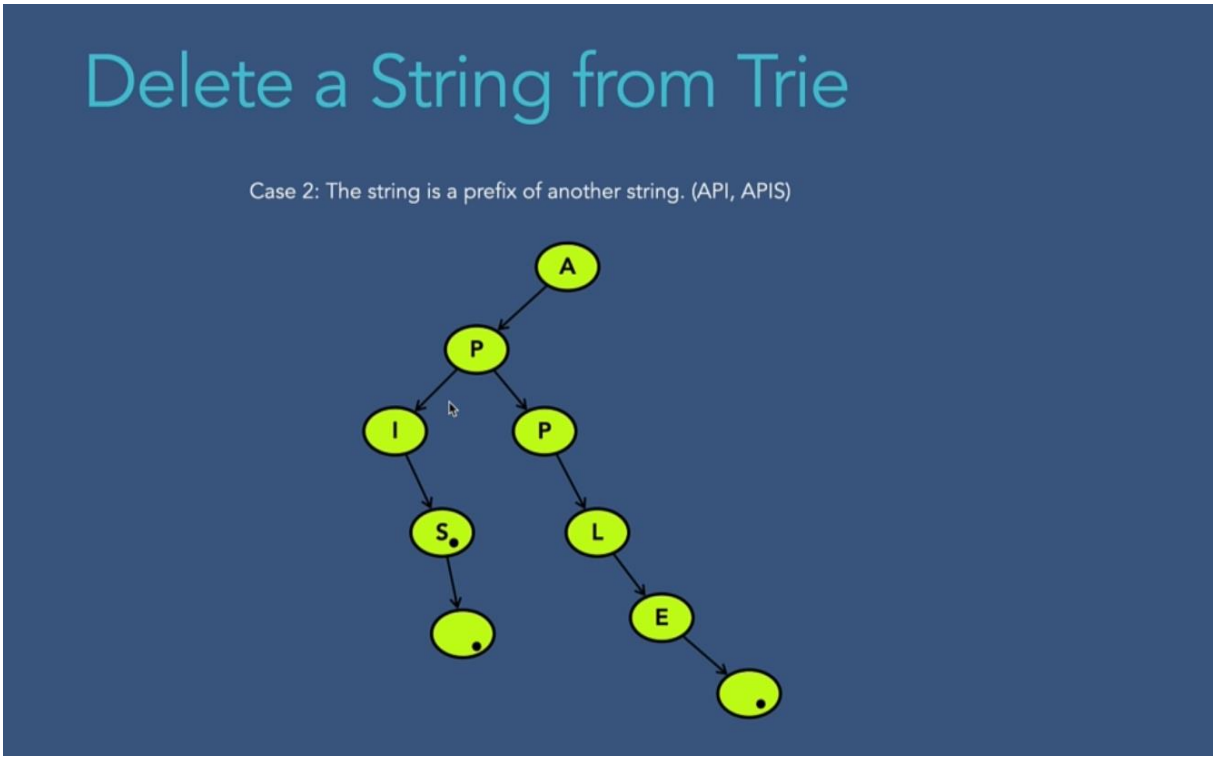




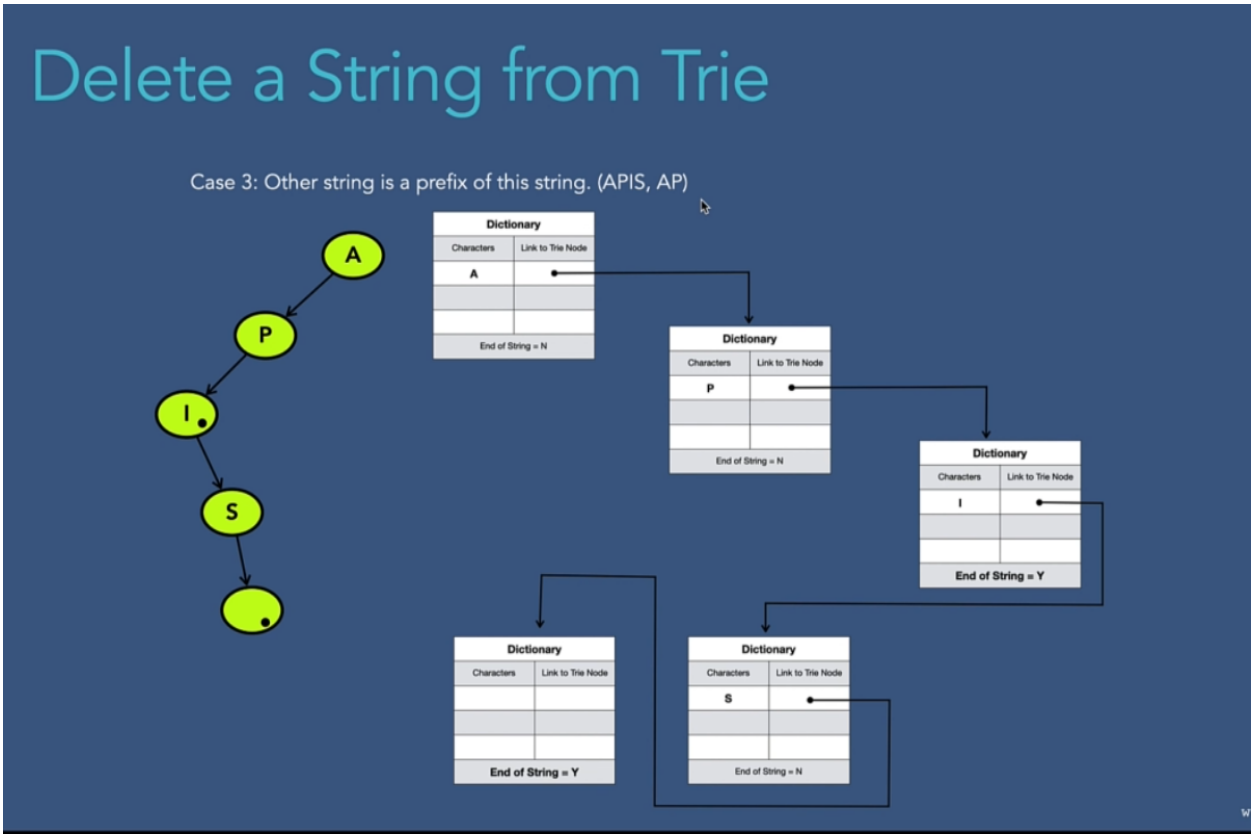
Here, we are returning false as only “A” and “P” is present in the Trie and it is prefix of the String “API“and we don’t find the End of String as Y (Which denotes YES) in the Trie. So, we return false.

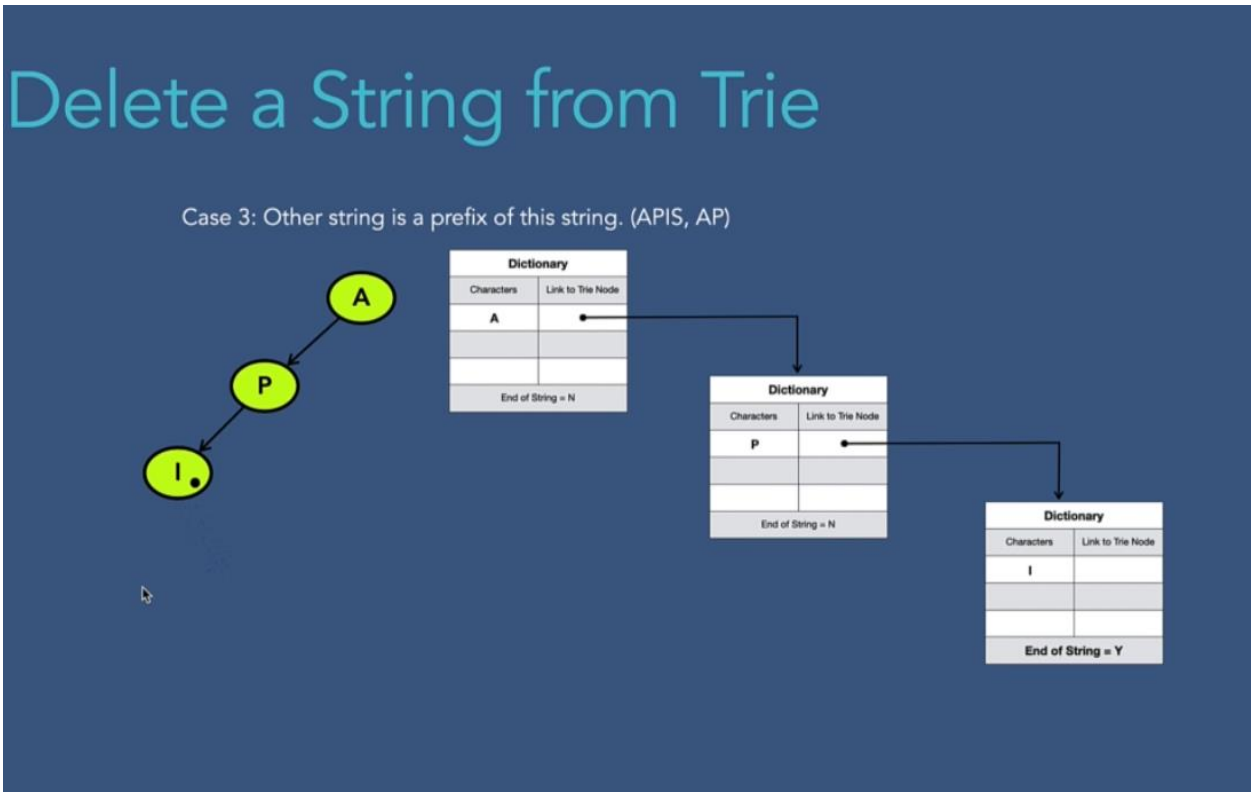
Deleting the API from the Trie. We first delete the End of the String as it does not depend on any other node and then we continue to the parent of the leaf node. And the parent node “I” is not needed for the other string so we can delete it.

After deleting of the End of String and the character “I” from the Trie. The other nodes “P” and “A” are required for the other string “APPLE” so we do not delete further.

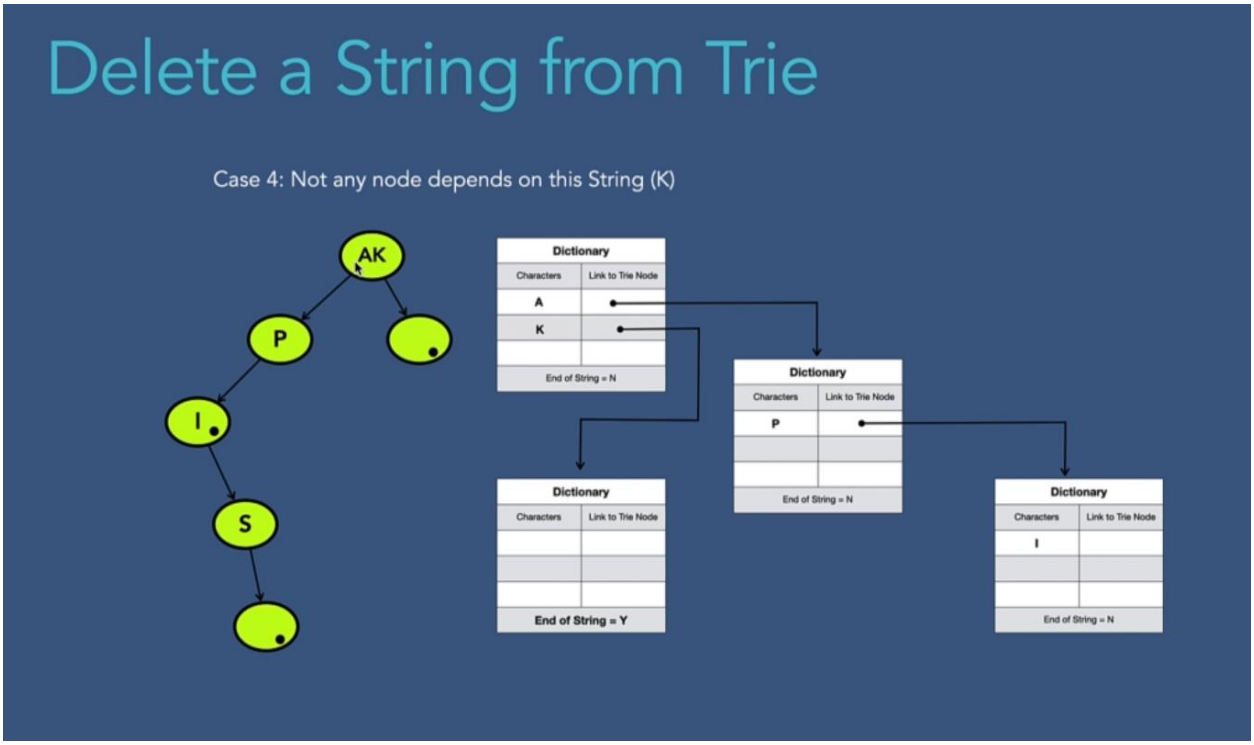


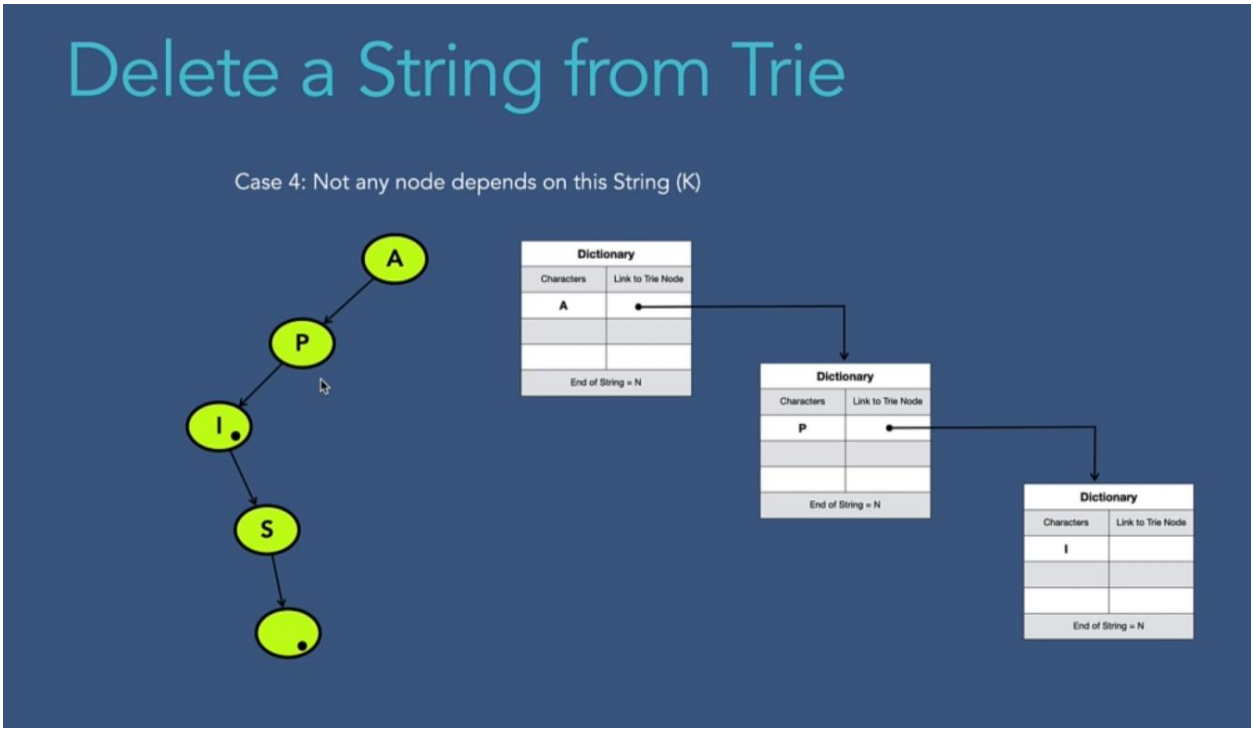
Our objective is to delete the String “API” from the Trie. The “API” is required for the String “APIS” so we only delete the End of String (“.”) for the String “API”.





We delete the End of String (“.”), the character “S” and the character “I” from the Trie.





Our Objective is to delete the String “K” from the Trie. This deletion process is simple. We delete K and the End of the String (“.”) which does not depend on any other node in the Trie.

Practical use of Trie

– Auto completion

– Spelling checker