

Forum:[Hack Forums](#)

Thread: [\[HFCC\] The Hack Forums Coding Contest - Week 4](#)

Haxalot - March 20, 2017, 11:07 am

[The Hack Forums Coding Contest: Week 4](#)

AboutThe HFCC is a weekly coding contest hosted by myself for the Hack Forums community. You will have 3 days to solve 5 programming challenges I have randomly selected from Hackerrank. The contest will start at 4pm (GMT) every Friday, and end at 4pm (GMT) every Monday.

The next thread detailing the next contest will be created every Monday, after the previous week's contest has finished.

You should not use this thread to post solutions or provide hints until the contest is over. Anyone violating this rule will be barred from the current and all future contests I host.

Previous weeks:[The Hack Forums Coding Contest - Week 1](#)

[The Hack Forums Coding Contest - Week 2](#)

[The Hack Forums Coding Contest - Week 3](#)

What's the Point?Coding challenges are a great way to improve upon your programming abilities. They encourage you to:research data structures and algorithms

think critically and analytically about problems by breaking them down into smaller (and therefore simpler) sub problems

Be performance-conscious of the code you write

This will help you to:be able to apply a variety of data structures and algorithms to any problems you encounter when programming

improve your problem solving abilities, enabling you to tackle increasingly complex problems

learn new techniques (such as memoisation) and utilise more performant algorithms

Another great advantage is that companies frequently test their candidates in interviews with similar challenges to what you will be solving in these contests. By practicing these problems, you are

effectively preparing yourself for future interviews, giving you the best chance possible to secure a great job at a top company. All of the largest companies in the software industry (Google, Facebook, Amazon, Microsoft, etc) will be looking for candidates who are great problem solvers, and the only way they can see if you're a great problem solver is by giving you programming challenges.

So by entering these contests, you will ultimately become a better programmer and improve your employability prospects, all whilst having fun solving problems with other members here.

Leaderboard
Kutabe (115)

jozefstodulski (53.18)

I'm always open to hearing feedback if members here have any thoughts on the difficulty or number of the challenges presented, the length of the contest, or the starting times.

[Sign Up and Get Involved!](#)

Haxalot - March 24, 2017, 4:10 am

This contest starts today (~5hrs time) for the interested.

vaiperboy - March 24, 2017, 4:15 am

sad to see that not much are interested,
anyways, just registered :) looking forward for the challenges :p

Nachfolger - March 24, 2017, 5:33 am

I was very interested until I had to sign up with my "actual" info. I can't be bothered to do that nonsense.

But gl!

vaiperboy - March 24, 2017, 5:40 am

(Mar 24, 2017 05:33 AM)Nachfolger Wrote: [\[CITE\]](#) I was very interested until I had to sign up with my "actual" info. I can't be bothered to do that nonsense.

But gl!just fake the name n w/e

Haxalot - March 24, 2017, 11:00 am

(Mar 24, 2017 05:33 AM)Nachfolger Wrote: [\[CITE\]](#) I was very interested until I had to sign up with my "actual" info. I can't be bothered to do that nonsense.

But gl!

As vaiperboy said, you can simply use a fake name if you're concerned about privacy.

Personally though, if you do want to get into solving programming challenges, then I'd use your actual name. There have been many cases where recruiters have (unsolicitedly) reached out to programmers who do well at problem solving on sites like this. Every programmer should have a GitHub account, so you can simply sign up using that to make things easier.

kutabe - March 24, 2017, 4:38 pm

I win. Changing Bits was hard enough.
First I did it with Golang, math/big, but
some cases didn't pass 'coz it was too slow.

[edit] :

With a little bit of investigation I found weak spot in my code and it was kinda surprising.
So, I was confused and upset when my code in Go took 30+ seconds in case #4.
I did some benchmarks and, gosh, it was so unexpected.
I fixed few lines and now this same test case perceed in 1.5s.
Don't use fmt.Scan when dealing with stdio, guys. Use fmt.Fscan and readers instead.
Wish you many years and prosperity.

[edit]:

Thread seems dead.

Haxalot - March 25, 2017, 3:45 pm

Congrats, you solved them all pretty quickly. And the thread's not dead - there's just not much to talk about whilst other's are still solving the challenges.

Haxalot - March 27, 2017, 8:22 am

This contest is now over. Feel free to post and/or discuss solutions to the challenges here.

kutabe - March 27, 2017, 12:16 pm

[HERE](#) is my solution to BitCh issue, if you interested.

Haxalot - March 27, 2017, 1:00 pm

(Mar 27, 2017 12:16 PM)kutabe Wrote: [CITE](#) [HERE](#) is my solution to BitCh issue, if you interested.
Nice. Go made the problem seem trivial, actually. I was using C and only managed to get 7/10 of the test cases to pass. I initially tried the eager approach of updating the array C whenever a bit was changed in either array A or array B:

Code #include <stdio.h>

```
int main()
{
    int n, q, carry = 0, key, value;

    scanf("%d %d", &n, &q);

    char a[n], b[n], c[n + 1], buffer[n], command[5], *a_or_b;

    scanf("%s", buffer);

    for (int i = n - 1, j = 0; i > -1; --i, ++j) {
        c[j] = a[j] = buffer[i] - '0';
    }

    scanf("%s", buffer);

    for (int i = n - 1, j = 0; i > -1; --i, ++j) {
        b[j] = buffer[i] - '0';

        switch (b[j] + c[j] + carry) {
```

```

        case 2:
            c[j] = 0;
            carry = 1;
            break;
        case 1:
            c[j] = 1;
            carry = 0;
    }
}

c[n] = carry;

while (q--) {
    if (2 == scanf("%s %d %d", command, &key, &value)) {
        printf("%d", c[key]);
    } else {
        a_or_b = command[4] == 'a' ? a : b;

        if (a_or_b[key] != value) {
            a_or_b[key] = value;

            while (c[key] == value) {
                c[key++] = !value;
            }

            c[key] = value;
        }
    }
}

return 0;
}

```

This meant that the lookup time for array C was a constant time operation, but if there were a lot of bits changing without much access to array C, then it would end up being quite expensive.

I then tried the lazy approach where I only computed part of array C when it was accessed. This yielded better performance overall, and made the updating of the arrays A and B constant time:

Code #include <stdio.h>

```

int main()
{
    int n, q, total, key, value;

    scanf("%d %d", &n, &q);

    char a[n + 1], b[n + 1], buffer[n], command[5];

    scanf("%s", buffer);

    for (int i = n - 1, j = 0; i > -1; --i, ++j) {
        a[j] = buffer[i] - '0';
    }
}

```

```

scanf("%s", buffer);

for (int i = n - 1, j = 0; i > -1; --i, ++j) {
    b[j] = buffer[i] - '0';
}

a[n] = b[n] = 0;

while (q--) {
    if (2 == scanf("%s %d %d", command, &key, &value)) {
        total = a[key] + b[key];

        while (--key > -1) {
            if (a[key] == b[key]) {
                total += a[key];
                break;
            }
        }

        printf("%d", total & 1);
    } else {
        if (command[4] == 'a') {
            a[key] = value;
        } else {
            b[key] = value;
        }
    }
}

return 0;
}

```

But it was still too slow for test cases #1, #4, and #5 though. I ran some benchmarks locally, where I had 100k lengths for arrays A and B, and 3 million queries (6 times the max on Hackerrank), which yielded execution times of below 1 second. The solution was just slow on Hackerrank... More thought is needed on this one, I think.