Problem C

THIS WAS SO ANNOYING. The cases whether either a, b, c or d are equal to 0 was very difficult to handle.

I spent so much time trying to solve a misunderstood version of the problem. A few things I didn't realise was that it was subsequences and not substrings. This means it doesn't have to be contiguous. This misunderstanding took me a while to clear up and it was getting late after thinking about the question for a while so I slept on it.

Before sleeping coming up with a few of the important observations. For a given 0, every 0 you add after will contribute to adding up to the value of a. This is the same for b, c and d.

This then becomes just counting the number of "matching pairs" (3121 task reference) and so to count the number of these pairs it's just $n(n-1)/2$. You can equate this to a and if you solve for n that will give the amount of:

$$n(n-1)/2 = a \implies n = (1 + \sqrt{1+8a})/2$$

You could just increment n from 0 until it becomes above or equal to a or just use the formula.

And then using the ideas from above, if we add a 0, any 1 we add after will contribute to b. So if b is larger than the amount of 1's we have left, we need to add that mean's you need to add a 0 so that you can achieve the total b necessary. Same idea works in reverse for c.

It also turns out that $b > numones$ then $c < numzeros$ i think so the condition just becomes and if else.

I think at this point, the main algorithm was working but the annoying part was figuring out the annoying base cases mentioned above. I think I tried a lot here, coding up cases for $b + c == 0$, $a + b + c + d == 0$ and it became a mess. After a lot of testing different implementations for this turns out I also missed the fact that the strings can't be empty.

There are more notes on this problem in my code solution but I found this problem very annoying. Personally did not enjoy it and think there are better problems that could have been chosen.