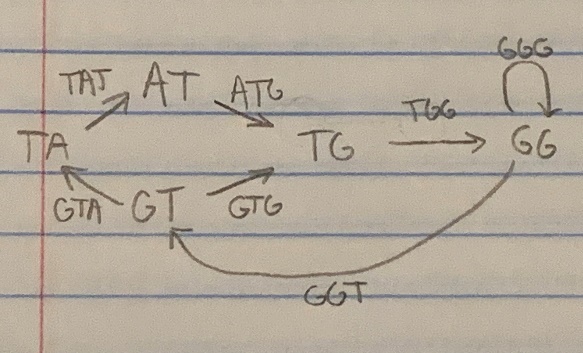
Aman Patel

September 22, 2020

CSCI-B 363

Homework 2

1. Dr. Smart is incorrect in his argument because it is possible that the consensus sequence is not the optimal median sequence. For example, given the set of sequences: , if the motifs selected are the final four characters of each sequence, the consensus sequence would be . If this were the median sequence, its total distance would be 4. However, the correct median sequence for these sequences is , with a total distance of 0.
2. The algorithm shown will find a k-mer that minimizes the total motif score of the set of DNA sequences, but it might miss the correct sequence. In this algorithm, the only k-mers tested are those found in the set of DNA sequences. However, the optimal solution might not be present in the set, thus all possible k-mers must be tested to ensure the true minimum is found. This would also remove some nested for loops and improve the runtime.
3. Modified Gibbs sampling algorithm
   1. Find the reverse complement of each DNA sequence in the set
   2. Select a k-mer from each sequence at random
   3. Remove a sequence at random
   4. Calculate the count matrix for the k-mers in the remaining sequences
   5. Convert the count matrix into a profile matrix
   6. Randomly select a new starting position using the probabilities in the profile matrix
   7. Repeat steps c-f as desired
4. Twin motif algorithm  
   minimumScore = MAX\_INTEGER  
   minimumSequence = “”  
   motif1 = []  
   motif2 = []  
   for sequence in AAAA…AAA to TTTT…TTT  
    create map from DNA sequence to score  
    for x in DNA  
    map x to calculateScore(x, sequence)  
    sort map and find sum of first half of elements  
    if sum < minimumScore  
    minimumSequence = sequence  
    minimumSum = sum  
    motif1 = first half of elements in map  
    motif2 = second half of elements in map  
   return motif1, motif2
5. Set B is the set that is more likely wrong because the score of its motifs is much higher than that of set A. The score of a set of motifs is the total number of differences between a set of DNA strings and a consensus string. The consensus string would be and its score would be 10. The consensus string of set B would be , and its score would be 19.
   1. Set of nodes:
   2. Graph
   3. Possible paths:
   4. All possible paths must start with because it is the only node with more outgoing edges than incoming edges. Also, all paths must end with because it is the only node with more incoming edges than outgoing edges.
6. Yes, the graph has a Eulerian Circle because each node has the same number of incoming edges as it has outgoing. A Eulerian Circle starts and ends at the same node, and it visits every edge exactly once. One path is .
7. Assuming all the reads are the same length, the reverse complement of each read can be found within the set of reads and they can be grouped together in an edge. Then, each edge can be divided into a prefix and suffix, and a De Bruijn graph can be formed. Starting with the node with outdegree > indegree, traverse the graph until the outdegree of the current node is greater than 1. At this point, traverse all outgoing paths and return the path that has the same length as the number of edges in the graph. Finally, condense the path into a double-stranded DNA sequence.
8. The reconstructed genome is . Starting with an empty array of characters, each node’s sequences are added to the array. For example, the first node is . Knowing that the two sequences are separated by a single base pair, the sequences can be added to the array as follows: . This process continues with the next node, These sequences are added starting at index 1, leading to the following array: . After this, the last character of each subsequent bottom sequence is appended to , yielding the final sequence.
9. Dr. Smart’s suggestion is based on the fact that if a vertex has a high indegree, it is likely to be the final node in the cycle. However, if the indegree of any vertex does not equal its outdegree, a Eulerian cycle cannot be formed for the graph. For example, the graph in Problem 7 has a Eulerian cycle because every vertex has indegree = outdegree.
10. De Bruijn bubbles
    1. Devise an algorithm for detecting bubbles  
       detectBubbles(DNA):  
       traverse(DNA, 0)  
       bubbles = []  
        if currentNode.outgoing > 1  
        traversal1 = traverse(DNA, node1.index)  
        traversal2 = traverse(DNA, node2.index)  
        if any(traversal1) in traversal2  
        bubbles.append(currentNode)
    2. Bubbles in de Bruijn graphs are caused by the addition of incorrect reads that form parallel paths from one node to another. Without further information, it is impossible to deduce which path is correct. The algorithm could either return both complete sequences or it could remove one of the sequences at random.