2002 Sun Microsystems and TopCoder Collegiate Challenge – Problem Statement

QuickBrownFox PROBLEM STATEMENT

Given a set of words, find a subset such that each letter from the alphabet is present in at least one word from the subset. When multiple such subsets exist, your method should return the one with the smallest combined length (i.e. the number of characters you get when you add up lengths of all words in the subset). See notes for additional instructions on breaking ties.

DEFINITION
Class: QuickBrownFox
Method name: smallest
Parameters: String[]
Returns: String[]
Method signature (be sure your method is public): String[] smallest(String[] words);

words specifies the words in the set. Your method should return its subset (which may include all words) that covers the entire alphabet. The returned words must be arranged in the same relative order as the input.

INPUT CONSTRAINTS
TopCoder will verify that all input constraints are met.
- words contains 0 to 20 elements, inclusive.
- Each word is composed of lowercase characters 'a' though 'z', inclusive.
- All words are 1 to 50 characters in length, inclusive.
- String[] words contains no duplicate entries.

NOTES
- In cases when multiple subsets of the same combined length cover the entire alphabet, your method should return the earliest subset in "binary counting" order. For each element of the original set, put '1' (one) if the subset contains the element, and '0' if the subset does not contain the element. The first element of the String[] corresponds to the least significant bit, and the last element corresponds to the most significant bit. For example, if the set is {"a","b","c","d"}, the binary string of 0101 corresponds to the subset {"a","c"}. To decide which subset comes first in the binary counting order, compare the integer values of the resulting binary strings, and return the subset that corresponds to the smaller integer value (see example 4).
- In cases when there are no subsets covering the entire alphabet, even when you use all words, your method should return a String[] with no elements.

EXAMPLES
1.
words={"the","quick","brown","fox","bear","cow","jumps","over","lazy","dog","cat","chicken"}. Your method should return
{"the","quick","brown","fox","jumps","over","lazy","dog"}.
2. words={"not","enough","words"}. Your method should return {}.
3.
words={"abb","cdd","eff","ghh","ijj","kll","mnn","opp","qrr","stt","uvv","wxx","yzz","ab","cd","ef","gh","ij","kl","mn"}. Your method should return {"opp", "qrr", "stt", "uvv", "wxx", "yzz", "ab", "cd", "ef", "gh", "ij", "kl", "mn"}. This example shows that your method should prefer subsets with smaller total character counts to subsets with larger total character count.
4.
words={"abcdefghijk","abcdefg","a","hijklmnopqrstuvwxyz","bcdefghijklmnopqrstuvwx

yz","lmnopqrstuvwxyz"}. There are three subsets that cover the entire alphabet, and all three have the same combined length:
("abcdefghijk","lmnopqrstuvwxyz") --> "100001",
{"a","bcdefghijklmnopqrstuvwxyz"} --> "010100", and
{"abcdefg","hijklmnopqrstuvwxyz"} --> "001010" (note how earlier words correspond to less significant bits of the binary string).
Your method should return {"abcdefg", "hijklmnopqrstuvwxyz"}, because the integer value of its binary string is the smallest.