Given a string, find its first non-repeating character

Given a string, find the first non-repeating character in it. For example, if the input string is “GeeksforGeeks”, then output should be ‘f’ and if input string is “GeeksQuiz”, then output should be ‘G’.

We strongly recommend that you click here and practice it, before moving on to the solution.

We can use string characters as index and build a count array. Following is the algorithm.

```
1) Scan the string from left to right and construct the count array.
2) Again, scan the string from left to right and check for count of each
 character, if you find an element who's count is 1, return it.
```

Example:

```
Input string: str = geeksforgeeks
1: Construct character count array from the input string.
   ....
  count['e'] = 4
  count['f'] = 1
  count['g'] = 2
  count['k'] = 2
  ……
2: Get the first character who's count is 1 ('f').
```

Implementation:

```
// C program to find first non-repeating character
#include<stdlib.h>
#include<stdio.h>
#define NO_OF_CHARS 256
```

```c
/* Returns an array of size 256 containg count
   of characters in the passed char array */
int *getCharCountArray(char *str)
{
   int *count = (int *)calloc(sizeof(int), NO_OF_CHARS);
   int i;
   for (i = 0; *(str+i);  i++)
      count[*(str+i)]++;
   return count;
}

/* The function returns index of first non-repeating
   character in a string. If all characters are repeating
   then returns -1 */
int firstNonRepeating(char *str)
{
  int *count = getCharCountArray(str);
  int index = -1, i;

  for (i = 0; *(str+i);  i++)
  {
    if (count[*(str+i)] == 1)
    {
      index = i;
      break;
    }
  }

  free(count); // To avoid memory leak
  return index;
}

/* Driver program to test above function */
int main()
{
  char str[] = "geeksforgeeks";
  int index =  firstNonRepeating(str);
  if (index == -1)
    printf("Either all characters are repeating or string is empty");
  else
   printf("First non-repeating character is %c", str[index]);
  getchar();
  return 0;
}
```

Run on IDE

Python

```python
# Python program to print the first non-repeating character
NO_OF_CHARS = 256

# Returns an array of size 256 containg count
# of characters in the passed char array
def getCharCountArray(string):
    count = [0] * NO_OF_CHARS
    for i in string:
        count[ord(i)]+=1
    return count

# The function returns index of first non-repeating
# character in a string. If all characters are repeating
# then returns -1
def firstNonRepeating(string):
    count = getCharCountArray(string)
    index = -1
```

```
    k = 0

    for i in string:
        if count[ord(i)] == 1:
            index = k
            break
        k += 1

    return index

# Driver program to test above function
string = "geeksforgeeks"
index = firstNonRepeating(string)
if index==1:
    print "Either all characters are repeating or string is empty"
else:
    print "First non-repeating character is " + string[index]

# This code is contributed by Bhavya Jain
```

Run on IDE

Output:

```
First non-repeating character is f
```

Can we do it by traversing the string only once?

The above approach takes O(n) time, but in practice it can be improved. The first part of the algorithm runs through the string to construct the count array (in O(n) time). This is reasonable. But the second part about running through the string again just to find the first non-repeater is not good in practice. In real situations, your string is expected to be much larger than your alphabet. Take DNA sequences for example: they could be millions of letters long with an alphabet of just 4 letters. What happens if the non-repeater is at the end of the string? Then we would have to scan for a long time (again).

We can augment the count array by storing not just counts but also the index of the first time you encountered the character e.g. (3, 26) for ‘a’ meaning that ‘a’ got counted 3 times and the first time it was seen is at position 26. So when it comes to finding the first non-repeater, we just have to scan the count array, instead of the string. Thanks to Ben for suggesting this approach.

Following is C implementation of the extended approach that traverses the input string only once.

```
#include <stdlib.h>
#include <stdio.h>
#include <limits.h>
#define NO_OF_CHARS 256

// Structure to store count of a character and index of the first
// occurrence in the input string
struct countIndex {
   int count;
   int index;
};

/* Returns an array of above structure type. The size of
   array is NO_OF_CHARS */
struct countIndex *getCharCountArray(char *str)
{
   struct countIndex *count =
        (struct countIndex *)calloc(sizeof(countIndex), NO_OF_CHARS);
   int i;
```

```
    for (i = 0; *(str+i);  i++)
    {
        (count[*(str+i)].count)++;

        // If it's first occurrence, then store the index
        if (count[*(str+i)].count == 1)
            count[*(str+i)].index = i;
    }
    return count;
}

/* The function returns index of the first non-repeating
    character in a string. If all characters are repeating
    then reurns INT_MAX */
int firstNonRepeating(char *str)
{
  struct countIndex *count = getCharCountArray(str);
  int result = INT_MAX, i;

  for (i = 0; i < NO_OF_CHARS;  i++)
  {
    // If this character occurs only once and appears
    // before the current result, then update the result
    if (count[i].count == 1 && result > count[i].index)
        result = count[i].index;
  }

  free(count); // To avoid memory leak
  return result;
}

/* Driver program to test above function */
int main()
{
  char str[] = "geeksforgeeks";
  int index =  firstNonRepeating(str);
  if (index == INT_MAX)
    printf("Either all characters are repeating or string is empty");
  else
   printf("First non-repeating character is %c", str[index]);
  getchar();
  return 0;
}
```

Run on IDE

Output:

```
First non-repeating character is f
```

Asked in: Amazon, Goldman Sachs, InfoEdge, MakeMyTrip, MAQ Software, OATS Systems, OLA, Payu, Tejas Network, Teradata

Related Problem : K'th Non-repeating Character

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

198 Comments Category: Strings

Related Posts:

- Count number of substrings with exactly k distinct characters
- Palindrome Substring Queries
- Remove extra spaces from a string
- Print shortest path to print a string on screen
- Longest Common Prefix | Set 5 (Using Trie)
- Longest Common Prefix | Set 4 (Binary Search)
- Lower case to upper case – An interesting fact
- Longest Common Prefix | Set 3 (Divide and Conquer)

(Login to Rate and Mark)

2.2 Average Difficulty : **2.2/5.0**
Based on **33** vote(s)

Add to TODO List
Mark as DONE

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.

198 Comments GeeksforGeeks 1 Login

Recommend 4 Share Sort by Newest

Join the discussion…

Anonymous · 6 days ago
Can be done in O(1) space by maintaining a vector of pair<char,int> .
For each char, store <str[i],i>
Sort acc. to char. Then traverse sorted vector to find chars with single freq.(easy due to sorted order) and update index = min(index,i).In the end, index will have index of char with 1 freq and is first one.
Time : O(NlgN) but Space : O(1)
Reply · Share ›

no_limit ➚ Anonymous · 4 days ago
1. Time is more important that space.
2. You are using vector. Than how is space O(1)??
Reply · Share ›

Anmol Varshney · 22 days ago
Simpler C++ Code:
http://ideone.com/uuVt2Y

Based on: http://www.geeksforgeeks.org/k...

Reply • Share ›

Sairanga • 25 days ago

java implementation
http://code.geeksforgeeks.org/...

Reply • Share ›

Shivam Mitra • a month ago

We can do it in one traversal using a double link list. The idea is to append the characters which have appeared once in DLL. Store the address corresponding to the inserted node. If we encounter a character twice, delete that node form DLL.
C++ code:http://ideone.com/6XeYt3

Reply • Share ›

DPS • a month ago

simple solution using hashing :
#include<bits stdc++.h="">
using namespace std;
int main()
{
char s[]="geeksforgeeks";
int l=strlen(s);
int a[26]={0};
for(int i=0;i<l;i++) a[s[i]-97]++;="" for(int="" i="0;i<l;i++)" {="" if(a[s[i]-97]="">1)
s[i]=' ';
}
for(int i=0;i<l;i++){ if(s[i]!=" "){="" cout<<s[i];="" break;="" }="" }="" }="">

1 Reply • Share ›

Shubham Chaudhary • a month ago

C++ implementation with map:
https://ideone.com/KUxpqK

Reply • Share ›

naman mishra • 2 months ago

We can use LinkedHashMap in Java to check the elements in the same order in which they were inserted. Hence the first element in the map that has a count of 1 will be the answer

Reply • Share ›

programmer • 2 months ago

can we do it using hashing?
like: hashmap<string,count>hm;
String means character of strings like
</string,count></bits>

g,0
e,0 etc
for first occurrence count will be 0, and when next occurrence appears, count will be incremented.
We will return the first one having count to be 0.
Can we do like this?
Sorry, i'm a naive user. Please guide.

Reply • Share ›

evolver programmer • 19 days ago
yes you can in O(n) time complexity.

Reply • Share ›

Bikash Maharjan • 2 months ago
public class firstString {

public static void main(String[] args) {

System.out.println("first non repeated char is ");

System.out.println(solution("assf asdfw asfcx tg"));

}

public static char solution(String s){

char c[]=s.toCharArray();

for(int i =0;i<s.length() ;i++){="" int="" count="0;" for(int="" j="0;j<s.length();j++") {="" if(c[i]="=c[j])" count++;="" }="" if="" (count="=1){" return="" c[i];="" }="" }="" return="" '1';="" }="" }="">

Reply • Share ›

Prashant Kumar • 2 months ago
using pointer to c++ class: http://code.geeksforgeeks.org/...

Reply • Share ›

Surya • 2 months ago
I guess , the String "ABCDD" would fail in case of 1st solution .

Expected answer : 'C'
Answer from Algo : 'A'

Correct me , if I am wrong

Reply • Share ›

venky Surya • 19 days ago
expected answer is : A(it is the first non repeating character in the string "ABCDD"

Reply • Share ›

navjot • 4 months ago

nice

Reply • Share ›

ADITYA DHANIWALA • 4 months ago

Check this method ->

```
#include<stdio.h>
#include<string.h>
#include<limits.h>
int main()
{
int hash_table[26]={0};
char str[]="zaaabdx cld jb";
int i;
for (i=1;i<=strlen(str);i++)
{
if (str[i-1]>=97 && str[i-1]<=122)
{
if (hash_table[str[i-1]-97]!=0)
hash_table[str[i-1]-97]=-1;
else if (hash_table[str[i-1]-97]>=0)
hash_table[str[i-1]-97]=i;
}
}
int j,min=INT_MAX;
for (j=0;j<26;j++)
{
if (hash_table[j]<min &&="" hash_table[j]!="0" &&="" hash_table[j]!="-1)"
min="hash_table[j];" }="" --min;="" printf="" ("first="" non="" repeating=""
char="%c",str[min]);" return="" 0;="" }="">
```

Reply • Share ›

invincible • 4 months ago

we can also check in one scan by traversing string from last...no need to maintain index variable and no need to scan count array... http://code.geeksforgeeks.org/...

Reply • Share ›

amor p ➚ invincible • 4 months ago

check for simple string first like " bba" :)

Reply • Share ›

Maverick92 • 4 months ago

Scanner S=new Scanner(System.in);

String string="mnnmposmsss";

String string= "hhhhmposhsss";

string = string.toLowerCase();

HashMap<character,integer> hmap=new HashMap<>();

for(int i=0;i<string.length();i++){ if(hmap.containskey(string.charat(i))){="" hmap.put(string.charat(i),hmap.get(string.charat(i))+1);="" }="" else="" hmap.put(string.charat(i),="" 1);="" }="" system.out.println(hmap.tostring());="" int="" count="0;" for(map.entry<character,integer=""> entry : hmap.entrySet()){

if(entry.getValue()==1){

count ++;

}

}

---

see more

• Reply • Share ›

Masaaki • 4 months ago

```
This my code below is O(n + m) solution in C++; n is a length of incoming st
and m is a number of unique char in the input string.
If code below has issue with format,
please go this link to see my whole code.
(Link http://code.geeksforgeeks.org/jvcb3N )

#include <iostream>
#include <unordered_map>
#include <vector>
using namespace std;

char findFirstNonRepeatedChar(char *str,int size) {
    unordered_map<char, int=""> mymap;
    vector<char> v;
    for(int i=0;i<size;i++) {="" if="" (mymap[str[i]]<="" 1)="" v.push_back(
```

• Reply • Share ›

Ram narayan Raigar • 4 months ago

import java.io.*;

class Firstunrepeatedchar

{

public static void main(String args[])

{

{

System.out.println("Enter String");

Console con=System.console();

String string;

string=con.readLine();

int i,j,flag=0;

for(i=0;i<string.length();i++) {="" for(i="0;i<string.length();i++)" {=""

see more

Reply • Share ›

anil kumar • 4 months ago

why dont we keep two indices. one will keep track current character and other will track first non repeating character. complexity O(N) only. no need to scan count array also.

```
cur=0;
final=0;
while(str[cur]){
count[str[cur++]]++;
while(str[final] && count[str[final]]>1)
final++;
}
if(!str[final])
printf("no non-repeating character in given string\n");
else
printf("%c is first non-repeating character in given string\n",str[final]);
```

1 Reply • Share ›

Radhika Sharma • 4 months ago

#include <stdio.h>

int main(void) {

char s[10] = {'m','u','m','b','a','i'};

char *p , *q;

int count = 0;

for(p = s; *p != '\0'; p++){

count = 0;

for(q = s ; *q != '\0'; q++){

```
if(*p == *q) count++;

}

if(count == 1) break;

}

if(*p != '\0')printf("First Non-Repeating Character : %c", *p);

else printf("No Non-Repeating Character found");

return 0;

}
```

Reply • Share ›

Ravi Kant Soni · 5 months ago

```
import java.util.HashMap;
import java.util.Map;

public class FirstNonRepeatingCharacter {

public static void main(String[] args) {

String str = "GeeksforGeeks";
firstNonRepeating(str);

String str2 = "GeeksQuiz";
firstNonRepeating(str2);

String str3 = "teeeeteeeer";
firstNonRepeating(str3);

}

private static void firstNonRepeating(String str) {
```

see more

Reply • Share ›

sudhan · 6 months ago

why use (str+i) in both and not just str ??

Reply • Share ›

Sidar · 6 months ago

Ignore my previous post
http://ideone.com/ckaBFy

Reply • Share ›

Sidar • 6 months ago

```
/* package whatever; // don't place package name! */

import java.util.*;

import java.lang.*;

import java.io.*;

/* Name of the class has to be "Main" only if the class is public. */

class Ideone

{

public static void main (String[] args) throws java.lang.Exception

{

String s1="palatte";

int ind=0;
```

see more

• Reply • Share ›

codemonk • 6 months ago

i think the soln is not giving first non repeating character...for e.g. "geekszf" gives output as 'f' but the first non repeating charcter is 'g'. my soln for above problem .. http://ideone.com/IU6cem

• Reply • Share ›

Avinash Arunkumar • 6 months ago

How about this..

```
import java.util.HashSet;

import java.util.LinkedHashMap;

import java.util.LinkedHashSet;

import java.util.Map;

import java.util.Set;

public class nonreeatingcharacter {

public static void main(String[] args) {

String s = "aabqqccxzfkbeddemqq";
char [] ch = s.toCharArray();
Set set1 = new LinkedHashSet();
```

```
for (char ch1 : ch) {
```

see more

Reply • Share ›

Arun ➚ Avinash Arunkumar • 6 months ago

Your solution fails for the input "aaabccdde"
Your solution returns "a"
But the actual first non repeating character is "b".

Reply • Share ›

Avinash Arunkumar ➚ Arun • 6 months ago

Thats true Arun. Thanks for pointing out. :) . Just adding to code by Mr. Patil below..

```
public class FirstNonRepeatingcharcter {

public static void main(String[] args) {

String s1 = "baab";

char stringArray[] = s1.toCharArray();

int i = 0;

boolean flag = false;

while (i < s1.length()) {

if(s1.indexOf(stringArray[i], i+1) == -1 && s1.lastIndexOf(stringArray[i], i-1) == -1 ) {
```

see more

Reply • Share ›

Devendra Patil • 7 months ago

In Java :

http://code.geeksforgeeks.org/...

Reply • Share ›

Rohit Pratap Singh ➚ Devendra Patil • 4 months ago

Hi Devendra,

Clever solution.

But the indexOf() operator you are using internally uses 3 loops, so the complexity will be more than o(n*n).

if(s1.indexOf(stringArray[i], i+1) == -1)

chetan kole • 8 months ago

My solution using Hash (Time Complexity o(n))

```
import java.util.HashMap;

import java.util.Map;

public class FirstNonRepeating {

public static void main(String[] args) {

int arr[] = { 1, 4, 2, 6, 3, 8, 1,4, 9 };

Map<integer, integer=""> hash = new HashMap<>();

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

if (hash.containsKey(arr[i]))

hash.replace(arr[i], hash.get(arr[i])+1 );

else
```


Rocky26 • 8 months ago

```
dic={}

str = "GeeksforGeeks"

for letter in str:

if dic.has_key(letter):

dic[letter] += 1

else:

dic[letter] = 1

lis=[]

for key in dic:

if dic[key] != " " and dic[key]<2:

print key,dic[key]

break
```

Reply • Share ›

Raunak Maheshwari • 9 months ago

Much simplified code using queue.

#include<bits stdc++.h="">

using namespace std;

int main()

{

int ascii[256]={0};

int arr[1000];

string a;

cin>>a;

queue<int> q;

for(int i=0;i<a.length();i++) {="" ascii[int(a[i])]++;="" q.push(int(a[i]));="" }="" while(!q.empty())="" {="" if(ascii[q.front()]="1)" {="" cout<<"the="" first="" character="" is="" :="" "<<char(q.front())<<endl;="" break;="" }="" else="" {="" q.pop();="" }="" }="" return="" 0;="" }="">

Reply • Share ›

Prashant • 9 months ago

A simpe approach using map in which we do not need to traverse the array for 256 (NO_OF_Char) again or the second time. The logic is traverse the array of chars from the reverse and take a variable say index which will keep updating the index when the count is 1. Finally we will have the index of the char which is having count as 1 and also the first non repeating one.
Please see the code.

#include<iostream>
#include<map>
using namespace std;

void firstUniqueElement(char*arr,int n){
map<char,int> HM;
int index=-1;
for(int i=n-1;i>=0;i--){
HM[arr[i]]+=1;
if(HM[arr[i]]==1) index=i;
}
//final check for uniqueness
if(HM[arr[index]] == 1)

cout<<arr[index]<<endl; else="" cout<<"not="" unique"<<endl;="" }="" int="" main() {="" char="" arr[]="{"geeksforgeeks"};" int="" n="sizeof(arr)/sizeof(char);" firstuniqueelement(arr,n);="" }="">

Reply • Share ›

Arvind • 10 months ago

The implementation of above written algorithm is wrong as we always get the first character in a-z which occurs only once in given string instead of getting the non repeating character which occurs first time in the given string...did u get my point ?

1 Reply • Share ›

Aman Aggarwal ➚ Arvind • 7 months ago

No.In the second implementation also,we check the entire count array,and output that character for which index is minimum.Hence we find the minimum index non repeating character.

For the first implementation,the logic is obvious.

Hope it helps.

Reply • Share ›

Rajan Kalra ➚ Arvind • 8 months ago

Indeed this was what I understood looking at the algo but the implementation takes care of your concern. Have a look!

Reply • Share ›

satish • 10 months ago

How to determine if String has all unique characters:-http://techno-terminal.blogspo...

Reply • Share ›

n20084753 • 10 months ago

An approach with linked list

http://code.geeksforgeeks.org/...

Reply • Share ›

Sarthak • 10 months ago

What does "for (i = 0; *(str+i); i++)" do??

Reply • Share ›

Michelle Lin ➚ Sarthak • 10 months ago

It's a pointer dereference.

*(str) is a pointer to the head of 'str', so the pointer points to str[0]. *(str+i) increments the pointer, moving it forward by sizeof(int) bytes (since 'i' is an int).

Reply • Share ›

Avadh Kishore • a year ago

nice work :)

Reply • Share ›

rahul gaur • a year ago

my implementation in python
http://laravel.io/bin/bEwLQ

Reply • Share ›

skillet rahul gaur • 9 months ago

Hi Rahul,
I like how concise ur code is.
could u please explain min(collection.iteritems(), key=operator.itemgetter(1)) part.
Which field is the minimum applied on? Will it break after finding fist minimum o traverse till he end?
Also you code doesn't handle when there is no such character of single occurence

Reply • Share ›

Krushna Kumar • a year ago

Simple and Easy

https://code.hackerearth.com/1...

Reply • Share ›

Shashank Shanker Khare • a year ago

How about traversing like this
http://ideone.com/gt2Rgd

Reply • Share ›

abhi divekar • a year ago

There's another approach: And at the end, we know ALL the non-repeating characters in the string, in chronological order.
use the hash map obviously, but also use a linked list or queue that stores the chronological order in which we get the characters. So, "Geeksforgeeks" would be stored as G->e->k->s->f->o->r->g. We only append to the end of this list when we find a new character (verified via the hashmap).
At any point, the head of the list is the first non-repeating character. If the next input character is the head's character, we remove the head and consider the next in list as the new "first non-repeating character". If that was repeated (it might have been somewhere in the previous input stream), we remove it too (again, verify with hashmap). Since there are a finite number of characters and we never re-enter a