Introduction to Biocomputing

Computing with DNA (and other molecules)

Biocomputing: major subfields

- *DNA Computing*
- *Bioinformatics*
- *"In vivo" Computations*
- *Computational Biology*

Computing with DNA (and other molecules)

- Biomolecules: DNA, RNA, protein
- Bio-tools: construct, measure, multiply, manipulate molecules
- Use these tools for computing

Why molecular computing ?

- Subjective (philosophical) reasons
  - It looks natural to do so
  - Another way to go beyond the barrier of human computing limits

Why molecular computing ?

- Objective reasons: very small, very precise, very specific, very cheap, and very energy efficient
  - Energy efficiency
    - On the scale of 10^{19} ligations/J vs. a scale of 10^{9} operations /J in electronic computers
  - Huge density of stored information
    - 1g DNA can store more than one trillion CDs
  - Massive parallelism

Other reasons for molecular computing

- Physical boundaries for the performances of the electronic computers
- Fast development of biotechnologies, genetics, and pharmaceutics
- (Theoretical) Understanding the essence of computation

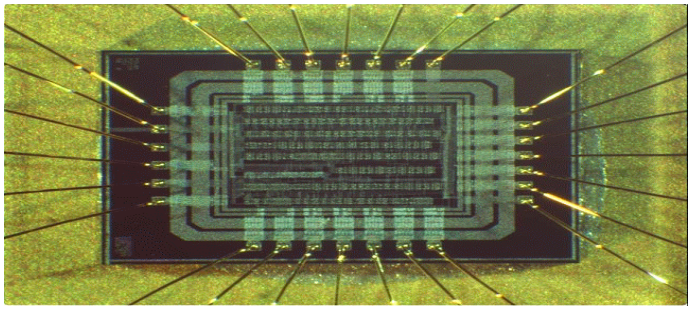
Computing is Easy

1. A METHOD FOR STORING INFORMATION

2. A FEW SIMPLE OPERATIONS FOR ACTING ON INFORMATION

Computing is Easy

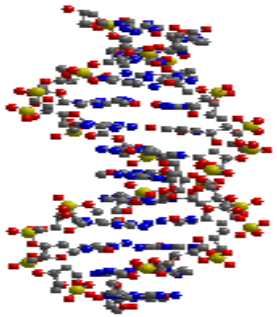
1. A METHOD FOR STORING INFORMATION



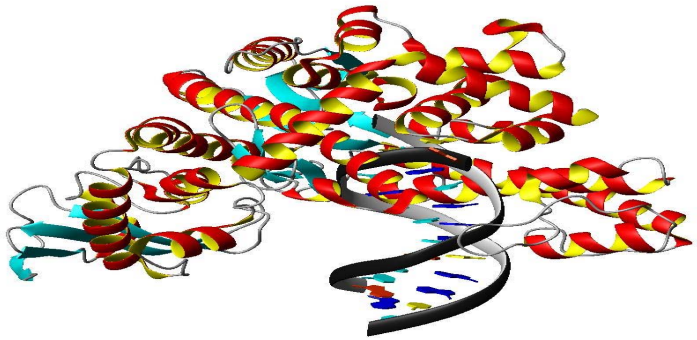
2. A FEW SIMPLE OPERATIONS FOR ACTING ON INFORMATION

Computing is Easy

1. A METHOD FOR STORING INFORMATION



2. A FEW SIMPLE OPERATIONS FOR ACTING ON INFORMATION



Demonstrating the potential of molecular computations

- Theoretical models – e.g., splicing systems
  - Universality results: equivalence with Turing machines
  - Consequence: any algorithm can in principle be implemented using biomolecular tools

Demonstrating the potential of molecular computations

- Practical demonstrations
  - Adleman's experiment
  - Satisfiability of logical formulas
  - Cryptanalyzing DES
  - Chess problems
  - Tic-tac-toe
  - Databases
  - DNA-based logical circuits
  - …
- Can DNA compute "everything" ?

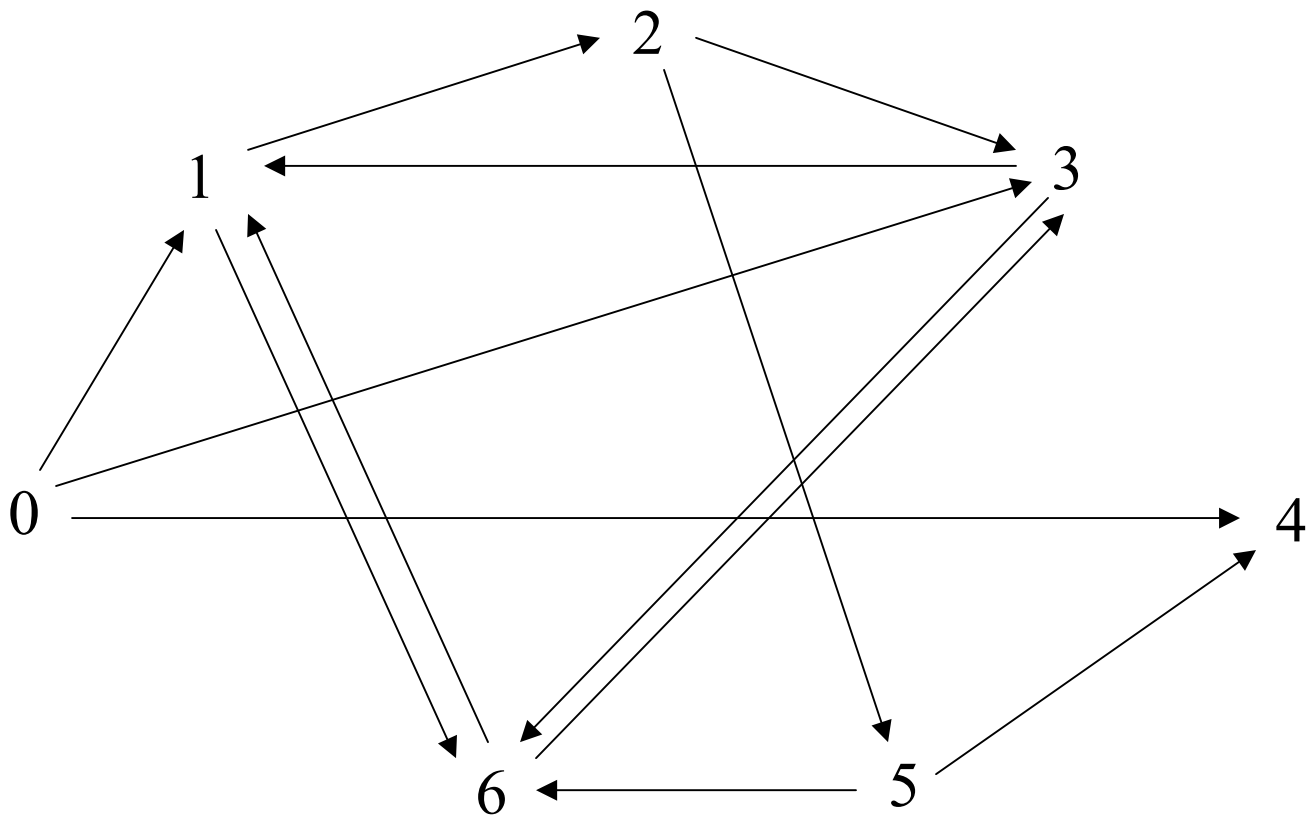
The beginning: Adleman's experiment

- L.M. Adleman: Molecular computation of solutions to combinatorial problems. *Science*, 226, 1021-1024, November 1994.
- Showed how DNA can be used to solve difficult math problems
- The problem of choice: the Hamiltonian Path Problem (HPP)

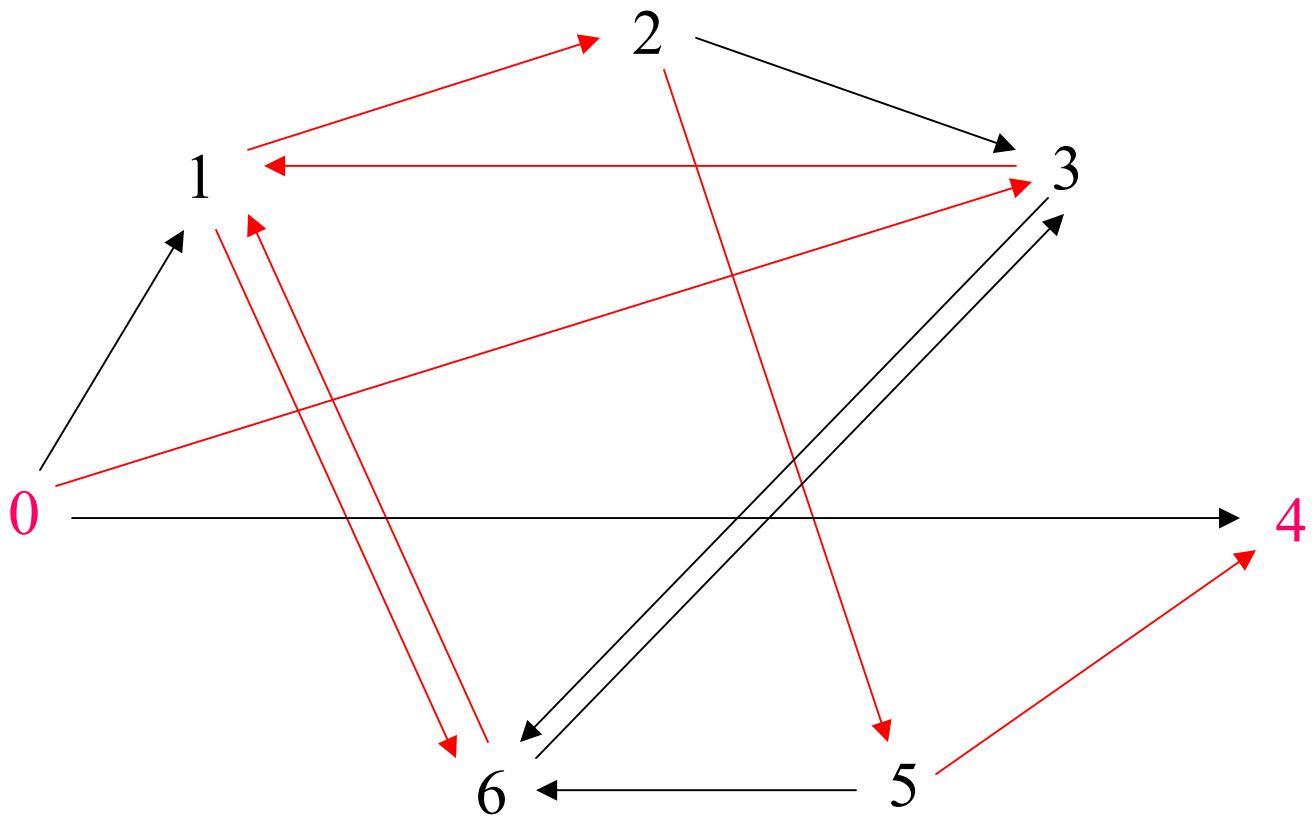
Hamiltonian paths

- Directed graphs: set of nodes and edges (arrows) among them
- Hamiltonian path from a node *s* to a node *e*: start in the node *s* and follow the edges to arrive in node *e*, such that all the other nodes have been visited on the way *exactly once*
- Hamiltonian path problem (*HPP*): *for a given graph, decide if there exists a Hamiltonian path*
- *Example: Adleman's graph*
- Several algorithms are known, but they all have exponential complexity in the worst case
- HPP is **NP**-complete

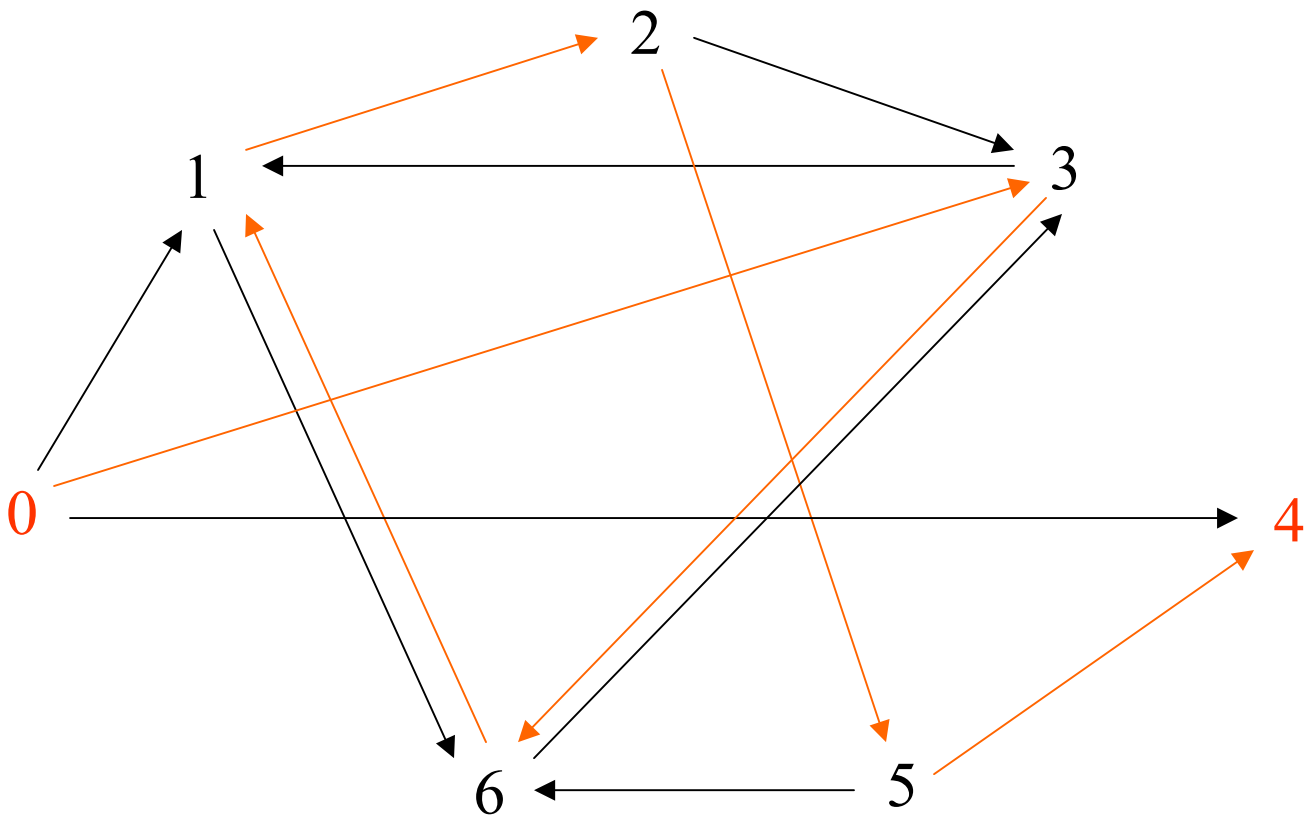
Adleman's graph



Adleman's graph: one path from 0 to 4 visiting all other nodes



Adleman's graph: a *Hamiltonian* path from 0 to 4



•Other Hamiltonian paths from 0 to 4 in this graph ?

Adleman's solution to HPP

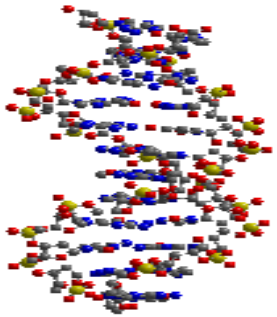
- Input: a directed graph with n nodes, v_{in}, v_{out}
- 1. Randomly generate paths in G
- 2. Reject all paths that do not begin in v_{in} and do not end in v_{out}
- 3. Reject all paths that do not involve exactly n nodes
- 4. For each node v of G, reject all paths that do not pass through v
- Output: YES if any paths remain, NO otherwise

Main idea

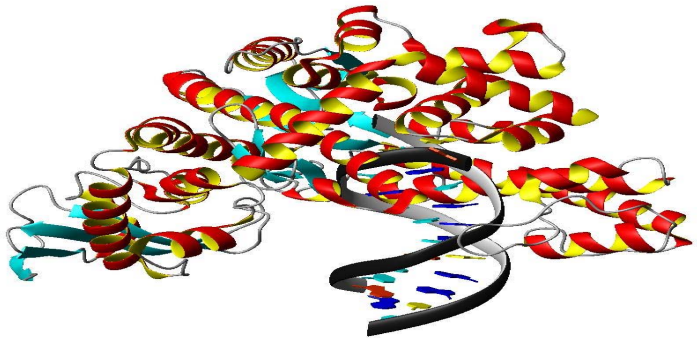
- Exhaustive search through all possible paths in G
- Background engine: the massive parallelism of bio-operations

Computing is Easy

1. A METHOD FOR STORING INFORMATION



2. A FEW SIMPLE OPERATIONS FOR ACTING ON INFORMATION



Experiment design: encoding the nodes

- A node: 20-mer DNA single strand

  s_2 = TATCGGATCGGTATATCCGA

  s_3 = GCTATTCGAGCTTAAAGCTA

  s_4 = GGCTAGGTACCAGCATGCTT

Experiment design: Encoding the edges

- Watson-Crick morphism h: a mapping applied on strings over the alphabet {A,C,T,G}

  h(A)=T, h(T)=A, h(C)=G, h(G)=C

- For a given string *u*, *h(u)* is the Watson-Crick complement of *u* (the single strands *u* and *h(u)* can form a perfect duplex DNA molecule)
- Note: h changes the orientation to 3'-5'
- Example:

  h(5'-CATTAG)=3'-GTAATC

Experiment design: Encoding the edges

- Each of the 7 strands for the nodes are

$$s_i = s'_i s''_i$$

- Edge from i to j: 20-mer DNA strand

$$h(s''_i s'_j)$$

(i=0: s_0 instead of s''_0; j=6: s_6 instead of s'_6)

Experiment design: Encoding the edges

- Examples:

s_2 = TATCGGATCG GTATATCCGA

s_3 = GCTATTCGAG CTTAAAGCTA

s_4 = GGCTAGGTAC CAGCATGCTT

$e_{2\rightarrow3}$ = CATATAGGCT CGATAAGCTC

$e_{3\rightarrow2}$ = GAATTTCGAT ATAGCCTAGC

$e_{3\rightarrow4}$ = GAATTTCGAT CCGATCCATG

Experiment design: encoding paths: double strands

- A strand encoding s_i fuses together (*annealing*) with a strand encoding $e_{i\to j}$: double strand with sticky ends
- A strand encoding s_j then fuses along forming the path from i to j
- It follows a strand corresponding to $e_{j\to k}$, and one corresponding to s_k, etc.

Adleman's solution to HPP

- Input: a directed graph with n nodes, v_{in}, v_{out}
- 1. Randomly generate paths in G
- 2. Reject all paths that do not begin in v_{in} and do not end in v_{out}
- 3. Reject all paths that do not involve exactly n nodes
- 4. For each node v of G, reject all paths that do not pass through v
- Output: YES if any paths remain, NO otherwise

Step 1: Generate random paths

- For each node i (except i=0,6) and each edge i→j: mix large quantities of s_i and $e_{i\to j}$ in one single ligation reaction
- Result: DNA molecules encoding random paths
- Note: huge scale (much larger than needed): each oligo present in 10^{13} copies

Step 2: Start in s_0 and end in s_6

- Multiply the result of Step 1
- PCR with s_0 and s_6 as primers
- Result: amplify those paths beginning in node 0 and ending in node 6

Step 3: Exactly *n* nodes on the path

- Run the result of Step 2 through gel electrophoresis
- The 140 bp band (7 nodes on the path) excised and DNA recovered
- Gel-purification and PCR
- Result: paths of 7 vertices from 0 to 6

Step 4: All nodes are on the path

- Denature the product of Step 3: single stranded DNA
- Testing: test for molecules s_0
- Repeat the testing for s_1, …, s_6
- Amplify by PCR and run on gel
- Result: *molecules encoding Hamiltonian paths from 0 to 6*

Discussion

- 7 days of work – the last step most time consuming (one full day)
- The molecular algorithm used here is rather primitive and inefficient
- The steps can be described in algorithmic way (bio-algorithm): easy to reason

Bio-algorithm for HPP

1. *Input*(N)
2. N←B(N,s_0)
3. N←E(N,s_6)
4. N←(N,<140)
5. **for** i=1 to 5 **do begin** n←+(N,s_i) **end**
6. *Detect*(N)

Scaling up the algorithm

- Quantity of oligos needed in the experiment: difficult isssue
- More edges: more oligos, linear growth
- More vertices: more oligos, exponential growth
- Errors: due to incorrect ligation, pseudo-paths may be fomed; unlikely to survive Step 4, check it !
- Other errors: losing the Hamiltonian path in Step 4 and getting some non-Hamiltonian ones

“It’s not that the bear dances so well, it’s that he dances at all”