Sort Characters By Frequency

Question body:

Given a string, sort it in decreasing order based on the frequency of characters.

First attempt:

With the wrong assumption that the string would only contain alphabet characters, it is natural to use a mapping array with ascii code of the characters.

This method is still doable even without the assumption. However, in that case, the solution would lack of elegance because it introduces a lot of unnecessary computation.

First attempt solution:

public class Solution {

public String frequencySort(String s) {

int[] bucketCount = new int [52]; //a-z A-Z count

int index\_shift\_upper = 65;

int index\_shift\_lower = 97;

int num\_alphabe = 26;

char[] charArray = s.toCharArray();

for(int i=0; i<24; i++){

bucketCount[i]=0;

}

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

int ascii = (int) charArray[i];

if(ascii>=index\_shift\_lower){

//lower

bucketCount[ascii-index\_shift\_lower] ++;

}else{

//upper

bucketCount[ascii-index\_shift\_upper+num\_alphabe]++;

}

}

StringBuilder to\_return = new StringBuilder();

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

for(int count=0; count<bucketCount[i]; count++){

if(i>25){

//upper case

char letter = (char) (index\_shift\_upper+ i-26);

to\_return.append(letter);

}

else{

//lower case

char letter = (char) (index\_shift\_lower+ i);

to\_return.append(letter);

}

}

}

return to\_return.toString();

}

}

Second attempt:

To generalize the solution so that it could be applied to all characters, a map structure is used instead of an array. The entries of the map is the bucket for the key, which is the character, containing the number of occurrence. After counting all characters in the string, we just need sort the map basing on the entry value. This way we will have a sequence of character sorted in occurrence number.

Solution:

public class Solution {

public String frequencySort(String s) {

char[] charArray = s.toCharArray();

HashMap<String,Integer> char\_bucket = new HashMap();

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

String letter = Character.toString(charArray[i]);

if(char\_bucket.containsKey(letter)!=true){

char\_bucket.put(letter, 1);

}else{

char\_bucket.put(letter, char\_bucket.get(letter) + 1);

}

}

List<Map.Entry<String, Integer>> entries =

new ArrayList<Map.Entry<String, Integer>>(char\_bucket.entrySet());

Collections.sort(entries, new Comparator<Map.Entry<String, Integer>>() {

public int compare(Map.Entry<String, Integer> a, Map.Entry<String, Integer> b){

return a.getValue().compareTo(b.getValue());

}

});

StringBuilder to\_return = new StringBuilder();

for(int index=entries.size()-1; index>=0; index--){

Map.Entry<String, Integer> entry = entries.get(index);

String letter = entry.getKey();

int count = entry.getValue();

for(int i=0; i<count; i++){

to\_return.append(letter);

}

}

return to\_return.toString();

}

}