What should the encryption be for the following?

Message: I AM

Key: 17

- R JV
- ZXRD
- Z RD

Correct

- R$JV

Continue

Developing an Algorithm

Have a question? Discuss this lecture in the week forums.

Interactive Transcript

Search Transcript English

0:03
Welcome back. Now you have all the concepts required to implement a Caesar Cipher. So let's get started developing the algorithm. As always, you should start with step one, working an instance of the problem yourself. Even though we have seen some instances worked, it is good to work a small instance, so you can write everything down and think it through. Let's encrypt the message I AM, which is really I, space, A-M with a key of 17, meaning you will shift each letter 17 positions through the alphabet. You might want to start by writing down the alphabet, and then writing down the alphabet shifted by 17 characters underneath it. For example, A has R beneath it, where R is 17 characters to the right of A in the original alphabet.

0:51
Next you would go through and replace each character in the message with the appropriate letter from the shifted alphabet. When you are done you have the encrypted message Z RD. Great, you have finished Step 1..

1:04
Now it is time to do Step 2, and write down exactly what you just did. The first thing you did was write down the alphabet. Then you computed the shifted alphabet. The third thing you did was to look at the 0th letter of the message. Don't forget, when you index into sequences such as strings and string builders, the first element is at index zero.

1:27
That letter was I, so you looked in the alphabet to find I, then you found the letter in the shifted alphabet at the same position which was Z. So, you replace the zero character of the message with Z.

1:41
Next, you look at the first letter of the message which is a space If you look for space in the alphabet, you would not find it, so you would not change the character at index one of the message.

1:52
Next, the second character is an A, for which you perform a very similar process as you did for the 0th character, and end up changing it to R.

2:01
Finally, you do the same thing for the third character which is an M that you turned into a D.

2:07
Now that you have thought through all of that, you have a list of the 17 things you did for this particular message, and this particular key. However, there's one more thing that is good to note here before you proceed. Notice that your algorithm calls for replacing characters in the message. If your message is a string, you cannot do that. As you recently learned, strings are immutable, meaning you cannot change them.

2:32
If you recognize this issue now you can adjust your algorithm to reflect the fact that you want to work with a string builder. Here we've added a step at the start to create a string builder from the string message.

2:44
And then we updated the algorithm to work on the string builder.

2:48
If you do not realize you need to do this now, you would figure it out at a later step. But the earlier you can figure it out, everything you need to do, the better.

2:59
Looking at this algorithm you can see that the first few steps are an initial setup, before you begin performing repetitive steps for each letter in the message.

3:09
If you focus on the steps after the initial set-up, you can see that you are doing almost, but not quite the same thing for each character n the message.

3:18
One significant difference is what you decide to do based on whether or not you find the letter in the alphabet. Replacing the current character if you find it.

3:29
Or doing nothing if you do not.

3:32
If you look at the steps for one particular character where the letter is in the alphabet, you will notice that the character you looked for in the alphabet is the current character in the string. And that the letter you used to replace the current character is what you found in the same position in the shifted alphabet.

3:50
Now that you have thought this all through, you can write down, a much more general algorithm.

3:56
Notice that the step number two here requires a little thought, and a couple of statements. But you've already seen how to do it.

4:04
When you are looking for patterns, you should examine any constants, such as zero here, and ask if you always use that constant. Or if you need to look for a more general pattern.

4:15
Here you always want to start from zero.

4:18
What about three? Do you always want to stop counting at three?

4:24
No. How high you count depends on the length of the message. Here we've written that you want to count to the length of encrypted. But noted that you want to count to less than it, not less than or equal to it. In our example, encrypted was four, and you only want to count to three.

4:42
Now it's time to test out the steps. Pause the video now and try to encrypt the message, a space bat with a key of 19.

4:53
Did you catch the subtle problem with this algorithm? Even though it computed everything you wanted, we never said what the final answer is.

5:02
You want to be sure to explicitly say this so that you know what to return from your method when you translate the code.

5:09
Your answer is the string inside of the string builder you called encrypted.

5:13
Now that you have fixed this detail of your algorithm you are ready to turn it into Java code. Thank you.

Downloads

- **Lecture Video** mp4
- **Subtitles (English)** WebVTT
- **Transcript (English)** txt

Would you like to help us translate the transcript and subtitles into additional languages?