Bioinformatics I
WS 15/16
Tutor: Alexander Seitz

1	2	3	4	$\sum$ (6+2)

Jonas Ditz
& Benjamin Schroeder

Blatt 9

(Abgabe am 18. January 2016)

Theoretical Assignment - *Overlap and Layout*

graphs

The graphs from the exercise are shown in Figure 1-2. Figure 1 is the overlap grap with minimal spanning tree (red edges) and Figure 2 shows the layout.

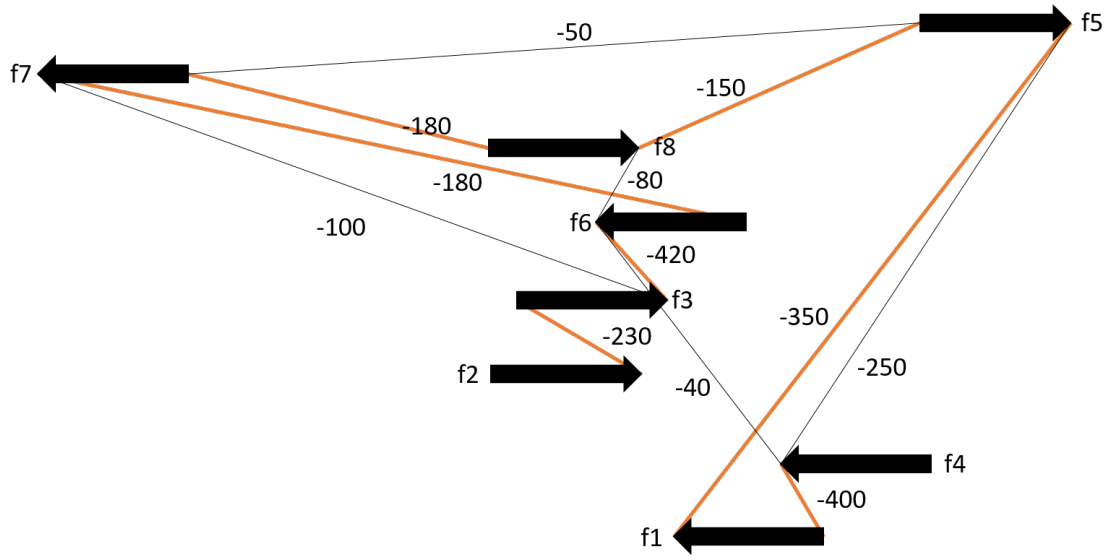

Figure 1: Minimal-spanning-tree

Is the Layout consitent?

One overlap (f_6 and f_8) is not consistent, because:

$$o(3'f_6 - 3'f_8) = 80$$

Theoretical Assignment - *Overlap graph figure*

A Figure for a paper would look like Figure 3.

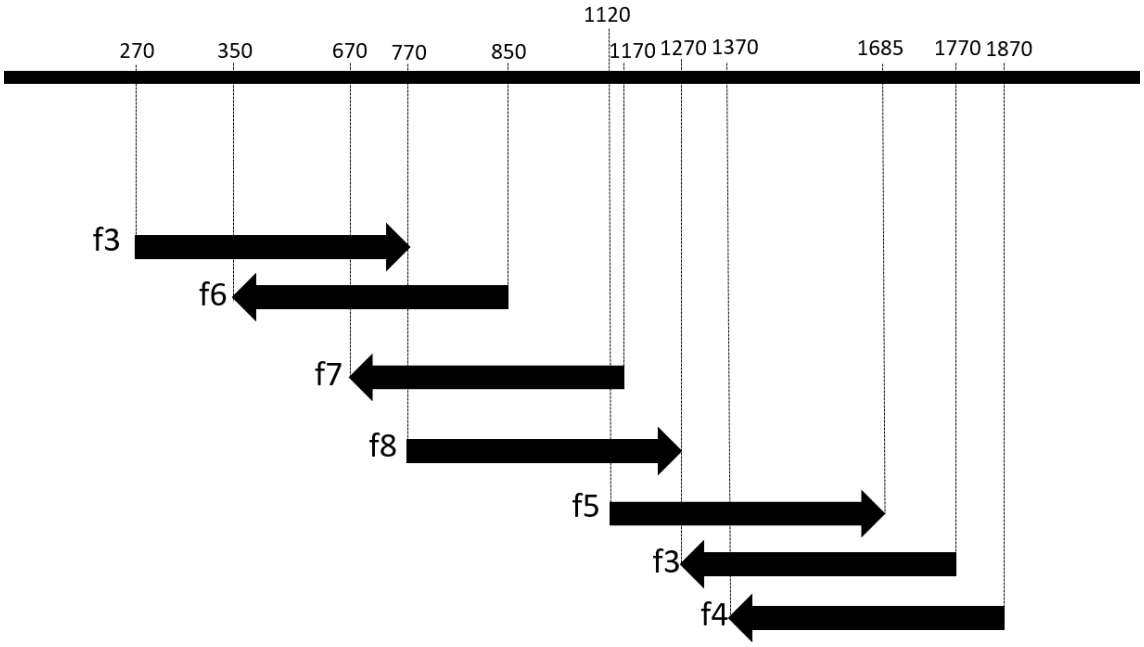

Figure 2: layout

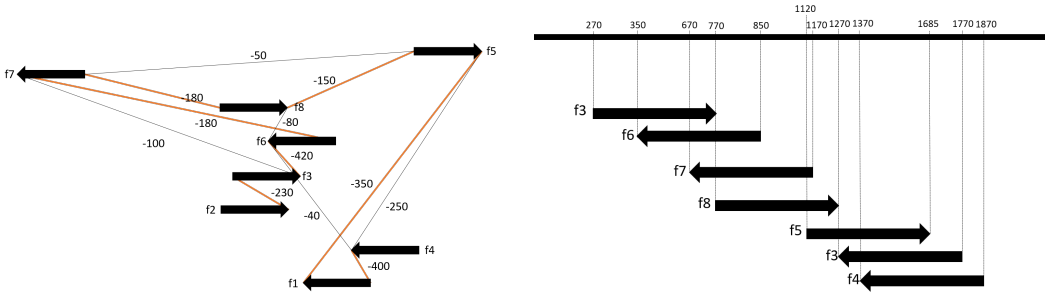

Figure 3: **left**: Overlap Graph with minimal spanning tree in red. **right**: Layout computed with the minimal spanning tree shown on the left site.

Theoretical Assignment - *Computing all overlaps using suffix tree*

Finding all overlaps is basically a APSP (All-pairs-suffix-prefix) problem. Solving such a problem with a suffix tree approach has a time complexity of $O(n + k^2)$ where n is the total length of all strings and k is the count of strings. First all reads will be concatenated with a separation character between all reads and a terminal character at the end. Now it is rather simple to check, whether the prefix of one read matches with the suffix of another read. This can be done using the suffix tree since we know that each prefix starts with an separation character (one could put one separation character at the beginning of the string to make this statement true) and every suffix ends ether with the separation character or the terminal character. Since one has to go trough the whole concatenated string and compare the prefix of each read with the suffix of every other read,

this approach has a total runtime complexity of $O(n + k^2)$.

Bonus - *Challenges of your Project*

One of the biggest challenges during our project was the framework we chose, Vaadin. We were not able to get it working of all computers we used. So in the end it was just one of us, who was able to work on the front-end layer. That slowed us down a lot. If we would start this project again from the scratch, Django would be a better choice. This framework comes with a development server and, therefore, would maybe make developing together easier. In the end we wasted a lot of time with Eclipse/Vaadin/Eclipse+Vaadin problems. Another challenge was the database schema for POODLE. We, especially Benjamin, talked a lot with people from Silke Wiesner's group at the MPI for Developmental Biology and that resulted in several schema changes during our project time.