Programming Homework 4

TOTAL POINTS 4

1. In a practical, we saw the `scs` function (copied below along with `overlap`) for finding the shortest common superstring of a set of strings. 1 point

```python
def overlap(a, b, min_length=3):
    """ Return length of longest suffix of 'a' matching
        a prefix of 'b' that is at least 'min_length'
        characters long.  If no such overlap exists,
        return 0. """
    start = 0  # start all the way at the left
    while True:
        start = a.find(b[:min_length], start)  # look for b's suffx in a
        if start == -1:  # no more occurrences to right
            return 0
        # found occurrence; check for full suffix/prefix match
        if b.startswith(a[start:]):
            return len(a)-start
        start += 1  # move just past previous match

import itertools

def scs(ss):
    """ Returns shortest common superstring of given
        strings, which must be the same length """
    shortest_sup = None
    for ssperm in itertools.permutations(ss):
        sup = ssperm[0]  # superstring starts as first string
        for i in range(len(ss)-1):
            # overlap adjacent strings A and B in the permutation
            olen = overlap(ssperm[i], ssperm[i+1], min_length=1)
            # add non-overlapping portion of B to superstring
            sup += ssperm[i+1][olen:]
        if shortest_sup is None or len(sup) < len(shortest_sup):
            shortest_sup = sup  # found shorter superstring
    return shortest_sup  # return shortest
```

It's possible for there to be multiple different shortest common superstrings for the same set of input strings. Consider the input strings `ABC`, `BCA`, `CAB`. One shortest common superstring is `ABCAB` but another is `BCABC` and another is `CABCA`.

What is the length of the shortest common superstring of the following strings?

`CCT`, `CTT`, `TGC`, `TGG`, `GAT`, `ATT`

11

2. How many different shortest common superstrings are there for the input strings given in the previous question? 1 point

Hint 1: You can modify the `scs` function to keep track of this.

Hint 2: You can look at these examples to double-check that your modified `scs` is working as expected.

4

3. Download this FASTQ file containing synthetic sequencing reads from a mystery virus: 1 point

https://d28rh4a8wq0iu5.cloudfront.net/ads1/data/ads1_week4_reads.fq

All the reads are the same length (100 bases) and are exact copies of substrings from the forward strand of the virus genome. You don't have to worry about sequencing errors, ploidy, or reads coming from the reverse strand.

Assemble these reads using one of the approaches discussed, such as greedy shortest common superstring. Since there are many reads, you might consider ways to make the algorithm faster, such as the one discussed in the programming assignment in the previous module.

How many As are there in the full, assembled genome?

Hint: the virus genome you are assembling is exactly 15,894 bases long

4633

4. How many Ts are there in the full, assembled genome from the previous question? 1 point

3723

I, **THOMAS JOHN JAMES**, understand that submitting another's work as my own can result in zero credit for this assignment. Repeated violations of the Coursera Honor Code may result in removal from this course or deactivation of my Coursera account.

Learn more about Coursera's Honor Code

Save Submit