

Forum:[Hack Forums](#)

Thread: [Programming Challenge -1~](#)

myselfandi - August 9, 2019, 10:36 am

We have an array of C strings, each string appears twice except for one string appears once.

You need to write an algorithm that return the one string that appears once. Write an algorithm that solve this in $O(n)$.

I'll post the solution in one day.

Securifier - August 9, 2019, 1:06 pm

You want each string in the array only appears once?

Soooo EZZZ :)

YouthInAsia - August 9, 2019, 1:17 pm

So what's the challenge....

myselfandi - August 9, 2019, 1:19 pm

(Aug 9, 2019 01:06 PM)Alpha Coder Wrote: [\[CITE\]](#) You want each string in the array only appears once?

Soooo EZZZ :)

Sorry edited, No return the string in the array that appears once.

(Aug 9, 2019 01:17 PM)YouthInAsia Wrote: [\[CITE\]](#) So what's the challenge....

Edited

Hoodie+ - August 9, 2019, 4:07 pm

I remember a problem like this in college.

I solved it like this by creating a hash of the current element i am checking and seeing if it is in a list of hashes, if it is I would delete it from the list and if it isn't I would add it to the hash list.

Not sure if it is fast/effective but it was working well.

myselfandi - August 9, 2019, 5:32 pm

(Aug 9, 2019 04:07 PM)Hoodie+ Wrote: [\[CITE\]](#) I remember a problem like this in college.

I solved it like this by creating a hash of the current element i am checking and seeing if it is in a list of hashes, if it is I would delete it from the list and if it isn't I would add it to the hash list.

Not sure if it is fast/effective but it was working well.

you are right, Hashing is a good way to solve the problem, traversing all array elements and putting them in a hash table, then can use the array elements as a key and for each occurrence you increment the value by 1. now you simply you can traverse the array and print the element with value 1 in the hash table. though this method require extra space. there is another way of solving this problem that uses $O(1)$.

6577657534735 - August 9, 2019, 10:40 pm

(Aug 9, 2019 05:32 PM)myselfandi Wrote: [\[CITE\]](#) you are right, Hashing is a good way to solve the problem, traversing all array elements and putting them in a hash table, then can use the array elements as a key and for each occurrence you increment the value by 1. now you simply you can traverse the array and print the element with value 1 in the hash table. though this method require

extra space. there is another way of solving this problem that uses $O(1)$.

xor every element of the array together, the result is the string that occurs once. same can be said of any data structure that can be represented as a byte array

the reason is if you xor a byte by itself is cancels out to 0, so if you xor every string twice they cancel out, and the one you only xor'd once is xor'd by 0, so it appears in the result

if anyone else wants more classic programming puzzles and interview questions like this, read Cracking The Code Interview

myselfandi - August 10, 2019, 5:17 am

(Aug 9, 2019 10:40 PM)Abraluna Wrote: [\[CITE\]](#) xor every element of the array together, the result is the string that occurs once. same can be said of any data structure that can be represented as a byte array

the reason is if you xor a byte by itself is cancels out to 0, so if you xor every string twice they cancel out, and the one you only xor'd once is xor'd by 0, so it appears in the result

if anyone else wants more classic programming puzzles and interview questions like this, read Cracking The Code Interview

? ?

Hoodie+ - August 10, 2019, 6:55 am

(Aug 9, 2019 10:40 PM)Abraluna Wrote: [\[CITE\]](#) xor every element of the array together, the result is the string that occurs once. same can be said of any data structure that can be represented as a byte array

the reason is if you xor a byte by itself is cancels out to 0, so if you xor every string twice they cancel out, and the one you only xor'd once is xor'd by 0, so it appears in the result

if anyone else wants more classic programming puzzles and interview questions like this, read Cracking The Code Interview

I don't believe I understand this way. Could you clarify it more?

I understand the fact that if you xor two equal strings that they will cancel out.

myselfandi - August 10, 2019, 7:56 am

(Aug 10, 2019 06:55 AM)Hoodie+ Wrote: [\[CITE\]](#) I don't believe I understand this way. Could you clarify it more?

I understand the fact that if you xor two equal strings that they will cancel out.

XOR is commutative so a if have this array {5,4,7,5,4}, and then xor all the element in it, the result will be 7

because $5^4 \wedge 7^5 \wedge 4 \Rightarrow 5^5 \wedge 4^4 \wedge 7 \Rightarrow 0^0 \wedge 7 \Rightarrow 7$

Hoodie+ - August 10, 2019, 4:09 pm

(Aug 10, 2019 07:56 AM)myselfandi Wrote: [\[CITE\]](#) XOR is commutative so a if have this array {5,4,7,5,4}, and then xor all the element in it, the result will be 7

because $5^4 \wedge 7^5 \wedge 4 \Rightarrow 5^5 \wedge 4^4 \wedge 7 \Rightarrow 0^0 \wedge 7 \Rightarrow 7$

Alright but how does that help us here?

6577657534735 - August 10, 2019, 9:27 pm

(Aug 10, 2019 04:09 PM)Hoodie+ Wrote: [\[CITE\]](#) Alright but how does that help us here?

we have a set of strings, every string in the set appears twice except one. we want to find the string that appears once. we can use a hash table to count the number of instances of each string and

find the one whose value is one, but this requires $O(n)$ space to store this info in the hash table. instead we want to decrease it to $O(1)$ space, as in space not related to the number of strings you're comparing. for this reason we can xor together every string, and the ones that appear twice cancel out, becoming 0, then the string that appears once is xor'd with 0, resulting in itself. thus if you xor every string in the set, the result is the string that appears once, requiring only $O(1)$ space on the number of strings in the set, and in fact this is the fastest method too since xor is wayy faster than hash tables

YouthInAsia - August 11, 2019, 3:34 am

```
Code arrayofstrings = {"hello","world","boost","start","world","boost","hello"}//start
for(index = 1 ; index < lengthofarrayofstrings; ++index)
    onlyone = arrayofstrings[index-1] ^ arrayofstrings[index]
print(onlyone)//start
```