Trie

Ex. 1

```
package B3Trie;

public class A1TrieDataStructure {
	static class Node{
	   Node[] children;
	   boolean eow; //eow means end of word

	   public Node() {
		  children = new Node[26];  //a to z therefore size=26
		  for(int i=0;i<26;i++) {
			children[i]=null;
		  }
		  eow=false;
	   }
	}

	static Node root = new Node(); //root Node which is empty

	//Time compleity of Trie to insert single word is O(L) where L is Length of
Word

	public static void insert(String word) {
	   Node curr =root;
	   for(int i=0; i<word.length();i++) {  //Time complexity: O(L)
		  int idx = word.charAt(i)-'a';

		  if( curr.children[idx]==null) {
			curr.children[idx]= new Node();
		  }

			curr =  curr.children[idx];
	   }
	   curr.eow=true;
	}


	//Time complexity to Search word in Trie is O(L) where L is length of the word
	public static boolean search(String key) {
	   Node curr = root;
	   for(int i=0; i<key.length();i++) {
		  int idx = key.charAt(i) - 'a';

		  if( curr.children[idx]==null) {
			return false;
		  }
		  if(i == key.length()-1 &&  curr.children[idx].eow==false) {
			return false;
		  }

		  curr = curr.children[idx];
	   }

	   return true;
	}
```

```
	public static boolean wordBreak(String key) {
		if(key.length()==0) {
			return true;
		}

		for(int i=1;i<=key.length();i++) {
			String firstPart = key.substring(0,i);
			String SecondPart = key.substring(i);
			if(search(firstPart) && wordBreak(SecondPart)) {
				return true;
			}
		}

		return false;
	}

	public static boolean IsStartsWith(String pref) {
		Node curr = root;
		for(int i=0;i<pref.length();i++) {
			int idx = pref.charAt(i)-'a';
			if(curr.children[idx]==null) {
				return false;
			}
			curr=curr.children[idx];
		}
		return true;
	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub
//		String words[]= {"the", "a", "there", "their", "any"};
//		for(int i=0;i<words.length;i++) {
//		insert(words[i]);
//		}
//		System.out.println("Is word is present: "+search("their")); //Ans: true
//		System.out.println("Is word is present: "+search("thor")); //Ans: false
//		System.out.println("Is word is present: "+search("an")); //Ans: false

		//Word Break Problem
//		String wordsBreakArr[]= {"i","like","sam","samsung","mobile"};
//		String key="ilikesamsung";
//		for(int i=0;i<wordsBreakArr.length;i++) {
//			insert(wordsBreakArr[i]);
//		}
//		System.out.println(wordBreak(key)); //Ans:true

		//StartsWith Problem que. Check prefix is present in word or not in
given words
		String startsWithWords[] = {"apple","app","mango","man","woman"};
		String prefix="app";
		for(int i=0;i<startsWithWords.length;i++) {
			insert(startsWithWords[i]);
		}
		System.out.println(IsStartsWith(prefix)); //true
```

```
	}

}
```

Ex.2

```
package B3Trie;
//count unique substrings of word here word="ababa" and ans is count=10
public class A2CountUniqueSubstrings {
   static class Node{
          Node[] children;
          boolean eow;
          public Node() {
               children = new  Node[26];
               for(int i=0;i<26;i++) {
                     children[i]=null;
               }

               eow=false;
          }
   }

   static Node root = new Node();

   public static void insert(String word) {
          Node curr=root;
          for(int i=0;i<word.length();i++) {
                int idx=word.charAt(i)-'a';
                if(curr.children[idx]==null) {
                      curr.children[idx]= new Node();
                }
                      curr=curr.children[idx];

          }
          curr.eow=true;
   }

  public static int countNoOfNodes(Node root ){
          if(root==null) {
                return 0;
          }
         int count=0;
          for(int i=0;i<26;i++) {
                if(root.children[i]!=null) {
                     count += countNoOfNodes(root.children[i]);
                }
          }
          return count+1;
   }
```

Trie

```
	public static void main(String[] args) {
		// TODO Auto-generated method stub
	 String word="apple";
	 for(int j=0; j<word.length(); j++) {
	    String suffix = word.substring(j);
	    insert(suffix);
	 }

	 System.out.println("Count No of Nodes=No of substrings:
"+countNoOfNodes(root));  //Ans:10
	}

}
```

Ex.3

```
package B3Trie;

public class A3LongestWordWithAllPrefix {

	static class Node{
		Node[] children;
		boolean eow;
		public Node() {
			children = new Node[26];
			for(int i=0;i<26;i++) {
				children[i]=null;
			}
			eow=false;
		}
	}

	static Node root = new Node();

	public static void insert(String word) {
		Node curr=root;
		for(int i=0;i<word.length();i++) {
			int idx = word.charAt(i)-'a';
			if(curr.children[idx]==null) {
				curr.children[idx]=new Node();
			}
			curr = curr.children[idx];
		}
		curr.eow=true;
	}

	static String finalAns="";

   public static void longestPrefixWord(Node root,StringBuilder temp ){
```

Trie

```
        if(root==null) {
              return;
        }
      for(int i=0;i<26;i++) {
            if(root.children[i]!=null && root.children[i].eow==true) {
                  temp.append((char)(i+'a'));
                  if(temp.length()>finalAns.length()) {
                        finalAns = temp.toString();
                  }
                  longestPrefixWord( root.children[i], temp );
                  temp.deleteCharAt(temp.length()-1);
            }
      }
    }

    public static void main(String[] args) {
          // TODO Auto-generated method stub
          String words[]= {"a","banana","app","appl","ap","apply","apple"};

          for(int i=0;i<words.length;i++) {
                insert(words[i]);
          }
          StringBuilder  tempStr= new StringBuilder("");
          longestPrefixWord(root,tempStr);

          System.out.println("Longest prefix String:"+finalAns); //Ans: apple

    }

}
```