package com.iimtiaz.day\_07;  
  
import java.util.Collections;  
import java.util.HashMap;  
import java.util.Map;  
  
public class ShortCharacter {  
 public static void main(String[] args) {  
 String s = "tree";  
 System.*out*.println(new Solution\_1().frequencySort(s));  
  
 }  
}  
  
*/\*\*  
 \* Time Complexity: O(n log n)  
 \* O(n): Iterating through the string characters once to count frequencies.  
 \* O(n log n): Sorting the Map.Entry objects based on their values using a stream and Collections.reverseOrder.  
 \* O(n): Iterating through the sorted entries and appending characters to the StringBuilder.  
  
 \* Space Complexity: O(n)  
 \* O(n): Stores a character-frequency pair for each unique character in the string.  
 \* O(n): Sorted entries stream: The stream likely uses a temporary data structure to hold the sorted entries,  
 \* contributing to space complexity.  
 \* O(n): StringBuilder: Holds the final sorted string, which can be up to the same size as the input string  
 \*/*class Solution\_1 {  
 public String frequencySort(String s) {  
 if (s.length() < 3) return s;  
 StringBuilder sb = new StringBuilder();  
 Map<Character, Integer> map = new HashMap<>();  
 for (char c : s.toCharArray()) {  
 map.merge(c, 1, Integer::*sum*);  
 }  
 map.entrySet().stream()  
 .sorted(Collections.*reverseOrder*(Map.Entry.*comparingByValue*()))  
 .forEach(entry -> {  
 for (int i = 0; i < entry.getValue(); i++) {  
 sb.append(entry.getKey());  
 }  
 });  
 return sb.toString();  
 }  
}