2002 Sun Microsystems and TopCoder Collegiate Challenge – Problem Statement

SetPartition PROBLEM STATEMENT

Set partitions divide elements A, B,..,<N-th letter> into non-empty subsets. For example, when N=4 (the set is ABCD), there are fifteen distinct partitions: {ABCD}, {ABC,D}, {ABD,C}, {AB,CD}, {AB,C,D}, {ACD,B}, {AC,BD}, {AC,B,D}, {AD,BC}, {A,BCD}, {A,BC,D}, {AD,B,C}, {A,BD,C}, {A,B,CD}, and {A,B,C,D}.
One way to define a partition is through a partitioning string. Each element of a partitioning string specifies the number of the subset to which the corresponding element of the set goes. For example, partitioning string {0, 1, 1, 2, 1} specifies the {A,BCE,D} partition of ABCDE: the first position specifies the subset number for A, the second position specifies the subset number for B, and so on. Therefore, A goes to subset 0, B,C, and E go to subset 1, and D goes to subset 2.
For a string to be a valid partitioning string, all its elements must be non-negative, its initial element must be 0, and the following limiting relation must hold for all i > 0 : Ai <= 1 + max(A0..Ai-1). For example, {1,0} is not a valid partitioning string because it does not start with 0, {0,-1} is invalid because it has negative numbers, and {0,3,1,2} is invalid because its second element violates the limiting relation. Note that there is a one-to-one correspondence between the partitioning strings and the set partitions.
You can order the partitions by ordering their corresponding partitioning strings. A natural order for strings is lexicographic, like words in a dictionary. For example, in lexicographic order {0,0,1,2} comes before {0,1,0,0}, but after {0,0,1,1}. If you order all possible partitioning strings of length N, {0, 0, 0, ..., 0} would be the first, and {0, 1, 2, ..., N-1} would be the last partitioning string. The partitions of ABCD in the example at the top of the problem are given in lexicographic order of their corresponding partitioning strings.
Write a method that, given a set partition, finds the set partition corresponding to the next partitioning string.

DEFINITION

Class Name: SetPartition
Method Name: nextPartition
Parameters: String[]
Returns: String[]

Method signature (be sure your method is public): String[] nextPartition(String[] partition);

TopCoder will ensure that:
- partition has between 1 and 25 elements, inclusive,
- Each element of the partition has between 1 and 26 elements, inclusive,
- Each element of the partition consists only of characters 'A' through 'Z', inclusive,
- Elements of partition and characters inside each element are sorted alphabetically in ascending order (this ensures that the partitioning string of the input partition is valid),
- There are no duplicate characters in the partition (this rule works across all elements),
- If a character <ch> is listed in an element of the partition, all characters from 'A' to <ch>, inclusive, are also listed, possibly in another element (this ensures that there are no gaps in the initial set),

- At least one element of the partition has two or more characters (this ensures that the next lexicographic partitioning string exists).

EXAMPLES
1. partition={"AB","C","D"}. The corresponding partitioning string is {0,0,1,2}; the next partitioning string in lexicographic order is {0,1,0,0}; your method should return {"ACD","B"}.
2. partition={"ABC","DEF"}. The corresponding partitioning string is {0,0,0,1,1,1}; the next partitioning string is {0,0,0,1,1,2}; your method should return {"ABC","DE","F"}.
3. If partition={"ADFHKM", "BZ", "CXY", "EOPVW", "GLN", "IJSTU", "Q", "R"}, your method should return {"ADFHKM", "B", "CXYZ", "EOPVW", "GLN", "IJSTU", "Q", "R"}.
4. If partition={"A","B","C","D","E","FG"}, your method should return {"A","B","C","D","E","F","G"}.
5. If partition={"ABCDEFG"}, your method should return {"ABCDEF","G"}.