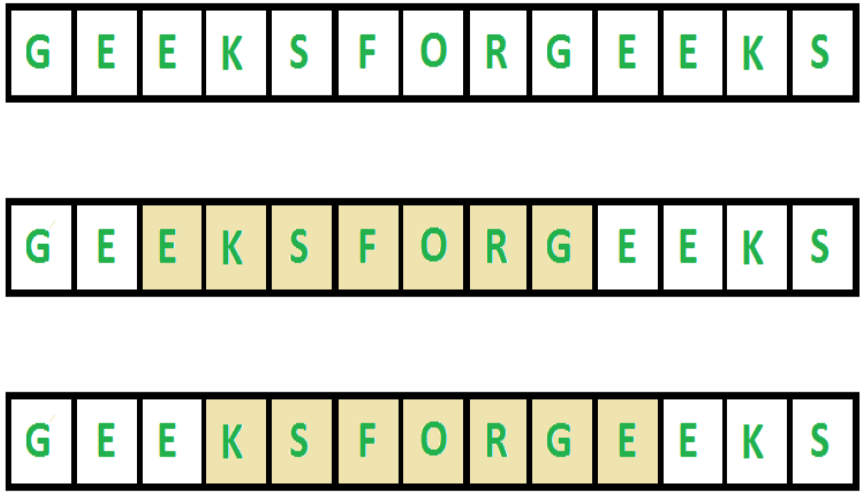
Length of the longest substring without repeating characters

Given a string, find the length of the longest substring without repeating characters. For example, the longest substrings without repeating characters for “ABDEFGABEF” are “BDEFGA” and “DEFGAB”, with length 6. For “BBBB” the longest substring is “B”, with length 1. For “GEEKSFORGEEKS”, there are two longest substrings shown in the below diagrams, with length 7.



The desired time complexity is O(n) where n is the length of the string.

Method 1 (Simple)
We can consider all substrings one by one and check for each substring whether it contains all

unique characters or not. There will be n*(n+1)/2 substrings. Whether a substirng contains all unique characters or not can be checked in linear time by scanning it from left to right and keeping a map of visited characters. Time complexity of this solution would be O(n^3).

Method 2 (Linear Time)
Let us talk about the linear time solution now. This solution uses extra space to store the last indexes of already visited characters. The idea is to scan the string from left to right, keep track of the maximum length Non-Repeating Character Substring (NRCS) seen so far. Let the maximum length be max_len. When we traverse the string, we also keep track of length of the current NRCS using cur_len variable. For every new character, we look for it in already processed part of the string (A temp array called visited[] is used for this purpose). If it is not present, then we increase the cur_len by 1. If present, then there are two cases:

a) The previous instance of character is not part of current NRCS (The NRCS which is under process). In this case, we need to simply increase cur_len by 1.
b) If the previous instance is part of the current NRCS, then our current NRCS changes. It becomes the substring staring from the next character of previous instance to currently scanned character. We also need to compare cur_len and max_len, before changing current NRCS (or changing cur_len).

Implementation

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

int min(int a, int b);
```

```
int longestUniqueSubsttr(char *str)
{
    int n = strlen(str);
    int cur_len = 1;  // To store the lenght of current substring
    int max_len = 1;  // To store the result
    int prev_index;  // To store the previous index
    int i;
    int *visited = (int *)malloc(sizeof(int)*NO_OF_CHARS);

    /* Initialize the visited array as -1, -1 is used to indicate that
       character has not been visited yet. */
    for (i = 0; i < NO_OF_CHARS;  i++)
        visited[i] = -1;
```


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

```
    /* Mark first character as visited by storing the index of first
       character in visited array. */
    visited[str[0]] = 0;

    /* Start from the second character. First character is already pro
       (cur_len and max_len are initialized as 1, and visited[str[0]]
    for (i = 1; i < n; i++)
    {
        prev_index =  visited[str[i]];

        /* If the currentt character is not present in the already pro
           substring or it is not part of the current NRCS, then do cu
        if (prev_index == -1 || i - cur_len > prev_index)
            cur_len++;

        /* If the current character is present in currently considered
           then update NRCS to start from the next character of previo
        else
        {
            /* Also, when we are changing the NRCS, we should also che
              length of the previous NRCS was greater than max_len or
            if (cur_len > max_len)
                max_len = cur_len;

            cur_len = i - prev_index;
        }

        visited[str[i]] = i; // update the index of current character
    }

    // Compare the length of last NRCS with max_len and update max_len
    if (cur_len > max_len)
        max_len = cur_len;


    free(visited); // free memory allocated for visited

    return max_len;
}

/* A utility function to get the minimum of two integers */
int min(int a, int b)
{
    return (a>b)?b:a;
}
```

```
/* Driver program to test above function */
int main()
{
    char str[] = "ABDEFGABEF";
    printf("The input string is %s \n", str);
    int len =  longestUniqueSubsttr(str);
    printf("The length of the longest non-repeating character substrin

    getchar();
    return 0;
}
```

Output

```
The input string is ABDEFGABEF
The length of the longest non-repeating character substring is 6
```

Time Complexity: O(n + d) where n is length of the input string and d is number of characters in input string alphabet. For example, if string consists of lowercase English characters then value of d is 26.
Auxiliary Space: O(d)
Algorithmic Paradigm: Dynamic Programming

As an exercise, try the modified version of the above problem where you need to print the maximum length NRCS also (the above program only prints length of it).

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

Recent Comments

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

Backtracking | Set 7 (Sudoku) · 18 minutes ago

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

Flipkart Interview | Set 6 · 38 minutes ago

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

Software Engineering Lab, Samsung Interview | Set 2 · 38 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

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

Writing code in comment? Please use **ideone.com** and share the link here.

63 Comments GeeksforGeeks

Sort by Newest

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

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

Ashish • 19 days ago

The longest NRCS can be printed by code at below link:
http://ideone.com/FFQRkw

Plz point out if any mistake.....

  • Reply • Share ›

Mohaan • 22 days ago

We can print and count like below.. If any mistake please specify...

http://ideone.com/lFZygQ

  • Reply • Share ›

Srikant Aggarwal • a month ago

This is giving wrong answer for ABCDEFCGHIJB

  • Reply • Share ›

mccullum Srikant Aggarwal • a month ago

its 9 correct

  • Reply • Share ›

shan • a month ago

A very simple solution to check and find longest substring is

#include<iostream>

using namespace std;

int main()

{

```
char s[]="geeksforgeeks";

bool visited[26]={false};

int max=INT_MIN,count=0,i=0;

while(s[i]!='\0')

{

if(visited[s[i]-'a']==false)
```

see more

Reply • Share ›

prashant jha • 2 months ago

here is the 0(nlogn) complexity using divide and conquer approach
http://ideone.com/eDWtc8

Reply • Share ›

Yogendra Singh Vimal • 3 months ago

```
#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define MAX 256

int lsubstring(char *s)

{

int count[MAX][2]={{0}};
```

```
int MLEN=0,CLEN=0,i,j,ptr=0;

for(i=0;*(s+i);i++)

{

if(count[*(s+i)][0]==0)
```

see more

Reply • Share ›

Harjit Singh • 5 months ago

This is wrong. We have to use array of size equal to string size in stead of 25

Reply • Share ›

navneet goel • 7 months ago

In the 2nd method program, run for the input string "geeksforkgeeks". here it s
longest non-repeating character substring is 7" though here longest non repea
geeks to k of fork).
Please correct me if i am wrong.

6 Reply • Share ›

chen navneet goel • 3 months ago

sforkge

1 Reply • Share ›

vikram • 7 months ago

reposting....
Code for printing the longest unique sub string. Also prints all the unique string
Please correct me, if something is wrong. Also please share if someone has b

```
#include "stdafx.h"
#include <iostream>
using namespace std;
```

```
int LongestUniqueSubString(char *str)
{
        if(str == 0)
                return 0;
        cout << "Input string = " << str << "\n";
        /*save the latest index at the character position*/
        int charIdx[256];
        for(int i=0; i < 256; i++) {
                charIdx[i] =-1;
        }
        /*current start and end of the unique sub string*/
```

see more

• Reply • Share ›

vikram • 7 months ago

Code for printing the longest unique sub strings.
Please correct me, if something wrong. Please share, if someone has tl

```
#include "stdafx.h"

#include <iostream>

using namespace std;

int LongestUniqueSubString(char *str)

{

if(str == 0)
```

```
return 0;
```

see more

Reply • Share ›

vishal • 9 months ago

O(n) time and O(26) space

```
 void initialise(int a[] , int n)
{
    int i =0;
    for(i = 0 ; i < n ;++i)
        a[i]  = 0;
}
int main()
{
        char arr[] = "GeeksforGeeks";
        int count[26] = {0};
        int i =0;
        int max = 0 , sum = 0;
        while( i < 13)
        {
                arr[i] = toupper(arr[i]);
                if( count[arr[i] - 65] == 0)
```

see more

1 • Reply • Share ›

timus ➦ vishal • 7 months ago

try this as input: "asdfgauvwx"

It'll give ans as 5, instead of 9

1 ^ | v • Reply • Share ›

vishal • 9 months ago

O(n) solution with o(26) space

```
 void initialise(int a[] , int n)
{
    int i =0;
    for(i = 0 ; i < n ;++i)
        a[i]  = 0;
}
int main()
{
        char arr[] = "GeeksforGeeks";
        int count[26] = {0};
        int i =0;
        int max = 0 , sum = 0;
        while( i < 13)
        {
                arr[i] = toupper(arr[i]);
                if( count[arr[i] - 65] == 0)
```

see more

^ | v • Reply • Share ›

Anonymous ↱ vishal • 9 months ago

Your algorithm fails for this case:
"GEEKSFORGEEKSXYZRLBHA"
Output is 7 but it should be 11
You should update max value every time new letter is added to NRCS

```
#include <iostream>
```

```
#include <cstring>
#include <cstdio>
using namespace std;

void initialise(int a[] , int n)
{
    int i =0;
    for(i = 0 ; i < n ;++i)
        a[i]  = 0;
}
int main()
{
```

see more

Reply • Share ›

vishal Anonymous • 9 months ago

Agreed I missed a case where the longest string is terminated
out

Reply • Share ›

vivek • 9 months ago

/*
#include
#include
void undo(int a[],char *s,int n)
{
int i;
for(i=0;i<=n;i++)

```
a[s[i]]=0;
}
int len(char *str)
{
int n =strlen(str);
int a[256]={0};
int i=0;
int maxcount=0;
int index=0;
int count=0;
while(i<n)
```

see more

Reply • Share ›

The King • 9 months ago

```
#include
#include
#define mchar 256

int main()
{
int i,j=0, len=0,max=0,*temp=NULL;
char str[20];
gets(str);

temp = (int *) calloc(sizeof(int) , mchar);

for(i=0;*(str+i);i++)
{
if(temp[*(str+i)] == 0)
{
len++;
```

```
temp[*(str+i)]=1;
}
```

see more

Reply • Share ›

me.abhinav • 11 months ago

Another O(n) solution.
<<expecting only="" lower="" case="" english="" alphabets="">>

```
 #include <iostream>
#define SIZ 100
#define index(c) c-'a'

using namespace std;

void process(char *str){
        int i, j, start, end, maxLen = -1, curLen;
        bool present[26];
        for(i=0 ; i<26 ; i++)
                present[i] = false;

        present[index(str[0])] = true;
        start = end = i = 0; curLen = j = 1;
        while(str[j]){
```

see more

Reply • Share ›

me.abhinav ➚ me.abhinav • 11 months ago

EXPECTING ONLY LOWER CASE ENGLISH ALPHABETS

Reply • Share ›

pritybhudolia • 11 months ago

I think this program is very simple and works in O(n) Complexity. I have used h
wrong. :)

```
#include<stdio.h>
#include<conio.h>
int longestUniqueSubsttr(char str[])
{
int hash[256]={0};
int start=0,max=0,i=0,j=0,end=0,begin=0;
for(i=0;i<strlen(str);i++) {="" if(hash[str[i]]="=1)" {="" start="end;" end="i;" if((en
{
max=end-start;
begin=start;
j=end;

}
}
hash[str[i]]=1;
```

see more

• Reply • Share ›

pritybhudolia ➚ pritybhudolia • 11 months ago

Previous code is not pasted properly. I have used hashing.Try this :)

```
 #include<stdio.h>
#include<conio.h>
int longestUniqueSubsttr(char str[])
{
     int hash[256]={0};
     int start=0,max=0,i=0,j=0,end=0,begin=0;
```

```
for(i=0;i<strlen(str);i++)
{
      if(hash[str[i]]==1)
      {
            start=end;
            end=i;
            if((end-start)>max)
            {
                max=end-start;
                begin=start;
```

see more

• Reply • Share ›

MVN Murthy ➚ pritybhudolia • 10 months ago

Take This Input - abcabcdabcdeabcdefga
Output should be - abcdefg

Your output is - abca

because you are not maintaining hash table.when encounter sc
hash table respective values.

Following is the Modification. Correct me if i m wrong.

#include

using namespace std;

void check(char *s,int n)
{
int *count,i,begin,end,m=-100,s_w;
s_w = begin = end = 0;
count = new int [256];

count = new int [256];

see more

Reply • Share ›

Akash ➚ pritybhudolia • 11 months ago

Hi, I guess the program returns length as 0 if all the characters because you are entering the if statement where the max varial there is a repeating character.

Also, in current scenario I guess you have missed to check wh are checking is included in length of NRCS or not. That is why i NRCS as 6 for the input string "GEEKS FORGEEKS".

I guess this might help. Please correct me if wrong.

```
/* Paste your code here (You may delete these lines if
```

Reply • Share ›

Prateek Sharma • a year ago

python code with o(n) time complexity based on approach similar to KMP algo

```
storingList =[]
def recursion(list1,tempArray,initial,i,len1):
    if i == len1:
        storingList.append(list1[initial:i])
        return 0
    else:
        if tempArray[list1[i]][0] !=1:
            tempArray[list1[i]] = [1,i]
            i = i+1
            recursion(list1,tempArray,initial,i,len1)
```

```
        else:
            storingList.append(list1[initial:i])
            initial = tempArray[list1[i]][1]+1
            tempArray[list1[i]][1]  =i
            i = i+1
            recursion(list1,tempArray,initial,i,len1)
```

see more

Reply • Share ›

keshav • a year ago

Each character need to refer its subsequent character for longest sub-string t
character has been found already then we need to check if its most previous i
string length of subsequent character.

```
 /* #include<stdio.h>
#include<conio.h>
#include<string.h>

int main()
{
    char str[100];
    int n,i,*ref,hash[256],max,Index;
    printf("enter the string\n");
    gets(str);
    n=strlen(str);
    ref= (int *)(malloc(n*(sizeof(int))));
    for(i=0;i<n;i++)
    hash[str[i]]=-1;
```

see more

Reply • Share ›

Code1101 • a year ago

```
    public int largestUniqueString(String s) {
       Set<Character> set = new HashSet<Character>();
       int[] w = new int[s.length()];
       for(int i=0; i<s.length(); i++) {
           if(set.contains(s.charAt(i))) set.clear();

           set.add(s.charAt(i));
               w[i] = set.size();
       }

       int max=-1;
       for(int i=0; i<w.length; i++) {
           if(w[i] > max) {
               max = w[i];
           }
       }
       return max;
   }
```

  • Reply • Share ›

rahul.titare • a year ago

```
int start=0,stop=0,longestStart=0,longestStop = 0;


            int[] counts = new int[65];

            int lastCheckedPosition = -1;

            String a = "GEEKSFORGEEKS".toUpperCase();
            counts[a.charAt(0) - 65] = 1;
```

```
            for(int i=1;i < a.length();i++){

                    if(counts[a.charAt(i)%65] == 0){

                            stop = i;

                            counts[a.charAt(i)%65] = i+1;

                    }else{
```

see more

Reply • Share ›

Sai Nikhil • a year ago

instead of

```
cur_len = i - prev_index;
```

the following would also work.

```
cur_len--;
```

Reply • Share ›

Sai Nikhil Sai Nikhil • a year ago

Also initialise curr_len to '0'

Reply • Share ›

Arun • a year ago

Java implementation for the same::

```
public static String findSubsString(String str){

HashSet set = new HashSet();

int max_curr = 0;
int max_overAll = 0;
int head = 0;
int tail = 0;
String longestSubstring = "";

while(tail < str.length()){

if(!set.contains(str.charAt(tail))){
max_curr++;
set.add(str.charAt(tail));
}else{
int i = 0;
```

see more

• Reply • Share ›

time pass • a year ago

the solution (linear time) gives output as 1 even in case of empty strings .. i th
begining of the function for n .. in case n is 0 , return 0

```
/* Paste your code here (You may delete these lines if not writing c
```

• Reply • Share ›

Saurabh Jain • 2 years ago

[sourcecode language="JAVA"]

import java.util.HashSet;
import java.util.Iterator;

```
import java.util.Iterator;
import java.util.Scanner;
import java.util.Set;

/**
*
* @author saurabh
*/
public class LongestSubstringWithoutRepeatingChars
{
String s;

public LongestSubstringWithoutRepeatingChars()
{
Scanner sc = new Scanner(System.in);
s = sc.nextLine();
```

see more

• Reply • Share ›

anubhav gupta • 2 years ago

[sourcecode language="C++"]

```
#include <iostream>
#include <map>
#include <set>
#include <algorithm>
using namespace std;

void compute(char *str,int *mins , int *maxs)
{
int begin,end,count=0,maxcount=-1;
map<char,int> hash;
hash.clear();
```

for(begin = 0,end = 0 ;end < strlen(str); end++){

hash[str[end]]++;

if(hash[str[end]]>1){

see more

• Reply • Share ›

Raghav • 2 years ago

A simple code as below would work for the O(n) solution

```
int retMaxLenSubString(char str[]){
    if(str == null || str.length==0)
            return 0;
    int charIndex[] = new int[256];
    for(int i=0;i<256;i++) {
      charIndex[i] = -1;
    }

    int maxLenStIndex = 0;
    int maxLen = 0;

    for(int i=0;i<str.length;i++) {
         int asciiValue = (int)str[i];
         if(index[asciiValue]>maxLenStIndex) {
             maxLenStIndex = index[asciiValue] + 1;
         }
         index[charIndex] = i;
         maxLen = max(maxLen, i-maxLenStIndex;
     }
}
```

}

kartikaditya • 2 years ago

[sourcecode language="C++"]
#include <iostream>
#include <stdio.h>
#include <string.h>

using namespace std;

void getLongestSubstringWithoutRepeatingChars(char* s) {
int n = strlen(s);
int lo = 0, hi = 0, t_lo = 0, t_hi = 0, max = 0, curr = 0;
int ht[256];
memset(ht, -1, 256 * sizeof(int));

for (int i = 0; i < n; ++i) {
if (ht[s[i]] != -1) {
for (; lo <= ht[s[i]]; ++lo) {
ht[s[lo]] = -1;
--curr;
}

see more

kartikaditya • 2 years ago

```
 #include <iostream>
#include <stdio.h>
#include <string.h>


using namespace std;
```

```
int getLongestSubstringWithUniqueChars(const char* s) {
    int n = strlen(s);
    bool ht[256];
    memset(ht, false, 256 * sizeof(bool));
    int max = -1, curr = 0;
    for (int i = 0; i < n; ++i) {
        if (ht[s[i]]) {
            curr = 0;
            memset(ht, false, 256 * sizeof(bool));
        }
        ++curr;
        ht[s[i]] = true;
```

see more

• Reply • Share ›

Rushi Agrawal • 2 years ago

Check the python code for the same. Looks quite elegant to me.

[sourcecode language="python"]

from string import ascii_lowercase

def maxss(s):
s = list(s)
maxlen = 0
maxsofar = 0
lastinstance = {}
for i in ascii_lowercase:
lastinstance[i] = -1
for i in range(len(s)):
if lastinstance[s[i]] < i - maxsofar:

```
if lastinstance[s[i]] < i - maxsofar:
lastinstance[s[i]] = i
maxsofar += 1
else:
maxsofar = i - lastinstance[s[i]]
lastinstance[s[i]] = i
if maxsofar > maxlen:
maxlen = maxsofar
return maxlen
```

• Reply • Share ›

Rushi Agrawal Rushi Agrawal • 2 years ago

Correction in the above code: The line [sourcecode language="python"

lastinstance is the dictionary to keep track of the last instance of the ch
works with smallcase alphabets, although can be scaled easily to large

• Reply • Share ›

vikram.kuruguntla • 2 years ago

Program to print the one of maximum sub string which has unique chars. Cha

Please correct me, if anything is wrong.

```
 #include "stdafx.h"
#include <iostream>
using namespace std;
#define NUM_CHARS 256

int _tmain(int argc, _TCHAR* argv[])
{
        char *input_string = "ABCabc";
        int input_length = strlen(input_string);
```

```
        char first_char_of_current_string = 0;

        char visited[NUM_CHARS];
        memset(visited, 0, sizeof(char) * NUM_CHARS);

        int current_start_ix = 0, final_start_ix = -1, final_end_ix =
```

see more

Reply • Share ›

syam ➦ vikram.kuruguntla • 2 years ago

program is wronge

Reply • Share ›

randy • 2 years ago

char str[] = "ABCDECQWERTYU";

The program does not work.

```
/* Paste your code here (You may delete these lines if not writing c
```

Reply • Share ›

GeeksforGeeks ➦ randy • 2 years ago

@randy: The program given in the post return 8 for "ABCDECQWERT
CQWERTYU is the longest string without repeating characters.

Reply • Share ›

randy ➦ GeeksforGeeks • 2 years ago

Yes It is correct, sorry for my mistake.

```
/* Paste your code here (You may delete these lines if
```

Reply • Share ›

itengineer • 2 years ago

```
 package com.sun.java.longestsubstring;


public class LongestSubstring
{
        private int visitedStartIndex = 0;
        private int previousIndex = 0;


        public static void main(String[] args)
        {
                LongestSubstring lsObject = new LongestSubstring();
                System.out.println("length = " + lsObject.findNRCSLen(
        }

        /**
         * @return  Non Repeating Character Substring length
         */
```

see more

^ | v • Reply • Share ›

itengineer ➚ itengineer • 2 years ago

Hey Geeks,

Forgot to mention that the code above uses JAVA language.

^ | v • Reply • Share ›

vkjk89 • 2 years ago

Here is one more code.

Venki/Kartik,
Plz suggest if anything wrong.

```
 #include<stdio.h>
#include<conio.h>
#include<string.h>
int longest(char * str ,int *pos)
{
int len;
int i,cur,max,*vis;
vis=(int*)malloc(sizeof(int)*256);
for(i=0;i<256;i++)
vis[i]=-1;
cur=max=0;
len=strlen(str);
for(i=0;i<len;i++)
{
```

see more

• Reply • Share ›

AKKICOOL • 2 years ago

Can you please give a case for

"when we are changing the NRCS, we should also check whether length of th
max_len or not" in else part

```
/* Paste your code here (You may delete these lines if not writing c
```

• Reply • Share ›

Arpit Gupta • 2 years ago

In this article complexity of brute force algorithm is written as exponential but it

be n*(n+1)/2 substrings and considering each of atmost length n the complexi
know how can it be exponential.

```
/* Paste your code here (You may delete these lines if not writing c
```

• Reply • Share ›

Load more comments

Subscribe Add Disqus to your site