GENOME-SCALE ALGORITHM DESIGN
by Veli Mäkinen, Djamal Belazzougui, Fabio Cunial and Alexandru I. Tomescu
Cambridge University Press, 2nd edition, 2023
www.genome-scale.info

## Exercises for Chapter 10. Alignment-based genome analysis

10.1 Calculate the probability of an SNP given a read pileup taking into account the measurement errormeasurement errors.

10.2 *Variant annotation* is the process of assigning a function for each detected variant. For example, a 1-base deletion inside an exon creates a frame-shift and may cause an abnormal protein product to be translated. Consider different variants that might appear and think about why some variants can be called *silent mutations*. Browse the literature to find out what *nonsense mutations*, *missense mutations*, and *regulatory variants* are.

10.3 Recall mapping quality Insight 10.1. Consider how it could be refined to take good alignments into account in addition to only the best. How would you in practice approximate such mapping quality under, for example, a $k$-errors search.

10.4 Consider splitting a pattern into fewer than $k+1$ pieces. How many errors should be allowed for the pieces in order to still obtain a lossless filter for the $k$-errors search?

10.5 Give the pseudocode for the algorithm depicted in Figure 10.2 to compute maximum scoring semi-local alignment along suffix tree paths.

10.6 Modify the above approach to solve the $k$-errors problem. Do you always need to fill the full dynamic programming matrix on each path? That is, show how to prune branches that cannot contain an occurrence even if the rest of the pattern $P_{j..m}$ exactly matches a downward path.

10.7 Modify the above approach to solve the $k$-errors problem using Myers' bitparallel algorithm instead of standard dynamic programming.

10.8 Modify all the above assignments to work with backtracking on the BWT index.

10.9 Most sequencing machines give the probability that the measurement was correct for each position inside the read. Let $\mathbb{P}(p_i)$ be such a probability for position $i$ containing $p_i$. Denote $M[c,i] = \mathbb{P}(p_i)$ if $p_i = c$ and $M[c,i] = (1 - \mathbb{P}(p_i))/(\sigma - 1)$ if $c \neq p_i$. Then we have a *positional weight matrix (PWM)* $M$ representing the read (observe that this is a profile HMM without insertion and deletion states). We say that the matrix $M$ *occurs* in position $j$ in a genome sequence $T = t_1 t_2 \cdots t_n$ if $\mathbb{P}(M,T,j) = \prod_{i=1}^{m} M[t_{j+i-1}, i] > t$, where $t$ is a predefined threshold. Give the pseudocode for finding all occurrences of $M$ in $T$ using backtracking on the BWT index. Show how to prune branches as soon as they cannot have an occurrence, even if all the remaining positions match $P_{1..j}$ exactly.

10.10 The goal in read alignment is typically to find the unique match if one exists. Consider how to optimize the backtracking algorithms to find faster a best alignment, rather than all alignments that satisfy a given threshold.

10.11 Some mate pair sequencing techniques work by having an adapter to which the two tails of a long DNA fragment bind, forming a circle. This circle is cut in one random place and then again $X$ nucleotides apart from it, forming one long fragment and another shorter one (assuming $X$ is much smaller than the circle length). The fragments containing the adapter are fished out from the pool (together with some background noise). Then these adapters containing fragments are sequenced from both ends to form the mate pair. Because the cutting is a random process, some of the mate pair reads may overlap the adapter. Such overlaps should be cut before using the reads any further.

a) Give an algorithm to cut the adapter from the reads. Take into account that short overlaps may appear by chance and that the read positions have the associated quality values denoting the measurement error probability.

b) How can you use the information about how many reads overlap the adapter to estimate the quality of fishing?

10.12 Construct the Burrows–Wheeler transform of `ACATGATCTGCATT` and simulate the 1-mismatch backward backtracking search on it with the read `CAT`.

10.13 Give the pseudocode for computing the values $\kappa(i)$ for prefix pruning applied on the prefixes $P_{1..i}$ of the pattern $P$.

10.14 Show that the values $\kappa(i)$ in prefix pruning are correct lower bounds, that is, there cannot be any occurrence missed when using the rule $k' + \kappa(i) > k$ to prune the search space.

10.15 Show that the computation of the values $\kappa(i)$ in prefix pruning is also feasible using the forward BWT index alone by simulating the suffix array binary search.

10.16 Give the pseudocode for the $k$-mismatches search using case analysis pruning on the bidirectional BWT. You may assume that a partitioning of the pattern is given together with the number of errors allowed in each piece. Start the search from a piece allowed to contain the fewest errors.

10.17 Show that suffix filtering is a lossless filter for a $k$-mismatches search.

10.18 Compute the minimal read length $m$ such that the expected number of occurrences of the read in a random sequence of length $n$ when allowing $k$ mismatches is less than $1/2$.

10.19 Compute the minimal read length $m$ such that the expected number of occurrences of the read in a random sequence of length $n$ when allowing $k$ errors is less than $1/2$ (use approximation, the exact formula is difficult).

10.20 Consider different large-scale variations in the genome, like gene duplication, copy-number variation, inversions, translocations, etc. How can they be identified using read alignment? Is there an advantage of using paired-end reads?

10.21 A greedy algorithm to solve Problem 10.1 is to start with empty $E'$, choose $h \in H$ with most incoming edges not in $E'$, add those unmatched edges to $E'$, and iterate the process until all $r \in R$ are incident to an edge in $E'$. Give an instance where this greedy algorithm produces a suboptimal solution.

10.22 Consider the following algorithm for the minimum-cost set cover problem. Start with an empty $C$ and iteratively add to $C$ the most cost-effective set $H_i$, that is, the set $H_i$ maximizing the ratio between $c(H_i)$ and the number of un-covered elements of $\{r_1, \ldots, r_m\}$ that $H_i$ covers. This is repeated until all elements are covered by some set in $C$. Show that this algorithm always produces a solution whose cost is $O(\log m)$ times the cost of an optimal solution.

## Additional exercises not in the book

10.23 Construct the Burrows–Wheeler transform of `ACATGATCTGCATT` and the Burrows–Wheeler transform of the *reverse* of `ACATGATCTGCATT`. Simulate the 1-mismatch search on the corresponding BWT indexes using case analysis pruning with the pattern `GTTC`.