Highest Occurring Character

Problem Description: Given a string, find and return the highest occurring character present in the given string.
If there are 2 characters in the input string with same frequency, return the character which comes first.

Sample Input:

baaabbcc

Sample Output:

b

How to approach?

This problem seems trivial if we had the frequency of each character with us. Then all we had to do was to find out the maximum frequency, iterate through the string from start to end, and if a character's frequency were equal to the maximum frequency, we could just return that character.

So now our problem boils down to finding the frequency of each character efficiently. Ideally we would want to update the frequency of a character in O(1) time. A HashMap is the ideal data structure for this.

So now, we will try this. Maintain a character to integer HashMap where all values are initialised with 0. Iterate through the string. As we encounter a character, we will increase that character's frequency by 1. We will also update a *maximumFrequency,* that stores the maximum frequency we've seen till now.

After we iterate through the string and have found the maximum frequency, we'll again iterate through the string from start to end. If a character's frequency matches the maximumFrequency, we can straight-away return that character.
The pseudo-code for this approach is shown on the next page.

```
function highestOccurringCharacter(str):

  HashMap(character, integer) frequency
  maximumFrequency <- 0
  answer <- ‘’

  for char in str:
    frequency[char] <- frequency[char] + 1
    maximumFrequency = max(maximumFrequency, frequency[char])

  for char in str:
    if(frequency[char] = maximumFrequency):
      answer <- char
      break

  return answer
```