Index Based Genome Mapper

Maarja Lepamets and Fanny-Dhelia Pajuste

Curriculum: Computer Science 2013/14
Institute of Computer Science
Faculty of Mathematics and Computer Science
Tartu University

Project repository: *https://github.com/maarjalepamets/read-mapping*

Introduction

One of the "hottest" terms in today's medical world is "personal medicine". For this to be possible, scientists need to sequence as many individuals as possible and map the reads back to the reference genome to find out the locations of the given sub-sequences, expression rates of the genes or the sequence of the full genome. This kind of biological data is, however, so large that using several software applications in its analysis is inevitable. When it comes to indexing and mapping to the genome, there are two widely-used algorithmic approaches. One involves modifications of suffix trees and Burrows-Wheeler transformation, another different kinds of hashing (Schbath *et al*, 2012). In this project we were implementing a latter type of genome indexer together with a mapping tool which relies on pairing every n-nucleotide sequence with its locations in a genome.

Indexing the Genome

The first step of mapping the sequencing reads is to index the full genome. For this we iterate over all chromosomes and extract each n-nucleotide ($n \leq 16$) word together with its genomic location. Then we sort the words using a linear time in-place radix sort combined with insertion sort for small buckets (Duvanenko, 2009) and keep only one copy of each word and a pointer to the starting position of its genomic locations. The overall time-complexity of the algorithm is $O(n)$ and altogether it takes $4 \times$ *size of the genome* $+ 8 \times$ *unique word in the genome* bytes.

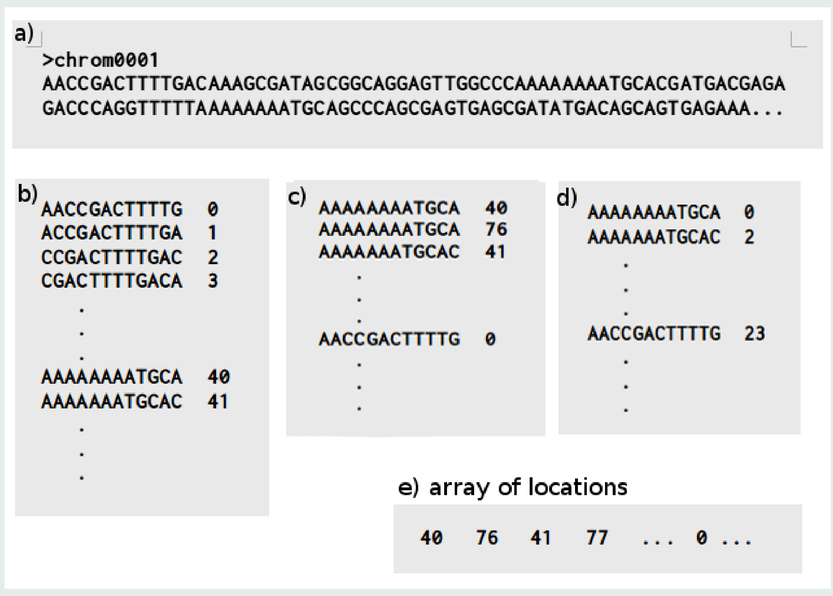

Figure 1: a) Genome sequence; **b)** All extracted n-mers; **c)** n-mers sorted with in-place radix sort; **d)** Only one copy of each n-mer and the pointer to the array of locations (**e**)).

The Evaluation

To evaluate the genome indexer we measured the time of creating indices for full genomes of four different organisms using two different word lengths (see Table 1). The time of writing a binary index file is excluded. We also noted the size of the index files but since we used no compressing the full human genome index took upto 20 GB for 16-nucleotide words and 11 GB for 10-nucleotide words.

Table 1: Measurements of run-time

	$n = 10$	$n = 16$
E. coli	0 min 0.5 s	0 min 0.6 s
P. aeruginosa	0 min 0.7 s	0 min 0.8 s
Mus musculus	16 min 22.1 s	26 min 15.4 s

Mapping Tool

The mapping process consists of two parts. In the first part the read is cut into over-lapping n-nucleotide seeds. The index created in the previous step is then used to find all locations of those seeds in the genome. When all seeds map to the similar location on the genome, we have found a perfect match. If only some map, we move to the second part of the mapping to find matches with errors by calculating simple edit distance between our read and the potential region in the genome. The time-complexity of the mapping tool is $O(m \log n)$ where m is the number of words and n the length of the genome.

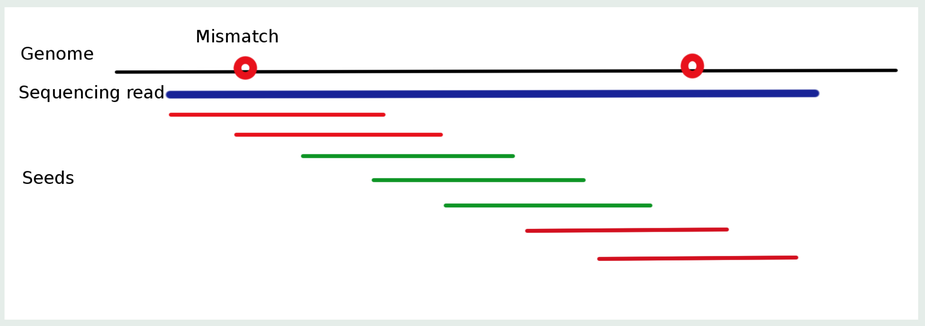

Figure 2: Read is cut into seeds. Green seeds map to the given genome location, red seeds do not because of the mismatches (red dots). The second part of the mapping tool is for deciding whether there are more or less mismatches in this genome region that what the user allows.

We ran the mapping tool on a data-set with over 48,000 reads. We looked for genome locations where given reads would map with less than four errors. Tools run-time was approximately 30 seconds.

References

Schbath, S., Martin, V., Zytnicki, M., Fayolle, J., Loux, V., Gibrat, J.-F. (2012) Mapping Reads on a Genomic Sequence: An Algorithmic Overview and a Practical Comparative Analysis. J Comput Biol., 19(6): 796-813.

Duvanenko, V. J. (2009) In-place Hybrid N-bit-Radix Sort. http://www.drdobbs.com/architecture-and-design/algorithm-improvement-through-performanc/221600153