

## TRIES SOLUTIONS

Solution 1:

We solve using Tries.

```
iclass TrieNode {
    List<String> data;
    TrieNode children[];
    boolean isEnd;

    TrieNode () {
        data = new ArrayList<>();
        children = new TrieNode[26];
        isEnd = false;
    }
}

class Solution {
    static TrieNode root;
    List<List<String>> ans;
    public List<List<String>> groupAnagrams(String[] strs) {
        ans = new ArrayList<>();

        root = new TrieNode();

        for(String word: strs){
            build(word);
        }

        dfs(root);

        return ans;
    }

    public void build(String s){
        TrieNode temp = root;
        char[] word = s.toCharArray();
        Arrays.sort(word);
        for(char c: word){


```

er.dm.zer03@gmail.com

```
        TrieNode child = temp.children[c-'a'];
        if(child == null) temp.children[c-'a'] = new TrieNode();
        temp = temp.children[c-'a'];
    }

    temp.isEnd = true;
    temp.data.add(s);
}

public void dfs(TrieNode rt){
    if(rt.isEnd){
        ans.add(rt.data);
    }

    for(int i = 0; i < 26; i++){
        if(rt.children[i] != null) dfs(rt.children[i]);
    }
}
}
```

## Solution 2:

```
class Solution {

    private static class Node {
        private char data;
        private String word;
        private boolean isEnd;
        private Node[] children;

        public Node(char data) {
            this.data = data;
            this.word = null;
            this.isEnd = false;
            this.children = new Node[26];
        }
    }

    private Node root = new Node('/');
    private String ans = "";
}
```

er.dm.zer03@gmail.com

```
private void insert(String word) {  
    Node curr = this.root;  
  
    for (int i = 0; i < word.length(); i++) {  
        int childIdx = word.charAt(i) - 'a';  
  
        if (curr.children[childIdx] == null) {  
            curr.children[childIdx] = new Node(word.charAt(i));  
        }  
  
        curr = curr.children[childIdx];  
    }  
  
    curr.isEnd = true;  
    curr.word = word;  
}  
  
private void dfs(Node node) {  
    if (node == null) {  
        return;  
    }  
  
    if (node.word != null) {  
        if (node.word.length() > ans.length()) {  
            ans = node.word;  
        } else if (node.word.length() == ans.length() &&  
node.word.compareTo(ans) < 0) {  
            ans = node.word;  
        }  
    }  
  
    for (Node child : node.children) {  
        if (child != null && child.word != null) {  
            dfs(child);  
        }  
    }  
}  
  
public String longestWord(String[] words) {  
    for (String word : words) {  
        insert(word);  
    }  
  
    Node curr = this.root;  
    dfs(curr);  
    return ans;  
}
```

er.dm.zer03@gmail.com

}

}

**APNA**  
**COLLEGE**

er.dm.zer03@gmail.com