SetPartition:

This problem is somewhat easier than a typical medium at this level, but only if attacked properly. The key to a solution is to use the partition string format to order the possible results.

First, get a translation between partitions of the style { "A", "BC", "D" } and partitions of the style { 0, 1, 1, 2 }. Once you have this in hand, you merely need to find the "smallest" partition string that is greater than the input partition string and is legal.

We can do this easily if we treat the partition string as a base-N number (where N is the 1-based alphabet index of the highest letter found in the input). If the input is { 0, 1, 1, 2 } then we can represent the partition as the number 0112 in base 4. The solution is guaranteed to exist (by the input limitations) and the solution will be one of 0113, 0120, 0200, which are formed by incrementing the 4th, 3rd and 2nd digits, respectively and assigning the value 0 to any digit to the right of the incremented digit. The correct solution is the first potential solution that is legal. 0113 is not legal since 3 > max(0, 1, 1) + 1 = 2. 0120 is legal, however, and corresponds to the partition { "AD", "B", "C" }.

Hence, a solution to this problem is given by

1. Translating the input to the "partition string" format.
2. Incrementing each digit of the partition string in order from lowest digit to highest digit until a legal solution is found.
3. Translating the partition string back to the standard format.