830. String Sort

* [Description](http://lintcode.com/en/problem/string-sort/" \l "description)
* [Notes](http://lintcode.com/en/problem/string-sort/#note)
* [Testcase](http://lintcode.com/en/problem/string-sort/#testcase)
* [Judge](http://lintcode.com/en/problem/string-sort/#judge)

Given a string, sort the string with the first keyword which is the most commonly used characters and the second keyword which is the dictionary order.

 Notice

* The length of string is less than 10000.
* All characters are lowercase

Have you met this question in a real interview?

Yes

**Example**

Given str = “bloomberg”, return “bbooeglmr”.

Explanation:

'b' appears the most frequently, and the dictionary sequence is the smallest, so it is ranked first, followed by 'o'.

Given str = “lintcode”, return “cdeilnot”.

Explanation:

All characters appear the same number of times, so directly in accordance with the dictionary order.

<http://lintcode.com/en/problem/string-sort/#>

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package javaapplication1;

import java.util.\*;

public class JavaApplication1 {

public static <K, V extends Comparable<? super V>> Map<K, V> sortByValue(Map<K, V> map) {

List<Map.Entry<K, V>> list = new LinkedList<>(map.entrySet());

Collections.sort(list, (Map.Entry<K, V> e1, Map.Entry<K, V> e2) -> (e2.getValue()).compareTo(e1.getValue()));

Map<K, V> result = new LinkedHashMap<>();

list.forEach((entry) -> {

result.put(entry.getKey(), entry.getValue());

});

return result;

}

public static String stringSort(String str) {

// Write your code here

HashMap<Character, Integer> hm = new HashMap();

for(int i =0; i<str.length(); i++) {

if(!hm.containsKey(str.charAt(i))) {

hm.put(str.charAt(i), 1);

} else {

hm.put(str.charAt(i), hm.get(str.charAt(i))+1);

}

}

Map<Character, Integer> ordenado = sortByValue(hm);

//printMap(ordenado);

ArrayList<ArrayList<Character>> matriz =

new ArrayList();

Iterator it = ordenado.entrySet().iterator();

int valueActual = -1;

ArrayList<Character> lista =

new ArrayList();

while (it.hasNext()) {

Map.Entry pair = (Map.Entry)it.next();

Character key = (Character)pair.getKey();

Integer value = (Integer)pair.getValue();

//System.out.println(pair.getKey() + " = " + pair.getValue());

if(value != valueActual ) {

if(lista.size() > 0) {

Collections.sort(lista);

matriz.add(lista);

}

lista = new ArrayList();

lista.add(key);

} else {

// lista.add(key);

lista.add(key);

}

valueActual = value;

it.remove(); // avoids a ConcurrentModificationException

}

if(lista.size() > 0)

{

Collections.sort(lista);

matriz.add(lista);

}

/\*

for(int i =0; i < matriz.size(); i++) {

for(int j = 0; j < matriz.get(i).size(); j++) {

System.out.print( matriz.get(i).get(j) + " " );

}

System.out.println();

}\*/

String ans = "";

for (int i = 0; i < matriz.size(); i++)

{

for (int j = 0; j < matriz.get(i).size(); j++)

{

/\*

for (int k = 0; k < hm[ matriz[i][j]]; k++)

{

ans += matriz[i][j];

}\*/

for (int k = 0; k < hm.get(matriz.get(i).get(j)); k++)

{

ans +=matriz.get(i).get(j);

}

}

}

return ans;

}

public static void printMap(Map mp) {

Iterator it = mp.entrySet().iterator();

while (it.hasNext()) {

Map.Entry pair = (Map.Entry)it.next();

System.out.println(pair.getKey() + " = " + pair.getValue());

it.remove(); // avoids a ConcurrentModificationException

}

}

public static void main(String[] args) {

// TODO code application logic here

String str = "bloomberg";

System.out.println( stringSort(str));

}

}