HiddenSeq PROBLEM STATEMENT

Secret messages may appear in text as follows. Starting at a given offset into
the text, examine each character by repeatedly skipping ahead a fixed number of
characters. For example, the text "a fresh red lollipop" contains the secret
message "hello". The message "hello" is found by starting at the zero based
offset 6 and taking every 3rd letter.

```
......*..*..*..*..*.
a fresh red lollipop
```

For the purposes of this problem, a secret message is defined as the longest
sequence of alphabetically ascending, consecutive letters, that can be found in a
given piece of text (for example, "abc", "mnopq", "xyz"). To make things more
interesting, the initial piece of text may repeat itself as often as necessary to
find the longest sequence. For example, the initial text "hello there", should be
treated as if it repeats itself (as in "hello therehello therehello there...").
See example E7.

The text may contain space characters. Space characters are significant and are
always counted when determining intervals. Space characters, however, are treated
specially when determining sequences. Though a space character may not begin a
sequence, a space character neither terminates a sequence nor counts towards the
total length of a sequence. Rather, the sequence simply continues skipping ahead
by the interval number of characters. For example, the text "mxnx xox x xp"
contains the sequence "mnop" of length 4 starting at offset 0 using every 2nd
character:

```
*.*.*.*.*.*.*
mxnx xox x xp
```

Write a method that, given a string representing a piece of text, finds the
longest hidden sequence of alphabetically consecutive characters of at least two
characters in length. Return the result in a int[] of three numbers representing
the zero based starting offset of the first character in the sequence, the
interval value used, and the total length of the sequence
({offset,interval,length}). If no sequence of length 2 or greater is found,
return a zero for each of the three values in the result ({0,0,0}). If there is
more than one hidden sequence having the longest length, select the final result
by using the following rules in order (see examples E0, E1, and E2):

R1: the alphabetically (dictionary order) lowest sequence (for example, "abc" is
lower than "bcd")
R2: the sequence with the smallest interval value
R3: the sequence with the smallest offset value

DEFINITION
Class: HiddenSeq
Method: findSequence
Parameters: String
Returns: int[]
Method signature: int[] findSequence(String text);
(be sure your method is public)

TopCoder will ensure the validity of the inputs. Inputs are valid if all of the
following criteria are met:
* text is from 0 to 50 characters in length inclusive
* text consists only of the space character and the lowercase letters 'a' through
'z' inclusive

EXAMPLES
E0: "aabbbcdc" ==> {1,2,3}
This example contains several sequences of length 2. At offset 1 using an
interval of 2 the sequence of length 3 "abc" is found, and again "abc" is found
starting at offset 1 using an interval of 3. Starting at offset 4 using an
interval of 1 the sequence "bcd" is found. So there are three sequences having
the longest length of 3. Rule R1 is applied first and eliminates the sequence
"bcd". Rule R2: is applied next and selects the sequence using the interval of 2
rather that the sequence using the interval of 3. And so the final result is the
sequence "abc" starting at offset 1, using an interval of 2, having length 3.

E1: "z abc" ==> {2,1,3}
The offset is 2 because a space cannot start a sequence)

E2: "aabbcc" ==> {0,2,3}
The sequence {1,2,3} has the same length, is alphabetically equivalent, has the
same interval value, but the returned sequence is the one with the smaller
offset.

E3: "" ==> {0,0,0}
E4: "a" ==> {0,0,0}
E5: "ace" ==> {0,0,0}
E6: "abc" ==> {0,1,3}
E7: "cba" ==> {2,2,3}
E8: "fhjbd ik eg lc" ==> {3,5,11}
E9: "better study this example" ==> {5,1,4}
E10: "bridge freezes before road surface" ==> {3,2,3}
E11: "the quick brown fox jumps over the lazy dog again" ==> {40,37,3}