Special course in Computer Science: Molecular Computing

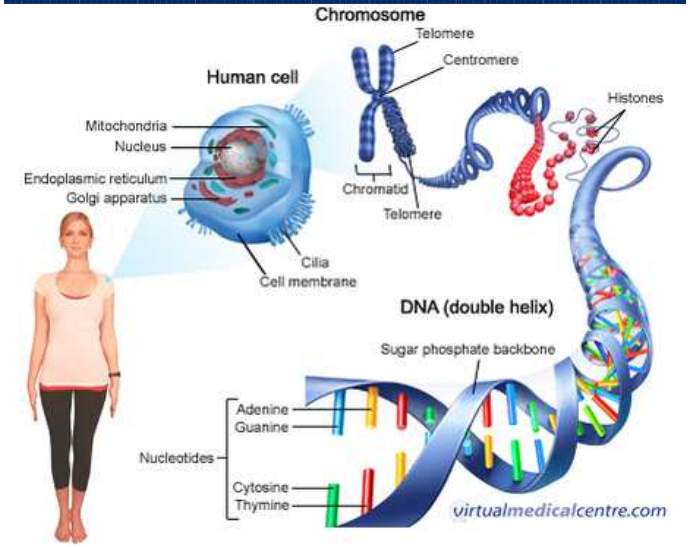
Lecture 3: Introduction to DNA Computing

Vladimir Rogojin

Department of CS, Abo Akademi

http://combio.abo.fi/teaching/special-course-in-computer-science-molecular-computing/

Fall 2015



forensics-science.weebly.com

DNA

DNA

- Base element of **Molecular Computing**

Importance of DNA

- Responsible for **inheritance**: encodes all genetic information of an organism
- Responsible for life processes: encodes all "**instructions**" needed for the functioning of the organism
- The most fundamental **mechanisms of DNA manipulation are the same** in all living organisms

www.javacr.com

DNA computing

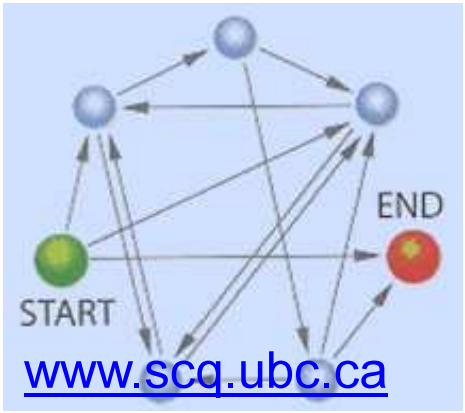
- Basic idea:
  - Molecular biology processes can be used to perform arithmetic and logic operations on information encoded as DNA strands
- Principle:
  - DNA could be synthesized, copied, cut/fused, filtered, measured, sequenced
  - Due to the complementarity principle it could form complex “programmable” nano-structures

DNA computing

- We will study today:
  - 2 historically important DNA experiments:
    - First experimental demonstration of DNA computing – Adleman's celebrated experiment, searching for HPP in directed graph with DNA
    - First experiment demonstrating the potential of molecular computing to outperform unaided human computational ability
  - Influence of DNA-based computation on other areas of computer science:
    - Formal language theory, coding theory,

"IT'S THE LATEST THING: a biological computer!"

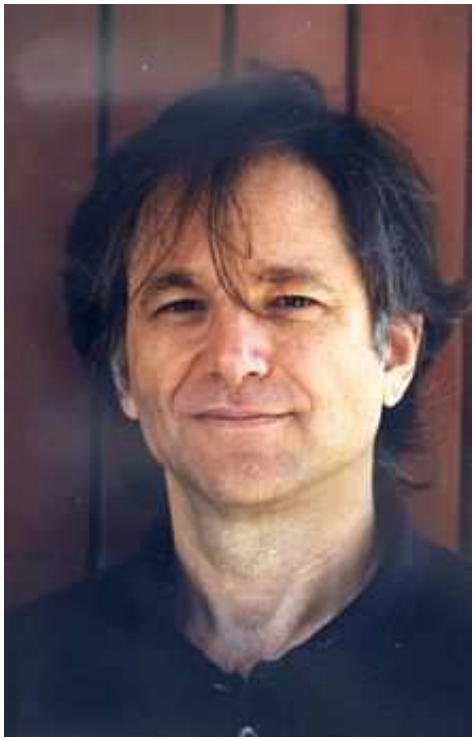
www.columbia.edu



www.scq.ubc.ca

First DNA computing experiment

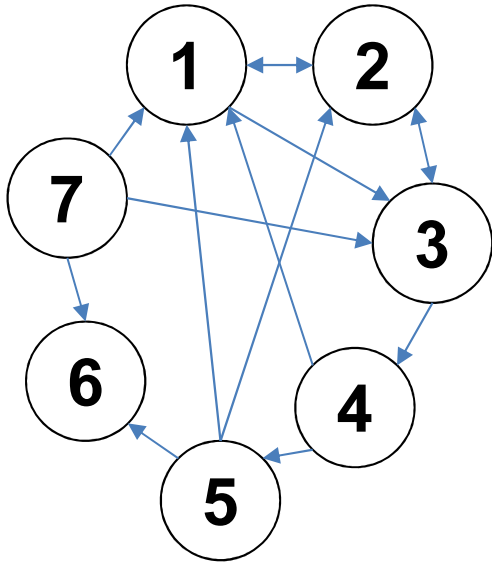
- Performed by Leonard Adleman in 1994, MIT
- Problem:
  - An instance of HPP with 7 nodes
- Hamiltonian path problem (HPP):
  - Search for a path (starting with some v_{start} and ending with some v_{end}) that contains all the vertices from the graph repeating exactly once
  - Known NP-complete problem



en.wikipedia.org

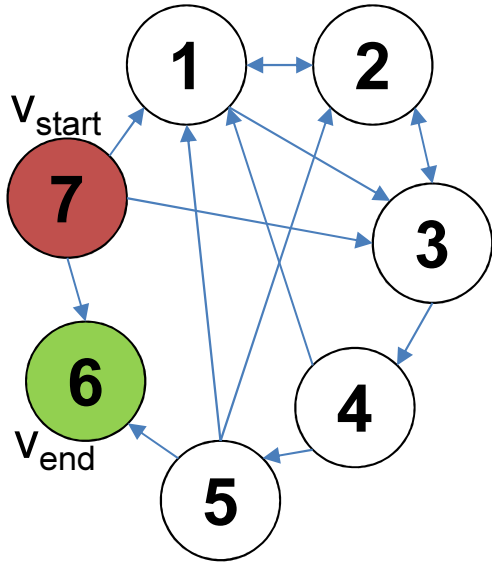
Solving HPP with DNA computing

- The algorithm:
  - *Input:* A directed graph *G* with *n* vertices and designated vertices v_{start} and v_{end}



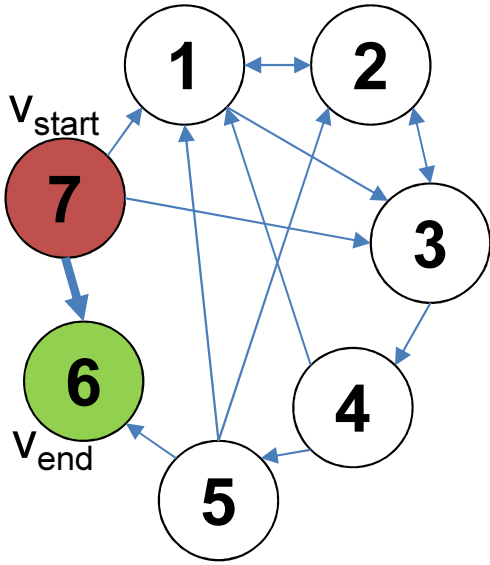
Solving HPP with DNA computing

- The algorithm:
  - *Input:* A directed graph *G* with *n* vertices and designated vertices v_{start} and v_{end}
  - *Step1:*



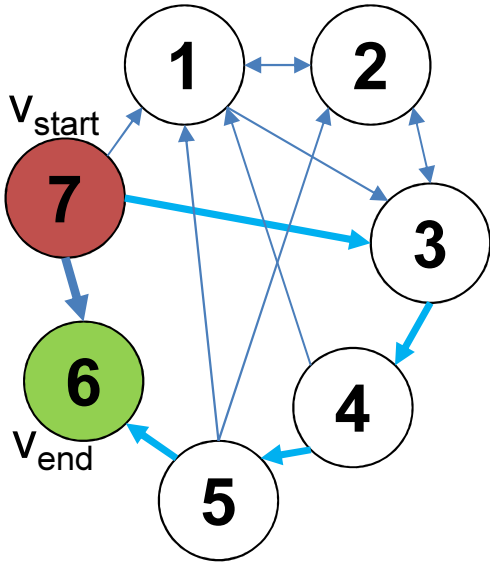
Solving HPP with DNA computing

- The algorithm:
  - *Input:* A directed graph *G* with *n* vertices and designated vertices v_{start} and v_{end}
  - *Step1:* Generate random paths through the graph
  - *Step2:* Keep only those paths that begin with v_{start} and end with v_{end}



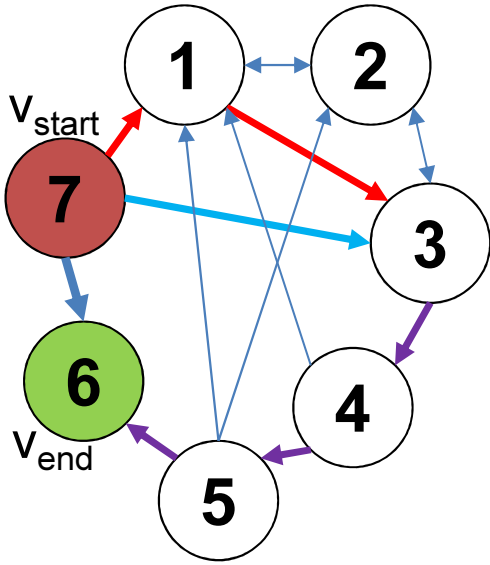
Solving HPP with DNA computing

- The algorithm:
  - *Input:* A directed graph *G* with *n* vertices and designated vertices v_{start} and v_{end}
  - *Step1:* Generate random paths through the graph
  - *Step2:* Keep only those paths that begin with v_{start} and end with v_{end}



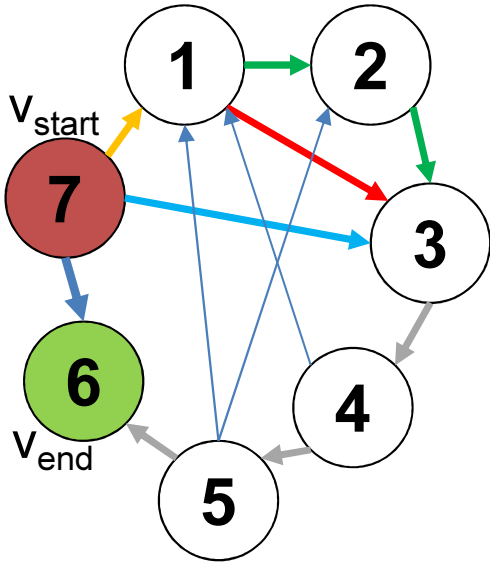
Solving HPP with DNA computing

- The algorithm:
  - *Input:* A directed graph *G* with *n* vertices and designated vertices v_{start} and v_{end}
  - *Step1:* Generate random paths through the graph
  - *Step2:* Keep only those paths that begin with v_{start} and end with v_{end}



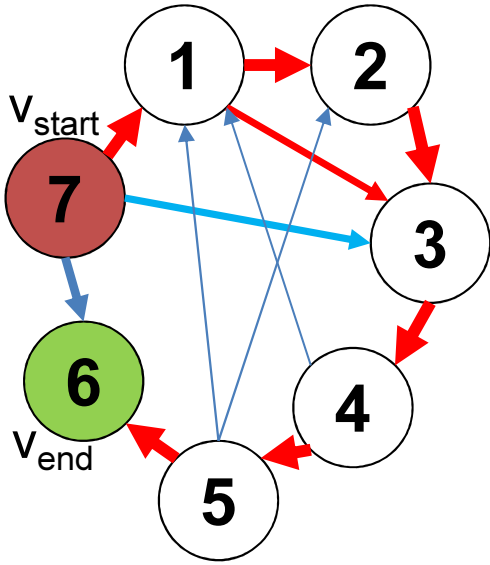
Solving HPP with DNA computing

- The algorithm:
  - *Input:* A directed graph *G* with *n* vertices and designated vertices v_{start} and v_{end}
  - *Step1:* Generate random paths through the graph
  - *Step2:* Keep only those paths that begin with v_{start} and end with v_{end}



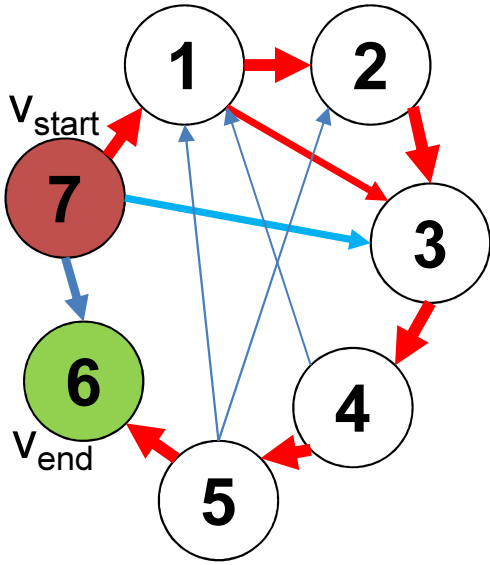
Solving HPP with DNA computing

- The algorithm:
  - *Input:* A directed graph *G* with *n* vertices and designated vertices v_{start} and v_{end}
  - *Step1:* Generate random paths through the graph
  - *Step2:* Keep only those paths that begin with v_{start} and end with v_{end}
  - *Step3:* Keep only those paths that enter exactly *n* vertices
  - *Step4:*



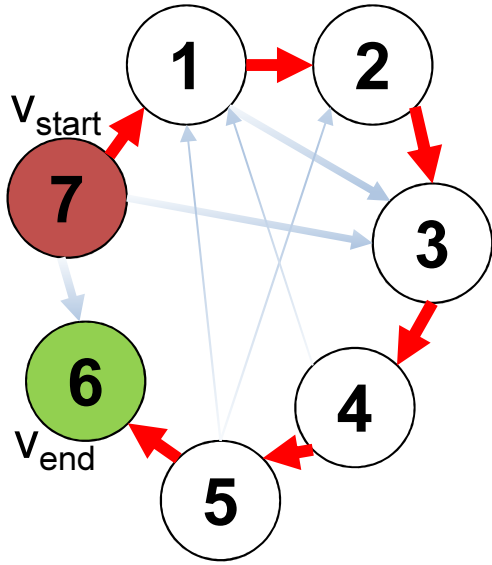
Solving HPP with DNA computing

- The algorithm:
  - *Input:* A directed graph *G* with *n* vertices and designated vertices v_{start} and v_{end}
  - *Step1:* Generate random paths through the graph
  - *Step2:* Keep only those paths that begin with v_{start} and end with v_{end}
  - *Step3:* Keep only those paths that enter exactly *n* vertices
  - *Step4:* Keep only those paths that enter all of the vertices of the graph at least once



Solving HPP with DNA computing

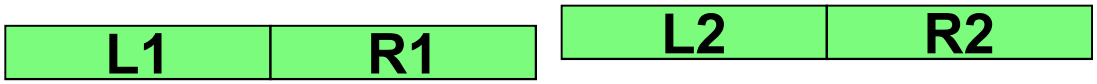
- The algorithm:
  - *Input:* A directed graph *G* with *n* vertices and designated vertices v_{start} and v_{end}
  - *Step1:* Generate random paths through the graph
  - *Step2:* Keep only those paths that begin with v_{start} and end with v_{end}
  - *Step3:* Keep only those paths that enter exactly *n* vertices
  - *Step4:* Keep only those paths that enter all of the vertices of the graph at least once
  - *Output:* If any paths remain, output "YES"; otherwise output "NO"

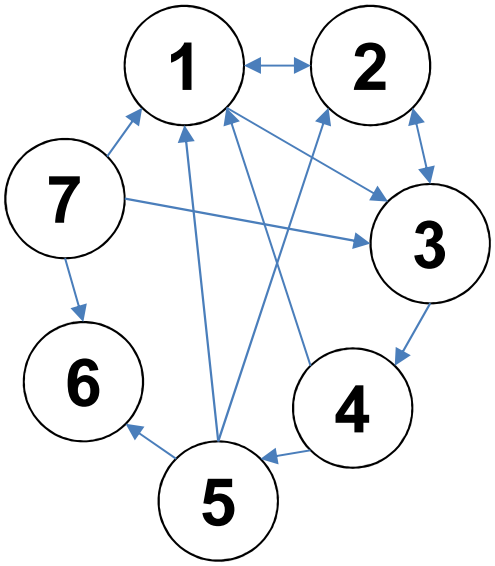


YES!

Solving HPP with DNA computing

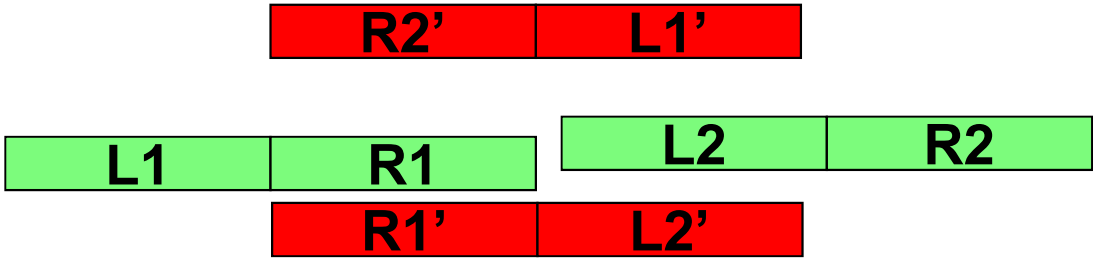
- Bio-algorithm:
  - Encoding the input:
    - Vertex → 20-mer single strand of

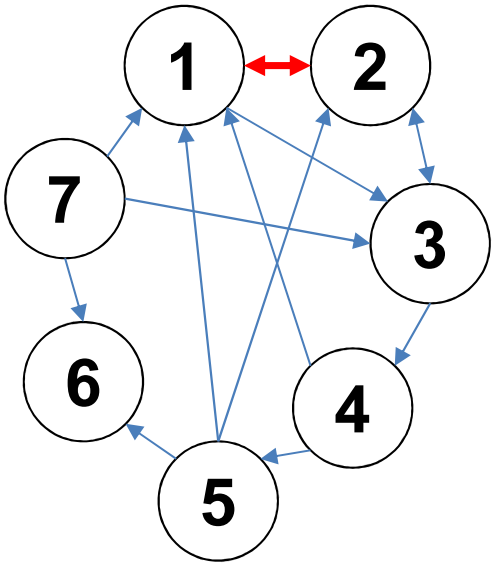




Solving HPP with DNA computing

- Bio-algorithm:
  - Encoding the input:
    - Vertex → 20-mer single strand of DNA
    - Directed edge →left half of the edge encoding is complementary to the right half of the source vertex encoding, and the right half of the edge encoding is complementary to the left half of the destination vertex encoding



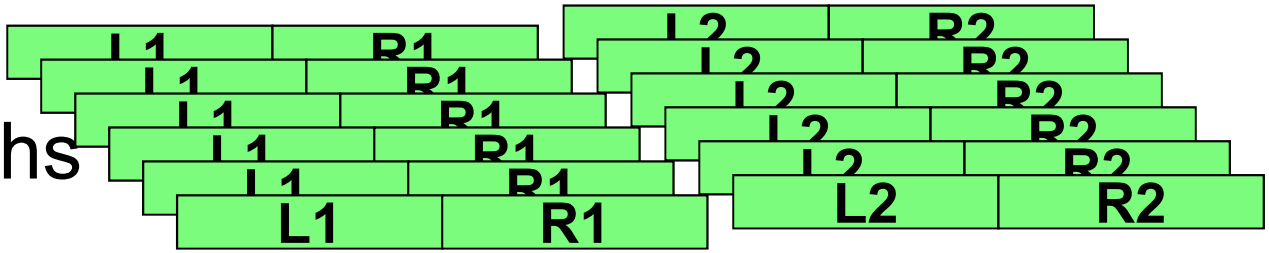


Solving HPP with DNA computing

- Example:
  - DNA sequences for the vertex 1 and the directed edges 1 → 2 and 2 → 3 encoded as:
    - O_2 = 5'-GCTATTCGAGCTTAAAGCTA-3'
    - $O_{1\rightarrow2}$ = 3'-GTATATCCGACGATAAGCTC-5'
    - $O_{2\rightarrow3}$ = 3'-GAATTTCGATGGCTAGGTAC-5'

Solving HPP with DNA computing

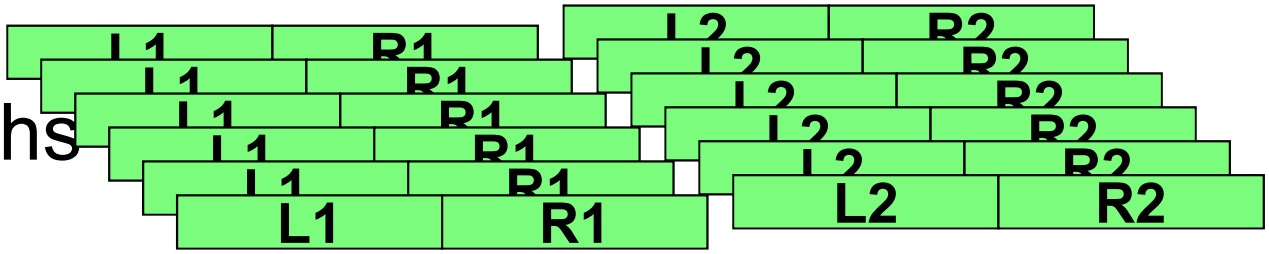
- Implementing *Step1* (Generate random paths through the graph):



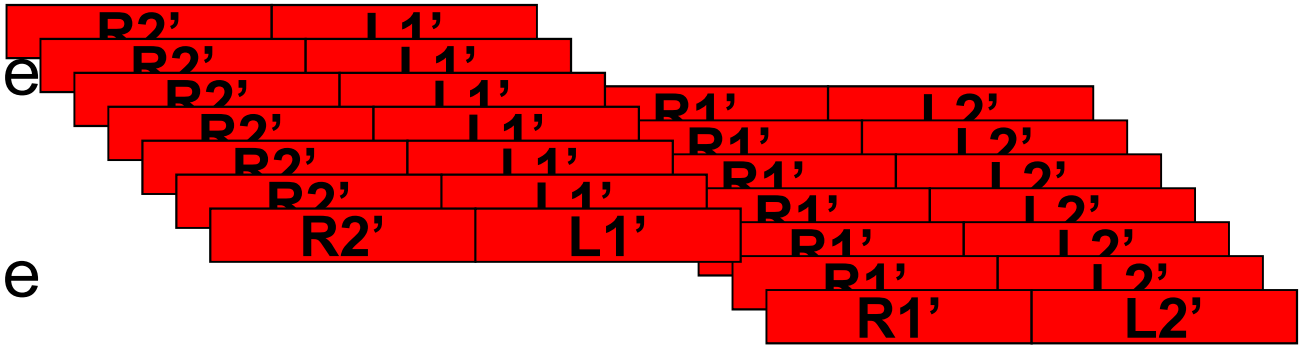
  - Amplify with PCR single strands O_i

Solving HPP with DNA computing

- Implementing *Step1* (Generate random paths through the graph):

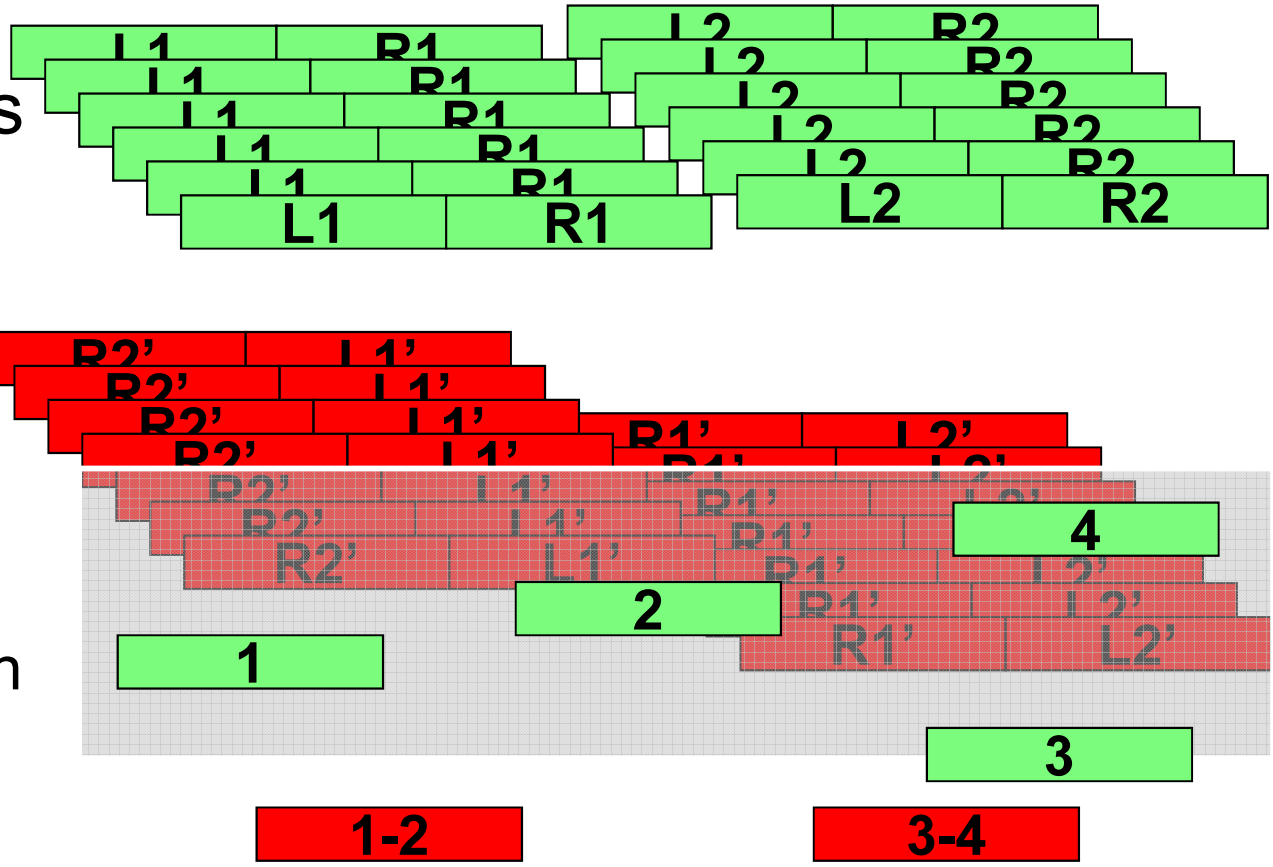


  - Amplify with PCR single strands O_i
  - Amplify with PCR single strands $O_{i \to j}$



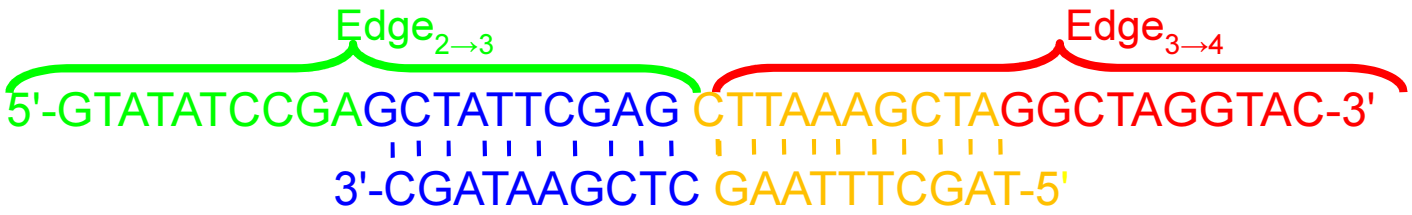
Solving HPP with DNA computing

- Implementing *Step1* (Generate random paths through the graph):
  - Amplify with PCR single strands O_i
  - Amplify with PCR single strands $O_{i \to j}$
  - Hybridize copies of O_i with copies of $O_{j \to i}$ and $O_{i \to k}$



Solving HPP with DNA computing

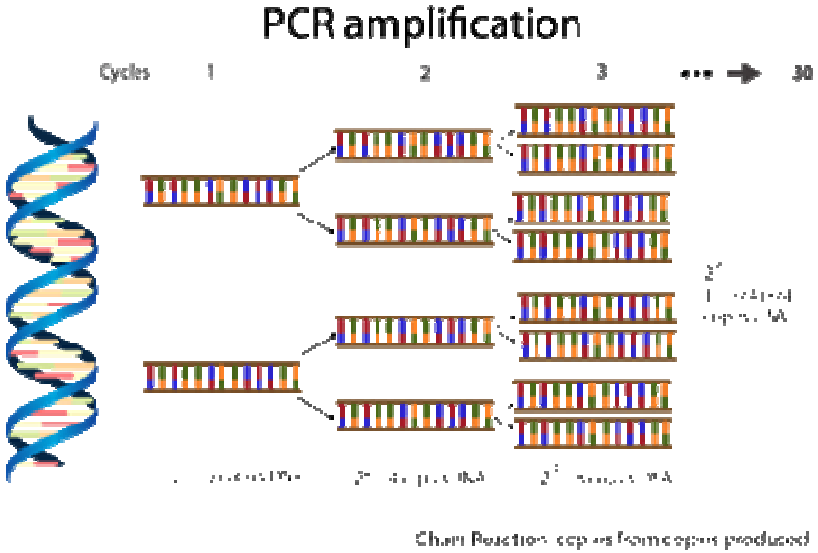
- Example:



- *Result:* formation of DNA molecules encoding random paths through *G*

Solving HPP with DNA computing

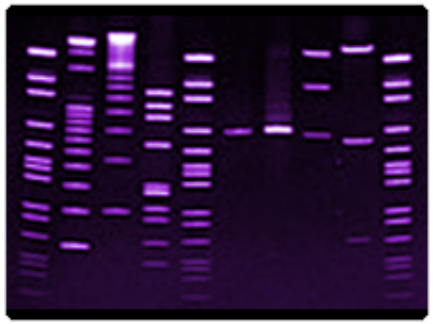
- Implementing *Step2*:
  - keep only paths that start with v_{start} and end with v_{end}
  - Amplify the product of Step1 with PCR and primers O_1 and O_7
  - Result: molecules encoding paths starting with 1 ending with 7 are amplified



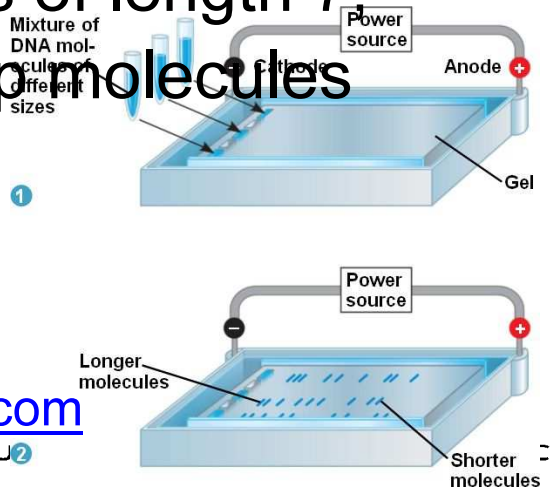
www.gmotesting.com

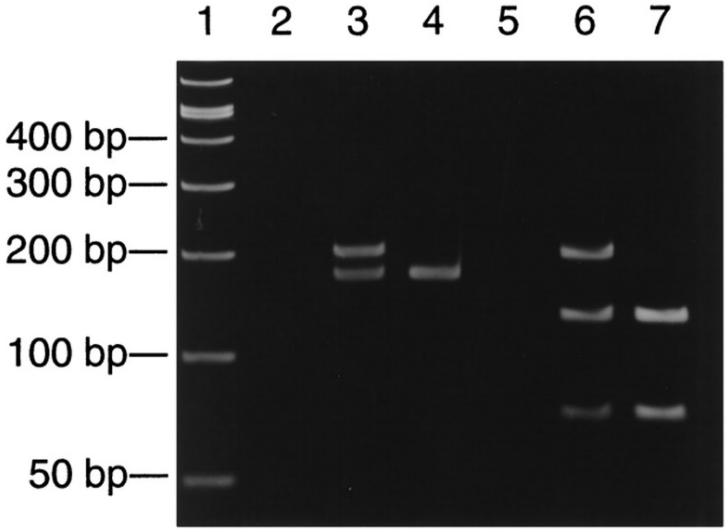
Solving HPP with DNA computing

- Implementing *Step3*:
  - keep only paths of the correct length
  - By use of gel electrophoresis separate DNA strands by their length
  - Filtering criteria: retain only molecules corresponding to paths of length 7
  - i.e., retain 20*7=140 bp molecules



www.thermofisher.com



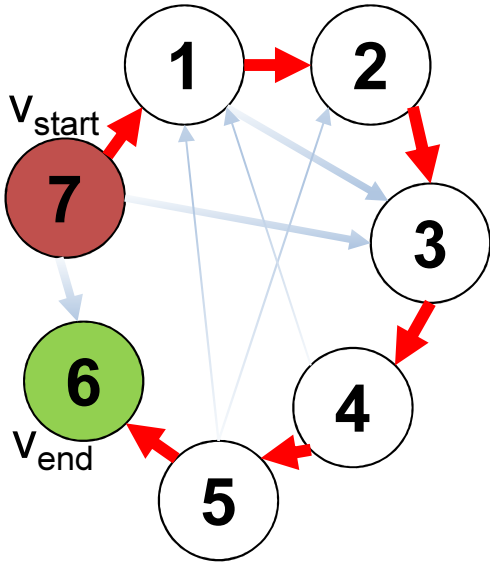


www.neurology.org

decodingdna.yolasite.com

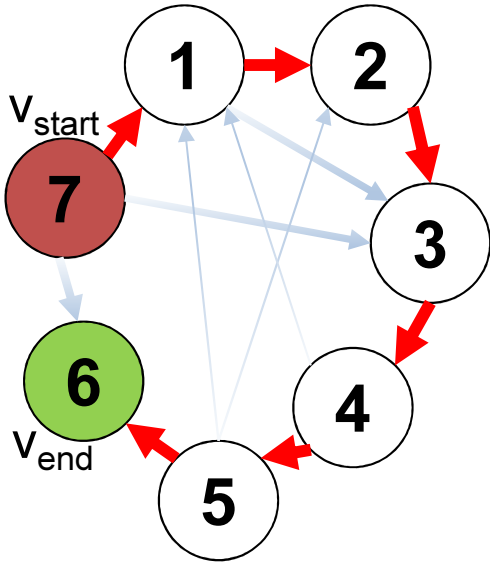
Solving HPP with DNA computing

- Implementing *Step4*:
  - keep only paths that pass through each vertex at least once
  - Filtering by affinity purification:
    - Generate single-stranded DNA from the product of *Step3*
    - For each vertex *i*:
      - Attach O_i to magnetic bead
    - Result:
      - The retained DNAs encode paths containing vertex *i*
      - By repeating successively the process with O_1, O_2, O_3, O_4 and O_5, one gets molecules encoding paths containing all the verticies



Solving HPP with DNA computing

- Implementing the *Output*:
  - are there any paths left?
  - The presence of a molecule encoding for a Hamiltonian path to be checked:
    - Amplify the result of Step4 by PCR with primers O_0 and O_6
    - *Output:* Sequence the result
    - In our case: 7-1-2-3-4-5-6

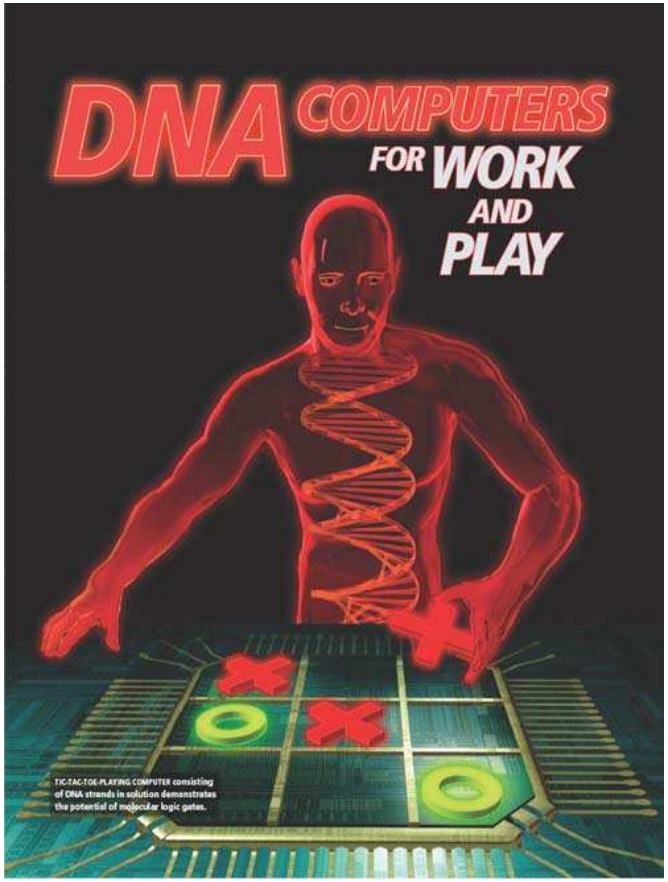


Discussion

- The entire computation required approximately **7 days** of wet lab work
- It was the **first proof-of-concept** experiment that DNA computation was possible
- Advantages:
  - *Step1* is cause of time-complexity exponential explosion:
    - **One time-step** with Adleman's algorithm
- Disadvantages:
  - Time and cost of the experiment
  - **Exponential explosion** for the **volume** of DNA material:
    - e.g., for 200 nodes one would need more than the Earth's weight of DNA

Milestone experiment demonstrating the potential of DNA computing

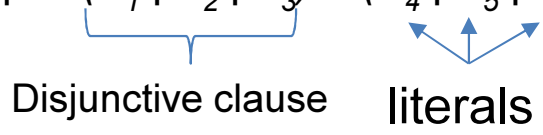
- In 2002 it was demonstrated that DNA computing devices can exceed computational capabilities of an unaided human (Braich et al. 2002)
- Solved with DNA 20-variable 3-SAT problem with 24 clauses
- Exhaustive search in solution space of size 2^{20}



www.nature.com

3-SAT problem

- Input:
  - Boolean formula in three-conjunctive-normal-form (3-CNF)
    - Conjunction of disjunctive clauses, where each clause is the disjunction of at most 3 literals
    - A literal – a boolean variable or its negation
    - Example: $(X_1 \mid X_2 \mid X_3)$ & $(X_4 \mid X_5 \mid X_6)$



  - The boolean formula is *satisfiable* if there exists a truth assignment what makes the formula *true*
- Output:
  - yes, if there exists variable truth value assignment satisfying the input boolean formula

Solving 3-SAT

- Nondeterministic algorithm:
  - Input:
    - A boolean formula *F* in *3-CNF*
  - *Step1:* Generate the set of all possible truth value assignments
  - *Step2:* Remove the set of truth value assignments that make the first clause false
  - *Step3:* Repeat Step 2 for all the clauses of the input formula
  - Output:
    - The remaining (if any) truth value assignments

Solving 3-SAT with DNA

- Encode variable:
  - For every variable X_k two 15-mer strands:
    - 1 strand X^T_k associated with *true* assignment of X_k
    - Another strand X^F_k associated with *false* assignment of X_k
- Generate library of all truth assignments by using mix-and-match combinatorial synthesis technique.
- In total all 2^{20} different assignments were generated – 300-mer DNA strand

Solving 3-SAT with DNA

- Filtering truth assignments satisfying the boolean formula *F*:
  - A truth assignment: $X_1X_2\ldots X_{20}$ where Xi either is equal to X^T_k or to X^f_k
  - 25 glass modules:
    - 0 glass module – initial library of all assignments
    - *i*'th glass module contains probes hybridizing to molecules representing the assignments satisfying clause *I*
    - By applying consecutively filtering from modules 1, 2, …, 24 to the library from the module 0 one gets molecules representing the truth assignments satisfying *F*

(X1|not(X2)|X3)

A truth asignment should contain at least one of X^T_1 or X^f_2 or X^T_3

Discussion

- Observation:
  - Only Watson-Crick complementarity-based annealing and melting was used to compute
  - Brute-force search: limit 60-70 variables
  - Breadth-first search algorithm: limit estimated 120 variables

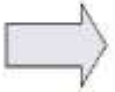
Discussion

- These two experiments are historically significant instances in the area of DNA computing experiments

- Potential applications:
  - Nanorobotics, nanocomputing, bioengineering, bio-nanotechnology, and micromedicine

Codding and DNA Computing

- It is natural to formalize DNA/RNA and their interactions when considering using them in computations and analysis



acgtcgtagttccagtc

www.macdevcenter.com

- One can abstract from all the biochemical properties and reactions related to DNA/RNA and focus solely on the DNA sequences

DNA vs electronic representation of information

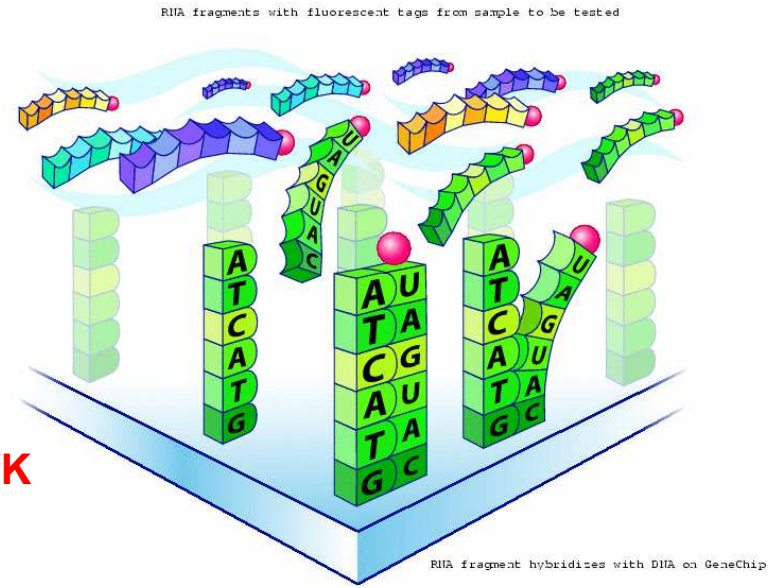
- Fundamental difference between **electronic information** and **DNA information**:
  - **Electronic-based** information has a **fixed address**, **freely accessible** and is **reusable**
  - **DNA-encoded** info is accessible in **less controllable** manner and **once used** in bio-operations **becomes unavailable**
- **Problems** with DNA-based information:
  - **Undesired/unplanned hybridization** between partially complementary DNA sequences is possible

en.wikipedia.org

geekychristian.com

Assesing DNA hybridization

- Standard biological methods **assessing hybridization** between DNA sequences rely on **thermodynamical** parameters:
  - For instance, the ***free energy*** *(energy needed to melt DNA)*
  - And **melting temperature** of DNA
- For calculation of these properties one needs:
  - **Not only WK complementarity** between single bases, but **also WK complementarity of neighboring bases** with their counterparts
- Approximate characterization of DNA hybridization:
  - WK complementarity
  - Other similarity measures



cswww.essex.ac.uk

Encoding of information with DNA

- Designing a set of “good” DNA strands:
  - ***Positive*** design problem:
    - Design a **set of input DNA molecules** such that there is a sequence of reactions that **produces the correct result**.
  - ***Negative*** design problem:
    - Design a **set of input DNA molecules** that **do not interact** in **undesirable ways**
      - i.e., **do not produce incorrect outputs**, and
      - **do not consume molecules** necessary **for other** “programmed” interactions

Encoding of information with DNA

- **The *positive* design problem** is highly related to a **specific experiment** and it is **hard** to **find** a generic **framework**

- **The *negative* design problem** can be solved on general basis – construct **a library** of molecules **not allowing** for **undesired** mutual **hybridization**:
  - no strand forms any undesired secondary structure such as hairpin loops
  - no string in the library hybridizes with any string in the library
  - no string in the library hybridizes with the complement of any string in the library

Encoding of information with DNA

- Uniqueness of the oligonucleotides:
  - individual **oligonucleotides** in a mixture **differ substantially** from each other
  - **Goal:** the **nucleotides** and **any longer sequences** containing the nucleotides should be easily **distinguishable**
  - In math terms those are **codes**
- DNA codewords
  - sets of unique oligonucleotides of a fixed length

Encoding of information with DNA

- DNA encoding design:
  - **Thermodynamical methods** provide the **most precise results** but are **computationally** the **most expensive**
  - The opposite approach: **WK complementarity** allows for **fastest** but **least-precise** methods
  - **Approximation** methods: **capture key aspects** of the nearest neighbor thermodynamic model
    - This is an **intermediate step** between these two methodologies
  - Various **discrete metrics** based often on **Hamming or Levenshtein distance** have been studied

Formalizing DNA

- Single-stranded DNA molecules →
  - strings over the DNA alphabet $D = \{A, C, T, G\}$
- Reactions on DNA →
  - Formal manipulation on the respective strings
- An alphabet:
  - A finite non-empty set of symbols
- A word:
  - A sequence of symbols from the alphabet

Hamming distance

- Hamming distance between two equal-length strings:
  - the number of positions at which the corresponding symbols are different
- Example:
  - "toned" and "roses":
    - 3
  - 1011101 and 1001001:
    - 2
  - 2173896 and 2233796:
    - 3

Levenshtein distance

- Levenshtein (edit) distance between two strings:
  - the minimum number of single-character edits (insertion, deletion, substitution) required to change one word into the other
- Example:
  - "kitten" and "sitting":
    - 3
    - **k**itten → **s**itten
    - sitt**e**n → sitt**i**n
    - sittin → sittin**g**

Intramolecular bonds in DNA

- Hairpin:
  - Self-hybridized DNA

- Hairpin-like secondary structures play important role in:
  - insertion/deletion operations with DNA
  - Whiplash PCR computing techniques
  - DNA RAM
  - *In vivo*: one of intramolecular operations in gene assembly process in ciliates

Intramolecular bonds in DNA

- Hairpin-freeness is crucial in the design of primers for the PCR reaction