Find the first non-repeating character from a stream of characters

Given a stream of characters, find the first non-repeating character from stream. You need to tell the first non-repeating character in O(1) time at any moment.

If we follow the first approach discussed here, then we need to store the stream so that we can traverse it one more time to find the first non-repeating character at any moment. If we use extended approach discussed in the same post, we need to go through the count array every time first non-repeating element is queried. We can find the first non-repeating character from stream at any moment without traversing any array.

We strongly recommend you to minimize the browser and try it yourself first.

The idea is to use a DLL (**D**oubly **L**inked **L**ist) to efficiently get the first non-repeating character from a stream. The DLL contains all non-repeating characters in order, i.e., the head of DLL contains first non-repeating character, the second node contains the second non-repeating and so on.
We also maintain two arrays: one array is to maintain characters that are already visited two or more times, we call it repeated[], the other array is array of pointers to linked list nodes, we call it inDLL[]. The size of both arrays is equal to alphabet size which is typically 256.

```
1) Create an empty DLL. Also create two arrays inDLL[] and repeated[] of size 256.
   inDLL is an array of pointers to DLL nodes. repeated[] is a boolean array,
   repeated[x] is true if x is repeated two or more times, otherwise false.
   inDLL[x] contains pointer to a DLL node if character x is present in DLL,
   otherwise NULL.
```


Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion
Geometric Algorithms

```
2) Initialize all entries of inDLL[] as NULL and repeated[] as false.

3) To get the first non-repeating character, return character at head of DLL.

4) Following are steps to process a new character 'x' in stream.
  a) If repeated[x] is true, ignore this character (x is already repeated two
      or more times in the stream)
  b) If repeated[x] is false and inDLL[x] is NULL (x is seen first time)
     Append x to DLL and store address of new DLL node in inDLL[x].
  c) If repeated[x] is false and inDLL[x] is not NULL (x is seen second time)
     Get DLL node of x using inDLL[x] and remove the node. Also, mark inDLL[x]
     as NULL and repeated[x] as true.
```

Note that appending a new node to DLL is O(1) operation if we maintain tail pointer. Removing a node from DLL is also O(1). So both operations, addition of new character and finding first non-repeating character take O(1) time.

```
// A C++ program to find first non-repeating character from a stream o
#include <iostream>
#define MAX_CHAR 256
using namespace std;

// A linked list node
struct node
{
    char a;
    struct node *next, *prev;
};

// A utility function to append a character x at the end of DLL.
// Note that the function may change head and tail pointers, that
// is why pointers to these pointers are passed.
void appendNode(struct node **head_ref, struct node **tail_ref, char x
{
    struct node *temp = new node;
    temp->a = x;
    temp->prev = temp->next = NULL;

    if (*head_ref == NULL)
    {
        *head_ref = *tail_ref = temp;
        return;
```


Popular Posts

All permutations of a given string
Memory Layout of C Programs
Understanding “extern” keyword in C
Median of two sorted arrays
Tree traversal without recursion and without stack!
Structure Member Alignment, Padding and Data Packing
Intersection point of two Linked Lists
Lowest Common Ancestor in a BST.
Check if a binary tree is BST or not
Sorted Linked List to Balanced BST

```
    }
    (*tail_ref)->next = temp;
    temp->prev = *tail_ref;
    *tail_ref = temp;
}

// A utility function to remove a node 'temp' fromt DLL. Note that the
// function may change head and tail pointers, that is why pointers to
// these pointers are passed.
void removeNode(struct node **head_ref, struct node **tail_ref,
                struct node *temp)
{
    if (*head_ref == NULL)
        return;

    if (*head_ref == temp)
        *head_ref = (*head_ref)->next;
    if (*tail_ref == temp)
        *tail_ref = (*tail_ref)->prev;
    if (temp->next != NULL)
        temp->next->prev = temp->prev;
    if (temp->prev != NULL)
        temp->prev->next = temp->next;

    delete(temp);
}
```

```
void findFirstNonRepeating()
{
    // inDLL[x] contains pointer to a DLL node if x is present in DLL.
    // If x is not present, then inDLL[x] is NULL
    struct node *inDLL[MAX_CHAR];

    // repeated[x] is true if x is repeated two or more times. If x is
    // not seen so far or x is seen only once. then repeated[x] is fal
    bool repeated[MAX_CHAR];

    // Initialize the above two arrays
    struct node *head = NULL, *tail = NULL;
    for (int i = 0; i < MAX_CHAR; i++)
    {
        inDLL[i] = NULL;
        repeated[i] = false;
    }

    // Let us consider following stream and see the process
    char stream[] = "geeksforgeeksandgeeksquizfor";
```

```
    for (int i = 0; stream[i]; i++)
    {
        char x = stream[i];
        cout << "Reading " << x << " from stream \n";

        // We process this character only if it has not occurred or oc
        // only once. repeated[x] is true if x is repeated twice or mo
        if (!repeated[x])
        {
            // If the character is not in DLL, then add this at the en
            if (inDLL[x] == NULL)
            {
                appendNode(&head, &tail, stream[i]);
                inDLL[x] = tail;
            }
            else // Otherwise remove this caharacter from DLL
            {
                removeNode(&head, &tail, inDLL[x]);
                inDLL[x] = NULL;
                repeated[x] = true;  // Also mark it as repeated
            }
        }

        // Print the current first non-repeating character from stream
        if (head != NULL)
            cout << "First non-repeating character so far is " << head
    }
}

/* Driver program to test above function */
int main()
{
    findFirstNonRepeating();
    return 0;
}
```

Output:

```
Reading g from stream
First non-repeating character so far is g
Reading e from stream
First non-repeating character so far is g
Reading e from stream
First non-repeating character so far is g
Reading k from stream
```


Recent Comments

affiszerv Your example has two 4s on row 3, that's why it...

Backtracking | Set 7 (Sudoku) · 13 minutes ago

RVM Can someone please elaborate this Qs from above...

Flipkart Interview | Set 6 · 33 minutes ago

Vishal Gupta I talked about as an Interviewer in general,...

Software Engineering Lab, Samsung Interview | Set 2 · 33 minutes ago

@meya Working solution for question 2 of 4f2f round....

Amazon Interview | Set 53 (For SDE-1) · 1 hour ago

sandeep void rearrange(struct node *head) {...

Given a linked list, reverse alternate nodes and append at the end · 2 hours ago

Neha I think that is what it should return as, in...

Find depth of the deepest odd level leaf node · 2 hours ago

```
First non-repeating character so far is g
Reading s from stream
First non-repeating character so far is g
Reading f from stream
First non-repeating character so far is g
Reading o from stream
First non-repeating character so far is g
Reading r from stream
First non-repeating character so far is g
Reading g from stream
First non-repeating character so far is k
Reading e from stream
First non-repeating character so far is k
Reading e from stream
First non-repeating character so far is k
Reading k from stream
First non-repeating character so far is s
Reading s from stream
First non-repeating character so far is f
Reading a from stream
First non-repeating character so far is f
Reading n from stream
First non-repeating character so far is f
Reading d from stream
First non-repeating character so far is f
Reading g from stream
First non-repeating character so far is f
Reading e from stream
First non-repeating character so far is f
Reading e from stream
First non-repeating character so far is f
Reading k from stream
First non-repeating character so far is f
Reading s from stream
```

```
First non-repeating character so far is f
Reading q from stream
First non-repeating character so far is f
Reading u from stream
First non-repeating character so far is f
Reading i from stream
First non-repeating character so far is f
Reading z from stream
First non-repeating character so far is f
Reading f from stream
First non-repeating character so far is o
Reading o from stream
First non-repeating character so far is r
Reading r from stream
First non-repeating character so far is a
```

This article is contributed by **Amit Jain**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Related Tpoics:

- Printing Longest Common Subsequence
- Suffix Array | Set 2 (nLogn Algorithm)
- Rearrange a string so that all same characters become d distance away
- Recursively remove all adjacent duplicates
- Dynamic Programming | Set 33 (Find if a string is interleaved of two other strings)
- Remove “b” and “ac” from a given string
- Dynamic Programming | Set 29 (Longest Common Substring)
- Write your own atoi()

36 Comments GeeksforGeeks

Sort by Newest

Subhash • 9 days ago

```
private static void FirstNonRepeatingChar()

{

string input = Console.ReadLine();

int length = input.Length - 1;

char[] inputCharArray = input.ToArray();

int j;

for (int i = 0; i < length; i++)
```

```
{

for ( j = i+1; j <= length; j++)

{

if (inputCharArray[i].Equals(inputCharArray[i]))
```

see more

• Reply • Share ›

Ankit Jain • 13 days ago

IS IT CORRECT ?

```
#include<stdio.h>

char s[100];
int flag[100];
char Output(char c)
{
static int i=0,j=0;
flag[c-97]++;
if(flag[c-97]==1)
{
s[j]=c;
j++;
}
while(flag[s[i]-97]!=1)
{
i++;
}
```

see more

Ravish Roshan • 14 days ago

What is the point of using doubly linked list here? Can't we do the same thing l

1 • Reply • Share ›

Rahul • a month ago

```
# include<stdio.h>

# include<stdlib.h>

# define NO_OF_CHARS 256

void removeDups(char str[] , int size)

{

int hash[NO_OF_CHARS] = {0};

for(int i=0 ; str[i] ; i++){

++hash[str[i]];

}

for(int i=0 ; i<size ;i++){="" if(hash[str[i]]="=1){" printf("%c",str[i]);="" break;="" }
to="" test="" removedups="" *="" int="" main()="" {="" char="" str[]="geeksgee
size="sizeof(str)/" sizeof(str[0]);="" removedups(str,size);="" getchar();="" retu
```

1 • Reply • Share ›

tamojit9 • a month ago

i think the repeated array is redundant. instead of initializing all the inDll entries
node(used to denote that the element is encountered the first time).

Following are steps to process a new character 'x' in stream.

a) If inDll is NULL, ignore this character (x is already repeated two
or more times in the stream)
b) If inDLL[x] is The Dummy Node (x is seen first time)
Append x to DLL and store address of new DLL node in inDLL[x].
c) If inDLL[x] is not NULL (x is seen second time)
Get DLL node of x using inDLL[x] and remove the node. Also, mark inDLL[x] a
please correct me if i am wrong.....

2 • Reply • Share ›

Monkey D Luffy • 2 months ago

Simple solution is every character you read from stream add it to Hashmap wi
read, index of character in stream). If add method returns false(That means th
value as very huge number for that key. After reading all characters from strea
at max 256. Now just order the hashmap ascending by values and choose firs
maximum size of hashmap here is constant. Let me know what you think.

• Reply • Share ›

vamsi • 3 months ago

Nice solution. The basic of using DLL is to maintain a queue of non-repeating
required here? why not we use single linked list for creating queue?

• Reply • Share ›

sri → vamsi • 2 months ago

if u use a linked list, deleting a node(non repeating character changes t
breaks the list.

• Reply • Share ›

sonu_nit • 4 months ago

How removing from a DLL is O(1) ? I am not getting it should be O(n) since m
for first repeated element we have to search for it where it is in DLL . it may be

• Reply • Share ›

Ganesh Babu • 5 months ago

we can use min heap also...here heap node structure
struct node{
char c;
int count; // maintain the count how many times character c is encounterd till
int index; // check for which character is first
}
If two character are with count 1, check their indexes, ans is the one with lowe

• Reply • Share ›

poonam → Ganesh Babu • 4 months ago

As far as I got you, you are trying to build heap over count property and
How will you handle when you encounter a character from a string, you
whether that character exists or not and this will be done processing al

• Reply • Share ›

Jinto Kuriakose • 6 months ago

Java code..

```
public class FirstNonRepeatingCharacterFromAStream {

public static void main(String[] args) {

find("abcdabcefghihid");

}

public static void find(String s) {

LinkedList<mynode> list = new LinkedList<mynode>();

MyNode[] nodes = new MyNode[256];
```

```
for (char c : s.toCharArray()) {

if (repeated[c] == false && nodes[c] == null) {
```

see more

Reply • Share ›

Bao Nhan → Jinto Kuriakose • 4 months ago

list.remove doesn't do it in O(1)

Reply • Share ›

neelabhsingh • 7 months ago

```
#include<stdio.h>
#define MAX 100
int main()
{
char s[100],*temp;
//char arr[26]={'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x
int arr[26]={0},i;
gets(s);
temp=s;
while(*temp!='\0')
{
arr[*temp-'a']++;
temp++;
}
i=0;
/* while(i<26)
{
printf("%d ",arr[i]);
```

see more

1 ^ | v • Reply • Share ›

Thien Nguyen • 7 months ago

As others suggested, we don't need the double link list. We need the simple q
fact potential non-repeate characters). When character becomes repeat chara
away if it is not first element of queue. Here is the algorithm

```
while(   queue is not empty && its head element is repeat character)
      remove head

#include <iostream>
#define MAX_CHAR 256
using namespace std;

void findFirstNonRepeating()
{
    // simple queue
    char queue[MAX_CHAR], *head, *tail;
    head = tail = queue; // use stl style

    int count[MAX_CHAR];
```

see more

2 ^ | v • Reply • Share ›

Ashok • 7 months ago

How about doing some thing like this. Maintain two linked Hash Sets (Java imp

1. uniqueSet, 2. dupSet.

and follow the below algo.

1. Read ch

2 if dupSet contains Ch go to Step 1

3. if uniqueSet contains Ch remove ch from uniqueSet and , it to dupSet nd go

4. Add Ch to uniqueSet and go to Step1

At any point of time the first character im uniqueSet will the required result.

Space complexity is 0(256) considering that steam follows unicode.

As we using LinkedHAshSet in Java, insertion order will be maintained and all

• Reply • Share ›

Agn B • 7 months ago

If we are interested in only finding out the first non -repeating element, then we stack. We push the 1st element to the stack. If there are no more elements in repeating element. Else, for the 2nd element in the stream we compare with th not do anything, if it does not match, then that element is the 1st non repeating

I think by this approach also we can solve the problem in O(1) time.Please let

• Reply • Share ›

Harish Kumar → Agn B • 6 months ago

Please explain ur algorithm for an input string 'certificate'

• Reply • Share ›

Agn B → Harish Kumar • 6 months ago

I think for the input 'certificate', 'c' is the first non repeating elem

• Reply • Share ›

Harish Kumar → Agn B • 6 months ago

I think u took the question wrong. The actual problem is

which has not repeated again in the whole string(i.e no
arrived twice answer is not c. Simply answer is we have
which contains the characters which are used only onc
'r'

• Reply • Share ›

Herman • 7 months ago

Just wanted to note something: Memory and time in this example is O(1) sinc
always have at most 256 unique chars. It's not dependent on the input size (w

The use of a DLL is nice, but you can also use any Queue, since you will alwa
queue, you can say that the time will be O(1) because queue size doesn't gro
not the size of the problem).
The magic of O-notation :)

• Reply • Share ›

Pramod → Herman • 7 months ago

The size may be more than 256 for unicode characters or some other

• Reply • Share ›

Herman → Pramod • 7 months ago

Yes, but it's constant nonetheless

• Reply • Share ›

Ankur Garg • 7 months ago

The approach here seems good. But I think we can do it this way as well .

1)Have a DLL ->Maintain its head and Tail Pointers .

2) Have a character array of size 256 to maintain count of each character as i
3) Now when u see a character . Check its count in the char array .
if (count==0)
append it to DLL at the back and update the tail pointer .

else
this should be the head of the linked List . Remove the element from the list .
4) Keep traversing the whole DLL . At the last the head of the DLL is our requi

Why do we need to keep an inDLL[MAX_CHAR] here . Just simply keeping a c
maintain the head and tail pointers .This would still give u the first repeating ch

• Reply • Share ›

Ankur → Ankur Garg • 7 months ago
This is wrong. Ignore
1 • Reply • Share ›

Sudarshan Singh • 7 months ago
Dynamic list is natural solution for such problem ,so we loose benefit of static array of node we can retain those benefits ,by the way good post.
• Reply • Share ›

Bharath • 7 months ago

```
#include <string>

#include <iostream>

#include <map>

using namespace std;

struct SNode {

char data;

struct SNode * next;

} ;
```

typedef struct SNode Node;

class List {

Node * head, *tail;

see more

• Reply • Share ›

prateek • 7 months ago

great solution for a common amazon interview question

• Reply • Share ›

Guest • 7 months ago

It seems that your analysis of O(1) time is incorrect. The solution you have is (
every element of this stream once.

• Reply • Share ›

guest → Guest • 7 months ago

its 0(N) for N elements so n/n is 0(1),Amortized analysis.
At every character you are just performing 0(1) operations

• Reply • Share ›

GeeksforGeeks Mod → Guest • 7 months ago

Please take a closer look. Time to query the first non-repeating charac
O(n).

• Reply • Share ›

Aniket → GeeksforGeeks • 7 months ago

hmmm.. got it wrong. You'r correct.

• Reply • Share ›

Aniket → GeeksforGeeks • 7 months ago

So overall time complexity is O(n).

• Reply • Share ›

Fate.AKong • 7 months ago

Great post. Thanks.
Just would like to add some trivial points here.
A hash structure (char=>Node*) can replace the two arrays since hash itself i
seen.

Following are the steps in detail: (for a new char 'x')
- if 'x' is not in hash (first occurrence)
append 'x' to DLL and store ('x'=>addr. of DLL node) into hash
- if 'x' is already in hash, and hash['x'] is not NULL (second occurrence)
remove the node from DLL and set hash['x'] to NULL
- if 'x' is already in hash, and hash['x'] == NULL (following occurrences)
ignore this character

If we don't take hash's overhead into consideration, then in this way, space is
N is the max # of possible chars), and the same time complexity is reserved (

Also hash can handle a much bigger size of charset. We need to know exactl
we could use a 256-size array.
But for hash there is no such restrictions. The stream can be UTF-8 chars, int
perfectly into a key.

Please point me out if anything goes wrong. Thanks.

6 • Reply • Share ›

naini ➦ Fate.AKong • 7 months ago

u wont be able to find out the FIRST non-repeating character with a ha

• Reply • Share ›

How will you check that 'x' is in hash or not ?

• Reply • Share ›