Biological Sequence Analysis (Spring 2015)

Exercise 3 (Thu 29.1, 10-12, B222, Veli Mäkinen)

Do any 5 assignments from below.

1. **Database search I.**

   `BLAST` is a heuristic aligner that makes local alignment feasible in large sequence databases. The following describes the main principles of `BLAST` with some simplifications. The database is indexed using a *k-mer index*, where each substring W of length k is associated with a list of pointers to the occurrence of W in the database. The lists of pointers of k-mers that are within Hamming distance 1 from some substring of a *query* sequence give the *candidate occurrences*. Dynamic programming is applied to extend candidate occurrences and to join nearby candidates, to form the final alignment results.

   Show that `BLAST` is a *lossy filter*, meaning that it might miss some optimal local alignments.

2. **Database search II.**

   Implement the k-mer index, e.g. by modifying first week's code for k-th order Markov chain. Basic version is enough without considering how to compress the lists of occurrences.

3. **Database search III.**

   Implement `BLAST`-like search on top of the k-mer index of the previous assignment to report candidate occurrences. You can ignore the dynamic programming part.

4. **Database search IV.**

   While there are many alternatives to `BLAST`, that obtain better filtering accuracy, speed, and space, the popularity of `BLAST` is mostly explained by its handling of statistical significance of the found alignments. Rather than ranking alignments directly by maximum score, it takes into account the probability of finding equally good alignments by chance. Get familiar with this process by reading this tutorial `http://www.ncbi.nlm.nih.gov/BLAST/tutorial/`. Be ready to explain the main concepts.

5. **Sparse dynamic programming I.**

   Show that set $M = M(A, B) = \{(i, j) \ : \ a_i = b_j\}$ needed for sparse dynamic programming LCS computation, sorted in reverse column-order, can be constructed in time $O(\sigma+|A|+|B|+|M|)$ time on constant alphabet and in $O((|B|+|M|)\log|A|)$ time on ordered alphabet. Observe also that this construction can be run in parallel with the main algorithm to improve the space requirement to $O(m)$.

6. **Sparse dynamic programming II.**

   The sparse dynamic algorithm for distance $D_{\mathtt{id}}(A, B)$ can be simplified significantly if derived directly for computing $|\mathsf{LCS}(A, B)|$. Derive this algorithm. *Hint.*

The search tree can be modified to answer range maximum queries instead of minimum.

7. **Affine gap scores I.**

   The lecture material defines gaps as runs of indels. Consider the alternative definition of gaps as runs of insertions or runs of deletions. Modify the basic recurrence for alignment under affine gap scores accordingly. What running time you obtain?

8. **Affine gap scores II.**

   Consider the setting above. Modify the Gotoh algorithm to handle the runs of insertions and runs of deletions separately. *Hint.* You will need three tables.

9. **Affine gap scores III.**

   Consider the setting above. Modify the invariant-based algorithm to handle runs of insertions and runs of deletions separately.