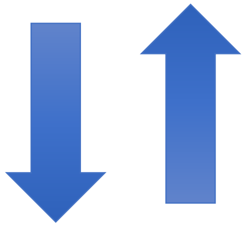
What would “`Hello World`” look like if stored in in a DNA “partition”?

Djordje Jevdjic

April 14th , 2021

DNA-Based Data Storage

Nature's way: storing information in the DNA format



01001010101011011101

DNA Storage "Drive"

Why DNA?

1. Incredible density
   - 6-7 orders of magnitude ahead of best alternatives!

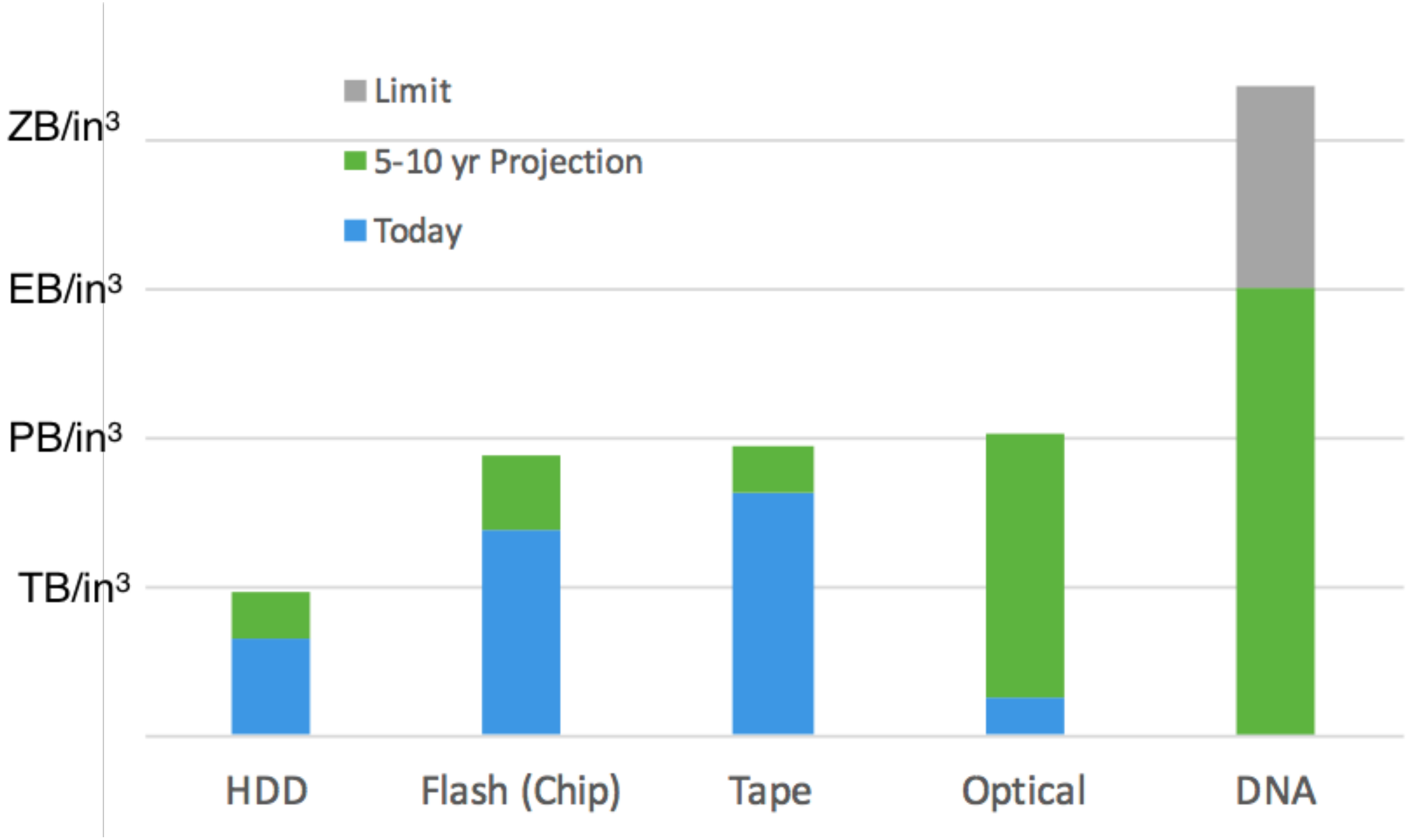
2. Unmatched durability
   - Thousands/millions of years (5 years for disks/flash)

3. Convenient for many data-parallel computations

4. Efficient random access (constant latency)

5. Never obsolete (read/write interfaces are eternal!)

Storage Density Projections*



*Credit: Luis Ceze & Karin Strauss

Why not DNA?

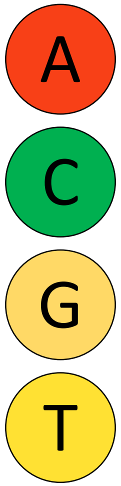
1. Prohibitive cost (but improving rapidly)
   - Write cost: $1000/MiB
   - Read cost: $10-$1000/MiB

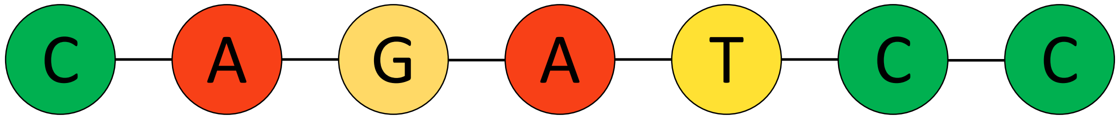
2. Access time in hours (milliseconds for disk)
   - OK for archival storage

3. Extremely error-prone
   - Especially with new, cheaper reading/writing technologies
   - Errors are nothing like we know to deal with

DNA Molecules

4 nucleotides:

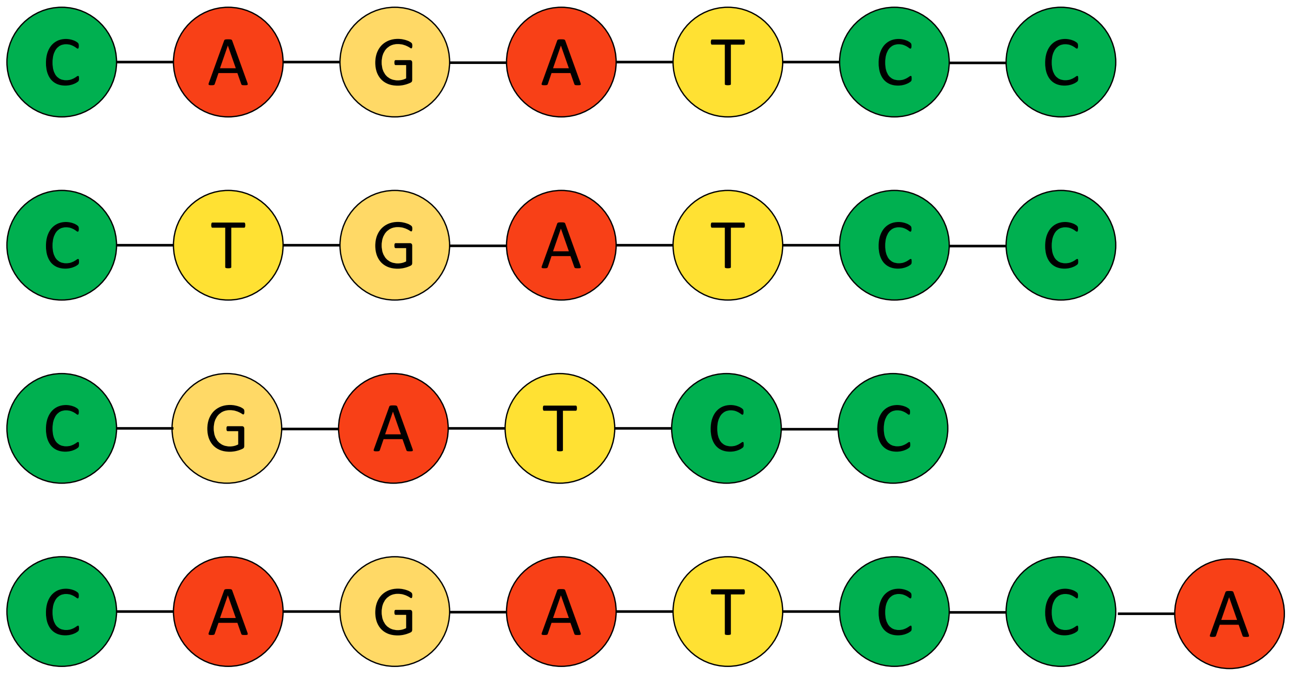




Synthetic DNA molecule (strand): a linear sequence of nucleotides created artificially (no biological meaning)

up to 2 bits of information per nucleotide

Errors in DNA storage



Original

Substitution (A→T)

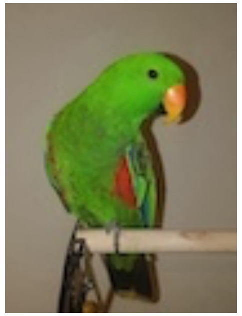
Deletion (of A)

Insertion (of A)

Def. Edit distance = minimum number of single-character operations (***substitution, deletion, insertion***) needed to convert one string into another

DNA storage

write process:

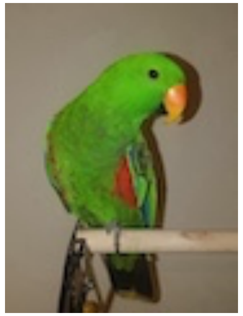


1011101… **encoding** → ACGTCAG… **synthesis** →

read process:

selecting →

sequencing → ACGTCAG… **decoding** → 1011101…



Encoding

01001000110101…

↓

CAGATCC…

encoding:

00 → A

01 → C

10 → G

11 → T

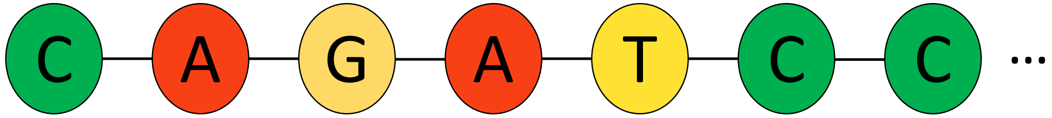
Synthesis (write)

Manufacturing artificial DNA strands.

many copies of each:

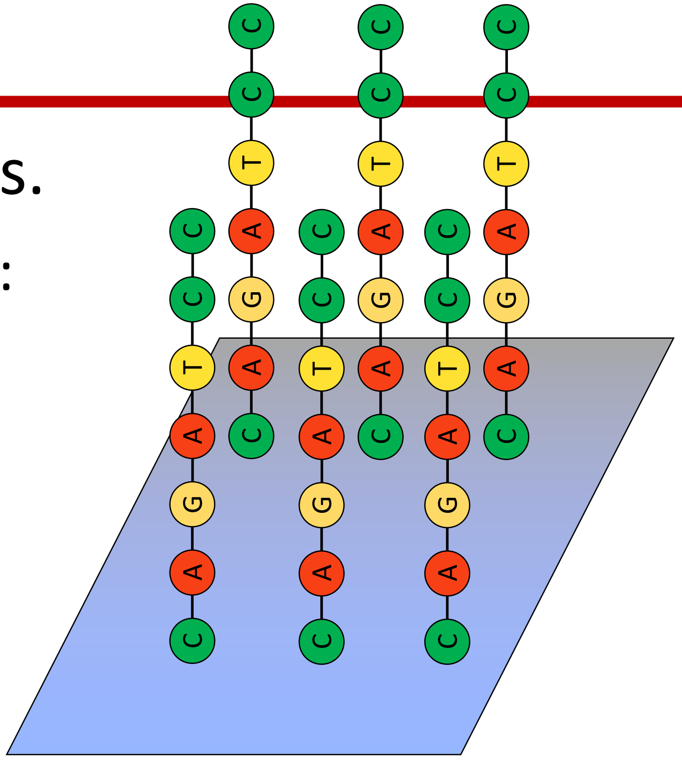
CAGATCC…

synthesis



99% 98% 97% 96.1% 95.1% 94.2%. …

P(the 100th nucleotide is correct) = 36%



Assume probability of attachment = 99%

Synthetic DNA molecules limited in length → chunk them up!

Breaking up molecules

01001000110101110111 0

encode

CAGATCCCGGGATAGCTACCA

break up

1. CAGATC
2. CCGGGA
3. TAGCTA
4. CCAATT

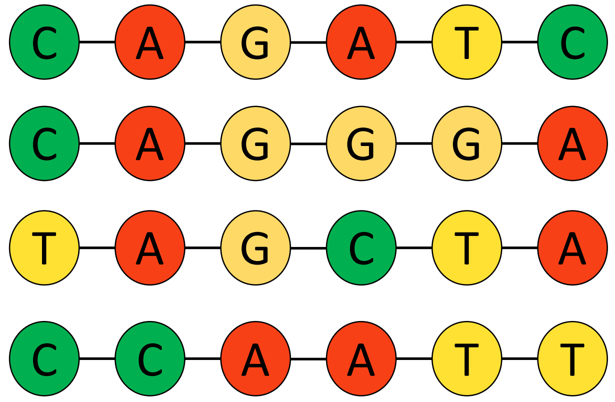
synthesize

Problem?

Ordering lost!

store

1.
2.
3.
4.



Encoding with Ordering

010010001101011101110

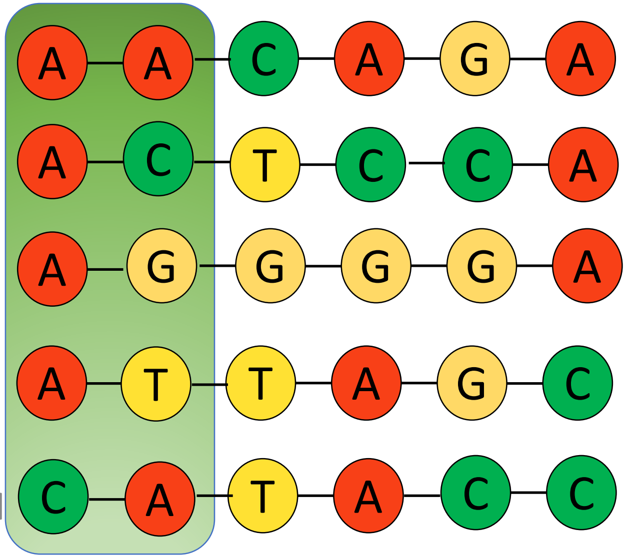
encode

CAGATCCCGGGATAGCTACC

break up

AACAGA
ACTCCA
AGGGGA
ATTAGC
CATACC

synthesize



store

Random Access

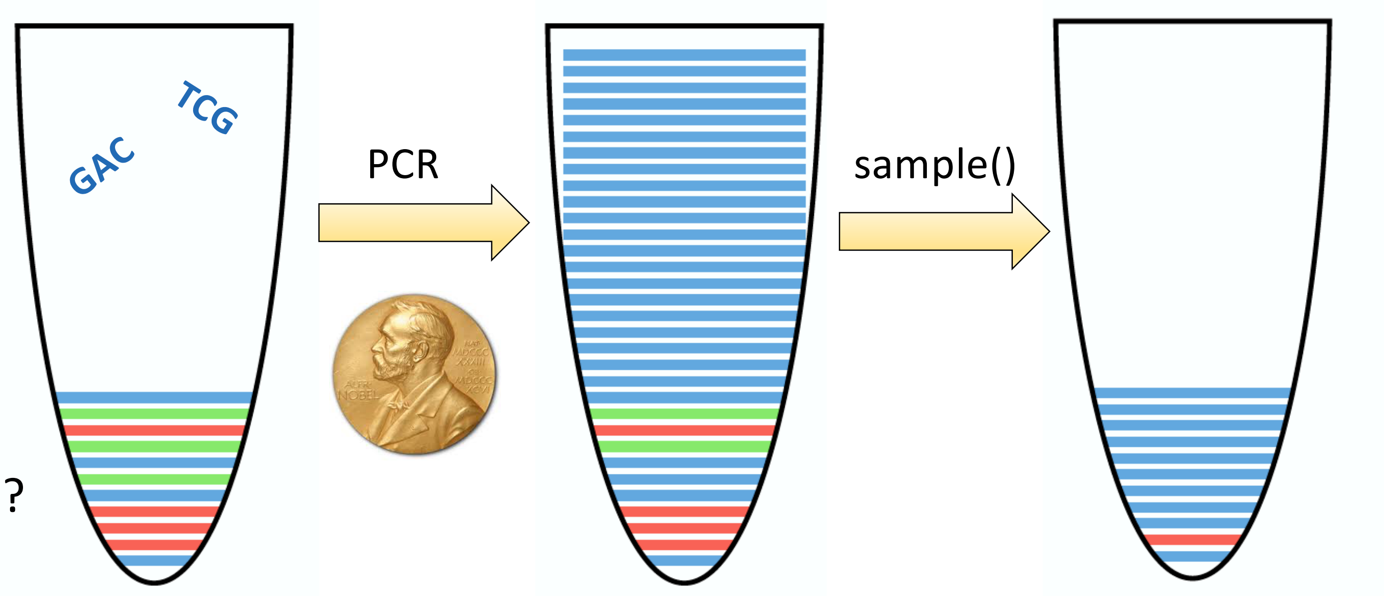
primers

GAC ACGAGGATTCAACC TCG
GAC ACCGAGGATTCAAC TCG
GAC CACACGGGGCCTTA TCG
GAC AAATCGGTTACCGG TCG
GAC TACCATGACGAAGC TCG
GAC GATTCAACACGAGT TCG
file #1

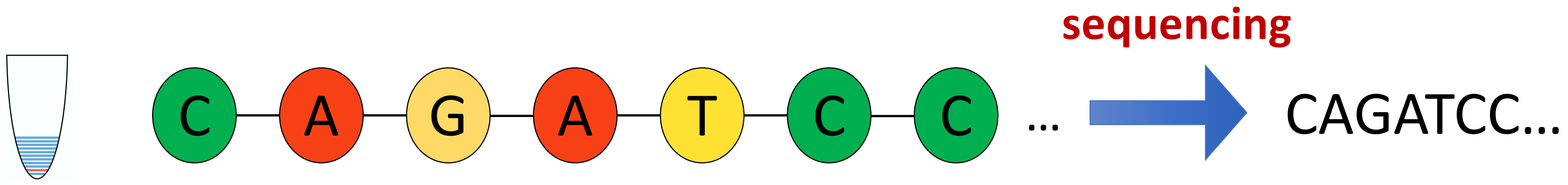
CTT GACCAGGATTCGT AGG
CTT CGATTCGATCGAC AGG
CTT TGATCGATCGAGC AGG
file #2

TAC AGCTTCGATTCGG GTA
TAC ATCGATCGTGCTA GTA
TAC CGTAATCGGACTC GTA
TAC GATCGGCTATTCC GTA
file #3

What if primers attach to some data?



Sequencing (reading)



Produces many (buggy) copies of each molecule:

synthesized	*sequenced*
CAGATCC	CAGATCC
	CAGATC
	AAGATCCA
	AGATTCC
	CAGGATCC

Decoding DNA

ACTTCCA	**AGGCGA**	**AGCTCCA**
TTAGC	**ATTAC**	**GGGGA**
CATACCG	**GAGGGGA**	**CATTAGC**
AACGA	**CATACCT**	**CAGACC**
AATAGA	**AACTGA**	**CCCA**
ACTCCCA	**AATCCA**	**TTAGC**
AGGGA	**TACAGA**	**AGGGA**
ATTAGC	**GGGGA**	**GATACC**
AACGA	**ATCTAGC**	**ATTAGCA**
CAGTACC	**CAACC**	**ACTCCA**
CAGA	**ACAGA**	**CGGGGA**
AGTCCA	**CATAC**	**CAGACCG**

Step 1: Clustering

ACTTCCA	AGGCGA	AGCTCCA
TTAGC	ATTAC	GGGGA
CATACCG	GAGGGGA	CATTAGC
AACGA	CATACCT	CAGACC
AATAGA	AACTGA	CCCA
ACTCCCA	AATCCA	TTAGC
AGGGA	TACAGA	AGGGA
ATTAGC	GGGGA	GATACC
AACGA	ATCTAGC	ATTAGCA
CAGTACC	CAACC	ACTCCA
CAGA	ACAGA	CGGGGA
AGTCCA	CATAC	CAGACCG

Step 1: Clustering

AGTCCA	CAGTACC	TTAGC	AACGA	AGGGA
ACTTCCA	CATACCT	CATTAGC	AATAGA	GGGGA
ACTCCCA	CAACC	ATCTAGC	AACGA	AGGCGA
AATCCA	CATAC	ATTAGC	CAGA	GAGGGGA
CCCA	CATACCG	ATTAGCA	AACTGA	GGGGA
ACTCCA	CAGACC	TTAGC	TACAGA	AGGGA
AGCTCCA	GATACC	ATTAC	ACAGA	CGGGGA
	CAGACCG			

Step 2: Finding Consensus per Cluster

AGTCCA	CAGTACC	TTAGC	AACGA	AGGGA
ACTTCCA	CATACCT	CATTAGC	AATAGA	GGGGA
ACTCCCA	CAACC	ATCTAGC	AACGA	AGGCGA
AATCCA	CATAC	ATTAGC	CAGA	GAGGGGA
CCCA	CATACCG	ATTAGCA	AACTGA	GGGGA
ACTCCA	CAGACC	TTAGC	TACAGA	AGGGA
AGCTCCA	GATACC	ATTAC	ACAGA	CGGGGA
	CAGACCG			
↓	↓	↓	↓	↓
ACTCCA	CATACC	AGGGGA	AACAGA	ATTAGC

Step 3: Reordering (assembly)

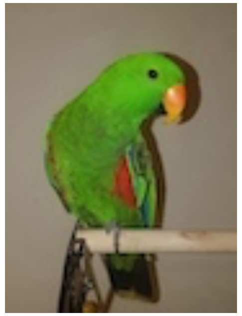
AGTCCA	CAGTACC	TTAGC	AACGA	AGGGA
ACTTCCA	CATACCT	CATTAGC	AATAGA	GGGGA
ACTCCCA	CAACC	ATCTAGC	AACGA	AGGCGA
AATCCA	CATAC	ATTAGC	CAGA	GAGGGA
CCCA	CATACCG	ATTAGCA	AACTGA	GGGGA
ACTCCA	CAGACC	TTAGC	TACAGA	AGGGA
AGCTCCA	GATACC	ATTAC	ACAGA	CGGGGA
	CAGACCG			
↓	↓	↓	↓	↓
ACTCCA	CATACC	AGGGGA	AACAGA	ATTAGC

CAGATCCAGGGATAGCTACC

Step 4: Error Correction and Decoding

AGTCCA	CAGTACC	TTAGC	AACGA	AGGGA
ACTTCCA	CATACCT	CATTAGC	AATAGA	GGGGA
ACTCCCA	CAACC	ATCTAGC	AACGA	AGGCGA
AATCCA	CATAC	ATTAGC	CAGA	GAGGGGA
CCCA	CATACCG	ATTAGCA	AACTGA	GGGGA
ACTCCA	CAGACC	TTAGC	TACAGA	AGGGA
AGCTCCA	GATACC	ATTAC	ACAGA	CGGGGA
	CAGACCG			
ACTCCA	CATACC	AGGGGA	AACAGA	ATTAGC

CAGATCCAGGGATAGCTACC → 01001000111010…



Summary

Basic steps in a DNA-storage pipeline:

- Chunking up data
- Encoding (binary to DNA strings)
- Synthesis of molecules (from DNA strings)
- Random access (PCR)
- Sequencing
- Clustering
- Consensus Finding
- Reordering (assembly)
- Decoding
- Error detection and correction…

wetlab