Remove all duplicates from the input string.

Below are the different methods to remove duplicates in a string.

METHOD 1 (Use Sorting)

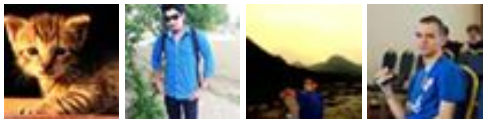
Algorithm:

```
1) Sort the elements.
2) Now in a loop, remove duplicates by comparing the
    current character with previous character.
3)  Remove extra characters at the end of the resultant string.
```

Example:

```
Input string:  geeksforgeeks
1) Sort the characters
   eeeefggkkosss
2) Remove duplicates
    efgkosgkkosss
3) Remove extra characters
     efgkos
```

Note that, this method doesn't keep the original order of the input string. For example, if we are to remove duplicates for geeksforgeeks and keep the order of characters same, then output should be geksfor, but above function returns efgkos. We can modify this method by storing the original order. METHOD 2 keeps the order same.



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

Implementation:

```
# include <stdio.h>
# include <stdlib.h>

/* Function to remove duplicates in a sorted array */
char *removeDupsSorted(char *str);

/* Utitlity function to sort array A[] */
void quickSort(char A[], int si, int ei);

/* Function removes duplicate characters from the string
   This function work in-place and fills null characters
   in the extra space left */
char *removeDups(char *str)
{
  int len = strlen(str);
  quickSort(str, 0, len-1);
  return removeDupsSorted(str);
}

/* Function to remove duplicates in a sorted array */
char *removeDupsSorted(char *str)
{
  int res_ind = 1, ip_ind = 1;

  /* In place removal of duplicate characters*/
  while(*(str + ip_ind))
  {
    if(*(str + ip_ind) != *(str + ip_ind - 1))
    {
      *(str + res_ind) = *(str + ip_ind);
      res_ind++;
    }
    ip_ind++;
  }

  /* After above step string is stringiittg.
     Removing extra iittg after string*/
  *(str + res_ind) = '\0';

  return str;
}

/* Driver program to test removeDups */
int main()
{
```

Geometric Algorithms

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

```
  char str[] = "eeeefggkkosss";
  printf("%s", removeDups(str));
  getchar();
  return 0;
}

/* FOLLOWING FUNCTIONS ARE ONLY FOR SORTING
    PURPOSE */
void exchange(char *a, char *b)
{
  char temp;
  temp = *a;
  *a   = *b;
  *b   = temp;
}

int partition(char A[], int si, int ei)
{
  char x = A[ei];
  int i = (si - 1);
  int j;

  for (j = si; j <= ei - 1; j++)
  {
    if(A[j] <= x)
    {
      i++;
      exchange(&A[i], &A[j]);
    }
  }
  exchange (&A[i + 1], &A[ei]);
  return (i + 1);
}

/* Implementation of Quick Sort
A[] --> Array to be sorted
si  --> Starting index
ei  --> Ending index
*/
void quickSort(char A[], int si, int ei)
{
  int pi;    /* Partitioning index */
  if(si < ei)
  {
    pi = partition(A, si, ei);
    quickSort(A, si, pi - 1);
    quickSort(A, pi + 1, ei);
```

```
    }
}
```

Time Complexity: O(nlogn) If we use some nlogn sorting algorithm instead of quicksort.

METHOD 2 (Use Hashing)

Algorithm:

```
1: Initialize:
    str  =  "test string" /* input string */
    ip_ind =  0          /* index to  keep track of location of next
                             character in input string */
    res_ind  =  0         /* index to  keep track of location of
                            next character in the resultant string */
    bin_hash[0..255] = {0,0, ….} /* Binary hash to see if character is
                                      already processed or not */
2: Do following for each character *(str + ip_ind) in input string:
             (a) if bin_hash is not set for *(str + ip_ind) then
                  // if program sees the character *(str + ip_ind) first time
                        (i)  Set bin_hash for *(str + ip_ind)
                        (ii)  Move *(str  + ip_ind) to the resultant string.
                             This is done in-place.
                        (iii) res_ind++
             (b) ip_ind++
  /* String obtained after this step is "te sringng" */
3: Remove extra characters at the end of the resultant string.
  /*  String obtained after this step is "te sring" */
```

Implementation:

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

/* Function removes duplicate characters from the string
   This function work in-place and fills null characters
```


Recent Comments

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

Backtracking | Set 7 (Sudoku) · 20 minutes ago

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

Flipkart Interview | Set 6 · 40 minutes ago

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

Software Engineering Lab, Samsung Interview | Set 2 · 40 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

```
   in the extra space left */
char *removeDups(char *str)
{
  bool bin_hash[NO_OF_CHARS] = {0};
  int ip_ind = 0, res_ind = 0;
  char temp;

  /* In place removal of duplicate characters*/
  while(*(str + ip_ind))
  {
    temp = *(str + ip_ind);
    if(bin_hash[temp] == 0)
    {
        bin_hash[temp] = 1;
        *(str + res_ind) = *(str + ip_ind);
        res_ind++;
    }
    ip_ind++;
  }

  /* After above step string is stringiittg.
     Removing extra iittg after string*/
  *(str+res_ind) = '\0';

  return str;
}

/* Driver program to test removeDups */
int main()
{
    char str[] = "geeksforgeeks";
    printf("%s", removeDups(str));
    getchar();
    return 0;
}
```

Time Complexity: O(n)

NOTES:

* It is assumed that number of possible characters in input string are 256. NO_OF_CHARS should be changed accordingly.
* calloc is used instead of malloc for memory allocations of counting array (count) to initialize allocated memory to '\0'. malloc() followed by memset() could also be used.
* Above algorithm also works for an integer array inputs if range of the integers in array is given. Example problem is to find maximum occurring number in an input array given that the input

array contain integers only between 1000 to 1100

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.

52 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

kinshuk chandra • 6 days ago

if characters lies in the range a...z, then we can use following O(n) method wit

```
public static void removeDuplicates(char[] str) {
int map = 0;
for (int i = 0; i < str.length; i++) {
if ((map & (1 << (str[i] - 'a'))) > 0) // duplicate detected
str[i] = 0;
else // add unique char as a bit '1' to the map
map |= 1 << (str[i] - 'a');
}
}
```

(borrowed from kodeknight)

• Reply • Share ›

Anurag • 3 months ago

```
#include<iostream.h>

#include<conio.h>

#include<string.h>

#include<stdio.h>

int i,j;

void no_rep(char s[50])

{
```

int l=strlen(s);

int flag[l];

for(i=0;i<l-1;i++) {="" for(j="i+1;j<l;j++)" if(s[i]="=s[j])" flag[j]="0;" }="" for(i="0;i
}="" void="" main()="" {="" clrscr();="" char="" str[50];="" cout<<"enter="" string
cout<<endl<<"string="" without="" repetitions="" is="" :";="" no_rep(str);="" get

• Reply • Share ›

Gaurav M • 3 months ago

Java program - in place

public static void main(String[] args)
{
String test="geeksforgeeks";
char[] charArray = test.toCharArray();
int uniqueIndex=0;
boolean[] recorder = new boolean[256];
for(int i =0 ; i<chararray.length;i++) {="" if(recorder[chararray[i]])="" {="" do="" r
recorder[chararray[i]]="true;" chararray[uniqueindex]="charArray[i];" uniqueinde
i="0;i<uniqueIndex;i++)" {="" system.out.print(chararray[i]);="" }="" }="">

• Reply • Share ›

ravi singh • 6 months ago

Remove duplicate char from a c++ string:

```
string removeDups(string str)
{
        map<char,int> my_map;
        int i=0, len = str.length()-1;
        while(i<=len)
        {
```

```
                if(!my_map[str[i]])
                {
                        my_map[str[i]] = 1;
                        i++     ;
                }
                else
                {
                        str.erase(i,1);
                        len--;
                }
        }
        return str;
}
```

1 • Reply • Share ›

Guest • 7 months ago

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

void remdupstr(char str[])
{
    int i,n=strlen(str),k=0,j;
    char *str2=(char *)malloc(sizeof(char)*n);
    for(i=0;i<n;) {="" str2[k]="str[i];" j="i+1;" while(j<n&&str[j]="
```

1 • Reply • Share ›

Avinash Nigam • 7 months ago

Java implementation of this algo is as following

```
private static String removeDuplicates(char[] charArray)

        {

                int inputIndex = 0, resiudalIndex = 0;

                Map<string, integer=""> hash = new HashMap<string, int

                while (inputIndex < charArray.length)

                {

                        if (!hash.containsKey(String.valueOf(charArray

                        {
```

see more

1 • Reply • Share ›

Mohini • 7 months ago

Following method gives only unique char of the string in O(n+k) time, where k the string but modifies the resulting string in the end. Do leave your comments better.

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

using namespace std;

void RemoveDups(char * str)
{
int tail = 0,i,j;
```

int found;
for(i = 0; str[i]!='\0';i++)
{
j =0; found = 0;
while(j<tail) {="" if(str[i]="=" str[j])="" {="" found="1;" break;="" }="" j++;="" }="" i
str[tail]="str[i];" if(found="=" 0)="" tail++;="" }="" for(int="" i="0;i<tail;i++)" cout
char="" str[]="geeksforgeeks" ;="" removedups(str);="" return="" 0;="" }="">

• Reply • Share ›

Arvind • 9 months ago

please note: isPresent] need to be isPresent[text language="[i"][/text]]

• Reply • Share ›

Arvind • 9 months ago

#include
#include
#define MAX 256

using namespace std;

void removeDuplicates(string text){
int isPresent[MAX]={0};
for(int i=0;i<text.length();i++){
if(isPresent[text language="[i"][/text]]==0){
cout<<text[i];
isPresent[text language="[i"][/text]]=1;
}
}
}

int main(){
string text="geeksforgeeks";

removeDuplicates(text);
return 0;
}

1 • Reply • Share ›

Zoha Khan • 9 months ago

can anybody expalin how did hashing work in general and in the above prograr

• Reply • Share ›

Unique • 10 months ago

```
/* public class RemoveDuplicates {
        public static void main(String[] args) {
                String s = "geekdforgeeks";
                String res = "";
                int count[] = new int[256];
                for (int i = 0; i < s.length(); i++) {
                        count[s.charAt(i)]++;
                        if (count[s.charAt(i)] > 0 && count[s.charAt(
                                res += s.charAt(i);
                }
                System.out.println(res);
        }
} */
```

• Reply • Share ›

kaushik • 11 months ago

```
//kaushik sahu

#include<stdio.h>
#include<conio.h>
```

```
int main ()
{
    char *str1 = "geeksforgeeks";
    char str2[50];
    int byte[26] = {0};
    int i=0,j=0;

    for(i=0;*(str1+i) != '&#092&#048';i++)
    {
        if(byte[*(str1+i)-97] == 0)
        {
            byte[*(str1+i) - 97] = 1;
            str2[j] = *(str1+i);
```

see more

• Reply • Share ›

Sahil → kaushik • 9 months ago

was there any need of second string?? this way, we can save the men

#include
#include

int main ()
{
char *str1 = "geeksforgeeks";

int byte[26] = {0};
int i=0;
clrscr();
for(i=0;*(str1+i) != '';i++)
{
if(byte[*(str1+i)-97] == 0)

```
{

byte[*(str1+i) - 97] = 1;
printf("%c",*(str1+i));

}
}

getch();
return 0;
}
```

1 • Reply • Share ›

Pratyay Pandey • 11 months ago

```
#include<stdio.h>
#include<stdlib.h>
char *removeDuplicate(char *str).
{
int i;
int k=0;
int *count = (int *)calloc(sizeof(int), 256);.
for (i=0;*(str+i);i++)
{
if (++count[*(str+i)] == 1).
{
*(str + k) = *(str + i);.
k++;
}
}
*(str + k) = &#039&#039;.
return str;
}
```

int main()
{
char str[] = {"This is a test string"};.
printf("%s", removeDuplicate(str));
getchar();
return;
}

• Reply • Share ›

miandfdy • 11 months ago

Please tell time and space complexity of my code

```
 /* Paste your code here (You may delete these lines if not writing c
import java.util.Arrays;
import java.util.Iterator;
import java.util.LinkedList;




public class Extthread
{
        public static void main(String[] args)
        {
                String s="inputing";char[] c=s.toCharArray();Arrays.s
                        LinkedList l=new LinkedList();
                                for (int i = 0; i < c.length; i++)
                                {
                                        l.add(c[i]);
```

see more

• Reply • Share ›

Ronak Hingar ➦ miandfdy • 3 months ago

Your space complexity is O(n) because of the linkedlist and time comp

• Reply • Share ›

ROHIT SINGHAL • a year ago

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

int main()
{
        static int stk[150],i,len;
        char a[150];
        printf("Enter the string\n");
        scanf("%s",a);
        len=strlen(a);
        for(i=0;i<len;i++)
        {
                if(stk[a[i]]==0)
                        stk[a[i]]++;
        }
        printf("Output is :");
        for(i=0;i<len;i++)
        {
```

see more

• Reply • Share ›

hemanthreddy • a year ago

It can be solved easily using bit vector

```
 //
//
```

```
#include "stdio.h"
#include "malloc.h"

#pragma warning(disable : 4996)    //to use gets function in //visual

int main()
{
        char *str=(char *)malloc(20*sizeof(char)),*write,*p;
        int *bitvector=(int *)calloc(8,sizeof(int));
        gets(str);
        write=str;
        p=str;
        while(*str)
```

see more

• Reply • Share ›

yelnatz ➦ hemanthreddy • a year ago

Can you explain the logic when you do this:

```
(bitvector[*str>>5] & (1<<(*str%32)))
```

Thanks.

• Reply • Share ›

abhishek08aug • a year ago

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

```
char * remove_duplicates(char * str) {
  int * char_count=(int *)calloc(sizeof(int), NO_OF_CHARS);
  int current_copy_index=0;
  char * temp=str;
  while(*temp!='&#92&#48') {
    if(*(char_count+*temp)==0) {
      *(char_count+*temp)=1;
      *(str+current_copy_index)=*temp;
      current_copy_index++;
    }
    temp++;
  }
  *(str+current_copy_index)='&#92&#48';
```

see more

• Reply • Share ›

itengineer • a year ago

[sourcecode language="language="Java"]
package com.rsquares.removechars;

public class CharacterRemover
{
public static int[] asciiChars = new int[256];

public static void main(String[] args)
{
CharacterRemover charRemover = new CharacterRemover();
charRemover.removeDuplicates("aaabbbccnnmmjjkkllooooouuuy");
}

public void removeDuplicates(String original)

```
{
setMarker(original);

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

see more

Reply • Share ›

PG • a year ago

Why are we always allocating a dynamic memory for count array. We can allc
allocate dynamic memory everytime we have to free memory to avoid memor
should be allocated. correct me if i am wrong.

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

Reply • Share ›

Rahul Kumar • a year ago

Hash will take smaller space.!

Reply • Share ›

Rahul Kumar • a year ago

all characters can be in range of uint8_t 0 to 256 max. Thats why. He is keepir
!

Reply • Share ›

Pradeep Kumar Choubey • a year ago

terrific logic dear.....but please explain it....please....

Reply • Share ›

Tushar Gaurav • a year ago

```
int main()
{
char a[50];
int b[256]={0};
int i=0;
clrscr();
printf("Enter string:n");
gets(a);
printf("n");
while(a[i]!=&#039&#039)
{
if(b[a[i]]>0)
{
b[a[i]]++;
}
else
{ b[a[i]]++;.
printf("%c", a[i]);
}
i++;
}
system("pause");
}
```

• Reply • Share ›

JP • 2 years ago

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

• Reply • Share ›

pradeep c.r • 2 years ago

```
#include
#include
int main()
{
char a[]="sssssssssssshhhhhhh";
int i,j,pos,k=0;
clrscr();
i=0;
while(a[i]!=''){
k++;
i++;
}
for(i=0;i<k;i++){
for(j=i+1;j<k;j++){
if(a[i]==a[j]){
pos=j;
a[i]=a[pos];
while(a[pos]!=&#039&#039){
```

see more

• Reply • Share ›

manish sahu • 2 years ago

```
#include
#include
int main()
{
char s1[30]="geeksforgeeks";
char s2[30];
int i,j;
int *count=(int *)calloc(sizeof(int),256);
for(i=0;s1[i];i++)
```

```
count[s1[i]]++;
}
for(i=0,j=0;s1[i];i++)
{
if(count[s1[i]]>0)
{
s2[j++]=s1[i];
count[s1[i]]=0;
}
}
s2[j]='';
printf("%s",s2);
getch();
return 0;
}
```

• Reply • Share ›

dejavu • 2 years ago

1.Make a count array and store the count of the characters in the string.
2.Iterate through the string and if count>0 the add it in the output string and set that it doesn't reappear in the string.

Implementation:

```
/* public class RemoveDuplicates {

        int[] count = new int[256];
        public String charPresent(String s)
        {
                String unique = "";
                //collect the count for characters
```

```
for(int i=0;i<s.length();i++)
{
        count[s.charAt(i)]++;
}
//iterate through the string and
```

see more

Reply • Share ›

tuhin • 2 years ago

```
#include
#include
#include
int count[256];

void makecount()
{ int i=0;
for(i=0;i<256;i++)
{
count[i]=0;
}
}

void removeduplicates(char *s)
{
makecount();
int i,j=0;
for(i=0;*(s+i)!=&#039&#039;i++)
{
```

see more

Reply • Share ›

```
 /* Paste your code here (You may delete these lines if not writing co
#include<stdio.h>
void remove_duplicates(char *c)
{
    printf("\nthe new string is:");
    int arr[256],i;
    for(i=0;i<256;i++)
    arr[i]=-1;
    for(i=0;c[i]!='&#092&#048';i++)
    {
        if(arr[c[i]]==-1)
        {
            arr[c[i]]=i;
            printf("%c",c[i]);
        }
    }
}
main()
{
    char str[20];
    printf("\nenter the string:");
    gets(str);
    remove_duplicates(str);
}
 */
```

• Reply • Share ›

venky • 2 years ago

#include

#include

int main()
{
char str[50];
char res[50];
scanf("%s",str);
int A[26]={0},i=0,k=0,len;
len=strlen(str);
for(i=0;i<len;i++)
{
if(A[str[i]-97]<1)
res[k++]=str[i];
A[str[i]-97]++;

}
res[k]=&#039'
printf("%s\n",res);

[sourcecode language="C"]
/* Paste your code here (You may delete these lines if not writing code) */

• Reply • Share ›

Aravindan • 2 years ago

```
 /* #include<stdio.h>
#include<conio.h>
void main()
{
        char a[100],b[100];
        int i,j,l,k=0;
        clrscr();
        for(i=0;(a[i]=getchar())!='$';i++);
        a[i]='&#092&#048';
```

```
        for(i=0;i=0;a[i];i++)
        {
        l++;
        }

        for(i=0;a[i];i++)
        {
                for(j=0;j!=l;j++)
```

see more

• Reply • Share ›

sudhansu sekhar nayak • 2 years ago

```
 /* Paste your code here (You may delete these lines if not writing co
[/import java.io.*;
/*
Enter String is :sudhansu
it remove su
print sudhan
*/
import java.util.*;
class RemoveDuplicate{
   public static void main(String[] args){
                try     {
                DataInputStream in = new DataInputStream(System.in);
                System.out.print(&quot;n n Enter the Sentence : &quot;
                String s = in.readLine();
                boolean flag = false;
                char c[] = s.toCharArray();
            Set set = new HashSet();
                for(int i = 0 ; i &amp;lt; c.length; i++){
```

see more

• Reply • Share ›

Arnab Sen Gupta • 2 years ago

```
 int main()
{
   char str[] = "geeksforgeeks";
   bool *a = (bool *)calloc(sizeof(bool), NO_OF_CHARS);
   int len = strlen(str);
   char *b = (char *)calloc(sizeof(char),len); //max size possible wh
   int i=0,k=0;
   for(i=0;i<len;i++)
   {
                    if(a[str[i]] == 0)
                    {
                                b[k]=str[i];
                                a[str[i]] = 1;
                                k++;
                    }
   }
   printf("%s",b);
   getchar();
   return 0;
}
```

1 • Reply • Share ›

arun ➦ Arnab Sen Gupta • a year ago

Good one ..

• Reply • Share ›

Arnab Sen Gupta ➦ Arnab Sen Gupta • 2 years ago

I wrote the calloc syntax wrong!! they should be as follows

```
 bool *a = (bool *)calloc(NO_OF_CHARS,sizeof(bool));
char *b = (char *)calloc(len,sizeof(char));
```

Reply • Share ›

Naveen Makwana • 3 years ago

```
 int main()
{
        char * p="";
        printf("enter the string :");
        gets(p);
        int i,j=1,k=1;
        for(i=0;p[i] ;i++){
                while(p[j]){
                        if(p[i]!=p[j]){
                                p[i+k]=p[j];
                                k++;
                        }
                        j++;
                }
                p[i+k]='&#092&#048';
                j=i+1;
                k=1;
        }
        printf(p);
return 0;

}
```

Reply • Share ›

```
#include

int main()
{
char str[]="aaaabbbccdbdbcd",str1[30];
int flag[127]={0};
int i,rem,j;
for(i=0,j=0;str[i]!='';i++)
{
rem=str[i]%'A';
if(flag[rem]==0)
{
str1[j++]=str[i];
flag[rem]=1;
}
}
str1[j]='';
printf("%s\n",str1);
}
```

• Reply • Share ›

intel • 3 years ago

can anyone tell me whats the use of the 3rd step in both the methods.. isn't it

• Reply • Share ›

sujay • 3 years ago

I found this solution online

[sourcecode language="java"]
public void removeDups(String str) {

```
// get the char array
char[] chArr = str.toCharArray();
System.out.println(chArr);
int tail = 1;
for(int i =0; i< chArr.length; i++)
{
int j;
for(j =0;j< tail; j++) {
if (chArr[i] == chArr[j])
break; // break if we find duplicate.
}
// if j reachs tail..we did not break, which implies this char at pos i
// is not a duplicate. So we need to add it our "unique char list"
// we add it to the end, that is at pos tail.
```

see more

• Reply • Share ›

picka • 3 years ago

we can use count array tech,
1. form a count array in d order of given string
2.print d indexes of d count array

Example,Geeks
a['g']=1
a['e']=2
a['k']=1
a['s']=1

jus print indexes,,geks

correct me if am wrong???

Venki → picka • 3 years ago

@picka, although count array works, printing characters corresponds t
of characters in string.

Or if the idea is to set up the count array in one pass across the string,
print only those characters corresponding to array value 1. However, it
above post.

• Reply • Share ›

yeskay • 3 years ago

```
 #include<string.h>

typedef struct{
        unsigned char flag:1;
}CHARMAP;

CHARMAP chars[26];

int main(){

        char* removeAllDuplicates(char*);
        char str[]="geeksforgeeks";
        clrscr();
        printf("Original String is %s\n",str);
        printf("Altered String is %s\n",removeAllDuplicates(str));
getch();
return 0;
}
```

see more

• Reply • Share ›

rajcools • 3 years ago

res_ind = 0 /* index to keep track of location of next
character in i/p string */
ip_ind = 0 /* index to keep track of location of
next character in the resultant string */

this is a typo. explanation of ip_end and res_ind has been reversed

• Reply • Share ›

GeeksforGeeks ➦ rajcools • 3 years ago

@rajcools: Thanks for pointing out the typo. We have corrected it.

• Reply • Share ›

divyaC • 4 years ago

```
removeDuplicates(char *str){
        bool setArray[256]={false};
        int i=0;
        while(str){
                if(!setArray[*str]){
                        setArray[*str]=true;
                        str[i]=*str;
                        i++;
                }
                str++;
        }
        str[i]='';
}
```

• Reply • Share ›

Rajendra Kumar Uppal • 4 years ago

1. Sorting takes O(nlogn) time.

2. Other hashing methods are dependent upon input string size, you are expe
be <= 256 characters.

What about using following algorithm, which:
1. platform or language independent;
2. O(n) time complexity worst-case;
3. O(1) space complexity no matter if your input string is a 10kb text file.

Algorithm:
Step 1. maintain two temporary pointers at starting char of the input string (ac
first pointer at first char and second pointer at next char;
Step 2. take 26 booleans or bits (in of c++), keep setting bits corresponding to
97
Step 3. when you see a bit is already set, then move forward, else copy that c
Step 4. Repeat until string finishes.

Rajendra.

• Reply • Share ›

geeksforgeeks • 5 years ago

@Snehal: You are right. We have added a note for this. Also, we have replace
very much. Keep it up!!

• Reply • Share ›

Load more comments

Subscribe Add Disqus to your site