CS 170 HW9

Yeung John Li, SID: 23665588, cs170-ia

Discussion:

DNA fragment assembly algorithm

Explanation:

The idea of the algorithm is to use greedy algorithm to reconstruct the whole DNA sequence by picking the most overlapped reads.

1. We create a list of nodes, where each node is each read.
2. We compare every pairs of reads, O(n) pairs. In each comparison, we recursively calculate the maximum overlap count and the corresponding overlapped section. We create a directed edge for each comparison, the two reads as the nodes, the weight is their overlap count and the first read points to the second read.
3. We sort the list of edges by their weight, the overlap count.
4. We loop through the sorted list of edges and try to add the nodes & edge to our list representation of connected components without creating a loop. (list of connected components where each connected component is a list of all edges & nodes in traverse order.)
5. After the loop, we obtain a list of edges selected. We reconstruct the original sequence from the list of edges.

Pseudocode:

```
def solver(reads):
        reads = truncate_reads(reads)        #remove duplicates and other unnecessary reads
        edges = []
        for read1, read2 of all possible pairs of reads:
                edges.add(get_overlap(read1, read2))
        edges = sort(edges)
        picked = []
        for edge in edges:
                pick edge if doesn't create a cycle
        return sum_sequence(picked)        #sum up sequence from edges in picked

def get_overlap(read1, read2):
        for index in k..1:
                if read1[1..index] == read2[-index..-1] or read2[1..index] == read1[-index..-1]:
                        return index
        return 0
```

Asymptotic Runtime Analysis:

1. Creating the list of edges takes a runtime involves $O(n^2)$ comparison of read pairs, each comparison checks for k possible overlap and each overlap takes O(k) to verify. Thus, the edges generating process takes a total of $O(n^2k^2)$.
2. Traversing this list of edge takes a runtime of $O(n^2)$ since there are only $O(n^2)$ edges.

Thus, the total runtime is $O(n^2k^2)$.

Efficiency

The algorithm I supplied is fairly efficient. The use of greedy algorithm in this case is much more efficient than in a regular graph as our sequence is on the same line that each read only points to one another read. Thus, to construct this sequence from a list of ordered edges by their overlap, we only need a runtime of $O(n^2)$, the length of edges. The checking of whether the edge produces a cycle is cleverly done by using a hash table and thus has a runtime of $O(1)$.

To calculate the overlap of two reads, we just need $O(k)$ checks where each check verifies $O(k)$ characters. Thus, the whole process takes only $O(k^2)$ and since there are $O(n^2)$ pairs of reads, we need a total of $O(n^2k^2)$.