In this article, we discuss a new problem of lexicographical order. Our aim is to print the lexicographically smallest string formed by using the input string after removing at most one character.

Lexicographical arrangement of string means the characters of the string are in sequence order. In this question, we have to find the lexicographically smallest string formed by removing at most one character from the given string. A lexicographically smallest string is a string which comes first when we sort the characters of the string in alphabetical order. This can be done by finding and removing the character which is greater than its immediate next character. Let’s first understand the question in an understandable way so that you can think to solve it with your own knowledge as well. 

Here are some examples to understand the question.

**Example 1:**

Suppose, the input string is “asdfghj”. Now, according to the question we have to remove at most one character from the given string and then return the output which should be in the lexicographically smallest string. So in this example ‘s’ character should be deleted to get the output string.

So, the output string is “adfghj”.

**Example 2:**

Let’s take another example to understand this problem statement. here , the input string is “abcda”. Now, try to solve this question and find its output. In this example the ‘d’ character should be deleted to get the output string.

So, the output string is “abca”.

**Example 3:**

If in the input string all the characters are the same then remove a character from the string which is our output string.

Input string: “aaaaa”

Output string: “aaaa”

Now, the problem statement is clear to you. Let’s move to solve this problem statement and code it in c++ language.

**Approach 1: Non-recursive Approach**

Let’s see the algorithm, explanation and program for this approach.

**Algorithm**

**Step 1:** Create a function named smallest which takes the input string and returns the output string.

**Step 2:** Using a for loop traverse through all the characters of the string.

**Step 3:** Check for the condition if s[i] > s[i + 1].

**Step 4:** If it happens, break the loop and erase this character from the string.

**Step 5:** After removing that character now our string is lexicographically the smallest string.

**Step 6:** Return the output string.

**Explanation**

In this algorithm we traverse through the input string and if any character which comes later in the order but comes before, then we delta that character and print the result. The time complexity of this approach is O(n). This is the basic simple approach to solve this problem statement. We can call it a non-recursive approach.

**Program**

|  |
| --- |
| #include <iostream>  using namespace std;   // Function to return the smallest string  string smallest(string s) {      int i = 0, s\_size = s.size();      string result(s);      for (; i < s\_size - 1; ++i) {           //check for the first point where s[i]>s[i+1]          if (s[i] > s[i + 1]) {              break;          }      }      result.erase(i, 1);      return result;  }  int main()  {  string s = "abcda";  cout << smallest(s);  return 0;  } |

**Output**

|  |
| --- |
| abca |

We discuss a simple approach now. Let's try to solve it with recursion using the same logic.

**Approach 2: Recursive Approach**

**Algorithm**

**Step 1:** Create a recursive function smallest which takes input as a string and returns a string which is lexicographically smallest string.

**Step 2:** It checks the conditions using the if statement.

**Step 3:** The function named smallest calls itself again and again to get the desired result.

**Step 4:** Then it returns the final string to the output.

**Explanation**

In this approach, we use a recursive function to find the lexicographically smallest string from the given input string. Here the function named smallest calls itself again and again to get the desired result. Then return the result to the main function. This approach is also called a recursive approach as in this algorithm the function calls itself again and again to get the job done.

**Program**

|  |
| --- |
| #include <iostream>  using namespace std;    // This function takes a string s and an integer i as inputs.  string smallest(string s, int i) {      if (i == 0){          s.erase(i, 1);          return s;      }      // If the character at index i is smaller than the character at index i-1,      // delete the character at index i-1 from the string s.      if (s[i] < s[i - 1]) {          s.erase(i - 1, 1);          return s;      }        // If neither of the above conditions is true, recursively call the smallest      // function with the string s and i-1 as inputs.      return smallest(s, i - 1);  }  int main()  {  string s = "abcda";  cout << smallest(s, s.size() - 1);  return 0;  } |

**Output**

|  |
| --- |
| abca |

Hope, it's clear the problem statement and you  also learn to solve this question.

**Conclusion**

In this article we see one more important question of lexicographical problems. We learn how to find the lexicographically smallest string formed by removing at most one character. We solve this problem and code  this problem in c++ language.