

## Biological Data Analysis (CSE 182) : Assignment 3

### Pattern matching

1. (20 pts.) Assume a random database of  $n$  nucleotide, with each base being equally likely. Similarly, assume that the query strings sum up to  $m$  nucleotides.
  - (a) Derive an expression for the expected number of  $\ell$ -mer matches between a query sequence and a database substring. Explain with examples how the expected number changes as  $m$  goes from a very large number to  $\ell$ ?
  - (b) Suppose the query sequence has exactly  $\frac{m}{r}$  strings, each of length  $r$ , and you are only interested in an  $\ell$ -mer match between a query string and a database substring. In other words, you do not allow  $\ell$ -mer matches that span multiple query strings. How does the expected number of  $\ell$ -mer change?
  - (c) Suppose that all nucleotides are not equally likely. Instead, the GC content of the database is 0.8. In other words  $\Pr[C] = \Pr[G] = 0.4$  and  $\Pr[A] = \Pr[T] = 0.1$ . Compute the expected number of matches of a 5-mer for the query string *AATAAGCCGC*, with  $m = 10$ .

2. (20 pts.) Design a seed-and-extend strategy to search for homologs, as follows: the database is a single DNA string of size  $n$ . You can assume that the database is random with each nucleotide occurring with equal probability. You query the database with many sequences, each of length  $r$ , for a total length of  $m$  bp (i.e., there are  $\frac{m}{r}$  query sequences). For each query sequence, your goal is to identify all database homologs that have  $\geq 85\%$  sequence identity with the query. Here, % identity refers to the percentage of alignment positions which have a 'match', or identical nucleotides. You can also assume that the number of indels is small and the total number of columns in the alignment with a true homolog is approximately  $r$ . Define the *sensitivity* of a search as  $\Pr(\text{any true homolog of any query is recovered})$ . The naive algorithm uses a local-alignment approach and has a sensitivity of 1.0, while taking  $O(nm)$  steps.

To speed-up the search, you use a seed-and-extend strategy: (a) search for an exact match of all query keywords of length  $\ell$ . (b) When you get a match between a keyword and a database sequence, you compute a global alignment of the query sequence against a size  $r$  sub-sequence from the database that contains the matching keyword. As discussed in class, this strategy improves speed but loses sensitivity. Define *speed-up* as the ratio of  $nm$  to the speed of a seed-and-extend search. Use the discussion from class to derive expressions for speed-up and of sensitivity (in terms of  $n, m, r, \ell$ ). Assume  $n = m = 10^7, r = 100$ . Using your method, compute a table with speed-up and sensitivity values for  $\ell \in \{5, 11, 15, 20, 25, 30, 35, 40\}$ . Submit the method or code used to compute speed-up and sensitivity, and the table.

3. (20 pts.) Build (using paper and pencil) an aho-corasick automaton for a dictionary containing 3 words. Show all failure links except the ones going to the root node, and all transition links. Submit a pdf with the automaton hand-drawn or drawn using some graphics toolkit. The words are: AAGTCGTATGC, AGTCTTAT, TCTTATAGC.
4. (40 pts) Write a program to implement an Aho-Corasick trie without failure links for the supplied dictionaries. Implement a search algorithm for searching the database against the two dictionaries, and report the number of matches to each keyword in the dictionary. Using E-value calculations for each keywords, say if the number of matches is identical to your expectation, less than what you expected, or greater than what you expected. Explain what might be happening. Below, please find some suggestions to help you with the computation.
  - First, run on small database sub-strings and get a time estimate of the entire computation.
  - Write your code to start searching from *any* position in the database, not just the first.
  - Create *checkpoints*, so that the program writes answers to a file as it is progressing, not just at the end. This allows you to check for errors, and re-start the search from a different point, not having to start from the beginning each time.

- Learn about Alu sequences on the human genome to understand discrepancy between expected hits and actual number of hits.