PhD Proposal

Davide Cozzi, 829827, d.cozzi@campus.unimib.it

Introduction

The problem of *pattern matching* is one of the most studied topics in the field of algorithmics and bioinformatics. For example the interest in such problems is due to the need to align sequences or search for specific patterns within the *DNA*.
In this context, a large number of data structure algorithms have been modeled. Among these, one of the most used is the **Burrows-Wheeler transform (*BWT*)** thanks to the studies of Ferragina and Manzini who proposed its use together with the so-called FM-index [1].
In recent years, in the field of bioinformatics, there is a change of interest. While until a few years ago the research was focused on the study of the *genome*, we are now starting to deepen the topic of *pangenome*. Currently, the need to take into account the high variability in population genomes as well as the specificity of an individual genome in a personalized approach to medicine is rapidly pushing the abandonment of the traditional paradigm of using a single reference genome [2].
In this context, various algorithms and various data structures have been implemented in order to study the so called *haplotypes* and *genotypes*. Briefly we could define *haplotypes* as a combination of allelic variants, inherited from a parent. Instead we can define the *genotype* as the complete set of genes contained in the DNA. From a biological point of view it is indeed interesting to note that each individual shares about 99% of the genetic code, while the remaining 1% differs mostly with **Single Nucleotide Polymorphisms (*SNPs*)**, i.e. variations of a single nucleotide in a precise *locus* of the DNA
Thanks to the last developments in sequencing technologies, which have led both to reduce the costs of single sequencing and to produce sequences of ever higher quality in less and less time, the researchers were able to theorize the *pangenome graph*, replacing the old representation of the single genome by a single sequence.
One of the most important data structure developed in order to handle the study of haplotypes sequences is the **positional Burrows-Wheeler transform (*PBWT*)**, proposed by Durbin in 2014 [3]. With this particular data structure it was possible both to study possible matches inside a haplotype panel and to search for set maximal matches between an external haplotype and a panel, as well as the set maximal matches inside the panel itself. Furthermore, for example, variants have also been studied for the management of the multiallelic case [4].
In 2021 Rossi et al. proposed *MONI* as a data structure to handle a **run-length version of BWT (*RLBWT*)** with the ultimate intention of indexing and using multiple genomes as a reference [5]. Together with this data structure the authors proposed the concept of *matching statistics* in order to efficiently compute the matches between a pattern and a text. A recent improvement, regarding the *RLBWT*, has been made through the implementation of *PHONI* [6], where where the so-called *longest-common-extension (LCE) queries* are used.
During the development of my master's thesis I worked in collaboration with the authors of MONI in order to create a run length encoded variant of the PBWT. In this work I used practically all concepts applied to the "classic" *BWT* in *MONI* and *PHONI*.
During my PhD I will focus my studies on the development of new algorithms in various topics such as variants of the *PBWT* (and any related uses), *haplotyping/genotyping* or solutions concerning the study of *SNPs*.It is also my intention to deepen the more experimental themes relating to pattern matching, in detail the new developments on BWT and new indexing structures, as well as the theme of *succinct data structures*, with particular attention to the use of *bitvectors*.

State of the art

I will now present a brief overview of the main algorithms, data structures, methods etc that will be the core of my studies during my PhD.

BWT

The **Burrows-Wheeler Transform (*BWT*)** [7] was introduced in 1994 in order to compress texts but it has had then wide use in bioinformatics, above all thanks to the already cited *FM-index*. Given a text T, \$-terminated, such that $|T| = n$, we can define the BWT_T, denoting with SA_T the *suffix array* of T, as: $BWT_T[i] = T[SA_T[i] - 1]$, if $SA_T[i] > 0$, and $BWT_T[i] =$ \$ otherwise. Less formally we can say that $BWT_T[i]$ is the character that precedes the i-th suffix in the lexicographically order. It is important to note that this transform is reversible so we can reconstruct the text T from its transform BWT_T using the so-called **LF-mapping**. Given BWT_T and an array, called F_T, with all the characters of T in the lexicographically order, we can say that, thanks to the *LF-mapping*, the j-th occurrence of a certain character in BWT_T corresponds to the j-th occurrence of the same character in F_T, so we can reconstruct T starting from its last character \$. With the use of the *LF-mapping* we can perform the *backward-search* in order to use the BWT_T to look for a pattern P within T. This can be done efficiently thank to the *FM-index* which consists of two functions. The first one is C function, such that, given an alphabet Σ (that includes the ending character \$), $C : \Sigma \to [1, n]$. This function, given a character $\sigma \in \Sigma$ returns the number of occurrences of characters lexicographically than the one given as argument in T. The second one is the Occ function, $Occ : \Sigma \times [1, n] \to [1, n]$, has as arguments a character $\sigma \in \Sigma$ and an index i of BWT_T and returns the count of occurrences of σ in $BWT_T[1, i]$ (**CONTROLLARE E RISCRIVERE MEGLIO**).
The use of BWT has allowed the construction of efficient algorithms both in the field of pattern matching and in that of sequence alignment.

Bitvectors

Bitvectors are ones of the most important data structure when mentioning succinct data structures.
A *bitvector* is an array on n bits which allows two particular operations, called **rank** and **select**, in addition to the classic operations possible on Boolean arrays, such as *random access* in *constant time* (**DA VERIFICARE**). More in detail the rank function allows you to calculate how many values of 1 are up to a certain index. Instead the select function allows to obtain the index of any one present in the bitvector. Formally, given a bitvector B, such that $|B| = n$, and given an index i, such that $0 \leq i < n$, we can define $rank_B(i) = \sum_{k=0}^{k<i} B[k]$. Instead, about the select function, given an integer i, such that $0 < i \leq rank_B(n)$, where $n = |B|$, we can define $select(i) = \min\{j \mid rank_B(j + 1) = i\}$.
From a purely theoretical point of view, with the additional cost of $\mathcal{O}(n)$ bits in memory, these two operations can be supported with constant time. In more practical terms there are several implementations of the same within **SDSL (*Succint Data Structures Library*)** [8], one of the most important C++ library used in bioinformatics. As the implementation changes (for example *plain bitvector, interleaved bitvector, sparse bivector* etc...) the computational time of the two operations varies (usually only one of the two is in constant time) and the amount of additional bits needed.
An example of the use of bitvectors is to track the runs in the run-length encoded implementations of BWT and $PBWT$, where we put one at each head of run.

RLBWT

Speaking of pangenome, linear indexing via FM-index is no longer the best solution as it does not handle the large repetitions present in this new type of sequences in the best possible way. In 2005 Mäniken and Navarro defined the **Run-Length Burrows–Wheeler Transform (*RLBWT*)**. Given a text T, $RLBWT_T$ is a rappresentation of BWT_T with a compact rappresentation of consecutive equal characters, the so-called *runs*. With this change of perspective the algorithms have gone from being linear over the length of the text, n, to being linear over the number of runs, r.
The new indexing method, which was then introduced by Gagie et al. [9], was called **r-index** and corresponds to the $RLBWT$ and a SA sampling at the begin and at the end of every run. The algorithm for querying

through the *RLBWT* take advantage of other methods, such as the use of **thresholds** (minimum *LCP* value between two consecutive runs of the same character) in *MONI*, and the use of **longest common extension (*LCE*) query** (to compute the right equal common extension between two position in the text) cd[10] and **straight-line programs (*SLP*)** [11] (for *random access* and *lce queries* over the grammar-compressed text) in *PHONI*. Both these solutions are used for the computation of the **matching statistics (*MS*)**. Given a pattern P and a text T the *MS* of P in respect to P is an array M of pairs position/length, $|M| = |P|$, such that $T[M[i].pos : M[i].pos + M[i].len - 1 = P[i : i + M[i].len] - 1]$ and $P[i : i + M[i].len]$ does not occur in T. Given *MS* we can compute every **Maximal Exact Match (*MEM*)** of a pattern in a text. Given a text T and a pattern P a substring of the pattern $P[i : i + l - 1]$, of length l, is a *MEM* of P in T if $P[i : i + l - 1]$ is a substring of T but but neither $P[i - 1 : i + l - 1]$ nor $P[i : i + l]$ are. At the end, using a particular function called φ (and φ^{-1}), based on the use of the **inverse suffix array (*ISA*)** as well, it was possible to start from the starting positon of a match extracted by *MS* and find all the starting positions of all the copies of P in T. Infact, formally, given a starting position p, we have $\varphi(p) = SA[ISA[p] - 1]$, $NULL$ if $ISA[p] = 0$, and $\varphi^{-1}(p) = SA[ISA[p] + 1]$, $NULL$ if $ISA[p] = |T| - 1$, where $ISA[i] = j$ iff $SA[j] = i$.
Thanks to these and other methods it was possible to perform pattern matching efficiently even on long sequences of nucleotides, such as those studied in a pangenomic context.

PBWT

Based on the theories of BWT Durbin, in 2014, he devised the positional **Burrows–Wheeler transform (*PBWT*)** [12], in order to solve the problem of pattern matching on panels (matrices) of haplotypes, denoted by X. In detail we are talking about a panel with M haplotypes and N biallelic sites. This data structure is based on a reversed-prefix ordering at each column k that produces two different multidimensional arrays. The first one is called **prefix array**, denoted by a, and, for each column k, contains, for each position i, the haplotype of index m in the original panel. More formally we can say that $a_k[i] = m$ iff X_m is the i-th haplotype in the reversed-prefix ordering at column k. Note that x_m such that $a_k[i] = m$ could be denoted by y_i^k. The second bidimensional array is called **divergence array**, denoted by d, and indicates the index of the starting column of the longest common suffix, ending in column k, between a row and its previous one, at reversed-prefix ordering at column k. More formally we can define $d_k[i] = h$ iff h is the smallest column index such that $y_i^k[h, k) = y_{i-1}^k[h, k)$. (**RISCRIVERE MEGLIO**)
Thanks to these two bidimensional arrays it is possible to compute all matches within X longer than a minimum length L, all set-maximal matches within X in linear time, all set-maximal matches (which we could also call "MEMs") from a new sequence z to X etc...

Research goals

For my master's thesis I therefore tried to combine the ideas related to the *RLBWT* with those of the *PBWT*, creating the **RLPBWT**. In order to obtain this result I have to rethink the concept of *Matching Statistics* for *PBWT*, how to compute the *SLP* for the panel, how to use *thresholds*/*LCE queries*, how to obtain the same behaviour of the φ function etc...
In detail, regarding *MS*, instead of the *pos* we track a *row* of the panel, regarding thresholds we use the index inside a run in a column where we have the minimum value of divergence array and, regarding the *SLP*, we "stretch" the reverse panel (from the right to the left) in in order to make *LCE queries* possible. Instead, regarding the φ function, I have implemented a new simple data structure to identify which row is above and which row is below each row in the panel permutated via the prefix array and stored in a run-length compressed way. (**RISCRIVERE MEGLIO ED EVENTUALMENTE APPROFONDIRE**).
Obviously there are some limitations, such as the study of biallelic panels only and the lack of management of any missing data, which are very frequent in the case of real data. First of all it will be interesting to implement a new version of the *RLPBWT* that can handle multiallelic data, adapting the current use of bitvectors to manage the *LF-mapping* also in that case. On the other hand, it will be more complicated to manage the missing data. This problem is known to be *hard* so I should probably deal with parametric algorithms or approximate algorithms, based on researches already developed in *BIAS*.

References

[1] Paolo Ferragina and Giovanni Manzini. Indexing compressed text. *Journal of the ACM (JACM)*, 52(4):552–581, 2005.

[2] Jasmijn A. Baaijens, Paola Bonizzoni, Christina Boucher, Gianluca Della Vedova, Yuri Pirola, Raffaella Rizzi, and Jouni Sirén. Computational graph pangenomics: a tutorial on data structures and their applications. *Natural Computing*, 21(1):81–108, Mar 2022.

[3] Richard Durbin. Efficient haplotype matching and storage using the positional Burrows–Wheeler transform (PBWT). *Bioinformatics*, 30(9):1266–1272, 01 2014.

[4] Ardalan Naseri, Degui Zhi, and Shaojie Zhang. Multi-allelic positional burrows-wheeler transform. *BMC bioinformatics*, 20(11):1–8, 2019.

[5] Massimiliano Rossi, Marco Oliva, Ben Langmead, Travis Gagie, and Christina Boucher. Moni: a pangenomics index for finding mems. *bioRxiv*, 2021.

[6] Christina Boucher, Travis Gagie, I Tomohiro, Dominik Köppl, Ben Langmead, Giovanni Manzini, Gonzalo Navarro, Alejandro Pacheco, and Massimiliano Rossi. Phoni: Streamed matching statistics with multi-genome references. In *2021 Data Compression Conference (DCC)*, pages 193–202. IEEE, 2021.

[7] Michael Burrows and David Wheeler. A block-sorting lossless data compression algorithm. 1994.

[8] Simon Gog, Timo Beller, Alistair Moffat, and Matthias Petri. From theory to practice: Plug and play with succinct data structures. In *13th International Symposium on Experimental Algorithms, (SEA 2014)*, pages 326–337, 2014.

[9] Dustin Cobas, Travis Gagie, and Gonzalo Navarro. A fast and small subsampled r-index. *arXiv preprint arXiv:2103.15329*, 2021.

[10] Lucian Ilie, Gonzalo Navarro, and Liviu Tinta. The longest common extension problem revisited and applications to approximate string searching. *Journal of Discrete Algorithms*, 8(4):418–428, 2010.

[11] Travis Gagie, Giovanni Manzini, Gonzalo Navarro, Hiroshi Sakamoto, Louisa Seelbach Benkner, Yoshimasa Takabatake, et al. Practical random access to slp-compressed texts. In *International Symposium on String Processing and Information Retrieval*, pages 221–231. Springer, 2020.

[12] Richard Durbin. Efficient haplotype matching and storage using the positional burrows–wheeler transform (pbwt). *Bioinformatics*, 30(9):1266–1272, 2014.