18.417 Introduction to Computational Molecular Biology

Problem Set 3 — Issued: October 5, 2004
Lecturer: Ross Lippert — Due: October 19, 2004

This is a short section, so the problem set will be likewise short.

1. 8.6 of JP.

2. 8.9 of JP.

3. 8.12 of JP. Note, the equation should read $1 - e^{-c}$ in this problem.

4. (*) Given a set of fragments, F, the shortest superstring can be found by solving a TSP problem on the *overlap graph*. Recall that the overlap graph has F as its nodes and an edge from v to w ($v, w \in F$) if $overlap(v, w) \geq T$ for some threshold T.

   Real assemblers do not attempt to find a best Hamiltonian path through this graph but are satisfied with obtaining a set of non-intersecting *confident* paths for some notion of confidence (usually having to do with the quality of overlaps along the edges of the path). These good paths are called *contigs*. Each path, $v_1 v_2 \cdots v_n$ corresponds to a superstring obtained by merging/concatenating the v_i. That string is treated as an indivisible unit during subsequent phases of assembly.

   If we have a hypothetical genome S and a set of fragments F, a contigging algorithm is called *consistent* for (S, F) when the contigs produced are all substrings of S. Consistency tells us that it is not impossible to reconstruct S from F.

   The *greedy contigging* algorithm proceeds by making each node a trivial contig and then joining pairs of contigs $v_1 v_2 \cdots v_n$ and $w_1 w_2 \cdots w_m$ together (making a new contig $v_1 \cdots v_n w_1 \cdots w_m$) when (v_n, w_1) is an edge and $overlap(v_n, w_1) = \max_{x,y} overlap(x, y)$ where x ranges over contig ends and y ranges over contig starts.

   This can produce inconsistencies on simple repeats. For example take

$T = 2$ and

f_1	t	a	c	a	t										
f_2			c	a	t	g	g	c							
f_3							g	c	t	t	g				
f_4									t	t	g	a	c		
f_5												a	c	a	t	c
S	t	a	c	a	t	g	g	c	t	t	g	a	c	a	t	c

and greedy contigging produces $f_1 f_5 \rightarrow tacatc$ and $f_2 f_3 f_4 \rightarrow catggcttgac$. We see that even though the S can be assembled from a contig of the overlap graph the contigs which are produced are inconsistent with S, meaning S cannot result from any subsequent assembly steps.

The Celera assembler used a more conservative algorithm. Given a contig $v_1 \cdots v_n$ a *best-suffix* is the contig $w_1 \cdots w_m$ such that $overlap(v_n, w_1) = \max_x overlap(v_n, x)$ and a *best-prefix* is the contig such that $overlap(w_m, v_1) = \max_y overlap(y, v_1)$, where x ranges over contig ends and y ranges over contig starts. We merge contigs A and B into the contig AB if and only if A is the only contig who has B for a best-suffix and B is the only contig having A as a best-prefix. This was called the *thickest edge* rule, though that is probably a misleading name.

For the example above, this is consistent: we do not merge F_1 and F_5 because F_1 is a best prefix of both F_2 and F_5. However, this algorithm is not consistent for every (S, F). Produce an example where the thickest edge rule will give inconsistent contigs, but where there exists a contig $f_1 f_2 \cdots f_k$ in the overlap graph whose superstring is S.[1]

5. 8.20 of JP. To clarify this problem: given a peptide, you have a set of theoretical masses which come from al the N-terminal prefixes and the C-terminal suffixes. If a mass modification happens at amino acid i, all prefixes after i and suffixes before i are modified likewise. Thus the problem is one of "Allowing k arbitrary mass modifications to the amino acids of the input peptide, find the highest shared peak count between the mass spectrum of the modified peptide and some arbitrary input mass spectrum (about which you know nothing).

[1] It is very possible that I have mis-written some part of this problem. If clarifications need to be made, I'd rather not make them the night before the problem set is due.

6. (*) 8.26 of JP. "Efficient" here means $O(n^d)$ for a smallish d. Don't reduce it to a "find a Hamiltonian path problem" and think that will do.

7. This is a simplification of the problem of identifying a substring of a large database like SwissProt or NRAA (tens of millions of amino acids long) that matches the mass of some peptide (5-30 amino acids long). n can be 10^8, N will be in the hundreds or thousands, while the m_i's are from 500 to $50k$ (that is beyond the upper limit of reliable mass measurement on a mass spec instrument).

   Devise an $O(n)$ algorithm which will locate all substrings of mass m in a string S (of length n) of amino acid sequences given a table of amino acid weights (positive integers). Extend this algorithm to allow the location of N masses $\{m_1, m_2, \ldots, m_N\}$ in S in time faster than $O(Nn + X)$, where X is the number of actual matches (that is, it should perform better than $O(Nn)$ when X is small but no worse that $O(X)$ when X is large).