CS170–Fall 2014 — Solutions to Homework 9

Steffan Voges, SID 23434518, cs170-cz

November 6, 2014

Collaborators: Ryan Flynn

1. Subsequence

Main idea. Create a directed graph with each read represented as a node. There exists an edge between a node a and another node b if the suffix of node a overlaps with the prefix of node b, with the weight of the graph labeled as the length of the overlap between the two reads. Then, use a greedy algorithm to reduce the graph down to one node. Select the edge with the greatest weight, merge the two nodes together, and adjust the corresponding edges. Repeat the process until you are left with only 1 node, which you select as your superstring.

Pseudocode.

```
def create_graph(reads):
    for read in reads:
        for node in reads:
            distance = overlap_distance(read, node)
            graph.add_weighted_edge(read, node, distance)
    return graph

def assemble(graph):
    while graph.nodes() > 1:
        edge = maximum edge in graph
        new_vertex = edge[0] + edge[1]
        add new_vertex to graph
        align edges previously connected to edge[0] and edge[1] to new_vertex
    return graph.nodes()

def overlap_distance(x, y):
    return overlap between suffix of x and prefix of y
```

Run time. $O(n^3k^2)$

Analysis of Runtime. Overlap_distance takes $O(k^2)$ since at worst, we compare every character in x to every character in y. Create graph will then take $O(n^2k^2)$ time- for each node, we look at the overlap distance to every other node. Next, we look at assemble. We look at only n nodes since at each step, we merge two nodes together. To pick the maximum node takes $O(logn)$ time when using a priority queue to keep record of the edges. We then take $4n^2k^2$ time to re-align all of the

edges previously coming in and out of both the first and second vertex. Since we run create_graph and assemble only once, our algorithm runs in $O(n^2k^2 + nlogn + 4n^3k^2) = O(n^3k^2)$