Finding common characters of two Strings in alphabetical /lexicographical order :

You are given two strings. You have to find and print all common characters in lexicographical order(alphabetical order). If there are no common letters print -1.
All letters are in lower case.

Let S1 = "maxinterview" and S2 = "internship"
The output should be "eiinrt"

Explanation :
The common alphabets scanning from left to right are i, n, t, e, r ;
where i appears twice.

Don't confuse it with LCS(longest common substring).
LCS would return the string "inter".

Method 1 : Brute Force or HIT & TRIAL method

1) Choose the smallest of the two strings. Create an array to store the occurrence of the letters alphabetically for the string.
2) Next use two loops. The outer loop runs from 0 to length of longer string and the inner loop runs from 0 to length of shorter string.
3) For each iteration of the outer loop we check all the characters of the smaller string and upon finding a match we increment a local counter.
Upon exiting from the inner loop we check if this counter value is equal to the value of the array of occurrences.

PSEUDOCODE:

```
//choose smaller string s2
for i = 0 to s2.length
    occ[i] = s2[i] - 'a'    //storing occurrences alphabetically
for i = 0 to s1.length
    counter = 0          //local counter
    for j = 0 to s2.length
        if s1[i] = s2[j]
            counter++
    end for
    if occ[i] = counter
```

```
        for k = 1 to counter //printing the common character
            print s1[i]
end for
```

The time complexity of this program would be O(mn), where m is the length of the string s2 and n is the length of the string s1.
The space complexity would also be O(N) as we are using an extra array to store the occurrences of the alphabets.

Method 2 : An asymptotically better solution:

Count occurrences of all characters from ‘a’ to ‘z’ in first and second strings.
Store these counts in two arrays occ1[] and occ2[].
Size of both occ1[] and occ2[] is 26.
Traverse occ1[] and occ2[]. For every index i, print character ‘a’ + i number of times equal min(occ1[i], occ2[i]).

Explanation :**Why min(occ1[i], occ2[i])**?
As we are storing the occurrences of all the alphabets of both the strings in two arrays. The maximum number of common occurrences of a particular character would be the corresponding minimum of the two array values occ1[i] and occ2[i].

Suppose, for the iteration i =3,
occ1[3] = 3 and occ2[3] = 2.
Then, min(occ1[3], occ2[3]) = 2, which is true as occ1[3] = 3 means that the character occurs thrice in string s1 and oc21[3] = 2 means that the character occurs twice in string s2. Thus the common occurrences would be two times.

PSEUDOCODE:

```
length1 = s1.length()
length2 = s2.length()
for i = 0 to s1.length
    occ[i] = s1[i] - ‘a’    //storing occurrences of string s1 alphabetically
for i = 0 to s2.length
    occ[i] = s2[i] - ‘a’    //storing occurrences of string s2 alphabetically
```

```
for i = 0 to occ1[].size        //occ1[].size = occ2[].size = 26
    if occ1[i] !=  0 and occ2[i]  !=  0
        //find the min of the occurrences and print that many times
        for j = 0 to min(occ1[i], occ2[i])
            print (i + 'a')
        end for
    dnd if
end for
```

The space complexity of this code would be O(N) as we are using extra arrays to store the occurrences, but the time complexity would be O(n), where n is the length of the larger string. This is **asymptotically much better** than the previous method where the running time was of the order of product of lengths of both the strings.