



Building a Primers Library for DNA Storage

#15006805

Saar Cohen

Shahar Trabelsi

Advisor: Dr. Sarel Cohen

In collaboration with: Prof. Dalit Naor.



1

DNA Storage - Introduction



Motivation

- Data production is rapidly increasing.
- Traditional storage methods are struggling with capacity.
- Traditional storage methods also face challenges with longevity.



What is DNA Storage?

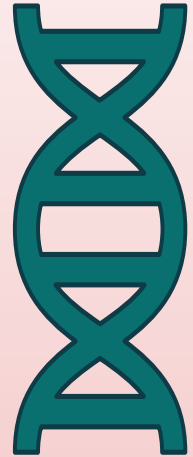
- Encoding digital information into DNA sequences.
- Preserves data within the structure of DNA molecules.
- Practical use is currently severely limited because of its high cost and very slow read and write times.



Why DNA Storage?

- **High Density:** Can store vast amounts of data in a very small physical space.
- **Longevity:** DNA is stable over thousands of years, ideal for long-term storage.
- **Durability:** Resistant to extreme environmental conditions.
- **Energy Efficiency:** Requires no power to maintain the stored information.
- **Scalability:** Potential to easily scale up storage capacity as needed.
- **Portability:** Highly portable due to its small physical size.

This makes it a promising technology for the future

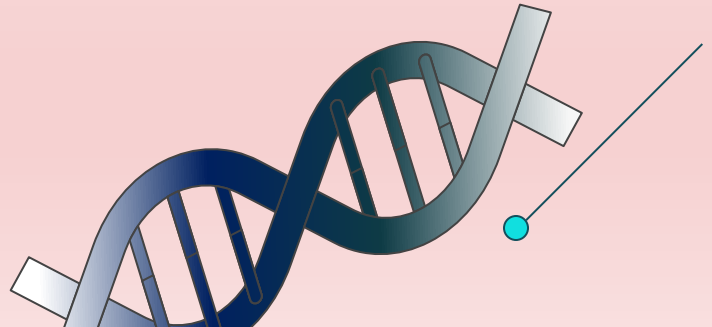




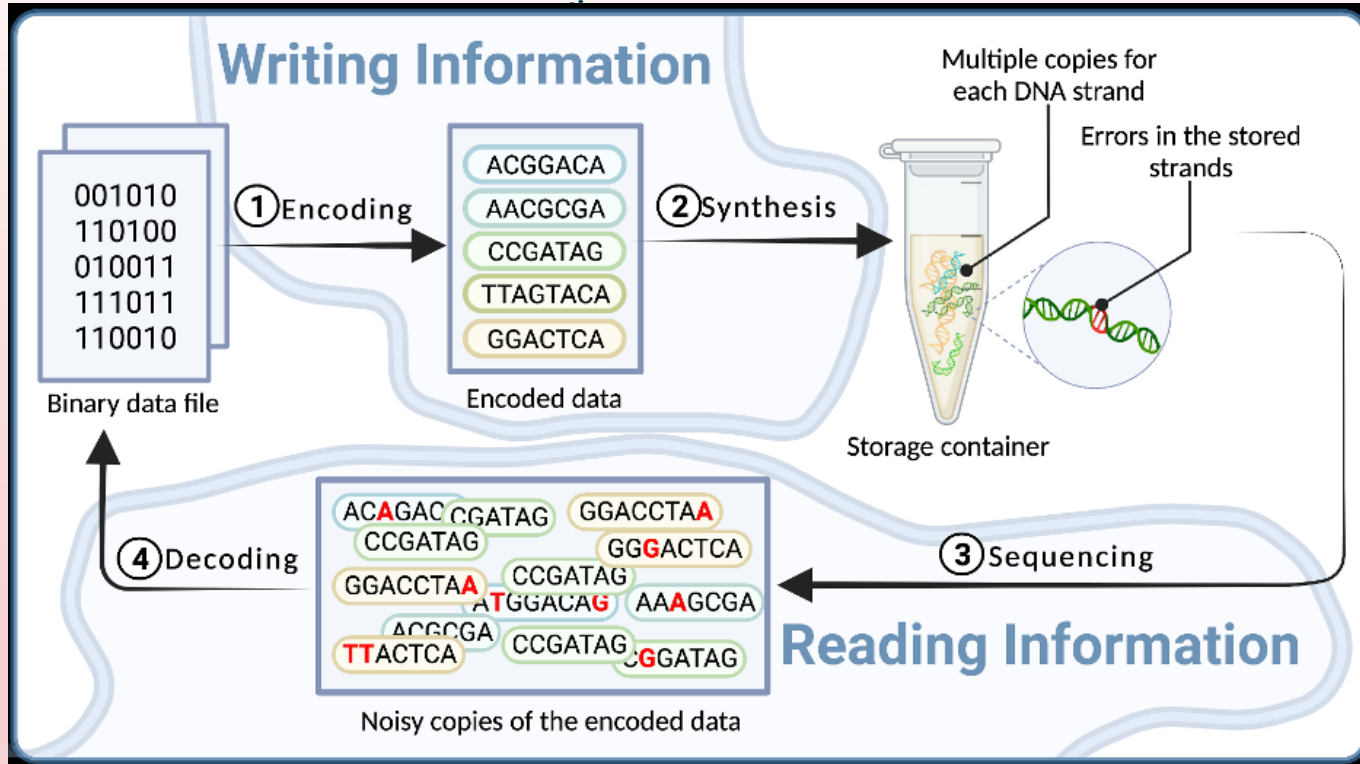
Background and Terminology

- **DNA:** Made of four building blocks: A, C, G, and T.
- **DNA Storage:** turn digital information into sequences of these letters.
- **DNA Strand:** A sequence of synthetic DNA.

Primer: A short (20 mer long) DNA sequence used as unique identifiers for files within a large DNA pool.



The Main Process



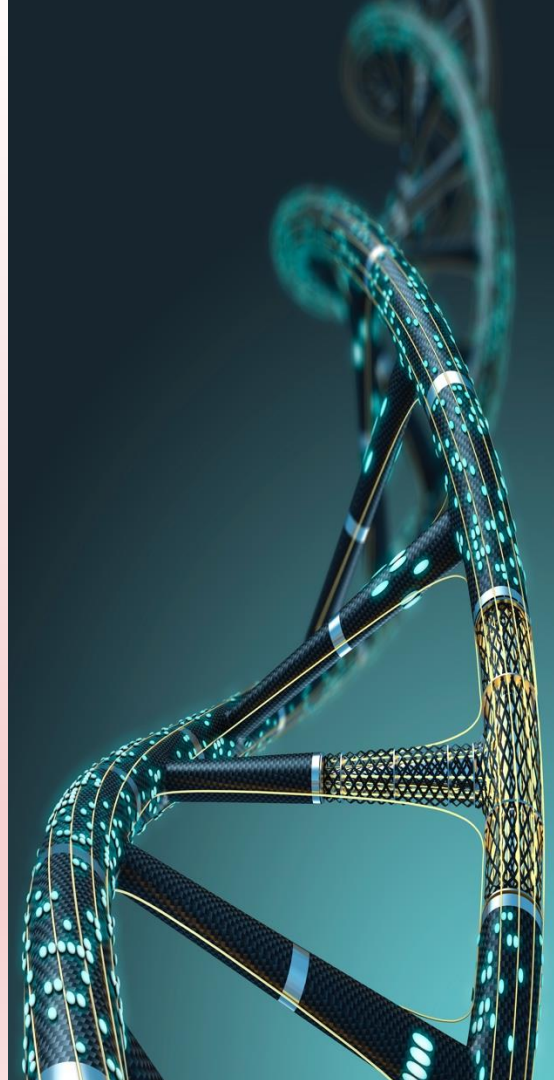
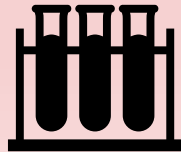
Today, the longest DNA strand of synthetic DNA that can be is 200-300.



All the world's data can be stored in just

204,651

DNA-Tubes





2

About the Article and Their Approach for Building a Primer Library for DNA Storage



The Article

"Random Access in Large-Scale DNA Data Storage"

Lee Organick, Siena Dumas Ang, Yuan-Jyue Chen, Randolph Lopez, Sergey Yekhanin, Konstantin Makarychev, Miklos Z Racz, Govinda Kamath, Parikshit Gopalan, Bichlien Nguyen, Christopher N Takahashi, Sharon Newman, Hsing-Yeh Parker, Cyrus Rashtchian, Kendall Stewart, Gagan Gupta, Robert Carlson, John Mulligan, Douglas Carmean, Georg Seelig, Luis Ceze & Karin Strauss

The topic of the Article:

- Advancements in DNA data storage technology
- Focus on encoding, storing, and enabling random access to digital data in large-scale DNA storage with primers designed for efficient retrieval and scalability.

Key Achievements:

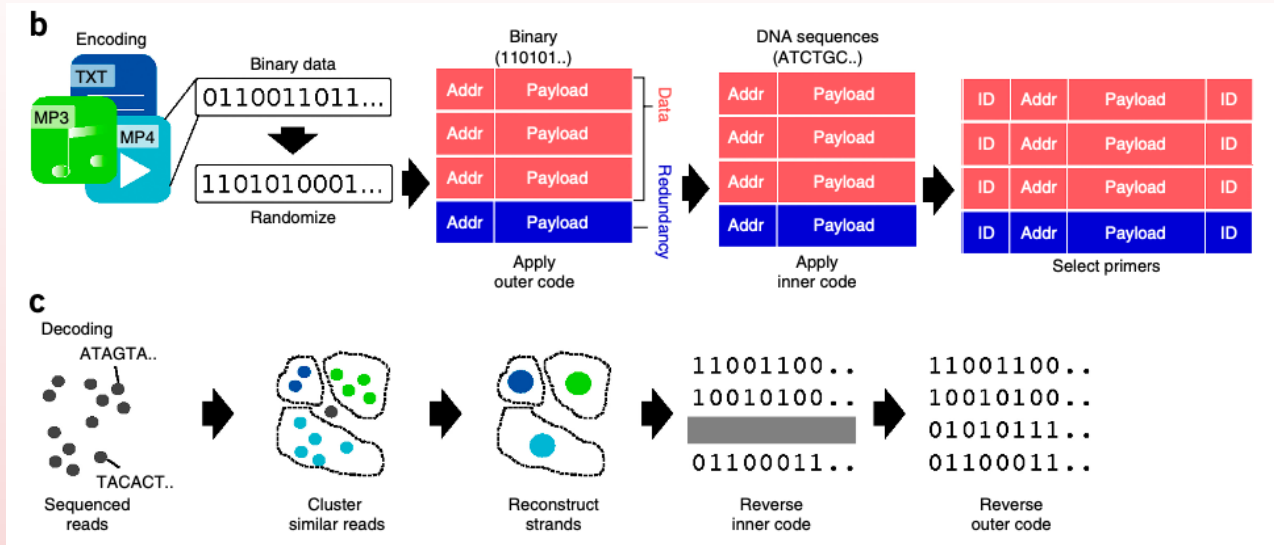
- Successfully stored 200 MB of data
- Demonstrated error-free retrieval using a new algorithm
- Developed a library of PCR primers for efficient data access

Highlighted Challenge:

- Building a robust library of unique DNA primers
- Meeting strict biological requirements

Objective Moving Forward:

- Build upon discoveries in developing a DNA primer library
- Address challenges related to primer uniqueness and biological constraints



- Primers are 20-mer sequences that must adhere to specific biological restrictions
- Theoretical number of possible primers: 4^{20} (~1.1 trillion)
- However, most sequences fail to meet the required biological constraints

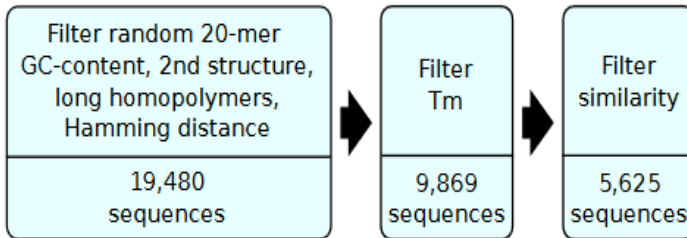
Their Findings

Our focus

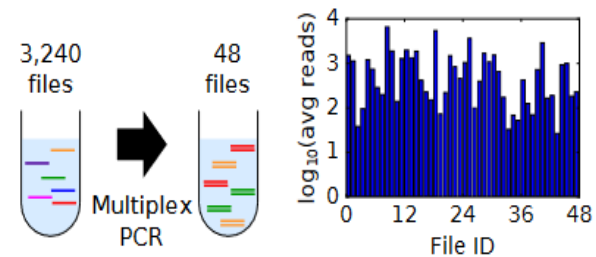
Biological process

a Primer library design

i. Design workflow

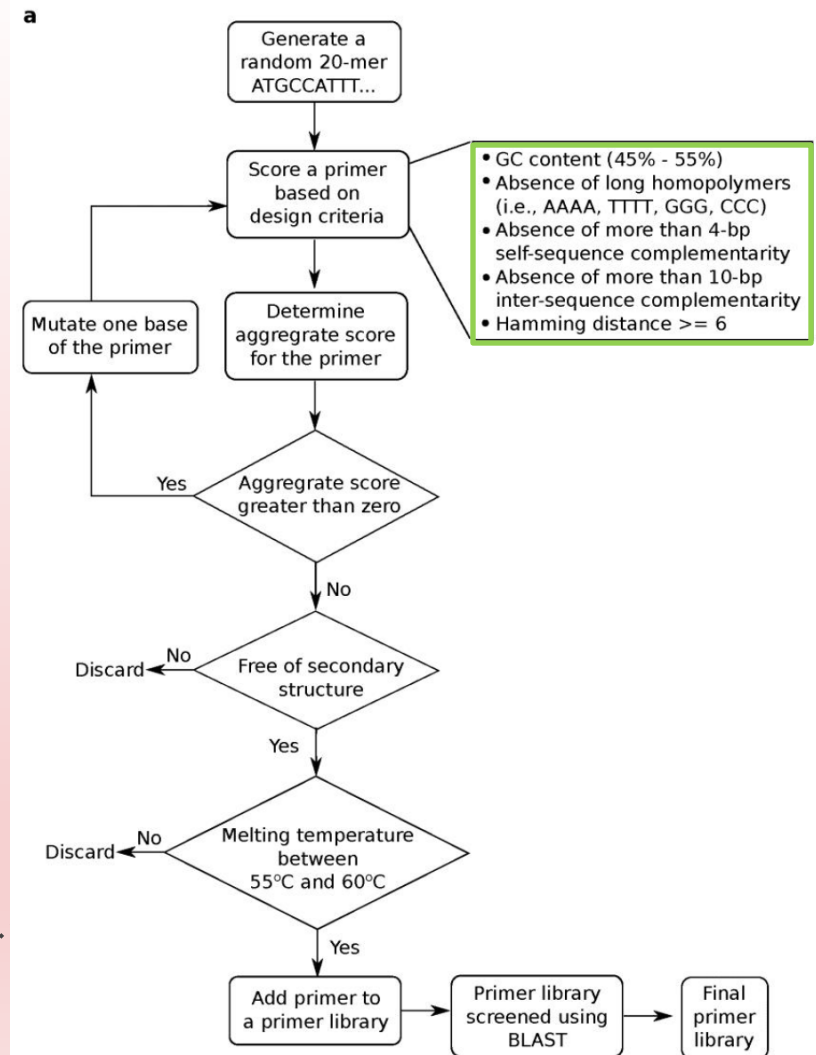


ii. Validation



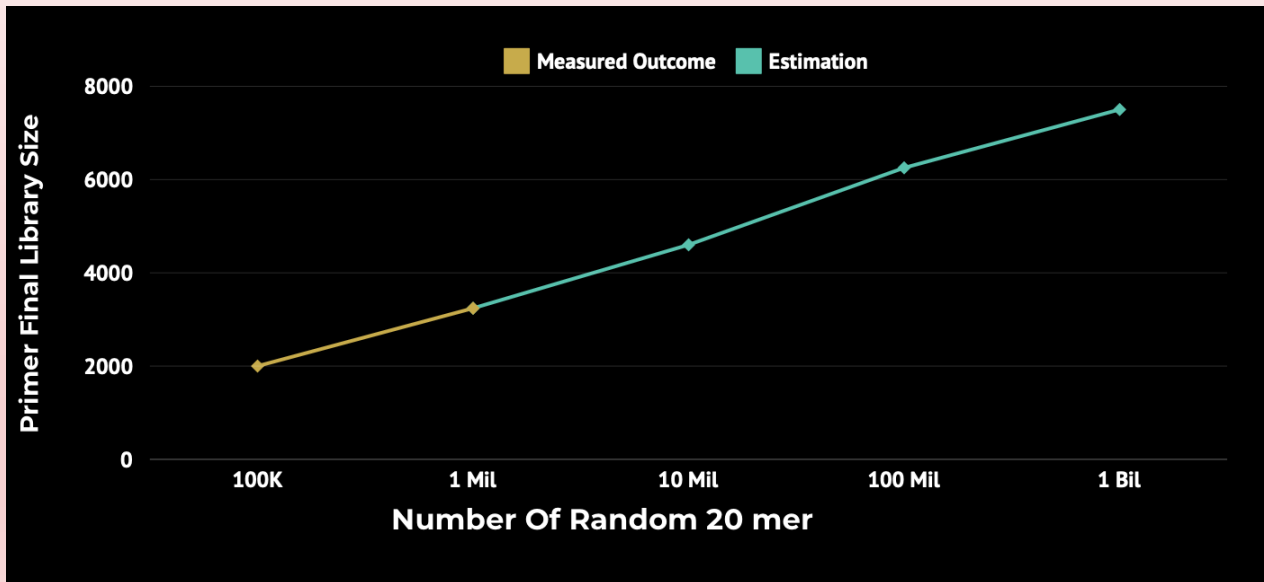
Their Process

- Repeat the flowchart **M** times:
Generate M random primers.
- For each generated random string:
 - **Mutate** strings that fail to meet the restrictions marked in green until one is found.
- The process inspects more than **M** primers:
If the average number of mutations per primer is **r**:
Total primers generated: $M' = M * r$.



Their Findings

The total number of primer pairs that pass the selection criteria increases with the log of the number of starting random 20-mers.





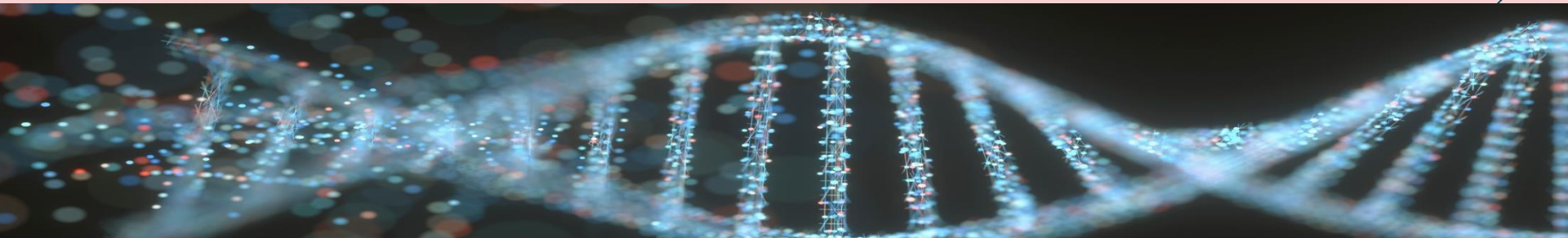
3

Our Goals & Our Approach



Our Goals

- 1) Develop an algorithm and implementation for building a primers library for DNA storage that fully meets biological and random access data system requirements as described in the article.
- 2) Enhance the algorithm's efficiency to ensure it runs within a reasonable timeframe.
- 3) Maximize the number of primers stored within a single primer tube.
- 4) Analyze and profile the algorithm.



Our Approach

Initial Approach:

- Started with a ***naive Python implementation** to build a primer library of **length 14**
- The algorithm checked all primers of length 14 against part of the **biological conditions** from the paper
- Realized that the algorithm was **too slow** for practical use

Bioinformatics tools:

- **Searched existing software** to handle the last 4 biological restrictions:
- Primer3, NUPACK, CD-HIT

New Approach:

- Adopted a **new strategy** (illustrated in the next slides)
- Focused on evaluating the number of primers in the final library **based on the number of generated random primers**

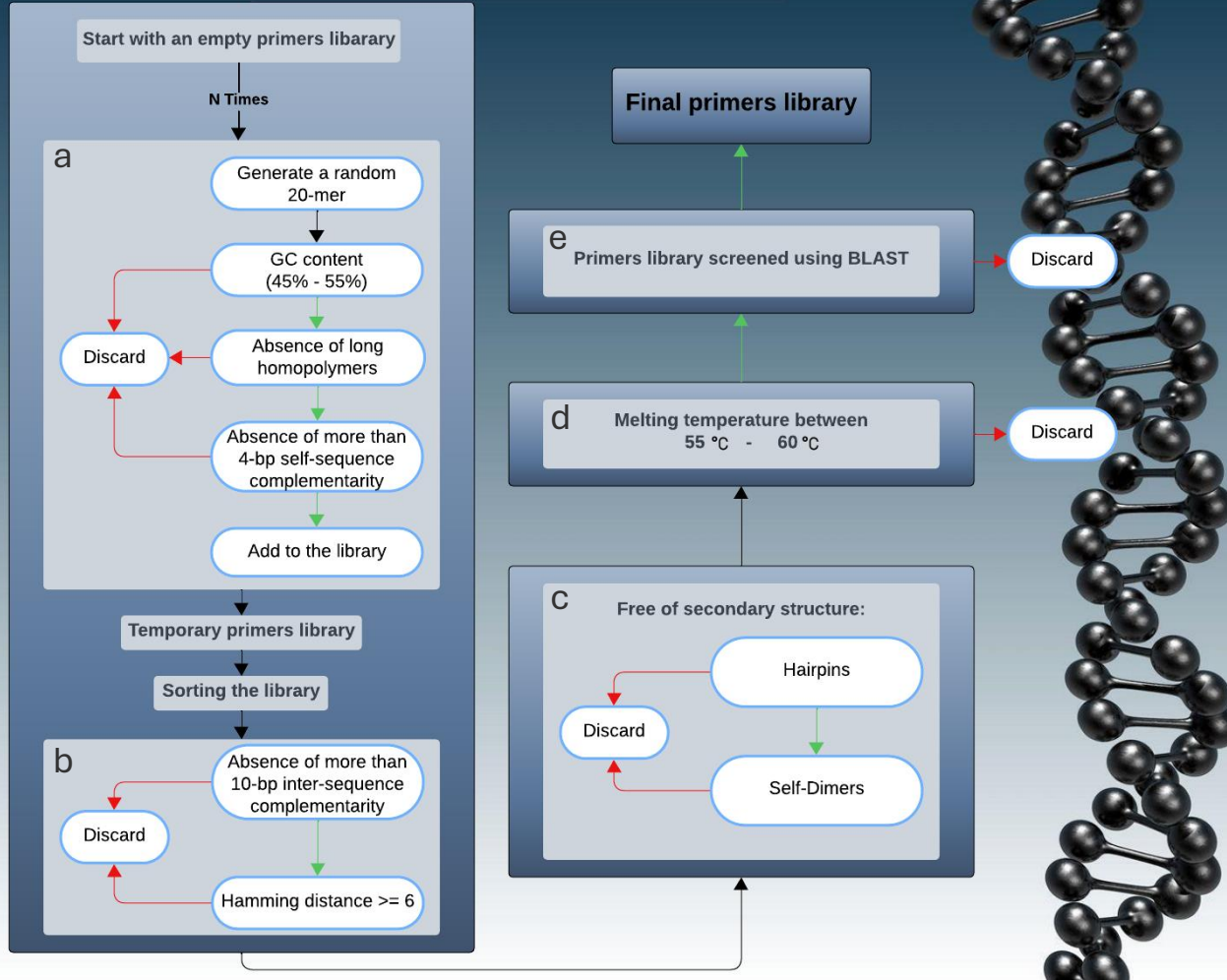


4

Our Solution



S&S Algorithm

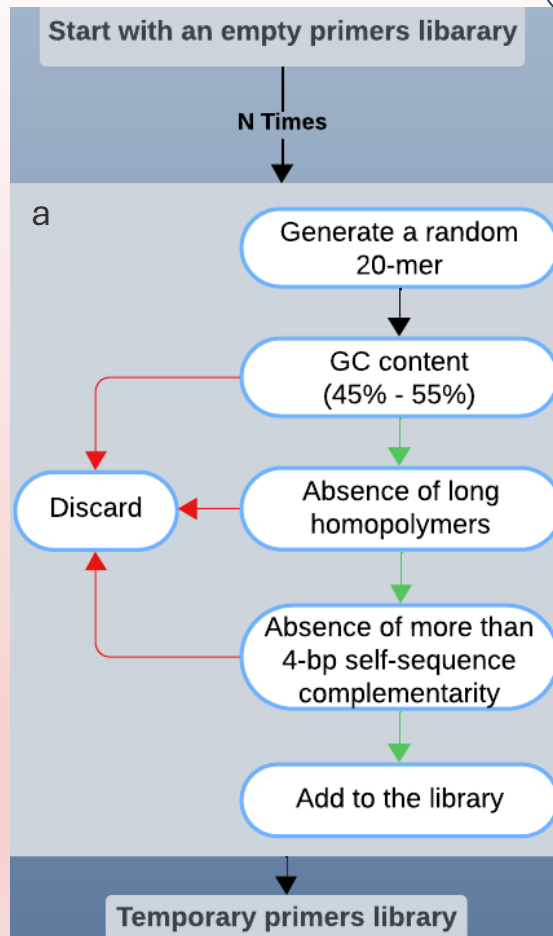


Our Solution - Deep Dive

- Intra-primer restrictions
- Linear time complexity check

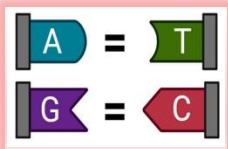
Process Overview:

- **Start** with an **empty primer library**
- **Step 1:** Generate **N random primers** for evaluation
- **Step 2: Internal DNA Restrictions Check (a):**
Check each primer for:
GC content
Long homopolymers
Long self-sequence complementarity
- **Step 3:** **Save primers** that pass the internal restrictions to a **temporary library**



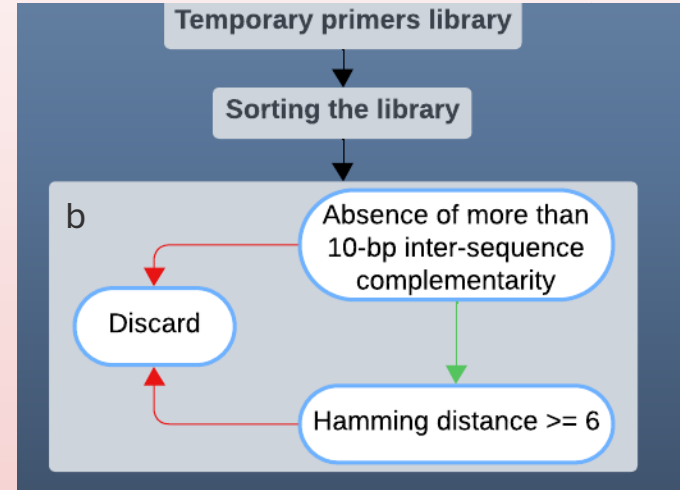
Our Solution - Deep Dive

Internal DNA restrictions (a):

GC Content	Ranging between 45% to 55% G and C content.	<p><u>For example:</u></p> <p>For a primer of length 20:</p> <ul style="list-style-type: none">• 'CATTGAGAATGAGCTGTTTT' is disqualified (35% GC).• 'CACCGAGGCTGAGCTGTCTT' is disqualified (60% GC).• 'ATGCGTACCGTAGCTAACGT' meets the condition (50% GC).
Long Homopolymers	Avoids primers with long repeats of the same nucleotide to prevent errors.	<p><u>For example:</u></p> <p>For a primer of length 20 without homopolymers greater than length 2:</p> <ul style="list-style-type: none">• 'TTGATAGTGATCTGAGTTTA' is disqualified (contains repeated 'T' 3 times).• 'ATCGTACGTAGCTAGCATGC' meets the condition.
Long Self-Sequence Complementarity	Avoids primers with long self-sequence complementarity. 	<p><u>For example:</u></p> <p>For a primer of length 20 without self complementing greater than 4:</p> <ul style="list-style-type: none">• 'ATCGATCGGCTAGCCGATCG' is disqualified (contains self-reverse-sequence complementarity – 'CGATCGGCTAGCCGA' – reverse complement).• 'ATCGGATCTAGGTCACGTAC' meets the condition.

Our Solution - Deep Dive

- Check the primer against all existing primers in the library.
- Sort the temporary library for efficiency.
- Reason: A sorted library allows us to run our Hamming distance test function more effectively.



Our Solution – Deep Dive

More DNA Restrictions (b):

Long Inter-Sequence Complementarity:

Avoids primers with long inter-sequence complementarity.

For example:

For a primer of length 20 and max inter-reverse-sequence complementarity > 10 and contain $p = \text{'ATCGTACGGTACGTTAGCGA'}$ in the library:

(p reverse complement =

$\text{'TCGCTAACGTACCGTACGAT'}$)

- **$\text{'TCGCTAACGTAGGTTAGCGA'}$** is disqualified.
- $\text{'TAGCATGCCTGCACCATCGT'}$ meets the condition.

Pseudo Code:

```
patterns = [] #set
```

#temp_primers_library - all primers that passed the 3 initial tests
for each primer in temp_primers_library:

```
    rev_complement = get_reverse_complement (primer)
```

```
    for each pattern of primer:
```

```
        if rev_complement contains pattern:  
            continue #primer failed
```

```
    for each pattern of primer:
```

```
        patterns.append(pattern)
```

Our Solution – Deep Dive

More DNA Restrictions (b):

Enhanced Long Inter-Primer Complementarity Checks:

- Store fixed-length reverse complement patterns of all primers in a set.
- **Efficiency:** Sets offer average $O(1)$ time complexity for lookups.
- **Improvement:** Compared to traditional $O(n)$ pairwise comparison, this method is significantly faster.
- **Scalability:** Quickly verify if a new primer's reverse complement matches any stored pattern.
- **Advantage:** Scales efficiently with larger primer libraries, enabling faster and more effective validation.



Our Solution – Deep Dive

More DNA Restrictions (b):

Hamming Distance:

Ensures that each primer is sufficiently different from the others in the library to risk of binding to the wrong targets to avoid decoding errors.

The hamming distance between two primers:

For two primers a and b:

If $a[i] \neq b[i]$ distance+1
distance++

For example:

For a primer of length 20 and Hamming Distance ≥ 6

and contain $P = \text{'ATCGTACGGTACGTTAGCGA'}$ in the library:

- $\text{'ATCGTACGGTACGTTAGCGA'}$ is disqualified (Hamming Distance from $P = \text{'ATCGTACGGTACGTTAGCGA'}$ is 1).
- $\text{'GTCATGCTGGGCGTTAACCA'}$ meets the condition (Hamming Distance from $P = \text{'ATCGTACGGTACGTTAGCGA'}$ is 8).


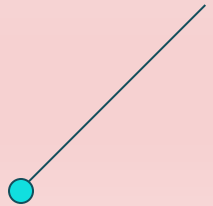



Our Solution – Deep Dive

More DNA Restrictions (b):

Efficient Hamming Distance Validation Using a Trie-Tree:



- **Method:** Utilize a Trie-Tree to validate primers against Hamming distance thresholds.
 - **Traversal:** Systematic character-by-character comparison with mismatch accumulation.
 - **Efficiency:** Reduces unnecessary comparisons compared to traditional linear checks.
 - **Advantage:** Quickly discards primers exceeding mismatch limits.
 - **Scalability:** Ensures faster validation, especially with larger primer libraries.
- 
- 
- 

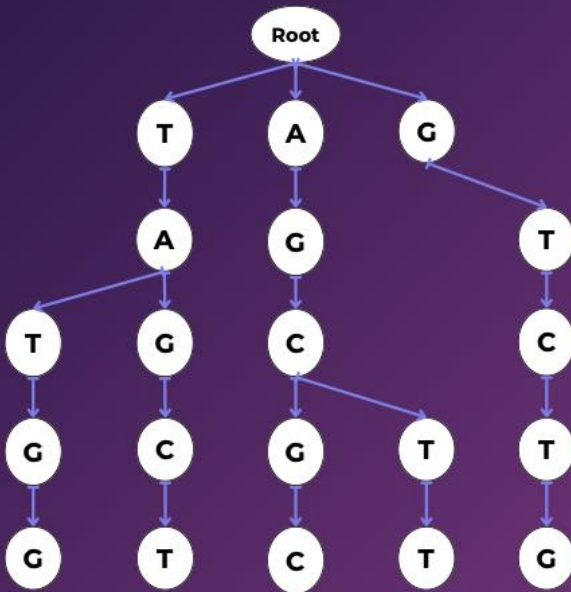
Hamming Distance - Trie-Tree Example

Min Hamming Distance - 3

Primer to add

AGCTA

every path from root to a leaf
is a primer in the final set
5 valid primers = 5 leaves



Primers already in the final set

AGCGC
AGCTT
TAGCT
TATGG
GTCTG

Green Node = min ham-distance found

The Trie-Tree enables efficient primer comparison by traversing character by character and counting mismatches. This approach reduces unnecessary checks and quickly eliminates primers exceeding the mismatch limit, making validation faster, especially for large primer libraries.

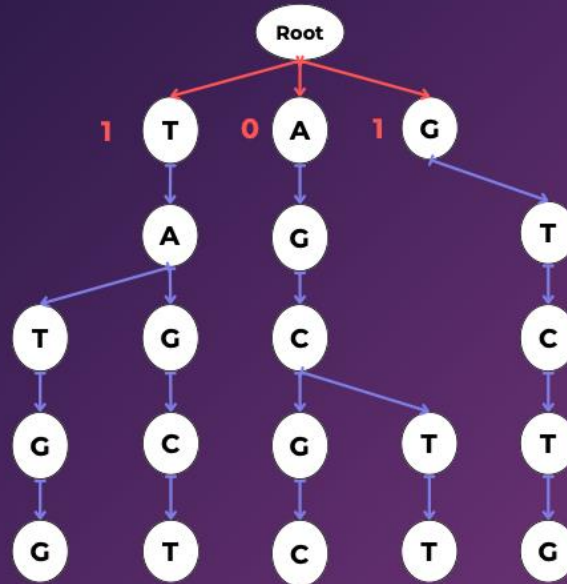
Hamming Distance - Trie-Tree Example

Min Hamming Distance - 3

Primer to add

AGCTA

Mismatches



Primers already in the final set

AGCGC
AGCTT
TAGCT
TATGG
GTCTG

every path from root to a leaf
is a primer in the final set
5 valid primers = 5 leaves

Green Node = min ham-distance found

keep track on mismatches between all primers in the set and the new primer we intend on adding
while searching in the tree

Hamming Distance - Trie-Tree Example

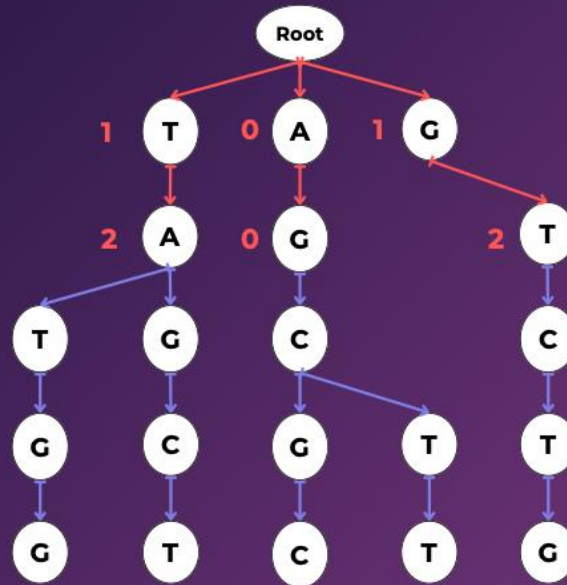
Min Hamming Distance - 3

Primer to add

AGCTA

Mismatches

Mismatches



Primers already in the final set

AGCGC
AGCTT
TAGCT
TATGG
GTCTG

every path from root to a leaf
is a primer in the final set
5 valid primers = 5 leaves

Green Node = min ham-distance found

Primers in the final set that share the same prefix will have the same path until the different BP
between them

Hamming Distance - Trie-Tree Example

Min Hamming Distance - 3

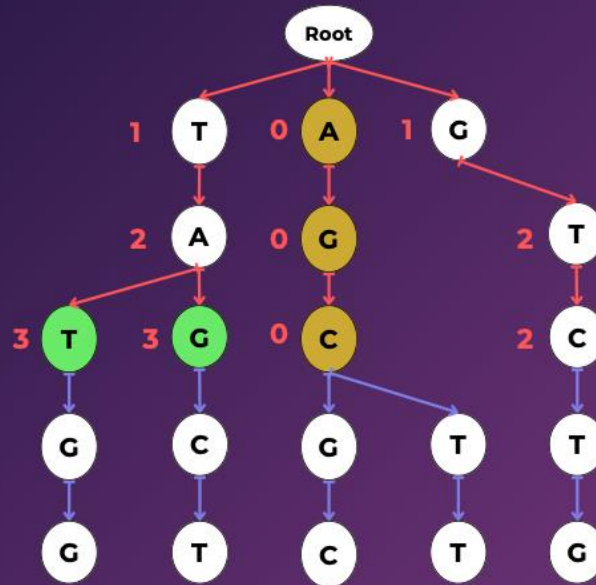
Primer to add

AGCTA

Mismatches

Mismatches

Mismatches



Primers already in the final set

AGCGC
AGCTT
TAGCT
TATGG
GTCTG

every path from root to a leaf
is a primer in the final set
5 valid primers = 5 leaves

Disqualify!!!

Green Node = min ham-distance found

found primer (AGCGC,AGCTT) with hamming distance < 6
new primer AGCTA disqualify

Inter Complementarity - Sliding Window Example

primer - AGCTTCGATTGCGTA

primer reverse complement - TACGCAATCGAAGCT

T A C G C A A T C G A A G C T



*if pattern already in the patterns set:
disqualify
else:
continue

check if all patterns **not** in the set, if so
in the final library we have zero presence of each one of the patterns of the reverse complement primer
and therefore passed this phase

Inter Complementarity - Sliding Window Example

primer - AGCTTCGATTGCGTA

primer reverse complement - TACGCAATCGAAGCT

T **A C G C A** A T C G A A G C T



*if pattern already in the patterns set:
disqualify
else:
continue

check if all patterns **not** in the set, if so
in the final library we have zero presence of each one of the patterns of the reverse complement primer
and therefore passed this phase

Inter Complementarity - Sliding Window Example

primer - AGCTTCGATTGCGTA

primer reverse complement - TACGCAATCGAAGCT

T A C G C A A T C G A A G C T



*if pattern already in the patterns set:
disqualify
else:
continue

check if all patterns **not** in the set, if so
in the final library we have zero presence of each one of the patterns of the reverse complement primer
and therefore passed this phase

Inter Complementarity - Sliding Window Example

primer - AGCTTCGATTGCGTA

primer reverse complement - TACGCAATCGAAGCT

T A C G C A A T C G A A G C T

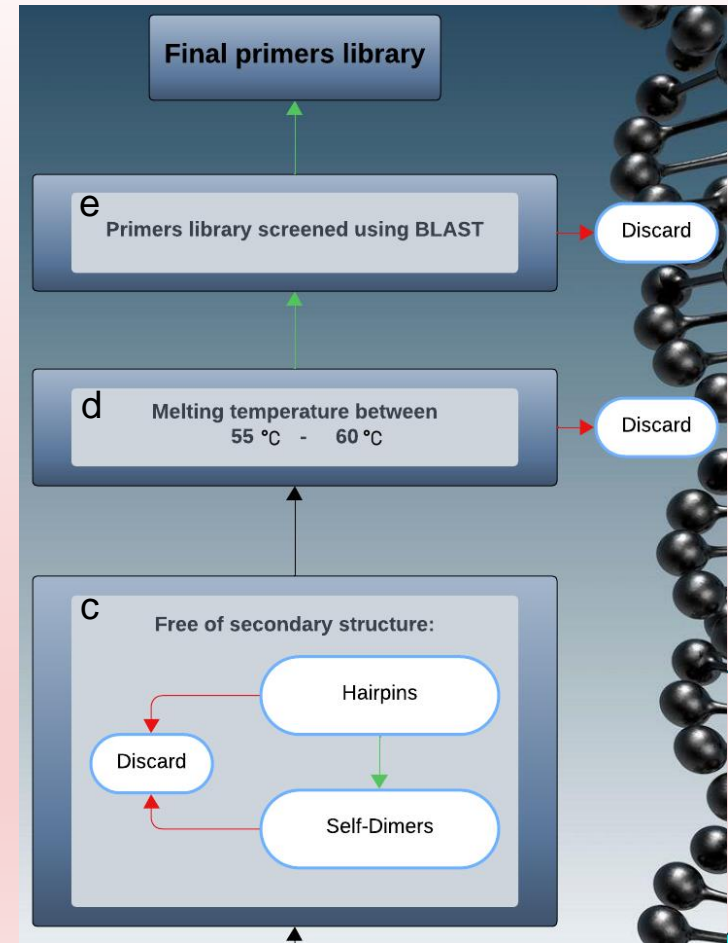
↙
*if pattern already in the patterns set:
disqualify
else:
continue

check if all patterns **not** in the set, if so
in the final library we have zero presence of each one of the patterns of the reverse complement primer
and therefore passed this phase

Our Solution - Deep Dive

More DNA Restrictions (c,d,e):

- Multiple filtering criteria were applied to ensure primer set quality.
- **Searched existing software** to handle the last 4 biological restrictions:
- **Secondary Structure (c):** Performed with NUPACK. NUPACK was also referenced in the article.
- **Melting Temperature (d):** Utilized Primer3.
- **Sequence Similarity Filtering (e):** Used CD-HIT for efficient screening, replacing BLAST.





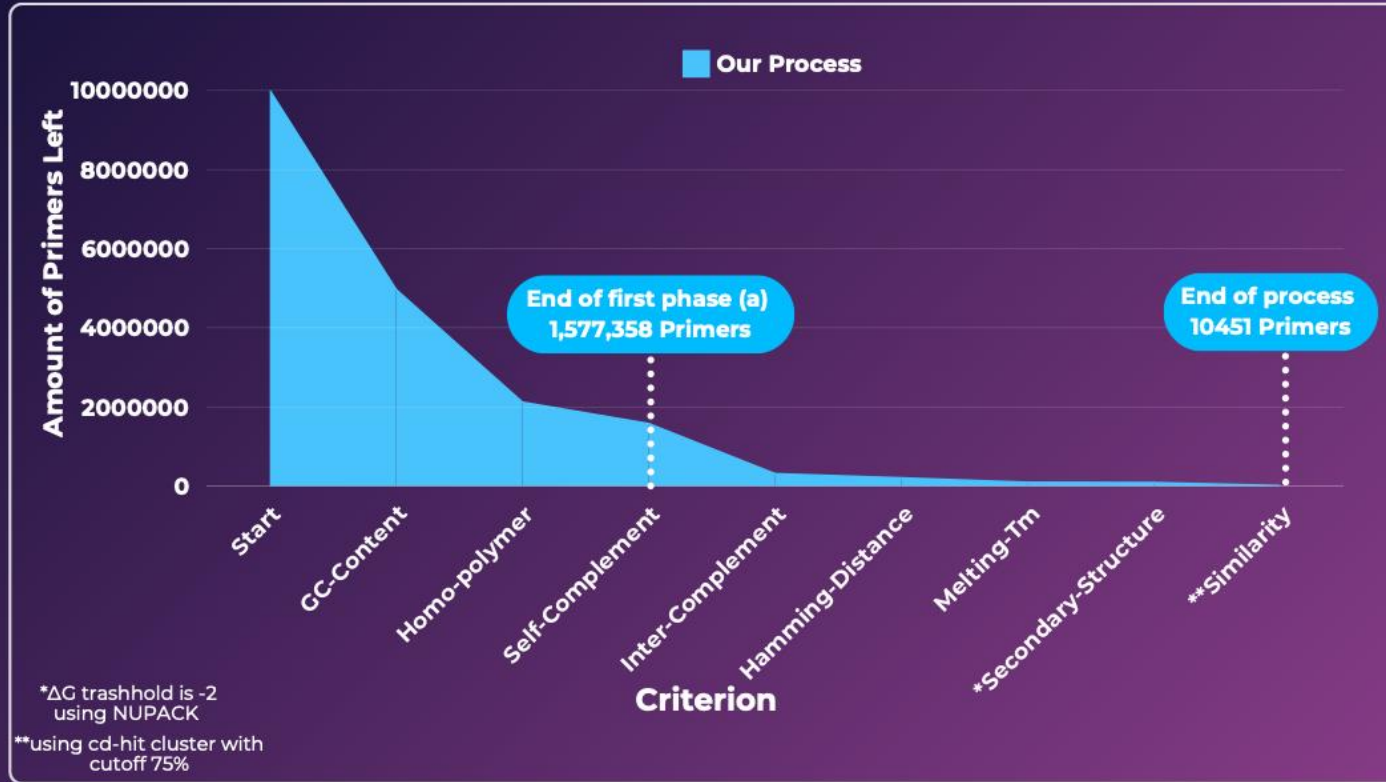
5

Our Findings



Primer Design and Filtration Pipeline

runtime - 54 minutes on laptop

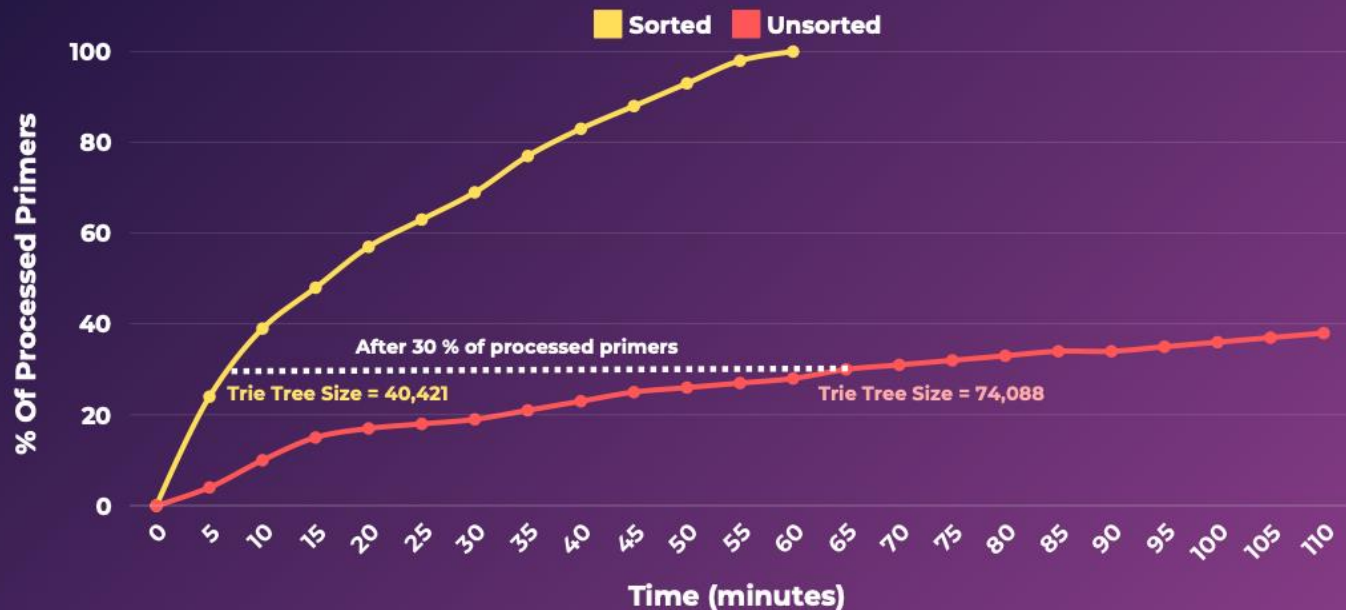


The minimum free energy (ΔG) threshold of -2, using NUPACK, filters out primers with unstable **secondary structures**, while CD-HIT clusters primers with over 75% **similarity** to reduce redundancy.

Sorted/Unsorted Primers Approach Effect On Trie-Tree

10 Million Random Primers
after intra-primer filters - 1,578,028 primers
rate of filtered invalids

Trie Tree Size = Primer Library Size
(no melting tm, similarity, secondary structure)



**both runs executed on laptop

When Sorted Trie-Tree Eliminate Similar Primers Faster

Profiling Of The Algorithms

- Our Algorithm -



6m 45s

7832

- Naive Algorithm -



45m 37s

7832

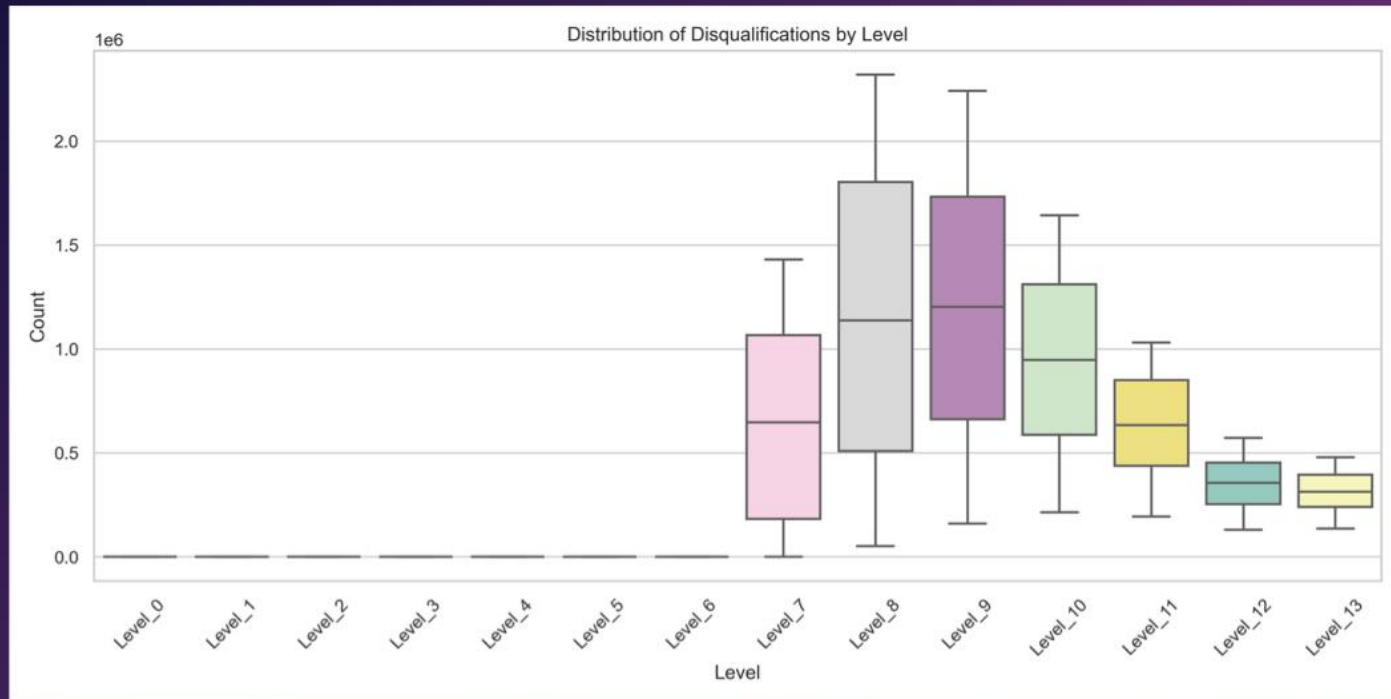
- Runtime -

- Primers -

			Homo-Polymer		Secondary Structure		GC Content
	Hamming-Distance		Self-Complement		Melting Tm		Inter-Complement

*timed for 1M random primers, as the amount of primers increase the hamming-distance share will get bigger and bigger (and even more drastically in the naive algorithm)

Primer Disqualifqation In Trie Tree By Level



14bp primers in 10 shuffled runs (4^{14} primers each)

Final Number Of Primers As Function Of N

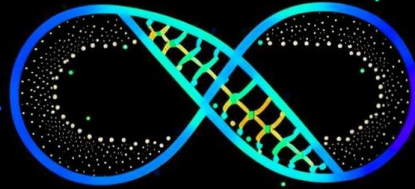


N	1 million	5 million	10 million
runtime	9 minutes	31 minutes	54 minutes
primer library size *75% cutoff	7832	9798	10451
primer library size *80% cutoff	8006	33,883	47,522

*cd-hit similarity cutoff percentage

Thank You!

DNA-Storage



Research Resources and Infrastructure:

- Michael Factor's code -
<https://colab.research.google.com/drive/1jbzcPQjKqABFnYJOv0SMsw8uANX7K6fl?usp=sharing>
- An Introduction to DNA Data Storage-
<https://dnastoragealliance.org/dev/publications/>
- Nature Biotechnology - https://static-content.springer.com/esm/art%3A10.1038%2Fnbt.4079/MediaObjects/41587_2018_BFnbt4079_MOESM4_ESM.pdf