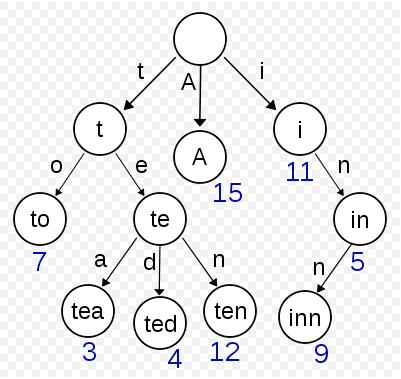
**Trie implementation Documentation**

A trie, or prefix tree, is a multi-way tree structure useful for storing strings over an alphabet. It is very efficient and handy and can be used for various purposes like auto-complete, spell-check, dictionary search, etc.

The **basic idea** of a Trie goes as below:

* We have a root node which represents an empty string.
* Every other node either represents a word or a prefix of one or more words.
* Each node can have atmost as many children as the size of the alphabet.



**Trie example**

The above diagram shows a trie which stores the words “A”, “to”, “tea”, “ted”, “ten”, “i”, “in”, and “inn”.

**Below is the list of function used and thier role is explaned :-**

**Trie = function()**  -- constructor of class Trie

**insert: function(str, pos)** --

\* Insert a word into the dictionary.

\* Recursively traverse through the trie nodes, and create new node if does not already exist.

\* @method insert

\* @param {String} str Word to insert in the dictionary

\* @param {Integer} pos Current index of the string to be inserted

\* @return {Void}

**remove: function(str, pos)** --

\* Remove a word from the dictionary.

\* @method remove

\* @param {String} str Word to be removed

\* @param {Integer} pos Current index of the string to be removed

\* @return {Void}

**update: function(strOld, strNew)**

\* Update an existing word in the dictionary.

\* This method removes the old word from the dictionary and inserts the new word.

\* @method update

\* @param {String} strOld The old word to be replaced

\* @param {String} strNew The new word to be inserted

\* @return {Void}

**countWord: function(str, pos)**

\* Count the number of times a given word has been inserted into the dictionary

\* @method countWord

\* @param {String} str Word to get count of

\* @param {Integer} pos Current index of the given word

\* @return {Integer} The number of times a given word exists in the dictionary

**countPrefix: function(str, pos)**

\* Count the number of times a given prefix exists in the dictionary

\* @method countPrefix

\* @param {String} str Prefix to get count of

\* @param {Integer} pos Current index of the given prefix

\* @return {Integer} The number of times a given prefix exists in the dictionary

**find: function(str)**

\* Find a word in the dictionary

\* @method find

\* @param {String} str The word to find in the dictionary

\* @return {Boolean} True if the word exists in the dictionary, else false

**getAllWords: function(str)**

\* Get all words in the dictionary

\* @method getAllWords

\* @param {String} str Prefix of current word

\* @return {Array} Array of words in the dictionary

**autoComplete: function(str, pos)**

\* Autocomplete a given prefix

\* @method autoComplete

\* @param {String} str Prefix to be completed based on dictionary entries

\* @param {Integer} pos Current index of the prefix

\* @return {Array} Array of possible suggestions