An in-place algorithm for String Transformation

Given a string, move all even positioned elements to end of string. While moving elements, keep the relative order of all even positioned and odd positioned elements same. For example, if the given string is "a1b2c3d4e5f6g7h8i9j1k2l3m4", convert it to "abcdefghijklm1234567891234" in-place and in O(n) time complexity.

Below are the steps:

1. Cut out the largest prefix sub-string of size of the form 3^k + 1. In this step, we find the largest non-negative integer k such that 3^k+1 is smaller than or equal to n (length of string)

2. Apply cycle leader iteration algorithm (it has been discussed below), starting with index 1, 3, 9…… to this sub-string. Cycle leader iteration algorithm moves all the items of this sub-string to their correct positions, i.e. all the alphabets are shifted to the left half of the sub-string and all the digits are shifted to the right half of this sub-string.

3. Process the remaining sub-string recursively using steps#1 and #2.

4. Now, we only need to join the processed sub-strings together. Start from any end (say from left), pick two sub-strings and apply the below steps:

….**4.1** Reverse the second half of first sub-string.
….**4.2** Reverse the first half of second sub-string.
….**4.3** Reverse the second half of first sub-string and first half of second sub-string together.

5. Repeat step#4 until all sub-strings are joined. It is similar to k-way merging where first sub-string is joined with second. The resultant is merged with third and so on.

Let us understand it with an example:

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

Please note that we have used values like 10, 11 12 in the below example. Consider these values as single characters only. These values are used for better readability.

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
a 1 b 2 c 3 d 4 e 5 f  6  g  7  h  8  i  9  j  10 k  11 l  12 m  13
```

After breaking into size of the form 3^k + 1, two sub-strings are formed of size 10 each. The third sub-string is formed of size 4 and the fourth sub-string is formed of size 2.

```
0 1 2 3 4 5 6 7 8 9
a 1 b 2 c 3 d 4 e 5

10 11 12 13 14 15 16 17 18 19
f  6  g  7  h  8  i  9  j  10

20 21 22 23
k  11 l  12

24 25
m  13
```

After applying cycle leader iteration algorithm to first sub-string:

```
0 1 2 3 4 5 6 7 8 9
a b c d e 1 2 3 4 5

10 11 12 13 14 15 16 17 18 19
f  6  g  7  h  8  i  9  j  10

20 21 22 23
k  11 l  12

24 25
m  13
```

After applying cycle leader iteration algorithm to second sub-string:

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

```
0 1 2 3 4 5 6 7 8 9
a b c d e 1 2 3 4 5

10 11 12 13 14 15 16 17 18 19
f  g  h  i  j  6  7  8  9  10

20 21 22 23
k  11 l  12

24 25
m 13
```

After applying cycle leader iteration algorithm to third sub-string:

```
0 1 2 3 4 5 6 7 8 9
a b c d e 1 2 3 4 5

10 11 12 13 14 15 16 17 18 19
f  g  h  i  j  6  7  8  9  10

20 21 22 23
k  l  11 12

24 25
m  13
```

After applying cycle leader iteration algorithm to fourth sub-string:

```
0 1 2 3 4 5 6 7 8 9
a b c d e 1 2 3 4 5

10 11 12 13 14 15 16 17 18 19
f  g  h  i  j  6  7  8  9  10

20 21 22 23
```

```
20 21 22 23
k  l  11 12

24 25
m  13
```

Joining first sub-string and second sub-string:
1. Second half of first sub-string and first half of second sub-string reversed.

```
0 1 2 3 4 5 6 7 8 9
a b c d e 5 4 3 2 1           <--------- First Sub-string

10 11 12 13 14 15 16 17 18 19
j  i  h  g  f  6  7  8  9  10  <--------- Second Sub-string

20 21 22 23
k  l  11 12

24 25
m  13
```

2. Second half of first sub-string and first half of second sub-string reversed together(They are merged, i.e. there are only three sub-strings now).

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
a b c d e f g h i j 1  2  3  4  5  6  7  8  9  10

20 21 22 23
k  l  11 12

24 25
m  13
```

Recent Comments

affiszerv Your example has two 4s on row 3, that's why it...
Backtracking | Set 7 (Sudoku) · 16 minutes ago

RVM Can someone please elaborate this Qs from above...
Flipkart Interview | Set 6 · 35 minutes ago

Vishal Gupta I talked about as an Interviewer in general,...
Software Engineering Lab, Samsung Interview | Set 2 · 36 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

Joining first sub-string and second sub-string:
1. Second half of first sub-string and first half of second sub-string reversed.

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
a b c d e f g h i j 10 9  8  7  6  5  4  3  2  1 <--------- First Sub-string

20 21 22 23
l  k  11 12                                      <--------- Second Sub-string

24 25
m  13
```

2. Second half of first sub-string and first half of second sub-string reversed together(They are merged, i.e. there are only two sub-strings now).

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
a b c d e f g h i j k  l  1  2  3  4  5  6  7  8  9  10 11 12

24 25
m  13
```

Joining first sub-string and second sub-string:
1. Second half of first sub-string and first half of second sub-string reversed.

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
a b c d e f g h i j k  l  12 11 10 9  8  7  6  5  4  3  2  1 <----- First Sub-string

24 25
m  13   <----- Second Sub-string
```

2. Second half of first sub-string and first half of second sub-string reversed together(They are merged, i.e. there is only one sub-string now).

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
```

```
a b c d e f g h i j k  l  m  1  2  3  4  5  6  7  8  9  10 11 12 13
```

Since all sub-strings have been joined together, we are done.

How does cycle leader iteration algorithm work?

Let us understand it with an example:

```
Input:
0 1 2 3 4 5 6 7 8 9
a 1 b 2 c 3 d 4 e 5

Output:
0 1 2 3 4 5 6 7 8 9
a b c d e 1 2 3 4 5

Old index    New index
0               0
1               5
2               1
3               6
4               2
5               7
6               3
7               8
8               4
9               9
```

Let len be the length of the string. If we observe carefully, we find that the new index is given by below formula:

```
if( oldIndex is odd )
        newIndex = len / 2 + oldIndex / 2;
else
        newIndex = oldIndex / 2;
```

So, the problem reduces to shifting the elements to new indexes based on the above formula.

Cycle leader iteration algorithm will be applied starting from the indices of the form 3^k, starting with k = 0.

Below are the steps:

1. Find new position for item at position i. Before putting this item at new position, keep the back-up of element at new position. Now, put the item at new position.

2. Repeat step#1 for new position until a cycle is completed, i.e. until the procedure comes back to the starting position.

3. Apply cycle leader iteration algorithm to the next index of the form 3^k. Repeat this step until 3^k < len.

Consider input array of size 28:

The first cycle leader iteration, starting with index 1:

1->14->7->17->22->11->19->23->25->26->13->20->10->5->16->8->4->2->1

The second cycle leader iteration, starting with index 3:

3->15->21->24->12->6->3

The third cycle leader iteration, starting with index 9:

9->18->9

Based on the above algorithm, below is the code:

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

// A utility function to swap characters
void swap ( char* a, char* b )
{
    char t = *a;
    *a = *b;
    *b = t;
}
```

```
// A utility function to reverse string str[low..high]
void reverse ( char* str, int low, int high )
{
    while ( low < high )
    {
        swap( &str[low], &str[high] );
        ++low;
        --high;
    }
}

// Cycle leader algorithm to move all even positioned elements
// at the end.
void cycleLeader ( char* str, int shift, int len )
{
    int j;
    char item;

    for (int i = 1; i < len; i *= 3 )
    {
        j = i;

        item = str[j + shift];
        do
        {
            // odd index
            if ( j & 1 )
                j = len / 2 + j / 2;
            // even index
            else
                j /= 2;

            // keep the back-up of element at new position
            swap (&str[j + shift], &item);
        }
        while ( j != i );
    }
}

// The main function to transform a string. This function mainly uses
// cycleLeader() to transform
void moveNumberToSecondHalf( char* str )
{
    int k, lenFirst;

    int lenRemaining = strlen( str );
    int shift = 0;

    while ( lenRemaining )
    {
        k = 0;

        // Step 1: Find the largest prefix subarray of the form 3^k +
        while ( pow( 3, k ) + 1 <= lenRemaining )
            k++;
        lenFirst = pow( 3, k - 1 ) + 1;
        lenRemaining -= lenFirst;

        // Step 2: Apply cycle leader algorithm for the largest subarr
        cycleLeader ( str, shift, lenFirst );

        // Step 4.1: Reverse the second half of first subarray
        reverse ( str, shift / 2, shift - 1 );

        // Step 4.2: Reverse the first half of second sub-string.
        reverse ( str, shift, shift + lenFirst / 2 - 1 );

        // Step 4.3 Reverse the second half of first sub-string and fi
        // half of second sub-string together
        reverse ( str, shift / 2, shift + lenFirst / 2 - 1 );

        // Increase the length of first subarray
        shift += lenFirst;
    }
}

// Driver program to test above function
int main()
{
    char str[] = "a1b2c3d4e5f6g7";
    moveNumberToSecondHalf( str );
    printf( "%s", str );
    return 0;
}
```

Click here to see various test cases.

Notes:

1. If the array size is already in the form 3^k + 1, We can directly apply cycle leader iteration algorithm. There is no need of joining.

2. Cycle leader iteration algorithm is only applicable to arrays of size of the form 3^k + 1.

How is the time complexity O(n) ?
Each item in a cycle is shifted at most once. Thus time complexity of the cycle leader algorithm is O(n). The time complexity of the reverse operation is O(n). We will soon update the mathematical proof of the time complexity of the algorithm.

Exercise:
Given string in the form “abcdefg1234567″, convert it to “a1b2c3d4e5f6g7″ in-place and in O(n) time complexity.

References:

A Simple In-Place Algorithm for In-Shu?e.

__Aashish Barnwal.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
- 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)

8 Tweet 1 0

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

74 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

Teegan. • 2 months ago

The algo in the paper is actualy different, which affects the time complexity. Yo
and then do the cycle leader for the first part etc.

Reply • Share ›

Careegan Teegan. • 2 months ago

Yes, as given the algorithm is not O(n). It is possibly Omega(n logn).

Reply • Share ›

Careegan Careegan • 2 months ago

(The paper has it right, though)

Reply • Share ›

Saurabh • 2 months ago

Since the input to be segregated is a String like "a5b6c7d8" we can use String
numbers from the string and append them at the end of the string in a single fo

Time complexity: O(n)
space complexity: O(1)

Example:

i here denotes the index of the character in the string

i=1 -- input: a5b6c7d8 (Append 5 to the end of string) output: ab6c7d85
i=2 -- input: ab6c7d85 (Append 6 to the end of string) output: abc7d856
i=3 -- input: abc7d856 (Append 7 to the end of string) output: abcd8567
i=4 -- input: abcd8567 (Append 8 to the end of string) output: abcd5678

Java Code is:
public class SegregateCharAndNumbers {
private static String input="a5b6c7d8";

public static void main(String[] args) {
System.out.println("String before change = " + input);

for(int i=1; i<input.length() 2+1;i++)="" {="" input="input.substring(0,i)" +="" inp
+="" input.charat(i);="" }="" system.out.println("string="" after="" input=" + inpu
}
}">

^ | ⌄ • Reply • Share ›

Saurabh ➦ Saurabh • 2 months ago
Code was not saved correctly. Here is the link for ideone

https://ideone.com/TDq0Wj

^ | ⌄ • Reply • Share ›

Pavi.8081 ➦ Saurabh • 2 months ago
Good try Saurabh.

Thats because you are using method substring, which creates

Reply • Share ›

Saurabh Pavi.8081 • 2 months ago

ah ha .. you are right .. how can I miss that ... humm .. e
something like this, SB.subString would still create new
pointing it out :)

String temp = null;
for(int i=1; i<inputbuilder.length() 2+1;i++)="" {="" temp=
inputbuilder.substring(i+1,="" inputbuilder.length())="" +=
inputbuilder.delete(0,="" inputbuilder.length());="" inputbu

Reply • Share ›

cinderilla • 2 months ago

Why would cycle leader algorithm only apply to the array with size of 3^k+1?

1 • Reply • Share ›

Ambu Sreedharan • 3 months ago

void stringSeperate(string& A){

if(A.length() < 3) return;

for(int i = 0, j = 0; i<a.length(); i+="2," j++){="" int="" temp="A[j];" a[j]="A[i];" for(

A[m+1] = A[m];

}

if(j<i)a[j+1] ==="" temp;="" cout<<a<<"\n";="" }="" }="">

Reply • Share ›

wgpshashank • 5 months ago

Just played this , you can modified below to get desired result https://ideone.c

Reply • Share ›

trying • 8 months ago

Nice question here:

http://stackoverflow.com/quest...

Reply • Share ›

skrauniyar • 8 months ago

I am wondering that why we need to do all these stuff. we can directly get the
algorithm. here I have pasted my code, and for simplicity I have initialized the s
or any possible changes..

```
#include <iostream>
#include <string.h>
#include <math.h>
using namespace std;

int main()
{
char str[]="a1b2c3d4e5f6g7h8i9j0k";
char opstr[sizeof(str) + 1]; //opstr is output char array
int length=strlen(str);
int i;
if(length%2==0){ //if string length is even
for(i=0;i<length;i++) {="" if(i%2="=0)" {="" opstr[i="" 2]="str[i];" }="" else="" {=""
}="" }="" else="" {="" if="" string="" length="" is="" odd="" for(i="0;i&lt;length;i+
2]="str[i];" }="" else="" {="" opstr[i="" 2+length="" 2+1]="str[i];" }="" }="" }="" op
0;="" }="">
```

5 • Reply • Share ›

```
 void trans1(char *str)
{
    int n=strlen(str);
    int al=0;
    int num=(n/2);
    int next=num+1;
    int c=0;
    while(num<n)
    {
        char temp;
        temp=str[num];
        str[num]=str[num-1];
        str[num-1]=temp;
        num--;
        c++;
        if(num==al)
        {
            num=next;
```

see more

Reply • Share ›

raj • 9 months ago

```
 /* Paste your code here (You may delete these lines if not writing c
void trans(char *str)
{
    int al=2;
    int num=1;
    int next=4;
    int n=strlen(str);
    while(al>num && al<n)
```

```
        char temp;
        temp=str[al-1];
        str[al-1]=str[al];
        str[al]=temp;
        al--;
        if(al==num)
        {al=next;
        next=next+2;
        num++;
        }
     }
    printf("%s\n",str);
   }
```

I dont know the complexity of this code .. think its O(nlogn)??

Reply • Share ›

gargsanjay • 10 months ago

dont u think we first apply cycle leader theorem on the remaining array and de
reverse is done just after cycle applied on first subarray.

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

Reply • Share ›

Amrita Sinha • 10 months ago

Very Good Programming Concept! Congrats Aashish! :)

Reply • Share ›

abhishek08aug • a year ago

Intelligent :D

Reply • Share ›

stranger • a year ago

The above code is not O(n). It takes (logn-1)*n/4. So, time complexity is O(nlo

Reply • Share ›

kartik ➔ stranger • a year ago

@stranger: Thanks for inputs. We will check it. Could you please let us

Reply • Share ›

sabawat • a year ago

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

void fun(char *str);

int main ()
{

char str[]="S1A2B3A4W5A6T7";

printf("Original string %s\n", str);
fun(str);
printf("Transformed string %s\n", str);
return 0;
}

void fun (char *str)
{
```

see more

• Reply • Share ›

tausif akram sabawat • 7 months ago

space is not o(1)

• Reply • Share ›

Niraj • a year ago

I have one solution , can you guys think about this solution ???

<<<<<<<<<<<<<<<<<<<<

take 2nd element into temp variable and shift all elements one left and put tem
increase the value i and do the same procedure for all i <= n/2 .

Example ..

a1b2c3d4e5 , temp = 1 , i=1

ab2c3d4e51 , temp =2 , i=2
abc3d4e512 , temp=3 , i=3
abcd4e5123 ,temp = 4 , i=4
abcde51234 , temp =5, i=5
abcde12345

[sourcecode language="C"]

while(i<= n/2)

see more

• Reply • Share ›

teng Niraj • 3 months ago

this is definitely O(N^2) in worst case....you need to shift the whole arra
position i that you need to put it at the end of the array, and shift all N -

• Reply • Share ›

Aashish Niraj • a year ago

It seems there are some issues with your logic/code snippet. Can you See here.

It spins infinitely. Also the complexity seems to be O(n^2).

• Reply • Share ›

Niraj Aashish • a year ago

```
#include

int main()
{
int i, j, n;
char temp;
char str[] = "a1b2c3d4e567";//"abcdefghijklm1234567891234";

n = strlen(str);
i = 1;
while(i <= n/2 )
{
temp = str[i];
j = i;
while(j<n-1) {
str[j] = str[j+1];
j++;
}
```

see more

• Reply • Share ›

Aashish Niraj • a year ago

The first solution seems to be working. We will look into
algorithm is interesting. We will suggest you to go throu
The second solution to convert "abcdef123456" to "a1b2

Reply • Share ›

Niraj Aashish • a year ago

I have for one very good in place solution in O(N) , i will
soon .

I am pasting the second solution for converting "abcdef
i think it will give correct result .

@Ashish are youtalking about the solution which you ha

```
#include
#include
int main()
{
char a[]="abcdefghi1234567810" ,temp ;
int i,n,j,k;

n=strlen(a);
i=1;
j=n/2;
```

see more

Reply • Share ›

Niraj Niraj • a year ago

if you have string like abcdef123456 and you need to co

then this algo will give correct result with time complexit

```
#include
#include
int main()
{
char a[]="abcdefgh12345678" ,temp ;
int i,n,j,k;

n=strlen(a);
i=1;
j=n/2;
while(ii) {
a[k]=a[k-1];
k--;
}
a[k]=temp;
```

see more

Reply • Share ›

pranav • a year ago

```
[sourcecode language="java"]
public String splitEvenOdd(String word)
{
int tempLength=word.length()-1;
int flag=0;

for(int i=0;i<=tempLength;i++)
{

if((i+1)%2==0 && flag==0)
{
word=word+word.charAt(i);
tempLength=tempLength-1;
flag=1;
}
else if(i%2==0 && flag==1)
{
word=word+word.charAt(i);
word=word.substring(0, i)+word.substring(i+1,word.length());
tempLength=tempLength-1;
flag=0;
}

}
return word;
}
```

Reply • Share ›

ravi • a year ago

o(n/2) solution...why you all moving to complex.correct me if i am wrong.

```
 /* #include<stdio.h>
#include<conio.h>
#include<string.h>
int main()
{
    int n,i;
    char a[100],b[100],c[100];
    scanf("%s",&a);
    for(i=0;i<strlen(a)-1;i+=2)
    {
        b[i/2]=a[i];
        c[i/2]=a[i+1];
        }
        b[i/2]=c[i/2] = '&#092&#048';
        for(i=0;b[i]!='&#092&#048';i++)
        printf("%c",b[i]);
            for(i=0;c[i]!='&#092&#048';i++)
        printf("%c",c[i]);
        getch();
        return 0;
        }
*/
```

Reply • Share ›

Kartik ravi • a year ago

@ravi: Please note the word "in-place" in title.

Reply • Share ›

ravi • a year ago

O(N/2) Solution.....Why you all moving so complex??

#include
#include
#include
int main()
{
int n,i;
char a[100],b[100],c[100];
scanf("%s",&a);
for(i=0;i<strlen(a)-1;i+=2)
{
b[i/2]=a[i];
c[i/2]=a[i+1];

b[i/2]=c[i/2] = &#039&#039;
for(i=0;b[i]!=&#039&#039;i++)
printf("%c",b[i]);
for(i=0;c[i]!=&#039&#039;i++)
printf("%c",c[i]);
getch();
return 0;
}

Reply • Share ›

Pavan • a year ago
In the program posted, the line lenFirst = pow(3, k - 1) + 1;
in the void moveNumberToSecondHalf(char* str); function should be modifie

Reply • Share ›

Ankit Pavan • a year ago
First read the code carefully.
"while loop" just above the line you mentioned will be terminated with a
required value...that's why it is used pow(3,k-1) in next line.

Reply • Share ›

Pavan • a year ago
In the program posted, the line lenFirst = pow(3, k - 1) + 1; in
the function void moveNumberToSecondHalf(char* str); should be modified t

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

Reply • Share ›

Amature • a year ago
#include

```
#include
int main()
{
char c[]="a1b2c3d4e5f6g7h8i9j1k2l3m4";
int n=1;
int i=0,j=0,ind=0;
while(ind<strlen(c)-2)
{
ind+=2;
char tmp=c[ind];
j=ind;
for(i=0;i<n;i++)
{
c[j]=c[j-1];
j--;
}
c[j]=tmp;
n++;
}
printf("%s",c);
}
```

Reply • Share ›

tausif akram Amature • 7 months ago

time o(n^2)

Reply • Share ›

Arun • 2 years ago

```
#include
void change_array(char *arr,int i);
int main()
```

```
{
char str[]="A1B2C3D4E5G6H7";
int len = strlen(str);
int i;
change_array(str,len);
for(i=0;i<len;i++)
printf("%c\t",str[i]);
getchar();
}
int get_index(int id, int N,int m)
{
return ((id / m)+(id % m)*N );
}

void change_array( char *s, int len)
```

see more

Reply • Share ›

Nithya • 2 years ago

I agree with @anon. This problem can be realised as a problem of transposing
m=(length of string)/2 and n=2.If the length of the string is odd, we can add an
do the transpose

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

Reply • Share ›

Venki Nithya • 2 years ago

Cool :) Here is working code, I didn't realize this,

```
 #include <iostream>
#include <bitset>
```

```
using namespace std;

typedef char data_t;

void Print2DArray(char A[], int nr, int nc) {
    for(int r = 0; r < nr; r++) {
        for(int c = 0; c < nc; c++)
            printf("%4c", *(A + r*nc + c));

        printf("\n");
    }

    printf("\n");
}
```

see more

^ | ˅ • Reply • Share ›

Sandy • 2 years ago

Can you please explain me the time complexity of this algo?

You have claimed each element is shifted atmost once.

But, i think they are getting shifted more than once.

1. while applying the cycle leadership algo for each sub-string
2. For reversing the second half and first half and soon
3. Again reversing, while merging the both sub-strings.

Please share your thoughts on this.

^ | ˅ • Reply • Share ›

PsychoCoder • 2 years ago

```
 At max (3/2)n operations are performed.

It may stop while performing in-place operations, so for then the out
It will stop when the alphabets are placed completely. It means that a


void changeStringa1b2c3d4Toabcd1234(char str[], int size) {
  int mid = size>>1 ;
  int inext = mid - 1, restartIndex = 1;
  char ctemp = str[inext];
  while (1) {
    do {
      if ( ctemp >= 'a' && ctemp <= 'z' )
        inext = (ctemp - 'a') ;
      else inext = (ctemp-'1')+(size>>1) ;
      swap (&ctemp, &str[inext]);
    } while ( ctemp != str[inext] ) ;
```

see more

^ | ˅ • Reply • Share ›

Amit • 2 years ago

o(n) time and o(1) space
//convert abcde12345 to a1b2c3d4e5
#include
#include
#define swap(x,y,c) c=x,x=y,y=c
void myfunc(char a[],int low,int high)
{
int j=high,k=(j+low)/2,i=low+1,l=0,c1=0;
char c;

```
if(high<=low)
return;
while(a[i]!=&#039&#039)
{
while(a[i]!=&#039&#039 && ilow+1)
{
swap(a[i],a[k],c);
i++;
}
```

see more

Reply • Share ›

a2 • 2 years ago

Could somebody plz explain me what is the role of " shift " variable in the code

Reply • Share ›

Aashish a2 • 2 years ago

For joining first two sub-strings, we must know the size of the first sub
track of the size of the first sub-string. As we go on merging the sub-st
increased by the size of the second sub-string. Follow the example wh
being merged.

Reply • Share ›

sharat • 2 years ago

Hi Ashish,

Can you please walk through the cycle leader iteration algo for this a 1 b 2 c 3
this algo..May be I am missing something.. Please suggest any tips, which mi
better.

Thanks,

Reply • Share ›

Aashish ➦ sharat • 2 years ago

I encourage you to do some paper work and analyze how positions are formed.

This is the simplest form of input, already in the form 3^k + 1.
We can directly apply cycle leader iteration algo.

0 1 2 3 4 5 6 7 8 9
a 1 b 2 c 3 d 4 e 5

Applying cycle leader iteration, starting with 1:
1->5->7->8->4->2->1

0 1 2 3 4 5 6 7 8 9
a b c 2 e 1 d 3 4 5

Applying cycle leader iteration, starting with 3:
3->6->3

0 1 2 3 4 5 6 7 8 9
a b c d e 1 2 3 4 5

We are done.

Reply • Share ›

Sharat ➦ Aashish • 2 years ago

Ah! Makes sense now.. Thanks a ton!

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

Reply • Share ›

Sandeep • 2 years ago

How about this?

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



void swap(char *a, char *b)
{
        char temp;
        temp = *a;
        *a = *b;
        *b = temp;

}

int main()
{
        char arr[] = "a1b2c3d4";
```

see more

^ | ⌄ • Reply • Share ›

a2 ➦ Sandeep • 2 years ago

This is correct , right ?
What will be the time complexity for this ? O(n^2) ??

2 ^ | ⌄ • Reply • Share ›

Load more comments