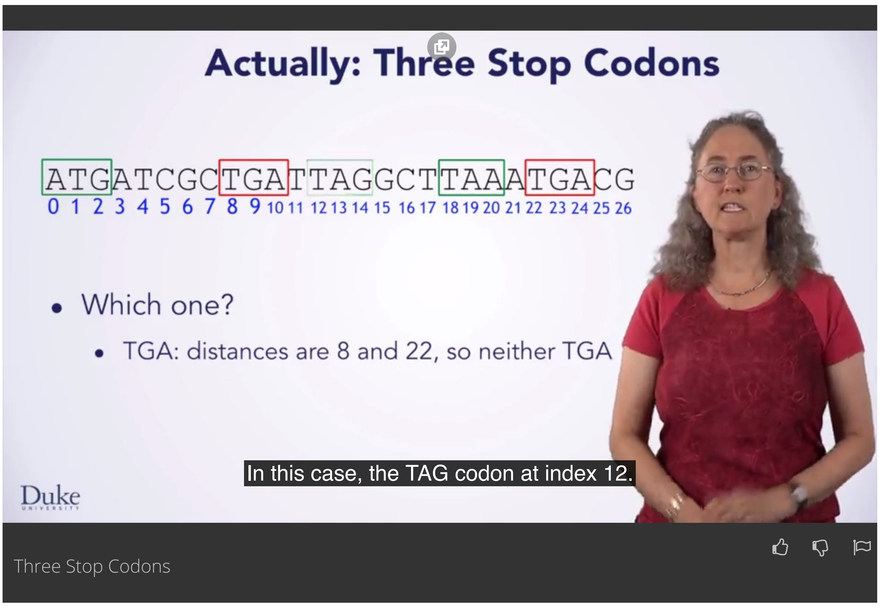

Have a question? Discuss this lecture in the week forums.

Interactive Transcript

Search Transcript

English

0:04
So far, you have developed an algorithm that would find the start and stop codons, figure out that the link is a multiple of three, and return the DNA string for this gene. Now it is time to add another layer of realism, and thus, complexity. There are actually three different stop codons, TAA, TGA, and TAG. So far, your algorithm has only looked for TAA, but now it is time to make it look for the first one of any of these three stop codons that is a multiple of three away from the start codon count.

0:37
Which one of these stop codons is the one you want? The first TGA is not a multiple of three from the start codon, so that isn't the one you want. The second TGA is also not a multiple of three from the start, so you don't want that either. That leaves you with TAG and TAA which are both multiples of three away. You want the one that comes first. In this case, the TAG codon at index 12.

1:03
Now you know what the problem is, let's solve it.

1:07
To do this, let's go back to our algorithm that we have from before. The only thing that was specific to looking for TAA was these two occurrences of TAA in steps 3 and 7. Could we start from this algorithm and had most of our work done for us?

1:23
The best way to do this is to split the problem up. We want to abstract out the part that searches for a stop codon into its own method.

1:33
We'll call it findStopCodon. And it will take the DNA string to search in, the index to start from, and the particular stop codon string, such as TAA, TGA, or TAG to search for. The algorithm will need a few changes, such as returning the index instead of the text and looking for any of the three stop codons, not just TAA. However, the basic mechanics of the algorithm, searching for a codon that is a multiple of three from the start, remain the same. We'll come back to those changes in a minute. For now, we'll just assume we have a findStopCodon method that works. Once we have extracted that out into its own function, we could use it to find the TAA stop codon, then call the method again to find the TAG stop codon. Then one more time, to find the TGA stop codon. Note how this corresponds to what we did by hand. We identified the positions of the TAA, TAG, and TGA stop codons. It is a bit different from our example by hand, since findStopCodon will only give us a position that is a multiple of three away from startIndex. Whereas, we showed a TGA that was not a multiple of three for the purposes of illustration.

2:50
Now that we have those three positions, we want the one that comes first. We would just need to take the minimum of three values, which we will call, minIndex.

3:02
Finally, our answer is the substring from startIndex to minIndex plus 3.

3:08
Now let's look at what we need to do the algorithm that we abstracted out into findStopCodon. First, these will no longer always be TAA.

3:20
Instead, they will be the stopCodon parameter which tells us the particular stopCodon we looking for.

3:27
The other change is that we want to give back the index where we found the stopCodon instead of the text between the start and stop codons.

3:36
Why? Our other algorithm needs these indices to compare to figure out which one to use before it gets the text.

3:46
Whenever we find a valid stopCodon in step 4, we could just give currIndex as our answer.

3:53
However, what should we give back to represent no valid index found in step 6? Negative 1 is often a good choice to indicate no valid index. But let's see how this would work with the way we are using the result of this function.

4:12
If we were to return -1, we could make it work, but we would have to change this code to do more complex comparisons than just taking the minimum of these three values.

4:25
You'll see this approach later, and learn a new concept in the process. But for now, let's make this work such that we can just take the minimum here.

4:34
Instead, we can return the length of the DNA string, since that is larger than any valid index.

4:43
Of course, now that we are doing that, we should explicitly check for the case where no valid stop codon was found. If minIndex is the length of the DNA string, none of the three stop codons was found.

4:56
Great. Now we have an algorithm. Let's go turn this into 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?