Tries

```
public class TrieNode {
	char data;
	boolean isTerminal;
	TrieNode[] children;
	public int childCount;
	public TrieNode(char data){
		this.data=data;
		this.isTerminal=false;
		this.children=new TrieNode[26];
		this.childCount=0;
	}
}

public class Trie {
	private TrieNode root;

	public Trie() {
		root = new TrieNode('\0');
	}

	// 1st method
	public void add(String word) {
		addHelper(root, word);
	}

	// helper for add method
	private void addHelper(TrieNode root, String word) {
		if (word.length() == 0) {
			root.isTerminal = true;
			return;
		}
		int childIndex = word.charAt(0) - 'a';
		TrieNode child = root.children[childIndex];
		if (child == null) {
			child = new TrieNode(word.charAt(0));
			root.children[childIndex] = child;
			root.childCount++;
		}
		addHelper(child, word.substring(1));
	}

	// 2nd method
	public boolean search(String word) {
		return searchHelper(root, word);
	}

	// helper method for search
	private boolean searchHelper(TrieNode root, String word) {
		if (word.length() == 0) {
			return root.isTerminal;
		}
		int childIndex = word.charAt(0) - 'a';
		TrieNode child = root.children[childIndex];
		if (child == null)
```

```java
                return false;
            return searchHelper(child, word.substring(1));
        }

        // 3rd method
        public void remove(String word) {
            removeHelper(root, word);
        }

        // helper method for remove
        public void removeHelper(TrieNode root, String word) {
            if (word.length()==0){
                root.isTerminal=false;
                return;
            }
            int childIndex=word.charAt(0)-'a';
            TrieNode child=root.children[childIndex];
            if (child==null)
                return;
            removeHelper(child,word.substring(1));
            if (!child.isTerminal && child.childCount==0){
                root.children[childIndex]=null;
                root.childCount--;
            }
        }
    }
```