AlignFix

Group Members: Siddharth Kaipa, Manish Sampath, Jason Chiu

Introduction: Our tool Alignfix is a seed and extend method designed to align short query sequences to a genome. The goal of this tool is to identify regions of similarity among the genome, even if the reads contain errors. This can be applied to various bioinformatics problems such as sequence alignment, genome assembly, and variant calling. The reason it's called a seed and extend method is because there are two subtools within Alignfix. The first is to identify a seed, or an exact match with part of the read and part of the target [1]. The second is to align the sequences around the seeds to save both time and space.

Methods: Our tool was split into multiple parts:

The first part is the suffix array. To generate this, we had to add the character "$" to the end of the input. For example if the input was the word "hello", the suffix array would change it to become "hello$". The next step was to generate all the suffixes of the input. Keeping the same example, this means that the suffixes would be [hello$,ello$,llo$,lo$,o$,$]. From here we just have to alphabetically order them => [$,ello$,hello$,llo$,lo$,o$]. To get the actual suffix array, identify the index in which these suffixes show up: [5,1,0,2,3,4]. This method will be done for the database genome. [2]

The second part of the tool implements binary search. Because a suffix array is generated from the genome, we used binary search to locate matches among the queries. Because the suffix array is ordered, we don't have to check every position. If we know it's not in the 5th suffix, we know that it's not in the 4th suffix or any other suffix below that. Thus binary search converges the search region and can search for a query in time O(log(n)) instead of O(n), which makes it considerably faster. Once the seed is found, binary search is used again to identify the first and last occurrences of the query. With the seed and the two locations, a boundary is placed in which we can align the query to the database.

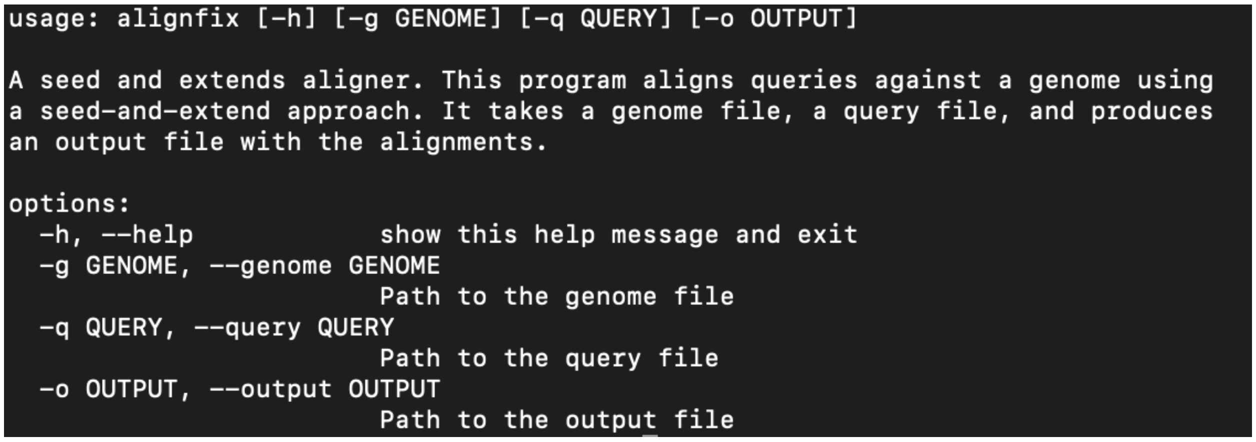
The alignment is done through the use of affine gaps, which means that extending a gap in the alignment will be penalized less than continuously opening up new gaps. The query and the region of the database are put into 3 2D matrices, which will be filled with scores based on if a gap is placed, a gap is extended, or if no gap is implemented (match and mismatches have different scores as well). Once the matrices are filled up, we implemented a simple backtracking method to generate the alignment based on where the max scores came from. This is the third part.

The fourth part of the tool is the command line argument. We utilized the argparse method to take in arguments via the command line. There will be three arguments, denoted by –genome, –query, and –output. This way, the user can input any genome they want along with what queries they want to search for, and our tool will output the closest match.

To periodically test our methods, we went to Gisaid and downloaded a COVID genome sample. From there, we wrote a method which would take reads of a specified length k from the genome and randomly mutate it. This way, we could check how the tool functions if there is no exact match.

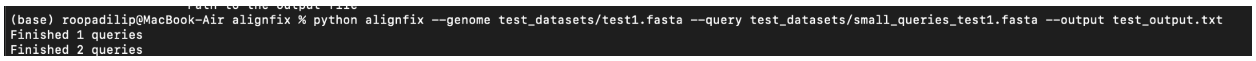
Results: Once the package is downloaded along with all the other background python packages, we can see the results.

Typing python alignfix –help will output text that will show the user how to use the alignfix tool. It also shows what inputs it takes in and what those inputs mean.



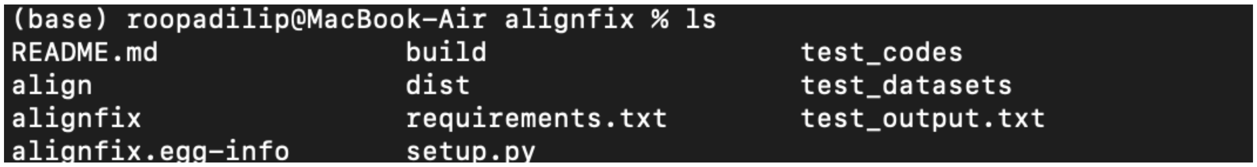
```
usage: alignfix [-h] [-g GENOME] [-q QUERY] [-o OUTPUT]

A seed and extends aligner. This program aligns queries against a genome using
a seed-and-extend approach. It takes a genome file, a query file, and produces
an output file with the alignments.

options:
  -h, --help            show this help message and exit
  -g GENOME, --genome GENOME
                        Path to the genome file
  -q QUERY, --query QUERY
                        Path to the query file
  -o OUTPUT, --output OUTPUT
                        Path to the output file
```

When the tool is run, it will tell the user how many queries have been processed. Then it will output the results in a file that the user can name.



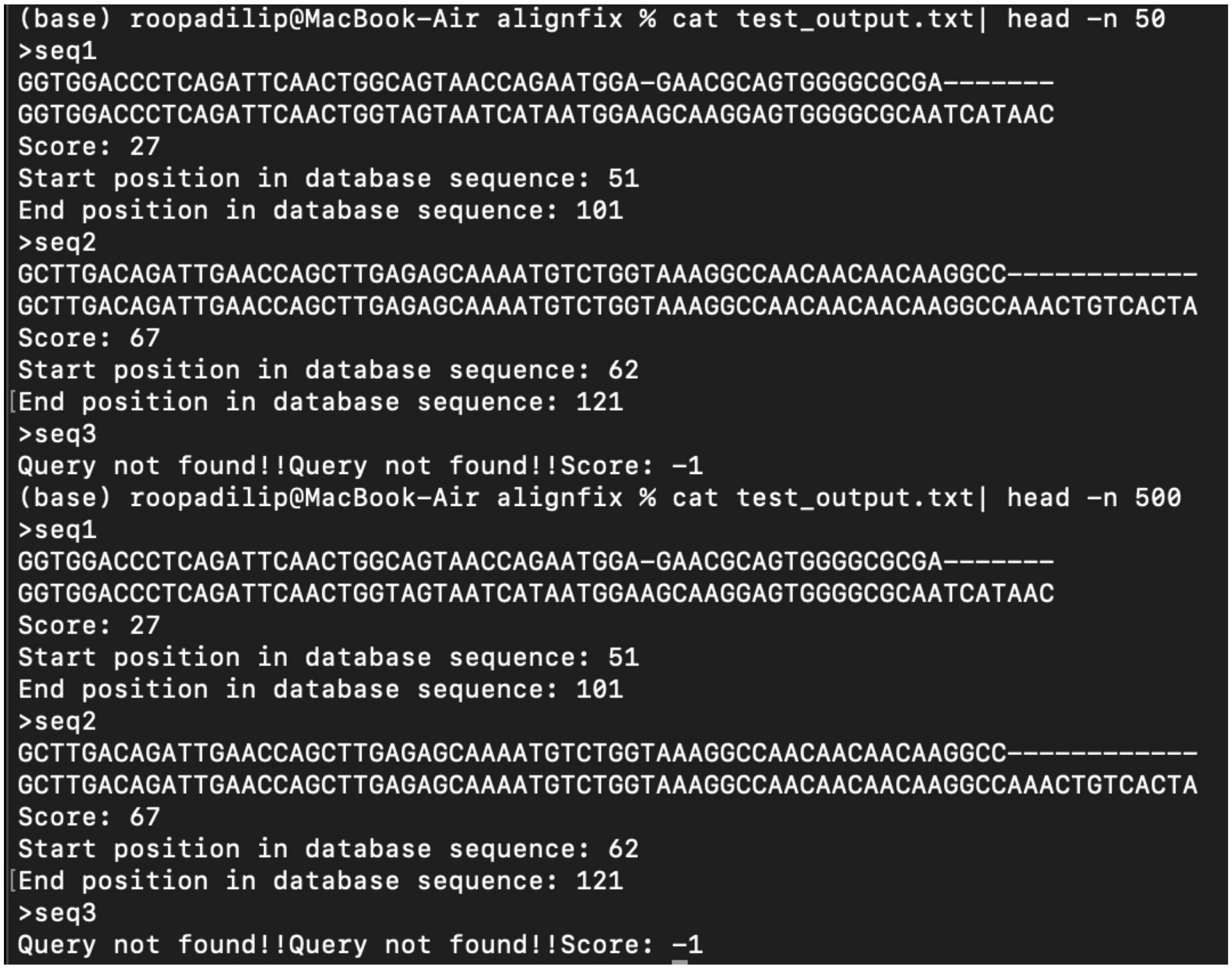
```
(base) roopadilip@MacBook-Air alignfix % python alignfix --genome test_datasets/test1.fasta --query test_datasets/small_queries_test1.fasta --output test_output.txt
Finished 1 queries
Finished 2 queries
```

In this case, we named our output file “text_output” and it’s a .txt file.



```
(base) roopadilip@MacBook-Air alignfix % ls
README.md               build                   test_codes
align                   dist                    test_datasets
alignfix                requirements.txt        test_output.txt
alignfix.egg-info       setup.py
```

When we open the file, we can see chunks of 6 lines. The first is the header for the query. The second and third are the alignments with the best score among the query and the database. The fourth line is the score of the alignment, and the fifth and sixth lines are the start and end position of the query in the database respectively. If a query is not found, there will only be 2 lines: the header and an error message stating that the query was not found and that the score is -1, which indicates that an alignment score for that doesn't exist.



```
(base) roopadilip@MacBook-Air alignfix % cat test_output.txt| head -n 50
>seq1
GGTGGACCCTCAGATTCAACTGGCAGTAACCAGAATGGA-GAACGCAGTGGGGCGCGA-------
GGTGGACCCTCAGATTCAACTGGTAGTAATCATAATGGAAGCAAGGAGTGGGGCGCAATCATAAC
Score: 27
Start position in database sequence: 51
End position in database sequence: 101
>seq2
GCTTGACAGATTGAACCAGCTTGAGAGCAAAATGTCTGGTAAAGGCCAACAACAACAAGGCC------------
GCTTGACAGATTGAACCAGCTTGAGAGCAAAATGTCTGGTAAAGGCCAACAACAACAAGGCCAAACTGTCACTA
Score: 67
Start position in database sequence: 62
End position in database sequence: 121
>seq3
Query not found!!Query not found!!Score: -1
(base) roopadilip@MacBook-Air alignfix % cat test_output.txt| head -n 500
>seq1
GGTGGACCCTCAGATTCAACTGGCAGTAACCAGAATGGA-GAACGCAGTGGGGCGCGA-------
GGTGGACCCTCAGATTCAACTGGTAGTAATCATAATGGAAGCAAGGAGTGGGGCGCAATCATAAC
Score: 27
Start position in database sequence: 51
End position in database sequence: 101
>seq2
GCTTGACAGATTGAACCAGCTTGAGAGCAAAATGTCTGGTAAAGGCCAACAACAACAAGGCC------------
GCTTGACAGATTGAACCAGCTTGAGAGCAAAATGTCTGGTAAAGGCCAACAACAACAAGGCCAAACTGTCACTA
Score: 67
Start position in database sequence: 62
End position in database sequence: 121
>seq3
Query not found!!Query not found!!Score: -1
```

The results are meant to look like BLAST. [3]

Comparing Against BWA: We compared our tool against BWA mem using a real world data set: A covid genome. The results were as followed:

Categories	**AlignFix**	**BWA MEM**
Time To Align	152.93 seconds	0.289 seconds
Percent of Queries Aligned	83%	99.93%

BWA is significantly faster than our tool. It also aligned more queries than AlignFix. However, since our original goal was to create an alignment tool that would align at least 80% of queries, we succeeded in our goal.

Discussion: One of the hardest parts of this project was figuring out how the affine gap penalties could be coded, as we needed to take into account 3 matrices, not just one. Another challenge was figuring out how to get the command line arguments such as pip install to work. To fix this, we looked at stackoverflow for advice [4]. Once those were fixed, the tool worked as intended. For future directions, it would be nice to see this tool work with other file formats other than fasta. We would also like to test it against existing aligner tools to compare the speed and performance of alignfix.

Code: https://github.com/msampath25/alignfix

References:

[1]https://www.sevenbridges.com/short-read-alignment-seeding/#:~:text=Alignment%3A%20a%20quick%20review&text=Many%20modern%20alignment%20algorithms%20rely,so%20seeding%20is%20very%20fast.

[2]https://usaco.guide/adv/suffix-array?lang=cpp

[3]https://blast.ncbi.nlm.nih.gov/Blast.cgi

[4]https://stackoverflow.com/questions/56534678/how-to-create-a-cli-in-python-that-can-be-installed-with-pip/66010978#66010978