## Previous Lectures

### Fall 2022 Meetings/Schedule

<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Livestream</th>
<th>Meeting</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1</td>
<td>13.10 Thu.</td>
<td>YouTube Live</td>
<td>L1: Intelligent Genomic Analyses</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(PDF) (PPT) YouTube Video</td>
</tr>
<tr>
<td>W2</td>
<td>27.10 Thu.</td>
<td>YouTube Live</td>
<td>L2: P&amp;S Course Introduction Logistics</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(PDF) (PPT)</td>
</tr>
<tr>
<td>W3</td>
<td>3.11 Thu.</td>
<td>Premiere</td>
<td>L3: Introduction to Sequencing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(PDF) (PPT)</td>
</tr>
<tr>
<td>W4</td>
<td>10.11 Thu.</td>
<td>Premiere</td>
<td>L4: Read Mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(PDF) (PPT)</td>
</tr>
<tr>
<td>W5</td>
<td>17.11 Thu.</td>
<td>Premiere</td>
<td>L5: GateKeeper</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(PDF) (PPT)</td>
</tr>
</tbody>
</table>

Livestream - P&S Genome Sequencing on Mobile
Onur Mutlu Lectures - 1 / 3

https://safari.ethz.ch/projects_and_seminars/fall2022/doku.php?id=bioinformatics
We need intelligent algorithms and intelligent architectures that handle data well.
Goal: Minimizing Alignment Time

Sequence Alignment is expensive

Our goal is to accelerate read mapping by reducing the need for dynamic programming algorithms
Key Idea

Genomic Strings

- **Dissimilar Strings**
  - Ignore them if the number of differences exceeds a threshold.

- **Similar Strings**
  - Find number, location, and type of differences?

**EXPENSIVE!**
1. Filter out most of dissimilar sequences.
2. Preserve all similar sequences.
3. Do it quickly.
MAGNET: Understanding and Improving the Accuracy of Genome Pre-Alignment Filtering

Alser, Mohammed; Mutlu, Onur; and Alkan, Can
On the False Positives of GateKeeper

We investigate four major sources of false positives:

- Leading and trailing zeros
- Random zeros
- Conservative counting
- Lack of backtracking
Leading and Trailing Zeros

Fig. 3: Examples of an invalid mapping that passes the SHD filter due to the leading and trailing zeros (first source of false positives). We use an edit distance threshold of 3 and an SRS threshold of 2. While the regions that are highlighted in green are part of the correct alignment, the wrong alignment provided by SHD is highlighted in red. The yellow highlighted bits indicate a source of false positive.
Fig 4: Examples of an incorrect mapping that passes the SHD filter due to random zeros (second source false positives). While the edit distance threshold is 3, a mapping of 4 edits (as examined at the end of the figure by Needleman-Wunsch algorithm) passes as a false positive.
Conservative Counting

The 3-bit ones are a result of substitutions and not the amendment.

| Query: AAAAAAACAACAACCCCATCAAAAGTGCCGTTAAGGAATATGATTCACACTTCTCAAAAGAGACATTTTCCAGCCAAAACACATGAAATATGCTC |
| Reference: AAAAAAAACAACAACCCCATCAAAAGTGCCGTTAAGGAATATGATTCACACTTCTCAAAAGAGACATTTTCCAGCCAAAACACATGAAATATGCT |
| Hamming Mask: 000000000000000000000000000000001110000000000000000000000000000011111111100011111111111111 |
| 1-Deletion Mask: 00000011111111000111110000111111111111111111111111111111111111111111111111111111111111111111111111111111111 |
| 2-Deletion Mask: 00000111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111 |
| 3-Deletion Mask: 00000011111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111 |
| 1-Insertion Mask: 00000111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111 |
| 2-Insertion Mask: 00000111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111 |
| 3-Insertion Mask: 00000111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111 |
| Final bit-vector: 0000000000000000000000000000000011100000000000000000000100000000000000000000000000000000000000000 |

Needleman-Wunsch Alignment:

AAAAAAAACAACCCCATCAAAAGTGCCGTTAAGGAATATGATTCACACTTCTCAAAAGAGACATTTTCCAGCCAAAACACATGAAATATGCT

\[\ldots\] : \ldots\]

AAAAAAAACAACCCCATCAAAAGTGCCGTTAAGGAATATGATTCACACTTCTCAAAAGAGACATTTTCCAGCCAAAACACATGAAATATGCT

Fig 5: An example of an incorrect mapping that passes the SHD filter due to conservative counting of the short streak of ‘1’s in the final bit-vector.
Lack of Backtracking

Fig 6: An example of an incorrect mapping that passes the SHD filter due to the lack of backtracking (overlapping identical subsequences).

Fig 7: An example of an invalid mapping that passes the SHD filter due to the lack of backtracking.
Can we improve the accuracy?
**Key observation:**
- Correct alignment always includes non-overlapping long identical subsequences.

**Key idea:**
- Count the **consecutive zeros** in each mask and select the longest in a divide-and-conquer approach.
MAGNET Walkthrough

Build Neighborhood Map  Track the Diagonally Consecutive Matches  ACCEPT iff number of ‘1’ ≤ Threshold

Read: TTTTACTGTTCTCCCTTTGAATACAATATATCTATATTTCCCTTGCTACATTTAAAATTTCCCCTTTATCTGTAATAATCAGTAATTACGGTTTTAAAA
Reference: TTTTACTGTTCTCCCTTTGAATACAATATATCTATATTTCCCTTGCTACATTTAAAATTTCCCCTTTATCTGTAATAATCAGTAATTACGGTTTTAAAA

Upper Diagonal-4:

| 110111111100111111110101100001010001011010011111101101100110110011010101011101111111101011000000 |

Upper Diagonal-3:

| 001101101011111111110101100110100111111011011001101100110101010111011111111111101011000010 |

Upper Diagonal-2:

| 00111101100101101110110000000000000000000000000000000000000000000000000010011110110111111111111111111111 |

Upper Diagonal-1:

| 0011111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111 |

Main Diagonal:

| 0000000000000000000001010000000000000000000000000000000000000000000000000000000000000000010001000000 |

Lower Diagonal-1:

| 0011111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111 |

Lower Diagonal-2:

| 0011111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111 |

Lower Diagonal-3:

| 0011111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111 |

Lower Diagonal-4:

| 0011111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111 |

MAGNET bit-vector:

| 0000000000000000000001010000000000000000000000000000000000000000000000000000000000000000010001000000 |

Find the longest segment of consecutive zeros
Exclude the errors from the search space
Divide the problem into two subproblems and repeat
Total number of edits = number of 1’s in MAGNET bit-vector
Fig. 1: Random edit distribution in a read sequence. The edits ($e_1$, $e_2$, ..., $e_E$) act as dividers resulting in several identical subsequences ($m_1$, $m_2$, ..., $m_{E+1}$) between the read and the reference.
MAGNET Walkthrough

**Build Neighborhood Map**

- Identifying $E+1$ non-overlapping subsequences

**ACCEPT iff number of ‘1’ ≤ Threshold**

- $\log_2 \text{ReadLength}$-bit counter.

### MAGNET Bit-vector

1. $\text{Pattern}$
2. $\text{Reference}$
3. $\text{Edit distance threshold}$

#### Neighborhood Map

- $2E+1$ bit-vectors

#### Comparator

- $\log_2 m$

#### Counter

- $m$

Proceed to the next Iteration
MAGNET (AACBB 2018, TIR 2017)

- Key observation:
  - Correct alignment always includes non-overlapping long identical subsequences.

- Key idea:
  - Count the **consecutive zeros** in each mask and select the longest in a divide-and-conquer approach.

- Key result:
  - MAGNET is 74x - 460x **faster** than its CPU implementation.
  - Contains up to **2 or 8 filtering units**, each of which has **10 folds the footprint** of that of GateKeeper on the FPGA.
  - MAGNET is 3.5x to 25552x (as GateKeeper stop filtering after E=4%[250bp] or 8%[100bp]) **more accurate** than GateKeeper and SHD.

- Weaknesses: Challenging to be implemented on FPGA due to random search.
More on MAGNET

- Download and test for yourself
  [https://github.com/BilkentCompGen/MAGNET](https://github.com/BilkentCompGen/MAGNET)

Can we do better? Scalability?
Sequence alignment

**Shouji: a fast and efficient pre-alignment filter for sequence alignment**

Mohammed Alser¹,²,³,*, Hasan Hassan¹, Akash Kumar², Onur Mutlu¹,³,*, and Can Alkan³,*

¹Computer Science Department, ETH Zürich, Zürich 8092, Switzerland, ²Chair for Processor Design, Center For Advancing Electronics Dresden, Institute of Computer Engineering, Technische Universität Dresden, 01062 Dresden, Germany and ³Computer Engineering Department, Bilkent University, 06800 Ankara, Turkey

*To whom correspondence should be addressed.

Associate Editor: Inanc Birol

Received on September 13, 2018; revised on February 27, 2019; editorial decision on March 7, 2019; accepted on March 27, 2019

Alser+, “**Shouji: a fast and efficient pre-alignment filter for sequence alignment**”, *Bioinformatics* 2019,

[https://doi.org/10.1093/bioinformatics/btz234](https://doi.org/10.1093/bioinformatics/btz234)
Key observation:

- Correct alignment always includes long identical subsequences.
- Processing the entire mapping at once (as in GateKeeper) is ineffective for hardware design.
"dot plot" or "dot matrix" is a visual representation of the similarities between two closely similar genomic sequences that is used in FASTA/FASTP (Lipman and Pearson, 1985).
**Key observation:**
- Correct alignment always includes *long identical subsequences*.
- Processing the entire mapping at once (as in GateKeeper) is ineffective for hardware design.

**Key idea:**
- Use **overlapping sliding window** approach to quickly and accurately find all long segments of *consecutive zeros*.
Given a text sequence $T[1...m]$, a pattern sequence $P[1...m]$, and an edit distance threshold $E$, the neighborhood map represents the comparison result of the $i^{th}$ character of $P$ with the $j^{th}$ character of $T$, where $i$ and $j$ satisfy $1 \leq i \leq m$ and $i-E \leq j \leq i+E$. The entry $N[i, j]$ of the neighborhood map can be calculated as follows:

$$N[i, j] = \begin{cases} 0, & \text{if } P[i] = T[j] \\ 1, & \text{if } P[i] \neq T[j] \end{cases}$$  \hspace{1cm} (1)$$
Shouji Walkthrough

Building the Neighborhood Map

Finding all common subsequences (diagonal segments of consecutive zeros) shared between two given sequences.

Storing it @ Shouji Bit-vector

Shouji Walkthrough

Building the Neighborhood Map

Finding all common subsequences (diagonal segments of consecutive zeros) shared between two given sequences.

Storing it @ Shouji Bit-vector

What Does Shouji Mean?

Named after a traditional Japanese door that is designed to slide open
The reason behind the selection of the window size is due to the minimal possible length of the identical subsequence that is a single match (e.g., such as `101').
• Counting is performed **concurrently** for *all* bit-vectors and all sliding windows in a single clock cycle using **multiple 4-input LUTs**.
Shouji

- **Key observation:**
  - Correct alignment always includes long identical subsequences.
  - Processing the entire mapping at once (as in GateKeeper) is ineffective for hardware design.

- **Key idea:**
  - Use overlapping sliding window approach to quickly and accurately find all long segments of consecutive zeros.

- **Key result:**
  - Shouji on FPGA is up to three orders of magnitude faster than its CPU implementation.
  - Shouji accelerates best-performing CPU read aligner Edlib (Bioinformatics 2017) by up to 18.8x using 16 filtering units that work in parallel.
  - Shouji is 2.4x to 467x more accurate than GateKeeper (Bioinformatics 2017) and SHD (Bioinformatics 2015).
More on Shouji

Sequence alignment

**Shouji: a fast and efficient pre-alignment filter for sequence alignment**

Mohammed Alser¹,²,³,*, Hasan Hassan¹, Akash Kumar², Onur Mutlu¹,³, * and Can Alkan³, *

¹Computer Science Department, ETH Zürich, Zürich 8092, Switzerland, ²Chair for Processor Design, Center For Advancing Electronics Dresden, Institute of Computer Engineering, Technische Universität Dresden, 01062 Dresden, Germany and ³Computer Engineering Department, Bilkent University, 06800 Ankara, Turkey

*To whom correspondence should be addressed.

Associate Editor: Inanc Birol

Received on September 13, 2018; revised on February 27, 2019; editorial decision on March 7, 2019; accepted on March 27, 2019

Alser+, “Shouji: a fast and efficient pre-alignment filter for sequence alignment”, *Bioinformatics* 2019,

https://doi.org/10.1093/bioinformatics/btz234

Download and test for yourself

[https://github.com/CMU-SAFARI/Shouji](https://github.com/CMU-SAFARI/Shouji)
Assignment #2

Which one out of the three pre-alignment filters, GateKeeper, MAGNET, and Shouji is more efficient? why?

- Hint:

<table>
<thead>
<tr>
<th></th>
<th>GateKeeper</th>
<th>MAGNET</th>
<th>Shouji</th>
<th>Justification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time complexity</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Space complexity</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FPGA resource</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accuracy</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Most speedup comes from **parallelism enabled** by **novel architectures** and **algorithms**
More on GateKeeper [Alser+, Bioinformatics 2017]

Mohammed Alser, Hasan Hassan, Hongyi Xin, Oguz Ergin, Onur Mutlu, and Can Alkan
"GateKeeper: A New Hardware Architecture for Accelerating Pre-Alignment in DNA Short Read Mapping"
Bioinformatics, [published online, May 31], 2017.
[Source Code]
[Online link at Bioinformatics Journal]
Mohammed Alser, Jeremy Rotman, Dhrithi Deshpande, Kodi Taraszka, Huwenbo Shi, Pelin Icer Baykal, Harry Taegyun Yang, Victor Xue, Sergey Knyazev, Benjamin D. Singer, Brunilda Balliu, David Koslicki, Pavel Skums, Alex Zelikovsky, Can Alkan, Onur Mutlu, Serghei Mangul

"Technology dictates algorithms: Recent developments in read alignment"

Genome Biology, 2021

[Source code]
Accelerating Read Mapping

Accelerating Indexing
- Reducing the number of seeds
- Reducing data movement during indexing

Accelerating Pre-Alignment Filtering
- q-gram filtering
- Pigeonhole principle
- Base counting
- Sparse DP

Accelerating Alignment
- Accurate alignment accelerators
- Heuristic-based alignment accelerators

Mohammed Alser, Zülal Bingöl, Damla Senol Cali, Jeremie Kim, Saugata Ghose, Can Alkan, Onur Mutlu

“Accelerating Genome Analysis: A Primer on an Ongoing Journey”
More on Fast Genome Analysis ...

- Onur Mutlu, "Accelerating Genome Analysis: A Primer on an Ongoing Journey"
Invited Lecture at Technion, Virtual, 26 January 2021.
[Slides (pptx) (pdf)]
[Talk Video (1 hour 37 minutes, including Q&A)]
[Related Invited Paper (at IEEE Micro, 2020)]
More on Intelligent Genome Analysis ...

https://www.youtube.com/watch?v=ygmQpdDTL7o
Detailed Lectures on Genome Analysis

- **Computer Architecture, Fall 2020, Lecture 3a**
  - *Introduction to Genome Sequence Analysis* (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=CrRb32v7SJc&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=5](https://www.youtube.com/watch?v=CrRb32v7SJc&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=5)

- **Computer Architecture, Fall 2020, Lecture 8**
  - *Intelligent Genome Analysis* (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=ygmQpdDTL7o&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=14](https://www.youtube.com/watch?v=ygmQpdDTL7o&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=14)

- **Computer Architecture, Fall 2020, Lecture 9a**
  - *GenASM: Approx. String Matching Accelerator* (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=XoLpzmNPas&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=15](https://www.youtube.com/watch?v=XoLpzmNPas&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=15)

- **Accelerating Genomics Project Course, Fall 2020, Lecture 1**
  - *Accelerating Genomics* (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=rgjl8ZyLsAg&list=PL5Q2soXY2Zi9E2bBVAgCqLgwiDRQDTyId](https://www.youtube.com/watch?v=rgjl8ZyLsAg&list=PL5Q2soXY2Zi9E2bBVAgCqLgwiDRQDTyId)

[https://www.youtube.com/onurmutlulectures](https://www.youtube.com/onurmutlulectures)
Prior Research on Genome Analysis (1/2)

- Alser + "SneakySnake: A Fast and Accurate Universal Genome Pre-Alignment Filter for CPUs, GPUs, and FPGAs.", *Bioinformatics*, 2020.


P&S Accelerating Genomics
Lecture 6: MAGNET & Shouji

Dr. Mohammed Alser
@mealser
ETH Zurich
Fall 2022
24 November 2022