Received September 25, 2020, accepted October 8, 2020, date of publication October 14, 2020, date of current version October 28, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3031159

A Review on Sequence Alignment Algorithms for Short Reads Based on Next-Generation Sequencing

JEONGKYU KIM, MINGEUN JI, AND GANGMAN YI, (Member, IEEE)
Department of Multimedia Engineering, Dongguk University, Seoul 04620, South Korea
Corresponding author: Gangman Yi (gangman@dongguk.edu)

This work was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant NRF-2019R1F1A1064019.

ABSTRACT With recent advances in next-generation sequencing (NGS) technology, large volumes of data have been produced in the form of short reads. Sequence assembly involves using initial short reads to produce progressively longer contigs, and then using scaffolds to produce the final sequence. These processes each require evaluation of the extent of homology between different sequences. However, because the NGS platforms currently being developed are diverse, and the data being produced are of different sizes and read lengths, numerous algorithms are being developed with unique methodologies to process this complex data. It is difficult for biologists to manipulate the different features involved in these algorithms. Therefore, to reduce experimental trial-and-error, different strategies are required depending on the performance and purpose of the optimal algorithm, thereby facilitating understanding of algorithm methodologies and effective use of their various features. This study is a review of the different short read alignment algorithms and NGS platforms that have been developed to date, in order to aid efficient selection of algorithms for reference sequences and mapping of DNA data.

INDEX TERMS NGS, sequence alignment, read alignment, FM-index, hashing.

I. INTRODUCTION

In the pursuit to uncover the secrets of life, DNA analysis suffered many technical difficulties at first, but recent advances in different sequencing technologies have enabled continual production of DNA sequences from many organisms. However, it is no small task to rapidly and accurately analyze the massive volume of sequence that has been expanding enormously of late. There are two methods of assembly after sequencing: *de novo* genome assembly and reference assembly. *De novo* sequence assembly does not use prior knowledge to produce the genome, whereas reference assembly, in which sequences are mapped to a reference, requires pre-existing data for comparison. Through these developments, various assembly algorithms are being used to reveal genome sequences. The commonality involving the functions of these two assembly methods is that they compare homology between different sequences.

The associate editor coordinating the review of this manuscript and approving it for publication was Daniel Grosu.

Sequence alignment aims to compare homology between two different sequences, and has developed from general methods for sequence alignment to diverse methodologies suited to specific purposes. Among methods to identify alignment products, optimal sequence alignment is a well-known technique. The method aims to determine an optimal solution for read alignment; however, the time required for alignment increases with longer sequences, as the amount of data to calculate grows. Therefore, improved methods with even faster run times are needed. Wilbur and Lipman [5] proposed a heuristic method to efficiently align query sequences in a database. Using a faster method than optimal alignment, they aimed to resolve the problems of increasing alignment run time. Nevertheless, the time required for alignment still increased with increasing data size; thus, the need to reduce run time persisted. Recently, GPU-based hardware architectures are being developed to produce even faster algorithms [8].

The first attempt to analyze base sequences using DNA sequencing technology was Sanger sequencing, for which various methods have been used. As a 1st generation sequencing technology, the chain termination method is very expensive and takes a long time, but is relatively accurate. Subsequently, next-generation sequencing (NGS) was developed to overcome issues with cost and calculation speed [44], [100]. NGS is a new sequencing technology that can be performed more rapidly and inexpensively than previous DNA sequence analysis techniques. NGS-based methods align short reads, extracted via an NGS platform, against a reference sequence [6]. A mapping algorithm is used to match parts of the reads that are similar to the reference, but, depending on the type or size of data being investigated, it is very difficult to decide upon a suitable mapping algorithm for the input data. Stable alignment and mapping of short reads is not yet a fully resolved problem [55], [101]. Many mapping algorithms have already been developed, and over 70 short read mapping tools have been proposed [9].

NGS platforms are needed to classify imprecise sequence from DNA as informative data. Some of the earliest companies performing sequencing, Applied Biosystems [10], Solexa [11], and Life Technologies [12], have also been developing new NGS platforms. Since then, many other companies, through the development of NGS platforms and introduction of the latest technology, are using sequencing strategies focused on large-scale parallel alignment of short reads instead of long sequences, and NGS platforms have undergone rapid advancement since their commercialization [58].

In this paper, we explain the efficient short read mapping algorithms used in NGS. Our aim is not to explain the methods of use or rate the algorithms, but rather to explain their features and possible uses. This paper is divided into Sections 1 to 6. Section 1 is the Introduction. Section 2 describes the trends in NGS technologies and platforms that have been developed from the initial Sanger method. Section 3 provides a simple introduction to the methodologies and algorithms involved in commonly used sequence alignment methods, specifically differentiating between optimal sequence alignments and heuristic sequence alignments. Section 4 compares algorithms between different strategies using FM-indexes or hash tables to classify short read alignment algorithms, and thus explores methods of mapping short reads to reference sequences. Section 5 shows a concise workflow of the overall features and strategies for the short read alignment algorithms discussed in this review paper. Finally, Section 6 is the conclusion.

II. CURRENT STATUS OF NGS TECHNOLOGY AND PLATFORMS

Sanger sequencing has been used for a long time as a method of DNA sequencing. However, the new method of NGS has been rapidly distributed, and enables low-cost, high-efficiency sequencing, producing a large quantity of short reads. In the first stage of NGS, the platform divides the DNA sequence into a very large number of fragments [1]. The fragments, which are the generated data, consist of short reads. Finally, the short reads are mapped back onto reference sequences. Thus, NGS technology is an inexpensive and efficient method for DNA sequencing, and for re-sequencing whole genomes that provides a high volume of data for processing [59]–[61].

NGS platforms extract short read, and sequencing techniques that use NGS platforms are referred to as NGS or 2nd generation sequencing [1]. In terms of the generations, 1st (Sanger) and 2nd (NGS) generation sequencing use PCR, and, depending on the amplification method, this step can be divided across the platform manufacturers Illumina [16], Roche [66], and Life Technologies [12], [96]. In 3rd generation sequencing (Next-NGS), the process of PCR amplification is omitted, and the base sequence is analyzed by sequencing single DNA molecules. This process is called single molecule real-time (SMRT) sequencing. SMRT sequencing is used in the PacBio RS platform, developed by Pacific Bio Sciences [13], [97]. In terms of Solexa (hereafter, Illumina) and 454 Life Sciences (hereafter, Roche) [66], the use of improved NGS platforms and technologies has been studied to extract short reads with low error rates [1]. Compared to 2nd generation sequencing, 3rd generation sequencing actually produces even shorter reads [98]. However, as the read length increases, the accuracy of the analysis can decrease and costs can rise. Even recently, the 2nd and 3rd generation NGS platforms that can be used are continuing to diversify [7], and this has been accompanied by changes in sequencing technology in an effort to improve NGS platform performance [95], [96].

III. METHODS FOR SEQUENCE ALIGNMENTS

A. OPTIMAL SEQUENCE ALIGNMENT

Over the last several decades, a large number of algorithms have been proposed for both local and global sequence alignment. First, these can be divided into two categories, optimal and heuristic. Optimal sequence alignment aims to find the best alignment, while heuristic sequence alignment seeks to find near-best alignments [20]. Representative optimal sequence alignment algorithms include those proposed by Needleman-Wunsch [2] and Smith and Waterman [3] using the principles of dynamic programming [4].

The optimal global alignment algorithm of Needleman-Wunsch operates similarly to the Smith-Waterman algorithm, but when two sequences are compared, if they have the same length and high homology, the global alignment method enables very efficient alignment. First, alignment is performed along the whole sequence, from start to end, to find the best alignment. Second, Smith and Waterman proposed an optimal local alignment in 1981 [3]. Unlike global alignment, which aims to find an optimal alignment for the entire sequence, local alignment aims to find shorter but more precisely homologous regions.

B. HEURISTIC SEQUENCE ALIGNMENTS

For extremely long sequences, alignment is a non-trivial problem, because, depending on the time complexity of

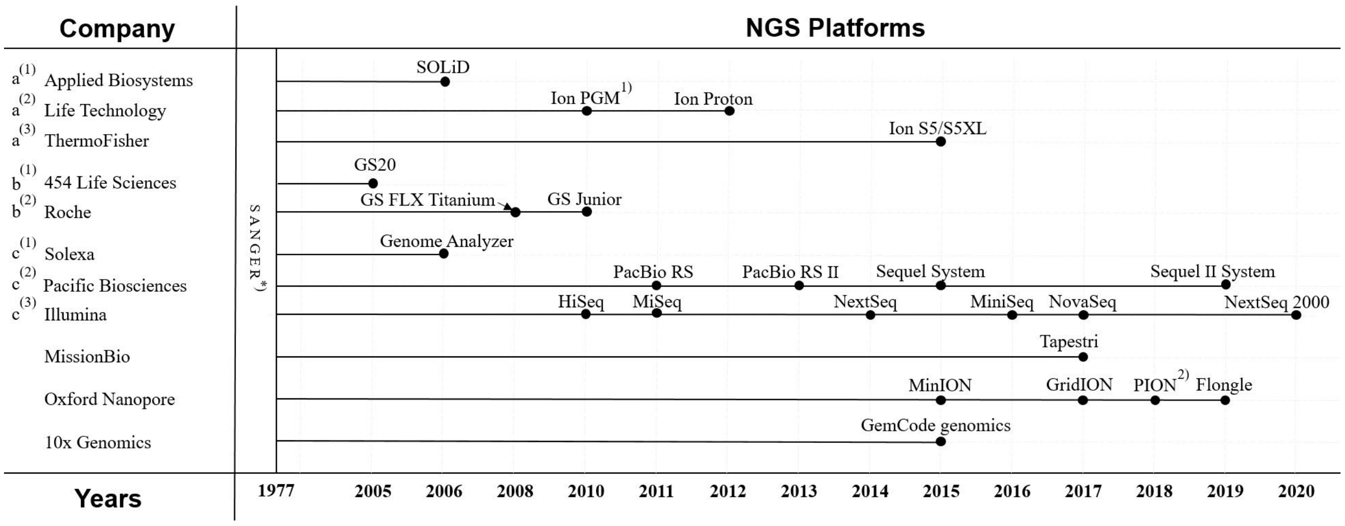

FIGURE 1. Figure 1 shows the development of NGS platforms from the appearance of NGS to recent times [13]–[18], [50]–[53]. First, NGS platforms have been classified based on their date of launch, in order to provide sales or services per year. SANGER*) indicates the development of 1st generation sequencing in 1977. Starting with 1st generation sequencing technology, the timeline shows NGS platforms and their corresponding companies arranged by year. First, the companies consist of 'a', 'b', and 'c'. 'a', 'b', and 'c' were taken over in the order of, as an example from the figure, a(1), a(2), and a(3), and have been displayed up to the current company. The orders for 'b' and 'c' follow the same principles as 'a'. Among NGS Platforms, 1) and 2) show the abbreviated names: 1) "Ion PGM" refers to the "Personal Genome Machine" and 2) "PION" refers to "PromethION" [62]–[65].

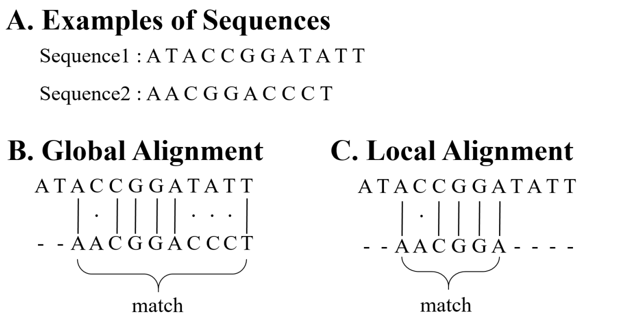

FIGURE 2. Example of global and local alignment in two sequence. In the figure, A is an example of two base sequences: Sequence 1 and Sequence 2. B shows the method for global alignment and C shows the mehod for local alignment. There are dots, long lines, and short lines. Non-matching pairs are indicated with dots '.', matching pairs are indicated with long vertical lines '|', and insertions or deletions, which show blank spaces, are indicated with short horizontal dashes '-'.

the algorithm, it requires a large number of calculations. Therefore, optimal alignment algorithms have been developed based on dynamic programming, and many researchers have attempted to produce faster algorithms. Heuristic algorithms do not always guarantee optimal results, but they produce sequence alignments that are generally optimal. These algorithms are an improvement upon optimal methods in terms of reducing the computing time required. Thereafter, algorithms have been developed using heuristic sequence alignment methods in accordance with the size of sequence and the sequencing method employed. Local sequence alignment methods include FASTA (Fast Accurate Search Tool-A) and BLAST (Basic Local Alignment Search Tool). Global sequence alignment methods include MUMmer [67] and LAGAN [68], and we provide a brief summary of MUMmer to describe global sequence alignment.

1) FASTA

FASTA [21] was the first database homology search tool, before the development of BLAST [22], and aims to find homology in DNA and protein sequences. Because of the restricted speed and memory of computers, a heuristic method was used to align the query sequence against the entire database. The approach of FASTA is as follows: first, identical fragments of length k in the two sequences are classified as strings. Here, the k-tuples classify the items required for matching. These are used to generate diagonal seeds between the two sequences. Matching parts are connected to make diagonals [69]. Second, a score is calculated using a scoring matrix, and matches with a score at least as high as a specified threshold are identified. Third, optimal local alignment is performed using the Smith-Waterman algorithm. While the speed is faster than optimal alignment algorithms, scores below the threshold are excluded from alignment, and hence the sequence alignment cannot be guaranteed to be as accurate as optimal alignment methods.

2) BLAST

Following FASTA, the Basic Local Alignment Search Tool (BLAST) program [22], [70] emerged. Compared to FASTA, BLAST enables faster alignment. BLAST is a tool that provides a rapid search function to find homologous sequences, and can be also accessed via the NCBI site [23]. This program has several options, and the program used differs depending on whether the query sequence is a nucleotide or protein sequence. From the query sequence, protein sequences of length $k = 3$ or nucleotide sequences of length $k = 11$ are combined. The combined sequences are compared with the database sequences. When database sequences are discovered that match the subword combinations, the homology search is expanded to neighboring subwords. Here, gaps are not allowed, and elongation continues until the score falls below the predefined threshold. After elongation ceases, from the database sequences, sequences are extracted with high-scoring segment pairs (HSP) that exceed the threshold [24], [25].

3) MUMMER

Maximal Unique Match-mer (MUMmer) is used to resolve the problem of BLAST alignment of very long sequences, such as whole genomes [68]. MUMmer is not used for existing DNA sequence alignment, but has been used to compare two related species of bacteria. Using an efficient suffix tree-based data structure, MUMmer is able to find exact matches within the whole genome according to the input sequence length. However, even though many bioinformatics researchers are making efforts to develop DNA analysis tools for more accurate comparison of genomes, this remains a difficult problem. Since then, several new methods have been proposed for comparison and visualization of genomes, aiming to overcome the difficulties in generating improved algorithms and visualization techniques required to compare genetic homology, including the Artemis comparison tool (ACT) [73], Mauve [74], BLAST ring image generator (BRIG) [75], and geneCo [76].

IV. SHORT READS ALIGNMENT ALGORITHMS

With the emergence of NGS technology, a large number of short reads are produced. Short read is comprised of very small units. The process of restoring the whole nucleotide sequence from these short reads can be divided into two steps. Reference sequencing is the process of mapping short reads by comparison with a reference sequence. *De novo* assembly is the process of restoring the whole sequence based on only short reads (i.e., without a reference sequence).

Reference sequencing is a mapping process in which short reads are compared with a reference sequence, to verify locations in the reference sequence corresponding to each of the short reads produced. The base sequence of interest is analyzed through this mapping process [99]. Meanwhile, the *de novo* assembly approach aims to derive information regarding a genome that has not yet been revealed. Reads are extracted by DNA sequencing, the reads are constructed into contigs, and the contigs are assembled to form scaffolds. Since there may be gaps between the contigs, scaffolding produces scaffolds that include such gaps. Finally, the scaffolds can be assembled into an entire genome sequence [26], [27].

Before performing alignments using short read, the most basic method of producing reads uses single-end sequencing. However, while single-end sequencing is relatively easy, it is difficult to find precise locations for mapping. For this reason, a method called paired-end reads or mate-paired reads has been used to produce both position and nucleotide information for the short reads [56]. In paired-end sequencing,

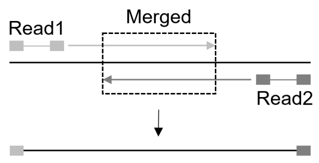

FIGURE 3. Paired-end sequencing.

the position and length of the reads can be predicted, which, compared to single-end sequencing, makes it possible to efficiently detect and amend alignment errors. In single-end sequencing, sequencing is performed from one end and reads are rapidly generated. Conversely, in paired-end sequencing, sequencing is usually performed from both ends of the DNA fragment [57]. As shown in Figure 3, When the sum of the read lengths is greater than the fragment length, there will be an overlap between the advancing ends of Read 1 (pale gray) and Read 2 (dark gray). When the overlapping parts are combined, a single, longer read can be produced. Compared to the single-end method, the paired-end approach produces a larger number of reads, and identifies the positions of the reads.

Short read alignment using reference sequencing, as depicted in Figure 4, can be divided into two methods. The first is FM-index-based algorithms [28] and the second is hashing-based algorithms [29]. Moreover, there are two types of methods to enable even faster computation speeds by improving computational performance at the hardware level. Various improved central processing unit (CPU) and graphics processing unit (GPU)-based algorithms have been developed to enable faster analysis of the alignment process.

A. HASH-BASED ALGORITHM

The hash function for each base in the sequence is indexed in a hash table that classifies keys and values according to consistent rules. Therefore, the process of using hash functions to index the hash table is referred to as hashing. The hash functions ensure that, for efficient management, data of an arbitrary length is mapped to data of a fixed length. In the hashing sequence, there will be a large number of intervals in which the same base sequence is repeated, and this can cause collisions in the hash table. Collisions restrict performance, and several methods have been proposed to resolve this problem. If there is an error or variant in one character in a read sequence, the hash value may be completely different from the appropriate value, and hash-based algorithms use various methods to account for this issue.

Figure 5 shows a general workflow for the hashing process. First, the same seed template is applied as in indexing of the reference sequence, and k-mers are generated from the query read. Fragments are obtained as continuous substrings with length 6. When the k-mers method is used, the required string can be obtained from the substrings [71]. The hash table is then used to check for congruency and incongruency with the reference sequence, as shown in processes C and D.

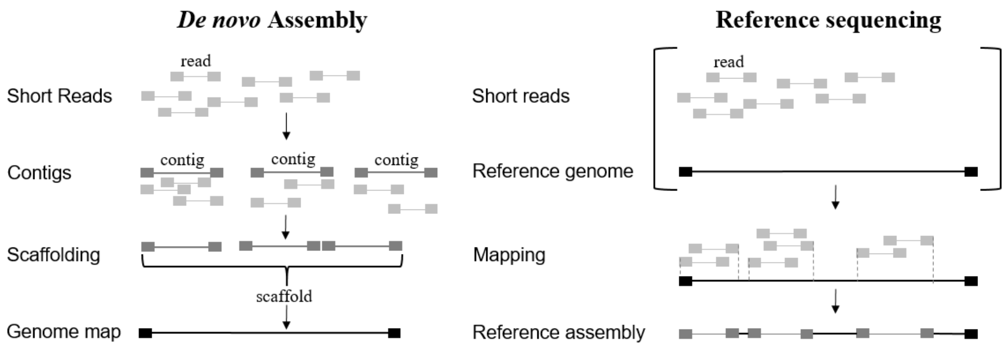

FIGURE 4. ***De novo*** **assembly and reference sequencing.**

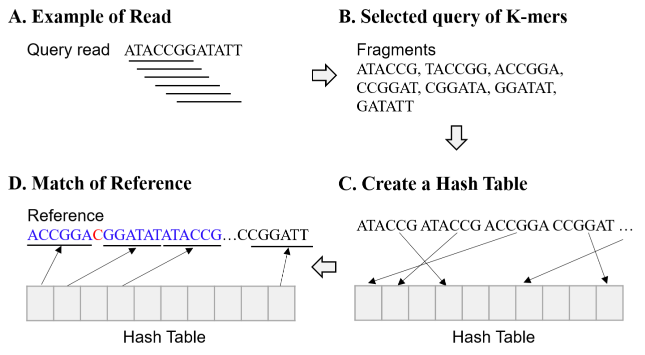
A. Example of Read
Query read ATACCGGATATT
B. Selected query of K-mers
Fragments
ATACCG, TACCGG, ACCGGA, CCGGAT, CGGATA, GGATAT, GATATT
D. Match of Reference
Reference
ACCGGACGGATATATACCG...CCGGATT
C. Create a Hash Table
ATACCG ATACCG ACCGGA CCGGAT ...
Hash Table
Hash Table

FIGURE 5. Hashing-based workflow. The hashing-based workflow proceeds in alphabetical order from Figure 5A to 5D. A is a simple example of a query read. B shows selection of k-mer sizes and fragments extracted from the query reads. The fragments are the substrings shown by the black lines below ATACCG. Processes C and D use the hash table. First, in C, the k-mer fragments extracted previously are used to construct a hash table that stores their hash values. In D, each of the fragments indexed in the hash table is compared with the reference for similarity. Continuous matching fragments are shown in color. Similar but non-continuous fragments are shown in black. A total of four fragments matched, of which 3 were continuous. A mismatch (red) is included with the three continuous matches (blue).

1) LOSSY FILTRATION

String matching problems are diverse, from the relatively simple task of searching for a single character to database searches involving complex patterns [36]. When the string is a sequence, and when trying to match a long sequence precisely, the accuracy of the aligned sequence may be reduced. However, fragments from most sequences can still be discovered. On the other hand, short reads exhibit an excessive number of hits that are not all aligned. Both accurate and inaccurate alignment are included, and to resolve this, methods such as PatternHunter [92] have been used to introduce spaced seeds, in order to match non-continuous sequences of characters. A seed template is constructed from only 1s and 0s, with 1s used for matches and 0s for mismatches. Seed template construction and seed matching reduce the number of random hits. However, they do not account for indels. Thereafter, the development of PatternHunter II [93], which uses several spaced seeds, increases the positions required for congruency. This can result in loss of sensitivity. One method to minimize the loss of sensitivity is to use more than one template, and perform matching at the required positions.

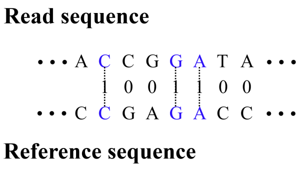

FIGURE 6. 3-mer CGA matched (1) and mismatched (0). When a read and the reference are compared, the matched pairs are highlighted in blue, with the matching of "cga" as an example of a seed template using 1 and 0.

Although there is still loss, the parts matched by the templates are identified, and mismatched parts are ignored. Expanding the range of choices is a process of loss, and spaced seeds can be used to discriminate sequences that allow errors and gaps. Several methods have been studied using seeds and filtration [37]–[39].

• BFAST (BLAT-like Fast Accurate Search Tool)

BFAST [47] was proposed for faster and more accurate sequence alignment of short reads. For this proposed method, first, indices need to be generated for the reference genome. Using an indexing method can reduce the time required for alignment. To make alignment more accurate, BFAST can use paired-end sequencing. Specifying a fixed length can improve the accuracy of alignment. Second, regarding the generation of several independent indexes, candidate alignment locations (CALs) need to be found for each read. In order to identify the CALs and the best matching reads, gaps are allowed and the best quality score can be used. Finally, alignment is performed using the Smith-Waterman algorithm for local alignment. As a simple workflow, first each index is used as a suffix array for the reference genome. A mask is applied to the suffix array, space-seeds are found, and the starting position of each seed can be determined. Afterwards, alignment of the CALs and short reads can be completed using the SW algorithm. Figure 7, below, is a simple example of applying a mask to the reference genome and obtaining space-seeds.

In addition to the short read aligners described above, there are also several hash table-based tools that have been proposed, including SeqMap [77], SHRiMP [78], ZOOM [79], MOSAIK [80], Stampy [81], FastHash [82], and SHRiMP2 [83].

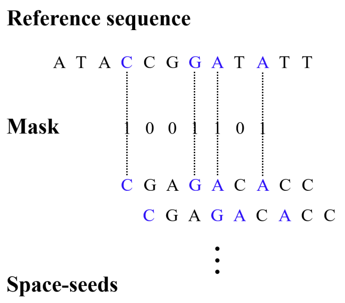

FIGURE 7. BFAST workflow. As a simple example, several masks are applied to indexed suffix arrays of the reference genome, and space-seeds are found.

2) LOSSLESS FILTRATION

In order to consider areas with true similarity matches without loss, lossless filtration algorithms need to remove the scope of potential mismatches from the range of potential similarities. Lossless filtration is referred to as lossless because, after identifying sequence fragments with potential similarities, it guarantees that only fragments that could be part of repeats are discarded [40]. Most methods depend on the pigeonhole principle, or minimum scores, for seed matching. The pigeonhole principle states that when there are n pigeons and n−1 pigeonholes, there cannot be a one-to-one matching between pigeons and pigeonholes [41]. In order to place all n pigeons into pigeonholes, at least one pigeonhole needs to contain at least two pigeons. This principle can be used in lossless filtration methods. Algorithms for the pigeonhole principle include SOAP [42], MAQ [43], and RMAP [46].

• SOAP (Short Oligonucleotide Alignment Program)

SOAP is an alignment tool that uses the concepts of seed-extend and a hash look-up table, and is capable of alignment either with or without gaps [42]. First, for the seed-extend concept, when aligning reads to a reference sequence, one continuous gap or up to two mismatches are allowed in the sequence. In terms of the hash look-up table, when seeds are generated for each read, an index table is made from the seeds, and candidate hits are searched and aligned to the reference sequence. In alignment, the best hit is considered to be the hit with the minimal number of mismatches, or with no gaps or a minimal number of gaps.

• MAQ (Mapping and Assembly with Qualities)

MAQ [43] is a sequence alignment algorithm that rapidly aligns short reads to a reference sequence, and precisely infers variants, including SNPs and indels. In order to efficiently map short reads, indices are assigned to the read sequences, the reference sequence is scanned, and extended or scored hits are identified. The best hits for possible read alignment positions are identified. The total quality score is minimized, including mismatched bases. If there are several reads in the best position, they are arbitrarily selected, and in order to improve alignment speed, only hits with 2 or fewer mismatches have to be considered.

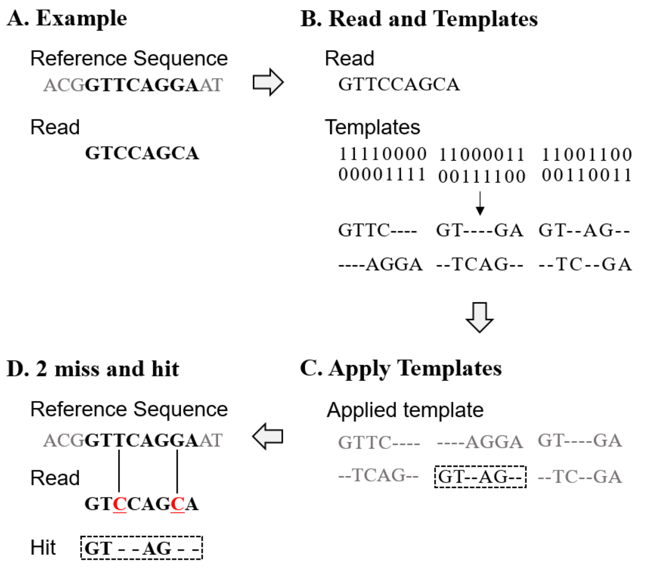

FIGURE 8. MAQ reference sequence template generation and construction of a hash table for the read sequence. This is an example of a reference sequence and a read sequence. The underlined parts of the read sequence '-' indicate the locations of mismatches due to variants or errors. The templates are constructed from the reference. Generally, given an 8-bp read, 6 templates are used. By comparing the seed templates with the reads in the hash table, the sequence can be found, including mismatches.

One disadvantage of this method is that the read sequence is hashed instead of the reference sequence. This means that whenever the read sequence is changed, a new hash table has to be constructed, and as the number of mismatches grows, the scanning process becomes longer. Referring to the example in Figure 8, first, using 6 non-adjacent templates, it is necessary to check whether they match with the 6 hash tables [43], [92]. The 6 templates used are 11110000, 00001111, 11000011, 00111100, 11001100, and 00110011, where 1s are indexed but 0s are not indexed. Using pair-end alignment of the read, MAQ stores only the two hash tables corresponding to two templates (11110000 and 00001111) in memory. When comparing the read sequence to several templates, each process is separate, and so after storing the read hashing results for the first template, hashing is performed for the next template and those results are stored. From the 6 templates, 6 hash values can be extracted, and the read sequence, after applying the templates, can be compared with the reference sequence.

• RMAP

The original RMAP [45] used a filtration method proposed by Baeza-Yates and Perleberg [54]. In this filtration strategy, when mapping is performed with reads of length n, allowing for k mismatches, each read is divided into k + 1 adjacent seeds. Quality score information is used, which can inform important decisions about the length and position of reads. Subsequently, RMAP [46] was adapted to use paired-end methods, quality scores could be used more precisely to improve accuracy, important positions could be identified, and wildcard matching methods could also be employed.

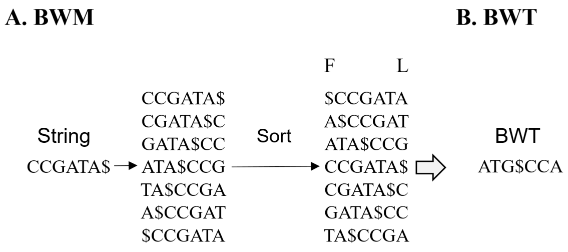

FIGURE 9. BWT construction process. The character $ indicates the end of the sequence, and sequence in string form is termed 'S'. S is defined to be "CCGATA". In order to construct the BWT, three processes are required. When proceeding in order from A to B in Figure 9, first, in A, the process of generating a suffix array from the original sequence, 'S' is achieved by sequential rotations of S, one character at a time. Second, the array is sorted to place the suffixes in alphabetical order (when "CC" and "CG" are arranged in alphabetical order, "CC" is first.) Third, in Figure 9 B, from the BWM on the left in A, using the LF method, the first and last columns are stored. Finally, the column corresponding to L is named the BWT.

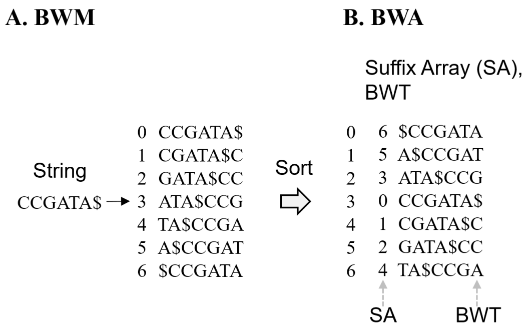

FIGURE 10. BWA suffix array construction and example.

Wildcards induce low quality locations, defined by the threshold value, to always guarantee matching. Mismatches are only calculated in high quality locations. First, a method with a seed structure is used to place the reads in positions of exact matches with the reference sequence, which represent all of the locations that can be mapped. Then, the algorithm allows precise mapping of even the parts that have not been mapped.

B. FM-INDEX-BASED ALGORITHM

An FM-index [48] is an algorithm based on the Burrows-Wheeler Matrix (BWM) and Last-First (LF) [30], and uses LF to enable backward searching and backtracking. The basic algorithm using BWM and LF is called the Burrows-Wheeler Transform (BWT). The BWT consists of a tree structure including all suffixes and prefixes in the string: although the structure allows very rapid calculations, it has a drawback in that, as the string becomes longer, the tree size also grows massively. Among efforts to overcome these difficulties, the suffix array of the BWA [32] has been converted to a data structure that stores the starting positions of suffixes, which are arranged alphabetically, reducing the storage space required compared to a suffix tree. String BWT was originally developed for data compression.

1) BWT (BURROW-WHEELER TRANSFORM)

For the composition of the BWT, when given data in a string format, the string is converted into a matrix, and the matrix is then sorted into alphabetical order. The last row of BWM after alignment is completed is called the BWT. This series of processes uses data in string form, and a simple example is shown in Figure 9.

• SOAP2

In SOAP2 [31], instead of a seed algorithm to index the reference sequence in the main memory, as in the original version of SOAP [42], the same BWT compression index is used as shown in Figure 10 [31]. SOAP2 can be used with both paired-end and single-end sequencing methods, and, like SOAP, prioritizes finding the best hit for reads. First, split-read alignment is used to handle mismatches and indels. The split-read method allows 1 mismatch when there are no matches, and searches for hits by splitting the read into 2 fragments before attempting mapping. If no hits are found, by the same method, 2 mismatches are allowed and the read is split into 3 fragments before attempting mapping. If no hits are found to the end, the Smith-Waterman algorithm is used for alignment. During alignment, one continuous gap or several mismatches (as many as 4) are allowed, but for optimal performance, allowing 2 mismatches provides suitable alignment. In addition, memory usage is reduced and the alignment speed is greatly improved.

2) BACKTRACKING

Backtracking refers to tracing back over part of the sequence that has already been aligned. A matrix is generated corresponding to the reference and query sequences, and the matrix is populated with scores. Scores are assigned according to whether or not there is a match, and scores are also given for gaps. Once the whole matrix has been filled, the highest scores are identified and backtracking is performed. Several aligners have been developed, such as Bowtie and BWA, which use backtracking to map read [94].

• BWA (Burrow-Wheeler Aligner)

BWA is an algorithm that uses a suffix array (SA) to perform alignment. In the process of generating the SA, in order to define new indexes through BWM alignment, indexes are assigned again according to their order. The altered SA index makes it possible to identify positions within the data in string form. BWA generates a BWM from the start, and stores the positions of the first and last columns; the starting positions are stored in the suffix array, and the end positions are stored in the BWT. Backtracking can be performed with methods such as Bowtie. An example of executing BWA is shown in Figure 10. First, data is used in the form a string stored in a BWM. If the string is named 'S', then S = "CCGATA". When trying to find "CC", displayed in bold in S, the BWT is ATG$CCA, and C is at indexes 3 and 4. In the SA, the 3rd and 4th values are 0 and 1, meaning that the string of interest, S, appears at the 0th or 1st position.

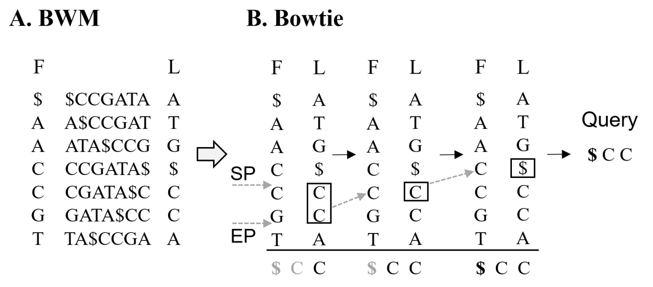

FIGURE 11. The Bowtie matching process using a BWM. The LF mapping method differentiates between L (Last) and F (First), and the L and F data stored in the BWM are used. The Bowtie matching process in Figure 11 involves two tasks. In Figure 11 A, the construction of BWM from Figure 9 is used. First, in order to find the example query ("$CC"), the SP (Start Point) and EP (End Point) columns are used. The SP and EP columns determine the part pertaining to the query suffix. Second, using the LF mapping method, the characters in L are checked to verify whether these are the characters located after the query. This series of processes is repeated. This backtracking procedure is used to find whether the query is present.

• Bowtie

Bowtie [33] rapidly aligns short reads to a reference sequence and efficiently stores the results in memory. Bowtie is an alignment algorithm that uses the BWM and LF mapping methods, and performs backtracking using the greedy algorithm to cope with inexact or uncertain matches. The following behaviors are possible for the query string. In the event of an inexact match, even if there is a problem at the time when the match is judged to be optimal, this is defined as an optimal but not a perfect match. If it is difficult to conclude that the overall alignment is optimal - an optimal solution can be obtained through partial optimal matching, but Bowtie does not guarantee finding the ideal alignment in these cases.

In addition to the sequence aligners described above, there are several other FM-index-based tools that have been proposed, such as GEM [84], BLASR [85], and BWA-SW [86].

C. THE SHIFT FROM CPU TO GPU ALIGNMENT

To rapidly analyze large numbers of short reads, generally, CPU methods have been used, with single or multiple processors. When short read is analyzed, the process that typically takes the longest time is short read alignment. Therefore, in order to analyze a large amount of short read loaded into memory, cost and speed need to be improved, and as the size of short read datasets has gradually grown, efficient, expandable short read aligners have increasingly become necessary [49]. One method that has been proposed to overcome these issues is to use a GPU for parallel processing of large-scale, short read [96], and recently CUDA, which uses a parallel programming language, can be used efficiently for parallel processing of large short read datasets [72]. GPU-based aligners follow the principles of the algorithms introduced in Sections 4.3 and 4.4. GPU algorithms can be classified into hashing-based GPU algorithms, and BWT-based GPU algorithms; here, we briefly introduce the upgraded GPU versions of the RMAP and SOAP algorithms, described above for CPU.

The hashing-based GPU aligner GPU-RMAP [35] is a version of the RMAP algorithm optimized for NVIDIA GPU products that is capable of parallel processing. First, hash tables for the whole genome and reads are sent from the CPU memory to the GPU memory. In the GPU, the genome is divided into several independent segments. The divided segments are scanned in parallel, and a score is assigned to the mapping locations of all of the reads. In the second stage, the reads are spread between GPU threads to select the positions with the highest scores. Finally, with regard to the hash tables sequentially constructed in RMAP, as the number of reads increases, the key-value pairs to be compared are loaded into the GPU memory, and the computational speed of searching may decrease. Here, if the tables were in the form of binary search trees, it could be an improvement upon conventional hashing methods. In binary search trees, the upper few levels of the tree, which are accessed frequently, are allocated to the faster cache memory to optimize performance. The BWT-based GPU algorithm SOAP3 Aligner is the GPU version of the BWT-based SOAP2 Aligner. SOAP3 Aligner allows reads to be aligned to a reference sequence with up to 4 mismatches.

The hashing-based GPU aligner GPU-RMAP [35] is a version of the RMAP algorithm optimized for NVIDIA GPU products that is capable of parallel processing. First, hash tables for the whole genome and reads are sent from the CPU memory to the GPU memory. In the GPU, the genome is divided into several independent segments. The divided segments are scanned in parallel, and a score is assigned to the mapping locations of all of the reads. In the second stage, the reads are spread between GPU threads to select the positions with the highest scores. Finally, with regard to the hash tables sequentially constructed in RMAP, as the number of reads increases, the key-value pairs to be compared are loaded into the GPU memory, and the computational speed of searching may decrease. Here, if the tables were in the form of binary search trees, it could be an improvement upon conventional hashing methods. In binary search trees, the upper few levels of the tree, which are accessed frequently, are allocated to the faster cache memory to optimize performance. The BWT-based GPU algorithm SOAP3 Aligner is the GPU version of the BWT-based SOAP2 Aligner. SOAP3 Aligner allows reads to be aligned to a reference sequence with up to 4 mismatches.

GPU-based aligners improve the speed of alignment for short read, and the process is expandable by using multiple GPUs instead of a single GPU. Other GPU-based aligners that have been proposed include CUSHAW [87], CUSHAW2-GPU [88], and SARUMAN [89].

V. SHORT READ ALGORITHM CHARACTERISTICS AND ANALYSIS OF STRATEGIES

A comparative analysis of the performance of the latest updated and existing algorithms was performed. Each short-read alignment algorithm was comparatively analyzed using the data provided by NCBI. Overall, the CPU-based

TABLE 1. Table 1 presents the reference sequence and SRA provided by NCBI to perform experiment under the same conditions as in the aligner. For a fair test at Intel(R) Xeon(R) CPU E5-2695 v4 @ 2.10 GHz, all the servers were run on a single core. All aligner versions used are the latest versions. There are three classification methods: 0,1 for the number of mismatches, and the default option for not selecting the number of mismatches.

Aligners	Version	Default		Zero mismatch		One mismatch	
		^{a}RA (%)	^{b}Time (h:m:s)	^{a}RA (%)	^{b}Time (h:m:s)	^{a}RA (%)	^{b}Time (h:m:s)
SOAP-v2	2.21	96.85	00:02:13	89.80	00:01:57	06.38	00:03:29
BWA	0.7.17-r1188	99.53	00:09:43	99.56	00:09:38	99.55	00:09:49
Bowtie	1.3.0	98.05	00:03:34	96.05	00:03:37	97.90	00:03:23
MAQ	0.7.1	96.94	00:06:47	89.61	00:06:44	89.61	00:06:47
RMAP	2.1	98.82	00:32:00	98.81	00:32:15	98.81	00:31:58
BFAST	0.7.0	99.09	00:37:27	18.42	00:24:56	98.64	00:31:42

RA: Read Aligned
Time: Total Elapsed Time

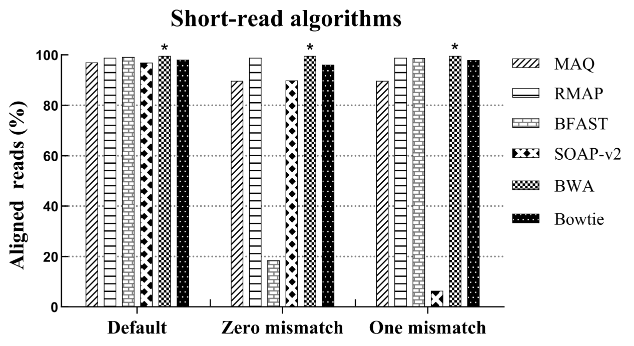

FIGURE 12. Table 1 shows the graph of the performance comparative analysis. Among the algorithms based on FM-index and hash, the algorithms with the most sorted reads are marked with the symbol '*'.

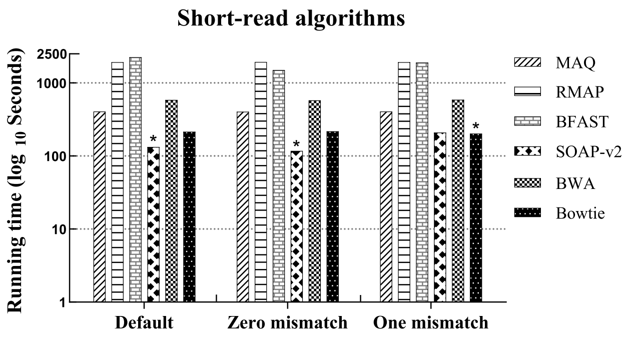

FIGURE 13. Table 1 shows the graph of the performance comparative analysis. Among the algorithms based on FM-index and hash, the algorithms with the least execution time are marked with the symbol '*'.

algorithms were compared. GPU-RMAP and SOAP3, which were updated to the GPU method in the existing CPU-based algorithms, were added to the Table 2. However, there were few objects available for comparison; hence, the analysis was performed based on CPU-based execution time and aligned reads.

For comparative analysis, each aligner used the same experimental data and server. Reference sequence NZ_CP05032 and Sequence Read Archive (SRA) SRR11461738 of Illumina Hiseq 4000 were used for analysis. All aligners, including MAQ and RMAP, were tested with a single core for a fair performance comparison. During the analysis, RMAP should only have one read line in the SRA file and the number of all reads must be equivalent to perform sorting. Therefore, upon finding the read with the maximum size, all the reads were filled with n bases equivalent to the maximum size, following which the experiment was performed. The result was remeasured for accuracy by arranging the start and end positions while excluding n bases. Figure 12 shows the alignment accuracy of the leads. Figure 13 shows the performance speed depending on the number of aligned leads.

TABLE 2. Overall details of short read alignment algorithms.

Category	Methods	Algorithms	Year	Hardware	Reference
Hash-based Algorithms	lossy	BFAST	2009	CPU	[47]
	lossless	MAQ	2008	CPU	[43]
		SOAP	2008	CPU	[42]
		RMAP	2008	CPU	[46]
	hashing	GPU-RMAP	2010	GPU	[35]
FM-index -based Algorithms	backtracking	BWA	2009	CPU	[32]
		Bowtie	2009	CPU	[33]
	BWT	SOAP2	2009	CPU	[31]
		SOAP3	2012	GPU	[34]

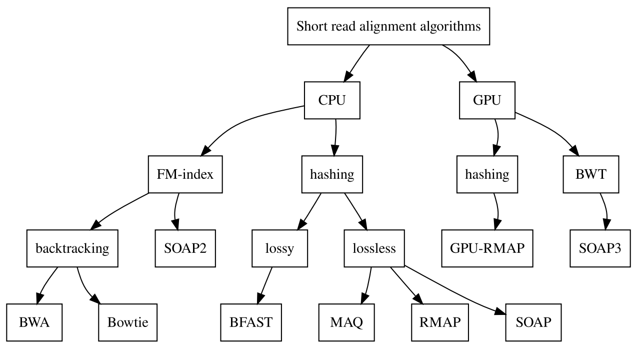

FIGURE 14. Short read alignment algorithm: hierarchical structure.

FM-index-based algorithms are generally faster than hash-based algorithms, and the number of aligned reads is also higher. Figures 12 & 13 and Table 1 demonstrate that the selection of an algorithm can vary depending on the speed and accuracy. To select an efficient short read alignment algorithm, it is necessary to confirm the execution time and read alignment performance, and establish an algorithm use plan according to the user's experiment.

VI. CONCLUSION

With the emergence of NGS technology, there have been rapid advances in read alignment algorithms using read. Unlike previous sequencing technologies, NGS enables high-speed analysis of DNA sequences. A large amount of NGS data has been extracted in the form of short reads, and short reads are used to obtain desired information by comparison with reference sequences through mapping [6]. Several algorithms using mapping methods have made important contributions to DNA analysis, even recently. However, since the algorithms used differ depending on the specific analytical methods and objectives involved, there are almost no generally-applicable algorithms. For this reason, the method for choosing the best short read alignment algorithm is to select the best algorithm in each specific circumstance. Short read alignment algorithms are still being actively proposed in order to develop new methods and improve existing methods.

When the short reads extracted by NGS platforms are mapped to reference sequences, the important aspects are high computing speeds and alignment accuracy. It is essential to select mapping algorithms that can perform optimal alignments according to the specific conditions. In this review paper, algorithm analysis has been divided into methods using hash tables, and methods using FM-indexing. The content regarding the algorithms in Section 4 has been arranged in the form of a hierarchical structure, as shown in Figure 14. It presents a workflow that concisely classifies the content of Section 4. In Section 5, to compare the performance of hash-based and FM-index-based algorithms, accuracy and execution time can be confirmed through default options and 0,1 mismatched experiment. As the user's choice of algorithm ultimately affects the speed and accuracy of the experiment, it is necessary to select an efficient algorithm by establishing an algorithm use plan suitable for the user's experiment. It is anticipated that Table 1 and Figures 12 & 13 will help in selection of an algorithm for more sorted reads and shorter execution time.

Recently, there have also been developments involving mapping algorithms using short reads, and enhancements to NGS platforms. Currently, well-known companies that can market NGS platforms generating short reads include Illumina, Oxford Nanopore, PacBio, and 10x Genomics. Since the Sanger DNA sequencing method was first published, there have been efforts to further advance sequencing technology and data analysis.

This study reviewed the types and functions of sequence alignment algorithms using short reads. Recently, computational methods have been shifting from CPU-based hardware to GPU-based platforms, but most algorithms handling bioinformatics data with high I/O still operate on CPU-based architectures. In the future, a review of GPU-based computational methods may also be very helpful.

REFERENCES

[1] X. G. Zhou, L. F. Ren, Y. T. Li, M. Zhang, Y. D. Yu, and J. Yu, "The next-generation sequencing technology: A technology review and future perspective," *Sci China Life Sci*, vol. 53, no. 1, pp. 45–57, 2010.

[2] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the aminoacid sequence of two proteins," *J. Mol. Biol.*, vol. 48, no. 3, pp. 443–453, 1970.

[3] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequence," *J. Mol. Biol.*, vol. 50, no. 1, pp. 195–197, 1981.

[4] S. Dreyfus, "Richard bellman on the birth of dynamic programming," *Oper. Res.*, vol. 50, no. 1, pp. 48–51, Feb. 2002.

[5] W. J. Wilbur and D. J. Lipman, "Rapid similarity searches of nucleic acid and protein data banks," *Proc. Nat. Acad. Sci. USA*, vol. 80, no. 3, pp. 726–730, Feb. 1983.

[6] J. R. Miller, S. Koren, and G. Sutton, "Assembly algorithms for next-generation sequencing data," *Genomics*, vol. 95, no. 6, pp. 315–327, Jun. 2010.

[7] T. C. Glenn, "Field guide to next-generation DNA sequencers," *Mol. Ecol. Resour.*, vol. 11, no. 5, pp. 759–769, Sep. 2011.

[8] D. R. Bentley, "Whole-genome re-sequencing," *Current Opinion Genet. Develop.*, vol. 16, no. 6, pp. 545–552, Dec. 2006.

[9] N. A. Fonseca, J. Rung, A. Brazma, and J. C. Marioni, "Tools for mapping high-throughput sequencing data," *Bioinformatics*, vol. 28, no. 24, pp. 3169–3177, Dec. 2012.

[10] *Applied Biosystems*. Accessed: May 1, 2020. [Online]. Available: https://www.thermofisher.com/kr/ko/home/brands/applied-biosystems.html

[11] *Hisroty of Sequenceing by Synthesis*. Accessed: May 1, 2020. [Online]. Available: https://www.illumina.com/science/technology/next-generation-sequencing/illumina-sequencing-history.html

[12] *Life Technologies*. Accessed: May 1, 2020. [Online]. Available: https://www.thermofisher.com/kr/ko/home/brands/life-technologies.html

[13] *Latest System Release*. Accessed: May 1, 2020. [Online]. Available: https://www.pacb.com/products-and-services/sequel-system/latest-system-release/

[14] *Company History*. Accessed: May 1, 2020. [Online]. Available: https://nanoporetech.com/about-us/history

[15] S. Budel. *10X Genomics Launches GemCode*. Accessed: May 1, 2020. [Online]. Available: https://www.decibio.com/2015/02/25/10x-genomics-launches-gemcode/

[16] *Experience NextSeq 2000*. Accessed: May 1, 2020. [Online]. Available: https://www.illumina.com/systems/sequencing-platforms/nextseq-1000-2000.html

[17] D. O'Neill. *The Ion Torrent S5 and S5 XL Next Generation Sequencing Systems*. Accessed: May 1, 2020. [Online]. Available: https://www.biosciences.ie/ion-torrent-s5-and-s5xl

[18] *Overview*. Accessed: May 1, 2020. [Online]. Available: http://www.dnalink.com/ngs-overview.html

[19] S. Yohe and B. Thyagarajan, "Review of clinical next-generation sequencing," *Arch. Pathol. Lab Med.*, vol. 141, no. 11, pp. 1544–1557, 2017.

[20] W. Haque, A. Aravind, and B. Reddy, "Pairwise sequence alignment algorithms: A survey," in *Proc. ISTA*, 2009, pp. 96–103.

[21] W. R. Pearson and D. J. Lipman, "Improved tools for biological sequence comparison," *Proc. Nat. Acad. Sci. USA*, vol. 85, no. 8, pp. 2444–2448, Apr. 1988.

[22] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *J. Mol. Biol.*, vol. 215, no. 3, pp. 403–410, Oct. 1990.

[23] M. Johnson, I. Zaretskaya, Y. Raytselis, Y. Merezhuk, S. McGinnis, and T. L. Madden, "NCBI BLAST: A better Web interface," *Nucleic Acids Res.*, vol. 36, no. 2, pp. 5–9, 2008.

[24] S. Karlin and S. F. Altschul, "Applications and statistics for multiple high-scoring segments in molecular sequences," *Proc. Nat. Acad. Sci. USA*, vol. 90, no. 12, pp. 5873–5877, Jun. 1993.

[25] S. F. Altschul, "Evaluating the statistical significance of multiple distinct local alignments," in *Theoretical and Computational Methods in Genome Research*, S. Suhai, Ed. Boston, MA, USA: Springer, 1997, pp. 1–14.

[26] K. Paszkiewicz and D. J. Studholme, "De novo assembly of short sequence reads," *Briefings Bioinf.*, vol. 11, no. 5, pp. 457–472, 2010.

[27] R. Li, H. Zhu, J. Ruan, W. Qian, X. Fang, Z. Shi, Y. Li, S. Li, G. Shan, K. Kristiansen, S. Li, H. Yang, J. Wang1, and J. Wang, "De novo assembly of human genomes with massively parallel short read sequencing," *Genome Res.*, vol. 20, no. 2, pp. 265–272, 2010.

[28] P. Ferragina and G. Manzini, "Opportunistic data structures with applications," in *Proc. 41st Annu. Symp. Found. Comput. Sci.*, 1998, pp. 390–398.

[29] L. Chi and X. Zhu, "Hashing techniques: A survey and taxonomy," *ACM Comput. Surv.*, vol. 50, no. 1, pp. 1–36, Apr. 2017.

[30] M. Burrows and D. J. Wheeler, "A block-sorting lossless data compression algorithm," Digit. Equip. Corp., Maynard, MA, USA, Tech. Rep. 124, 1994.

[31] R. Li, C. Yu, Y. Li, T.-W. Lam, S.-M. Yiu, K. Kristiansen, and J. Wang, "SOAP2: An improved ultrafast tool for short read alignment," *Bioinformatics*, vol. 25, no. 15, pp. 1966–1967, Aug. 2009.

[32] H. Li and R. Durbin, "Fast and accurate short read alignment with burrows-wheeler transform," *Bioinformatics*, vol. 25, no. 14, pp. 1754–1760, Jul. 2009.

[33] B. Langmead, C. Trapnell, M. Pop, and S. L. Salzberg, "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome," *Genome Biol.*, vol. 10, no. 3, p. R25, 2009.

[34] C.-M. Liu, T. Wong, E. Wu, R. Luo, S.-M. Yiu, Y. Li, B. Wang, C. Yu, X. Chu, K. Zhao, R. Li, and T.-W. Lam, "SOAP3: Ultra-fast GPU-based parallel alignment tool for short reads," *Bioinformatics*, vol. 28, no. 6, pp. 878–879, Mar. 2012.

[35] A. M. Aji, L. Zhang, and W.-C. Feng, "GPU-RMAP: Accelerating short-read mapping on graphics processors," in *Proc. 13th IEEE Int. Conf. Comput. Sci. Eng.*, Dec. 2010, pp. 168–175.

[36] G. Navarro and M. Raffinot, *Flexible Pattern Matching in Strings: Practical On-Line Search Algorithms for Texts and Biological Sequences*. Cambridge, U.K.: Cambridge Univ. Press, 2002.

[37] G. Kucherov, L. Noe, and M. Roytberg, "Multiseed lossless filtration," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 2, no. 1, pp. 51–61, Jan. 2005.

[38] S. Burkhardt and J. Karkkainen, "Better filtering with gapped q-grams," in *Combinatorial Pattern Matching* (Lecture Notes in Computer Science), vol. 2089, A. Amir, Ed. 2001, pp. 73–85.

[39] P. A. Pevzner and M. S. Waterman, "Multiple filtration and approximate pattern matching," *Algorithmica*, vol. 13, nos. 1–2, pp. 135–154, Feb. 1995.

[40] P. Peterlongo, G. T. Sacomoto, A. P. D. Lago, N. Pisanti, and M. F. Sagot, "Lossless filter for multiple repeats with bounded edit distance," *Algorithms Mol. Biol.*, vol. 4, no. 3, pp. 1–20, 2009.

[41] M. Ajtai, "The complexity of the pigeonhole principle," *Combinatorica*, vol. 14, no. 4, pp. 417–433, Dec. 1994.

[42] R. Li, Y. Li, K. Kristiansen, and J. Wang, "SOAP: Short oligonucleotide alignment program," *Bioinformatics*, vol. 24, no. 5, pp. 713–714, Mar. 2008.

[43] H. Li, J. Ruan, and R. Durbin, "Mapping short DNA sequencing reads and calling variants using mapping quality scores," *Genome Res.*, vol. 18, no. 11, pp. 1851–1858, Nov. 2008.

[44] X. Gao and P. Dai, "Impact of next-generation sequencing on molecular diagnosis of inherited non-syndromic hearing loss," *J. Otol.*, vol. 9, no. 3, pp. 122–125, Sep. 2014.

[45] A. D. Smith, Z. Xuan, and M. Q. Zhang, "Using quality scores and longer reads improves accuracy of solexa read mapping," *BMC Bioinf.*, vol. 9, no. 1, p. 128, 2008.

[46] A. D. Smith, W.-Y. Chung, E. Hodges, J. Kendall, G. Hannon, J. Hicks, Z. Xuan, and M. Q. Zhang, "Updates to the RMAP short-read mapping software," *Bioinformatics*, vol. 25, no. 21, pp. 2841–2842, Nov. 2009.

[47] N. Homer, B. Merriman, and S. F. Nelson, "BFAST: An alignment tool for large scale genome resequencing," *PLoS ONE*, vol. 4, no. 11, p. e7767, Nov. 2009.

[48] P. Ferragina and G. Manzini, "Indexing compressed text," *J. ACM*, vol. 52, no. 4, pp. 552–581, Jul. 2005.

[49] R. Wilton, T. Budavari, B. Langmead, S. J. Wheelan, S. L. Salzberg, and A. S. Szalay, "Arioc: high-throughput read alignment with GPU-accelerated exploration of the seed-and-extend search space," *PeerJ*, vol. 3, p. e808, Mar. 2015.

[50] *SmidgION*. Accessed: May 1, 2020. [Online]. Available: https://nanoporetech.com/products/smidgion

[51] *Ion GeneStudio S5 Series*. Accessed: May 1, 2020. [Online]. Available: https://www.thermofisher.com/kr/ko/home/life-science/sequencing/next-generation-sequencing/ion-torrent-next-generation-sequencing-workflow/ion-torrent-next-generation-sequencing-run-sequence/ion-s5-ngs-targeted-sequencing.html

[52] *The Next Era in Sequencing Starts Now*. Accessed: May 1, 2020. [Online]. Available: https://www.illumina.com/systems/sequencing-platforms/novaseq.html

[53] *Sequel Ii System*. Accessed: May 1, 2020. [Online]. Available: https://nanoporetech.com/about-us/history

[54] R. A. Baeza-Yates and C. H. Perleberg, "Fast and practical approximate string matching," *Inf. Process. Lett.*, vol. 59, no. 1, pp. 21–27, Jul. 1996.

[55] M. Smolka, P. Rescheneder, M. C. Schatz, A. von Haeseler, and F. J. Sedlazeck, "Teaser: Individualized benchmarking and optimization of read mapping results for NGS data," *Genome Biol.*, vol. 16, no. 1, p. 235, Dec. 2015.

[56] K. F. Au, H. Jiang, L. Lin, Y. Xing, and W. H. Wong, "Detection of splice junctions from paired-end RNA-seq data by SpliceMap," *Nucleic Acids Res.*, vol. 38, no. 14, pp. 4570–4578, Aug. 2010.

[57] H. Li and N. Homer, "A survey of sequence alignment algorithms for next-generation sequencing," *Briefings Bioinf.*, vol. 11, no. 5, pp. 473–483, Sep. 2010.

[58] S. C. Schuster, "Next-generation sequencing transforms today's biology," *Nature Methods*, vol. 5, no. 1, pp. 16–18, Jan. 2008.

[59] J.-S. Lim, B.-S. Choi, J.-S. Lee, C. Shin, T.-J. Yang, J.-S. Rhee, J.-S. Lee, and I.-Y. Choi, "Survey of the applications of NGS to whole-genome sequencing and expression profiling science," *Genomics Inf.*, vol. 10, no. 1, pp. 1–8, 2012.

[60] E. R. Mardis, "Next-generation DNA sequencing method," *Annu. Rev. Genomics Hum. Genet.*, vol. 9, no. 1, pp. 387–402, 2008.

[61] M. Mielczarek and J. Szyda, "Review of alignment and SNP calling algorithms for next-generation sequencing data," *J. Appl. Genet.*, vol. 57, no. 1, pp. 71–79, Feb. 2016.

[62] *Ion Personal Genome Machine*. Accessed: May 1, 2020. [Online]. Available: https://www.thermofisher.com/order/catalog/product/4462921

[63] *PromethION*. Accessed: May 1, 2020. [Online]. Available: https://nanoporetech.com/products/promethion

[64] *Products*. Accessed: May 1, 2020. [Online]. Available: https://www.10xgenomics.com/

[65] *Mission Bio*. Accessed: May 1, 2020. [Online]. Available: https://missionbio.com/

[66] *Roche Milestones*. Accessed: May 1, 2020. [Online]. Available: https://www.roche.com/about/history.htm

[67] A. L. Delcher, S. Kasif, R. D. Fleischmann, J. Peterson, O. White, and S. L. Salzberg, "Alignment of whole genomes," *Nucleic Acids Res.*, vol. 27, no. 11, pp. 2369–2376, Jan. 1999.

[68] M. Brudno, "LAGAN and multi-LAGAN: Efficient tools for large-scale multiple alignment of genomic DNA," *Genome Res.*, vol. 13, no. 4, pp. 721–731, Apr. 2003.

[69] D. Lipman and W. Pearson, "Rapid and sensitive protein similarity searches," *Science*, vol. 227, no. 4693, pp. 1435–1441, Mar. 1985.

[70] S. Altschul, "Gapped BLAST and PSI-BLAST: A new generation of protein database search programs," *Nucleic Acids Res.*, vol. 25, no. 17, pp. 3389–3402, Sep. 1997.

[71] Y. Liao, G. K. Smyth, and W. Shi, "The subread aligner: Fast, accurate and scalable read mapping by seed-and-vote," *Nucleic Acids Res.*, vol. 41, no. 10, p. e108, May 2013.

[72] M. Alser, H. Hassan, H. Xin, O. Ergin, O. Mutlu, and C. Alkan, "Gate-Keeper: Enabling fast pre-alignment in DNA short read mapping with a new streaming accelerator architecture," *Bioinformatics*, vol. 33, no. 21, pp. 3355–3363, 2017.

[73] T. J. Carver, K. M. Rutherford, M. Berriman, M.-A. Rajandream, B. G. Barrell, and J. Parkhill, "ACT: The artemis comparison tool," *Bioinformatics*, vol. 21, no. 16, pp. 3422–3423, Aug. 2005.

[74] A. C. E. Darling, "Mauve: Multiple alignment of conserved genomic sequence with rearrangements," *Genome Res.*, vol. 14, no. 7, pp. 1394–1403, Jun. 2004.

[75] N.-F. Alikhan, N. K. Petty, N. L. Ben Zakour, and S. A. Beatson, "BLAST ring image generator (BRIG): Simple prokaryote genome comparisons," *BMC Genomics*, vol. 12, no. 1, p. 402, Dec. 2011.

[76] J. Jung, J. I. Kim, and G. Yi, "GeneCo: A visualized comparative genomic method to analyze multiple genome structures," *Bioinformatics*, vol. 35, no. 24, pp. 5303–5305, Dec. 2019.

[77] H. Jiang and W. H. Wong, "SeqMap: Mapping massive amount of oligonucleotides to the genome," *Bioinformatics*, vol. 24, no. 20, pp. 2395–2396, Oct. 2008.

[78] S. M. Rumble, P. Lacroute, A. V. Dalca, M. Fiume, A. Sidow, and M. Brudno, "SHRiMP: Accurate mapping of short color-space reads," *PLoS Comput. Biol.*, vol. 5, no. 5, May 2009, Art. no. e1000386.

[79] H. Lin, Z. Zhang, M. Q. Zhang, B. Ma, and M. Li, "ZOOM! Zillions of oligos mapped," *Bioinformatics*, vol. 24, no. 21, pp. 2431–2437, Nov. 2008.

[80] W.-P. Lee, M. P. Stromberg, A. Ward, C. Stewart, E. P. Garrison, and G. T. Marth, "MOSAIK: A hash-based algorithm for accurate next-generation sequencing short-read mapping," *PLoS ONE*, vol. 9, no. 3, Mar. 2014, Art. no. e90581.

[81] G. Lunter and M. Goodson, "Stampy: A statistical algorithm for sensitive and fast mapping of illumina sequence reads," *Genome Res.*, vol. 21, no. 6, pp. 936–939, Jun. 2011.

[82] H. Xin, D. Lee, F. Hormozdiari, S. Yedkar, O. Mutlu, and C. Alkan, "Accelerating read mapping with FastHASH," *BMC Genomics*, vol. 14, no. S1, p. S13, Jan. 2013.

[83] M. David, M. Dzamba, D. Lister, L. Ilie, and M. Brudno, "SHRiMP2: Sensitive yet practical short read mapping," *Bioinformatics*, vol. 27, no. 7, pp. 1011–1012, Apr. 2011.

[84] S. Marco-Sola, M. Sammeth, R. Guigó, and P. Ribeca, "The GEM mapper: Fast, accurate and versatile alignment by filtration," *Nature Methods*, vol. 9, no. 12, pp. 1185–1188, Dec. 2012.

[85] M. J. Chaisson and G. Tesler, "Mapping single molecule sequencing reads using basic local alignment with successive refinement (BLASR): Application and theory," *BMC Bioinf.*, vol. 13, no. 1, p. 238, Dec. 2012.

[86] H. Li and R. Durbin, "Fast and accurate long-read alignment with burrows–wheeler transform," *Bioinformatics*, vol. 26, no. 5, pp. 589–595, Mar. 2010.

[87] Y. Liu, B. Schmidt, and D. L. Maskell, "CUSHAW: A CUDA compatible short read aligner to large genomes based on the burrows-wheeler transform," *Bioinformatics*, vol. 28, no. 14, pp. 1830–1837, 2012.

[88] Y. Liu and B. Schmidt, "CUSHAW2-GPU: Empowering faster gapped short-read alignment using GPU computing," *IEEE Des. Test. IEEE Des. Test. Comput.*, vol. 31, no. 1, pp. 31–39, Feb. 2014.

[89] J. Blom, T. Jakobi, D. Doppmeier, S. Jaenicke, J. Kalinowski, J. Stoye, and A. Goesmann, "Exact and complete short-read alignment to microbial genomes using graphics processing unit programming," *Bioinformatics*, vol. 27, no. 10, pp. 1351–1358, May 2011.

[90] B. Langmead and S. L. Salzberg, "Fast gapped-read alignment with Bowtie 2," *Nature Methods*, vol. 9, no. 4, pp. 357–359, 2012.

[91] P. Klus, S. Lam, D. Lyberg, M. Cheung, G. Pullan, I. McFarlane, G. S. Yeo, and B. Y. Lam, "BarraCUDA–a fast short read sequence aligner using graphics processing units," *BMC Res. Notes*, vol. 5, no. 1, p. 27, 2012.

[92] B. Ma, J. Tromp, and M. Li, "PatternHunter: Faster and more sensitive homology search," *Bioinformatics*, vol. 18, no. 3, pp. 440–445, 2002.

[93] M. Li, B. Ma, D. Kisman, and J. Tromp, "PatternHunter II: Highly sensitive and fast homology search," *J. Bioinf. Comput. Biol.*, vol. 2, no. 3, pp. 417–439, Sep. 2004.

[94] V. Makinen, N. válimaki, A. Laaksonen, R. Katainen, T. Elomaa, H. Mannila, and P. Orponen, "Unified view of backward backtracking in short read mapping," in *Algorithms and Applications* (Lecture Notes in Computer Science), vol. 6060. Berlin, Germany: Springer-Verlag, 2010, pp. 182–195.

[95] D. Khodakov, C. Wang, and D. Y. Zhang, "Diagnostics based on nucleic acid sequence variant profiling: PCR, hybridization, and NGS approaches," *Adv. Drug Del. Rev.*, vol. 105, pp. 3–19, Oct. 2016.

[96] M. L. Metzker, "Sequencing technologies—The next generation," *Nature Rev. Genet.*, vol. 11, no. 1, pp. 31–46, 2010.

[97] H. E. Check, "Genome sequencing: The third generation," *Nature*, vol. 457, pp. 768–769, Feb. 2009.

[98] C. Ye, C. M. Hill, S. Wu, J. Ruan, and Z. S. Ma, "DBG2OLC: Efficient assembly of large genomes using long erroneous reads of the third generation sequencing technologies," *Sci. Rep.*, vol. 6, no. 1, p. 31900, Oct. 2016.

[99] C. Trapnell and S. L. Salzberg, "How to map billions of short reads onto genomes," *Nature Biotechnol.*, vol. 27, no. 5, pp. 455–457, May 2009.

[100] D. Muzzey, E. A. Evans, and C. Lieber, "Understanding the basics of NGS: From mechanism to variant calling," *Current Genet. Med. Rep.*, vol. 3, no. 4, pp. 158–165, Dec. 2015.

JEONGKYU KIM received the bachelor's degree in computer engineering from Dongguk University, in 2019, where he is currently pursuing the master's degree with the Department of Multimedia Engineering. His research interests include computational biology and bioinformatics.

MINGEUN JI received the bachelor's degree in computer science and engineering from Gangneung–Wonju National University, South Korea, in 2018. He is currently pursuing the master's degree with Dongguk University, Seoul, South Korea. His research interests include data science and computational biology.

GANGMAN YI (Member, IEEE) received the master's and Ph.D. degrees in computer sciences from Texas A&M University, USA, in 2007 and 2011, respectively. In 2011, he joined the System Software Group, Samsung Electronics, Suwon, South Korea. He was with the Department of Computer Science and Engineering, Gangneung–Wonju National University, South Korea, in 2012. Since 2016, he has been with the Department of Multimedia Engineering, Dongguk University, Seoul, South Korea. He was researched in an interdisciplinary field of researches. His research interests include development of computational methods to improve understanding of biological systems and its big data. He serves as the Chair for the international conferences and workshops. He also serves as a Managing Editor and a Reviewer for the international journals.