Programming Homework 3

TOTAL POINTS 4

1. What is the edit distance of the best match between pattern `GCTGATCGATCGTACG` and the excerpt of human chromosome 1? (Don't consider reverse complements.) 1 point

   3

2. What is the edit distance of the best match between pattern `GATTTACCAGATTGAG` and the excerpt of human chromosome 1? (Don't consider reverse complements.) 1 point

   2

3. In a practical, we saw a function for finding the longest exact overlap (suffix/prefix match) between two strings. The function is copied below. 1 point

```
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 prefix 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
```

   Say we are concerned only with overlaps that (a) are exact matches (no differences allowed), and (b) are at least **k** bases long. To make an overlap graph, we could call `overlap(a, b, min_length=k)` on every possible pair of reads from the dataset. Unfortunately, that will be very slow!

   Consider this: Say we are using k=6, and we have a read **a** whose length-6 suffix is `GTCCTA`. Say `GTCCTA` *does not occur in any other read in the dataset*. In other words, the 6-mer `GTCCTA` occurs at the end of read **a** and nowhere else. It follows that **a**'s suffix cannot possibly overlap the prefix of any other read by 6 or more characters.

   Put another way, if we want to find the overlaps involving a suffix of read **a** and a prefix of some other read, we can ignore any reads that don't contain the length-k suffix of **a**. This is good news because it can save us *a lot* of work!

   Here is a suggestion for how to implement this idea. You don't have to do it this way, but this might help you. Let every k-mer in the dataset have an associated Python `set` object, which starts out empty. We use a Python dictionary to associate each k-mer with its corresponding `set`. (1) For every k-mer in a read, we add the read to the `set` object corresponding to that k-mer. If our read is `GATTA` and k=3, we would add `GATTA` to the `set` objects for `GAT`, `ATT` and `TTA`. We do this for every read so that, at the end, each `set` contains all reads containing the corresponding k-mer. (2) Now, for each read **a**, we find all overlaps involving a suffix of **a**. To do this, we take **a**'s length-k suffix, find all reads containing that k-mer (obtained from the corresponding `set`) and call `overlap(a, b, min_length=k)` for each.

   The most important point is that we *do not* call `overlap(a, b, min_length=k)` if **b** does not contain the length-k suffix of **a**.

   Download and parse the read sequences from the provided Phi-X FASTQ file. We'll just use their base sequences, so you can ignore read names and base qualities. Also, no two reads in the FASTQ have the same sequence of bases. This makes things simpler.

   https://d28rh4a8wq0iu5.cloudfront.net/ads1/data/ERR266411_1.for_asm.fastq

   Next, find all pairs of reads with an exact suffix/prefix match of length at least 30. Don't overlap a read with itself; if a read has a suffix/prefix match to itself, ignore that match. Ignore reverse complements.

   - Hint 1: Your function should not take much more than 15 seconds to run on this 10,000-read dataset, and maybe much less than that. (Our solution takes about 3 seconds.) If your function is much slower, *there is a problem somewhere*.
   - Hint 2: ***Remember not to overlap a read with itself***. If you do, your answers will be too high.
   - Hint 3: You can test your implementation by making up small examples, then checking that (a) your implementation runs quickly, and (b) you get the same answer as if you had simply called `overlap(a, b, min_length=k)` on every pair of reads. We also have provided a couple examples you can check against.

   Picture the overlap graph corresponding to the overlaps just calculated. How many edges are in the graph? In other words, how many distinct pairs of reads overlap?

   904746

4. Picture the overlap graph corresponding to the overlaps computed for the previous question. How many nodes in this graph have at least one outgoing edge? (In other words, how many reads have a suffix involved in an overlap?) 1 point

   7161

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