Adleman's DNA algorithm for Hamiltonian Path

The Travelling Salesman Problem

Brief History

- In 1994, Leonard M. Adleman released an article in Science, in which he introduced the NP complete problem of the Hamilton path with DNA molecules.

- He solved the directed Hamiltonian Path problem in 7 days using 7 cities.
- Record: 13,509 cities using 3 Digital AlphaServer 4100's and 32 Pentium II's, and it took only 3 months.

Example: Hamiltonian Graph

- Given a *directed* graph can we find an hamiltonian path (*a more complex problem than the TSP*).
- In this experiment there are 2 keywords:

  massive parallelism (all possibilities are generated)

  complementarity (to encode the information)

- This experiment proved that DNA computing wasn't just a theoretical study but could be applied to real problems like **cryptanalysis** (breaking DES).

The Hamiltonian path problem

- **This kind of problems are abstracted as graphs. Graphs has nodes and edges. Graphs are oriented (like the above) and non-oriented.**

The Hamiltonian path problem:

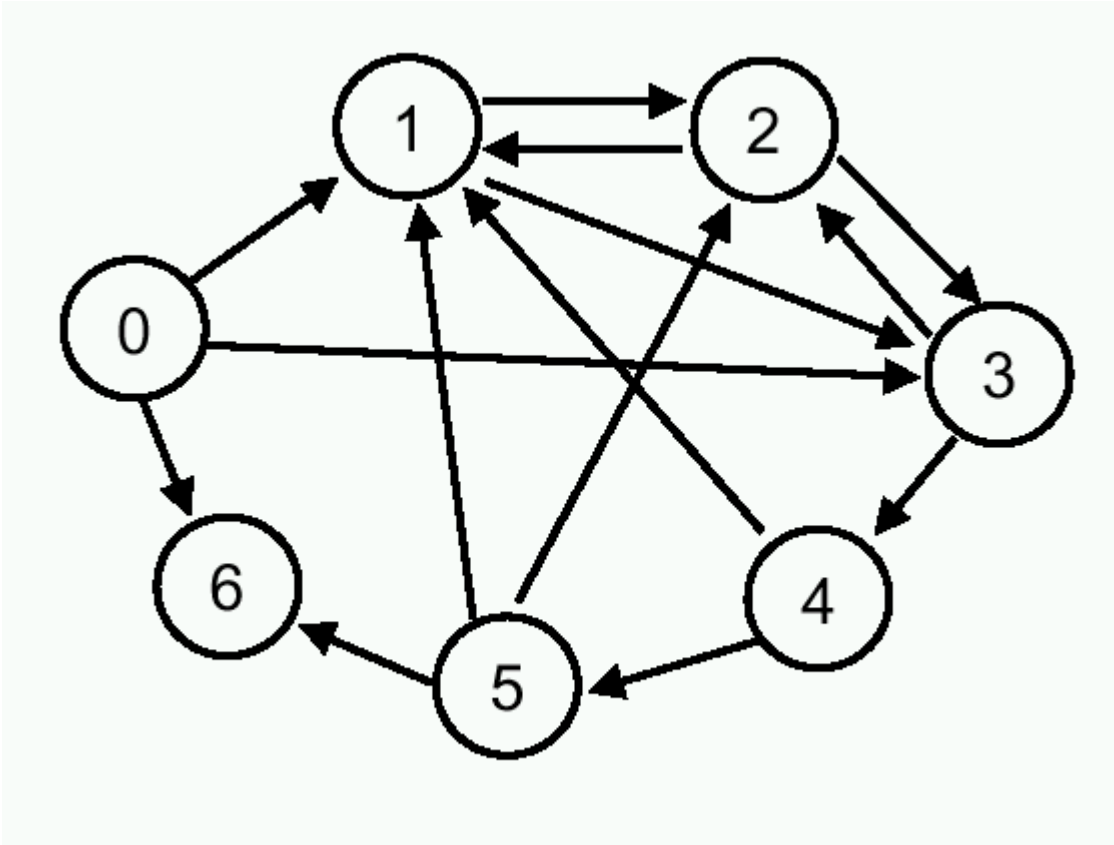
In a directed graph,

find a path from one node that visits (following allowed routes) each node exactly once.

NP-complete

The DNA "computer" can solve it by enumerating all valid paths in parallel

Hamiltonian path as an example of graph theory problem



Adelman's DNA algorithm for Hamiltonian Path

- Input:
  - A directed graph with ***n*** nodes including a **start node *A*** and an **end node B.**
- **Step 1.** Generate random paths through the graphs of the above form, randomly, and in large quantities .
- **Step 2.** Remove all paths that do not begin with *start node A* and end with end node B.
- **Step 3.** If the graph has n nodes, then keep only those paths that enter exactly n nodes.
- **Step 4.** Remove any paths that repeat nodes. This is done by *filtering out* all paths that have no node V_i, for all V_i
- **Step 5**. If any path remains then answer "yes"otherwise answer "no".

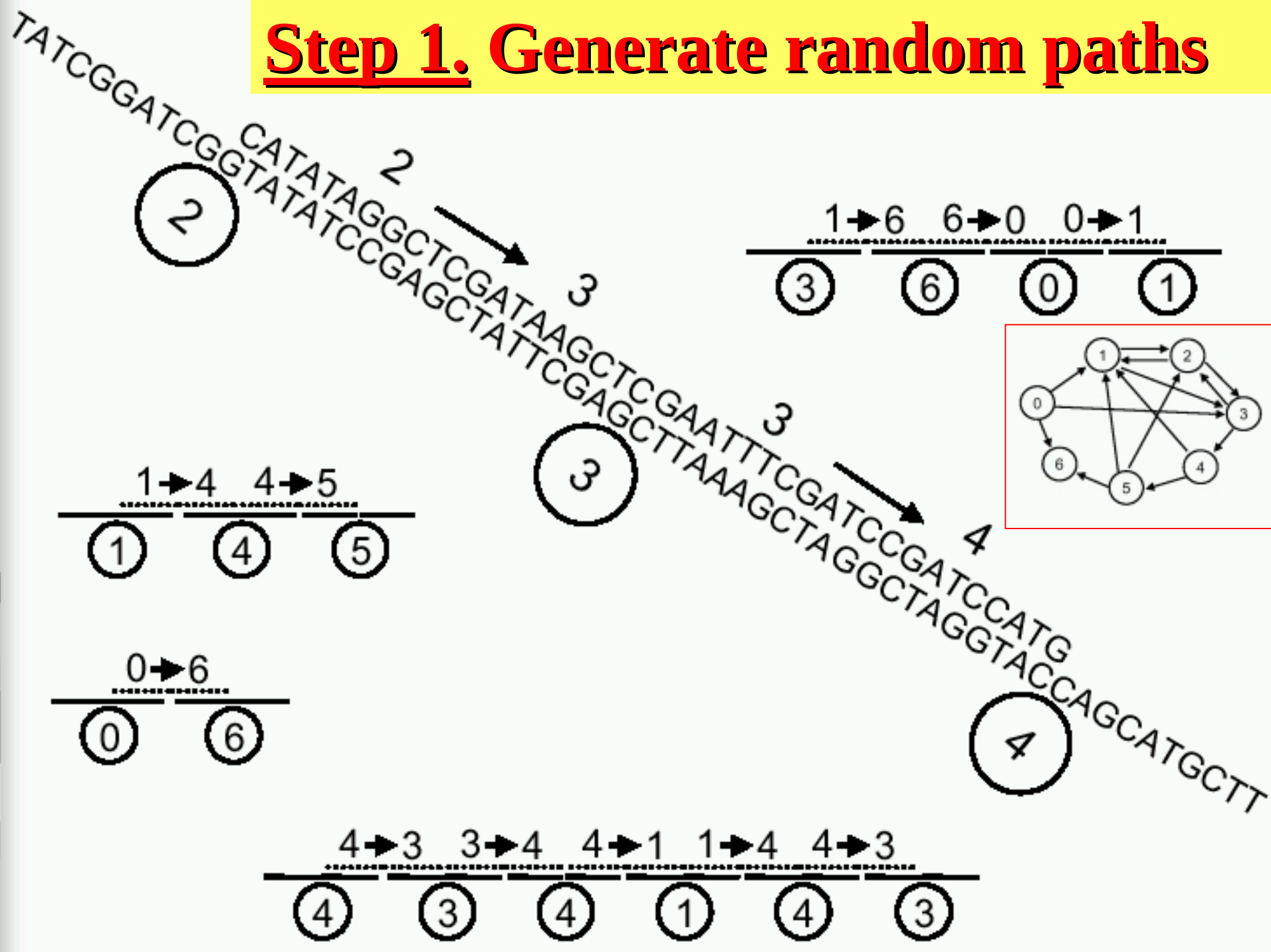
This is a nondeterministic algorithm.

Explain this idea using many children and Lego blocks.

- **Step 1.** Generate random paths through the graph.
  - Mix solutions of nodes and edges, **Ligation**
- **Step 2.** Remove all paths that do not begin with *start node A* and end with end node B.
  - **Polymerase Chain Reaction**
- **Step 3.** If the graph has n nodes, then keep only those paths that enter exactly n nodes.
  - **Gel-Electrophoresis** to get Solution length
    - solution length = (number of nodes) * (20 bp per node)
- **Step 4.** Remove any paths that repeat nodes
  - **Magnetic beads and filtering**
  - To keep paths that have no repeated nodes, filter out all paths that do not have some of nodes (since if a node is missing, some other is repeated). (this is repeated for 5 nodes)
    - **anneal node complements to bio-avidin beads**

Step 5. If any path remains then answer “yes”otherwise answer “no”.

Step 1. Generate random paths



Step 1:
Generate all routes using Ligase

- Synthesizing a short strand of DNA is an easy process using a DNA synthesizing machine.
- Generate all routes.

DNA is connected using an enzyme called ligase

Graph Encoding with DNA

- **Vertex *i***
  - **Random 20-mer DNA sequences: O_i**
  - **Watson-Crick complement $\underline{O_i}$**
- **Edge $i \rightarrow j$**
  - **3‘ 10-mer of O_i (For $i = 0$ take all of O_0)**
  - **5‘ 10-mer of O_j (For $j = 6$ take all of O_6)**
  - **Preserves edge orientation**

Vertex	Encoding
O_2	5‘-TATCGGATCGGTATATCCGA-3‘
O_3	5‘-GCTATTCGAGCTTAAAGCTA-3‘
$O_{2 \rightarrow 3}$	5‘-GTATATCCGAGCTATTCGAG-3‘
$\underline{O_3}$	3‘-CGATAAGCTCGAATTTCGAT-5‘

DNA Computer for this problem

DNA can implement this algorithm! (Uses 10^{15} DNA strings)

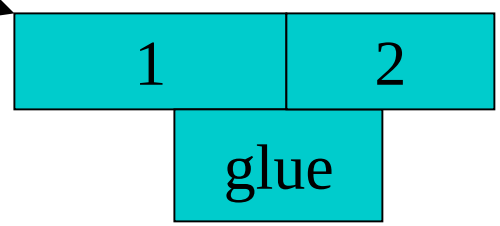
Step 1 : To each node "i" of the graph is associated a ***random* 20 base string** (of the 4 bases A,G,C,T), e.g.

TATCGGATCGGTATATCCGA

Call this string "S-i".

(It is used to "glue" 2 other strings, like LEGO bricks).

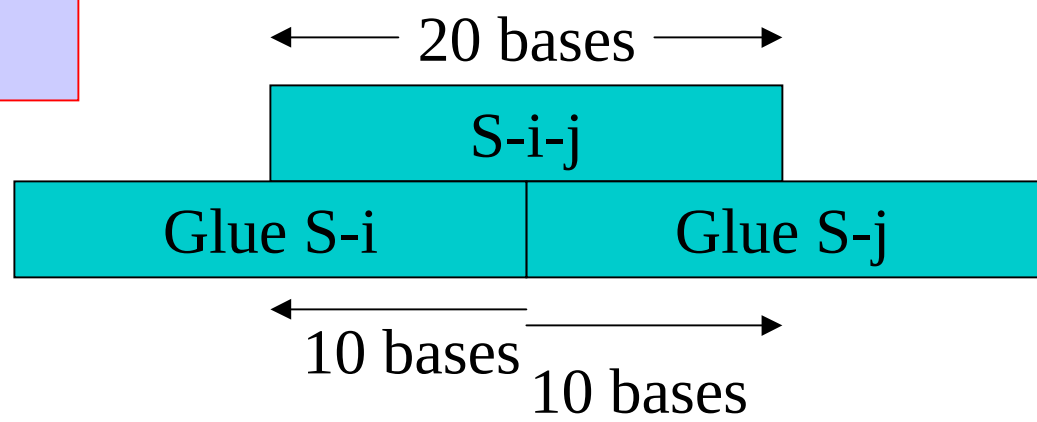
Representation of nodes

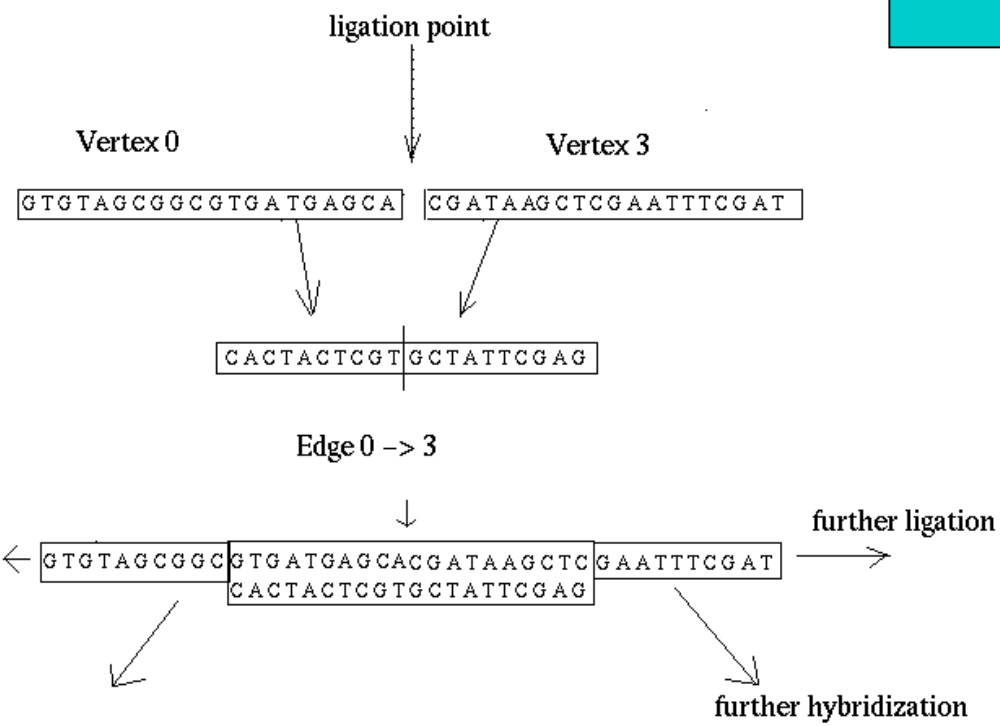


For each directed (arrowed) edge (node “i” to node “j”) of the graph, associate a 20 base DNA string, called “S-i-j” whose -

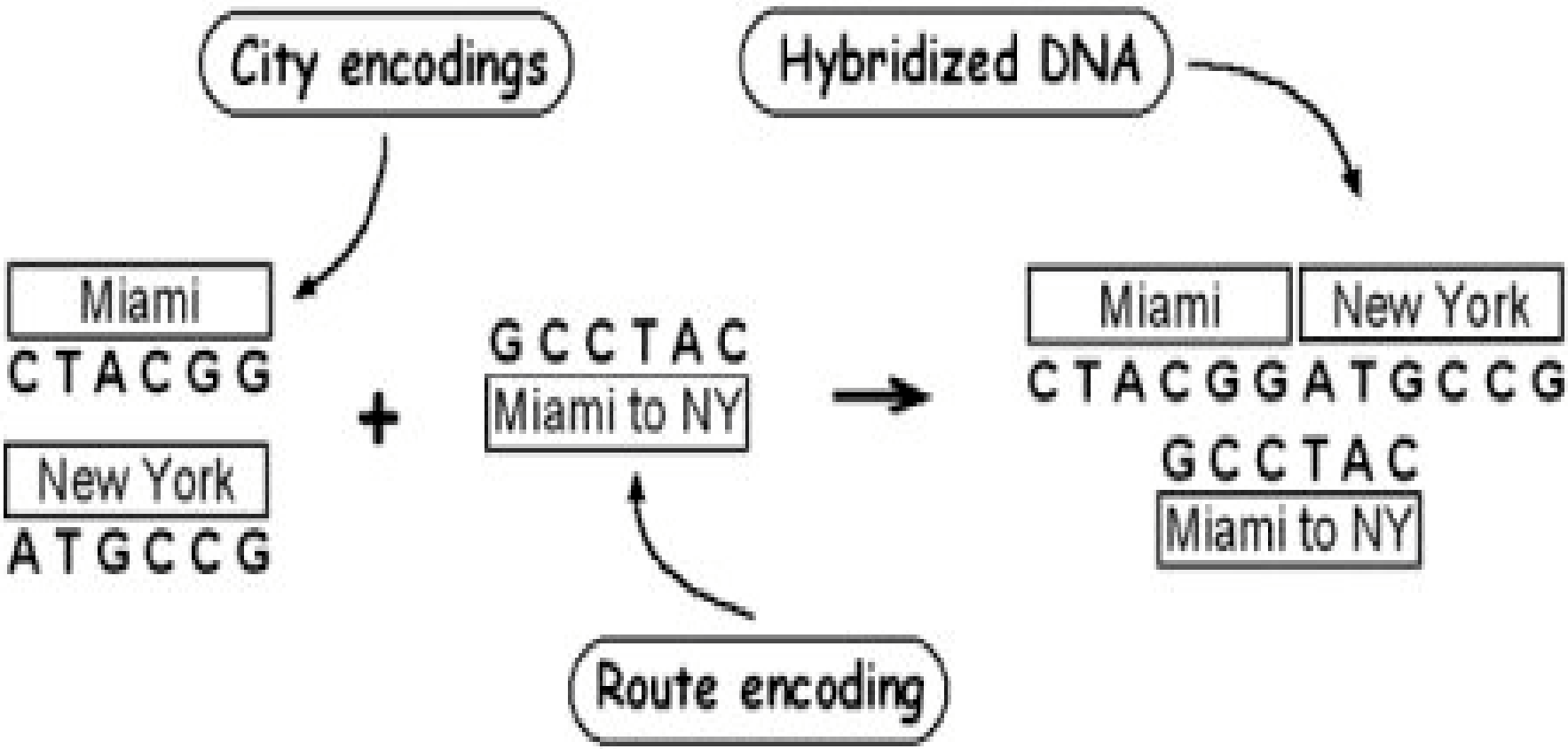
a) **left half** is the DNA complement (i.e. c) of the **right half of S-i**,
b) **right half** is the DNA complement of the **left half of S-j**.

Representation of edges





Generating routes

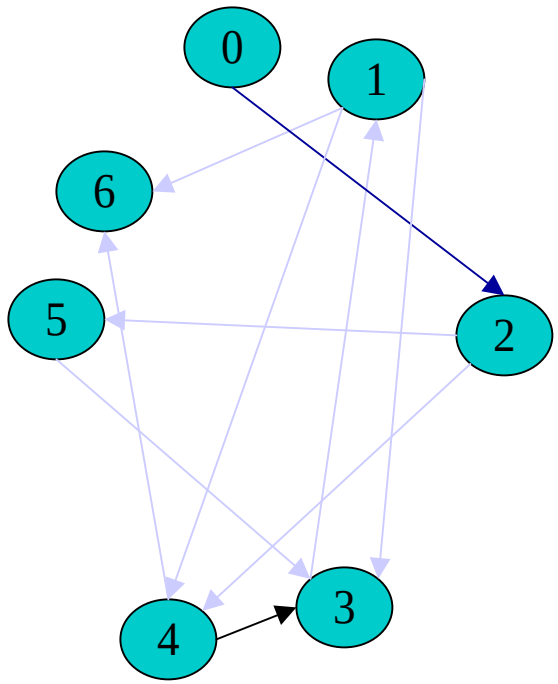


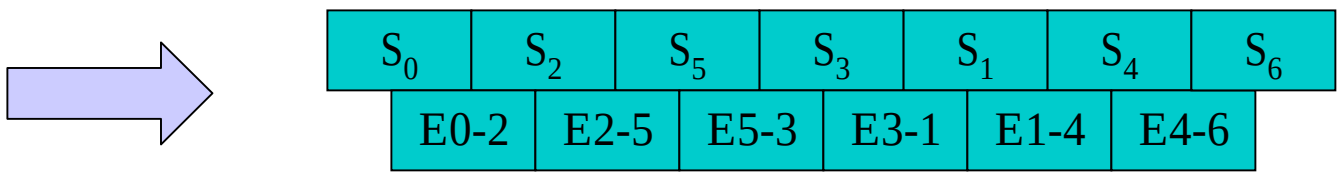
Generating routes

Generating routes

Filtering was the most time-consuming aspect (taking 1 week for 6 nodes)

Multiple purifying and amplifying steps were executed to ensure “good” results



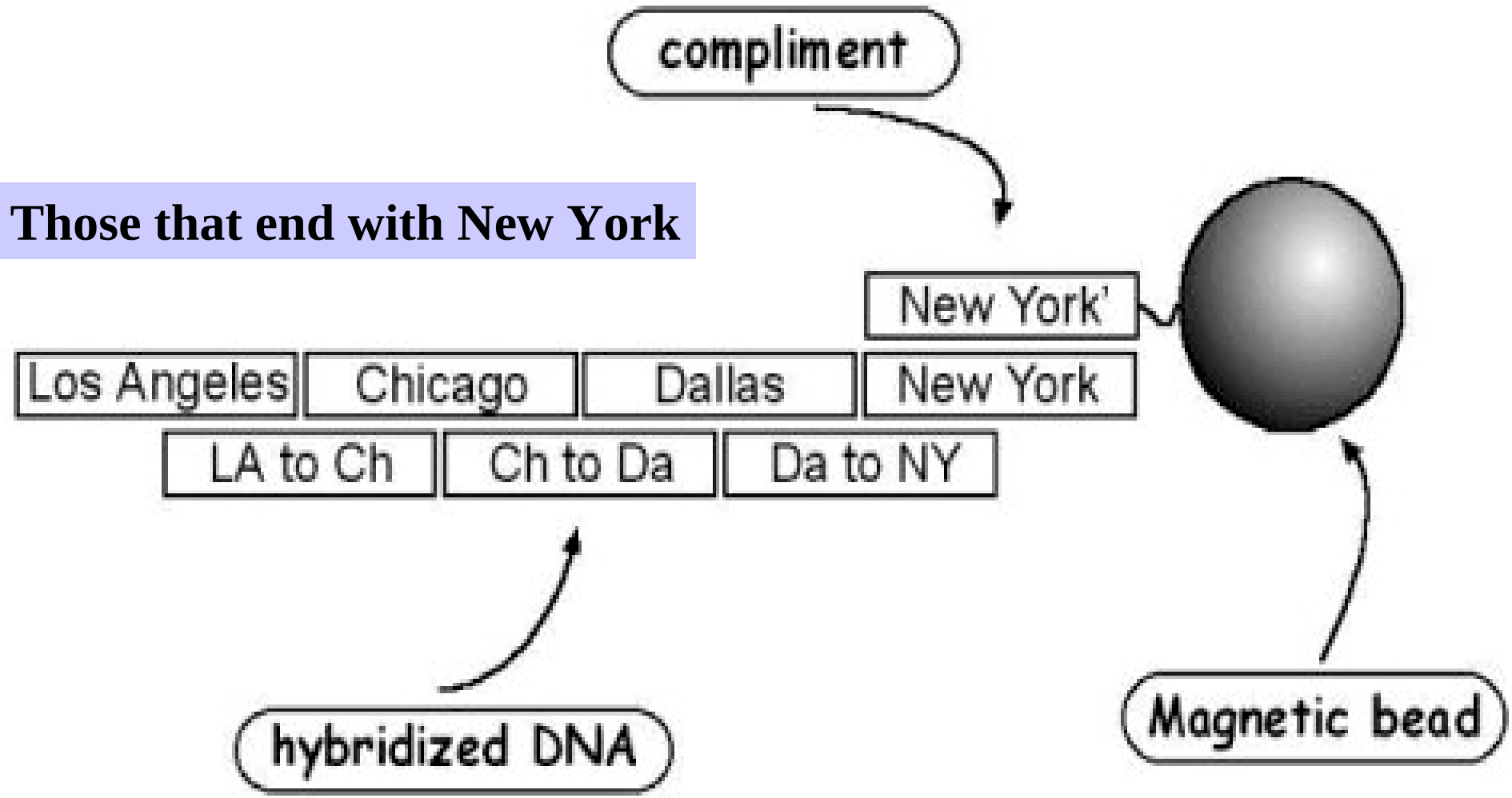


Step 2: Use PCR to remove bad starts and ends

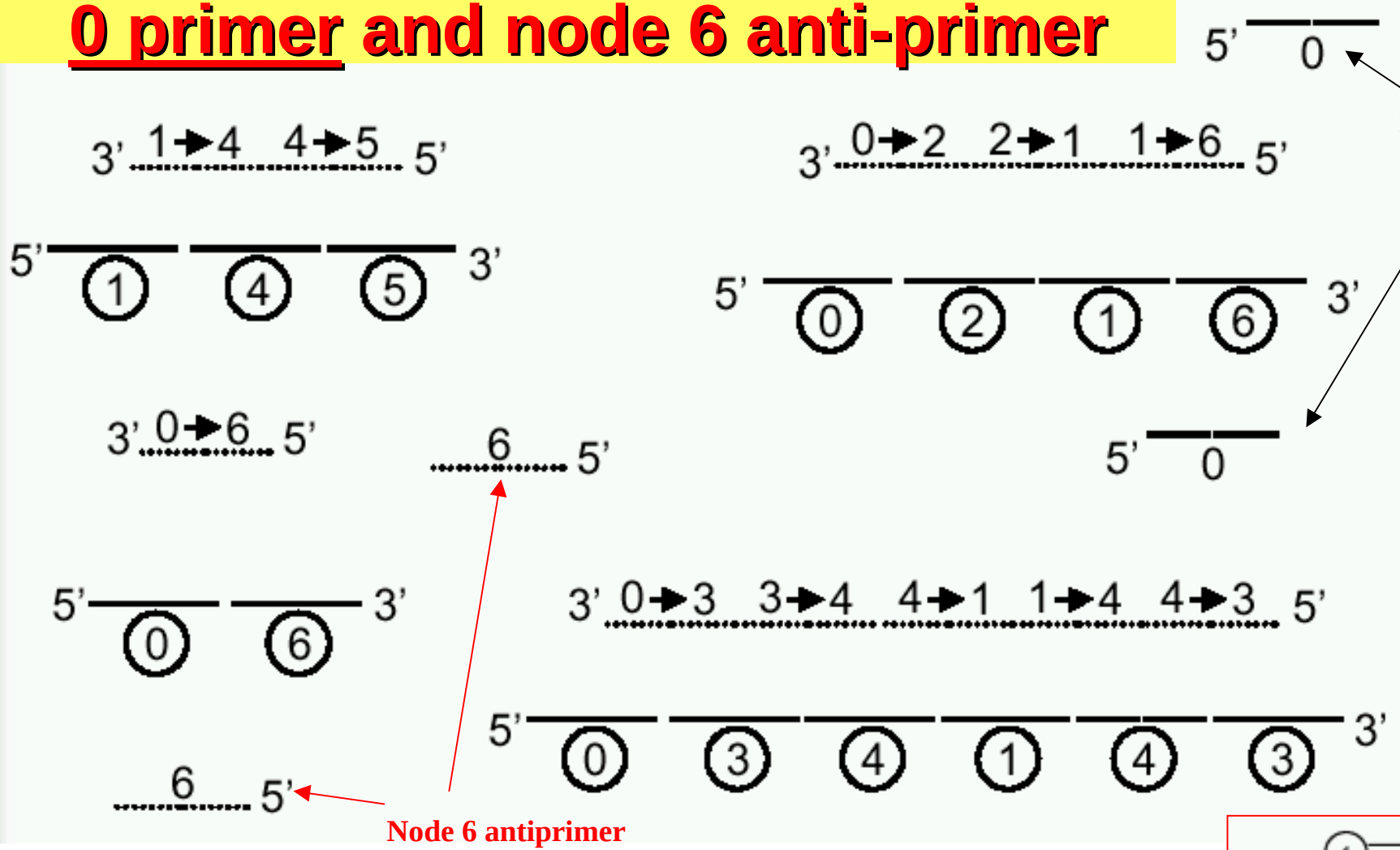
- Selectively amplify DNA strands that represent paths from correct starting city (node) A to destination city B (use *Polymerase Chain Reaction—PCR*).
- **Number of other paths is negligible**

Only those molecules encoding paths that began with node A and ended with node B were amplified.

Affinity Purification

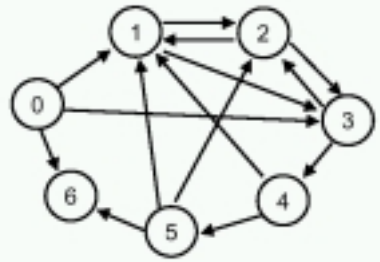


Step 2a. Denature and add node 0 primer and node 6 anti-primer

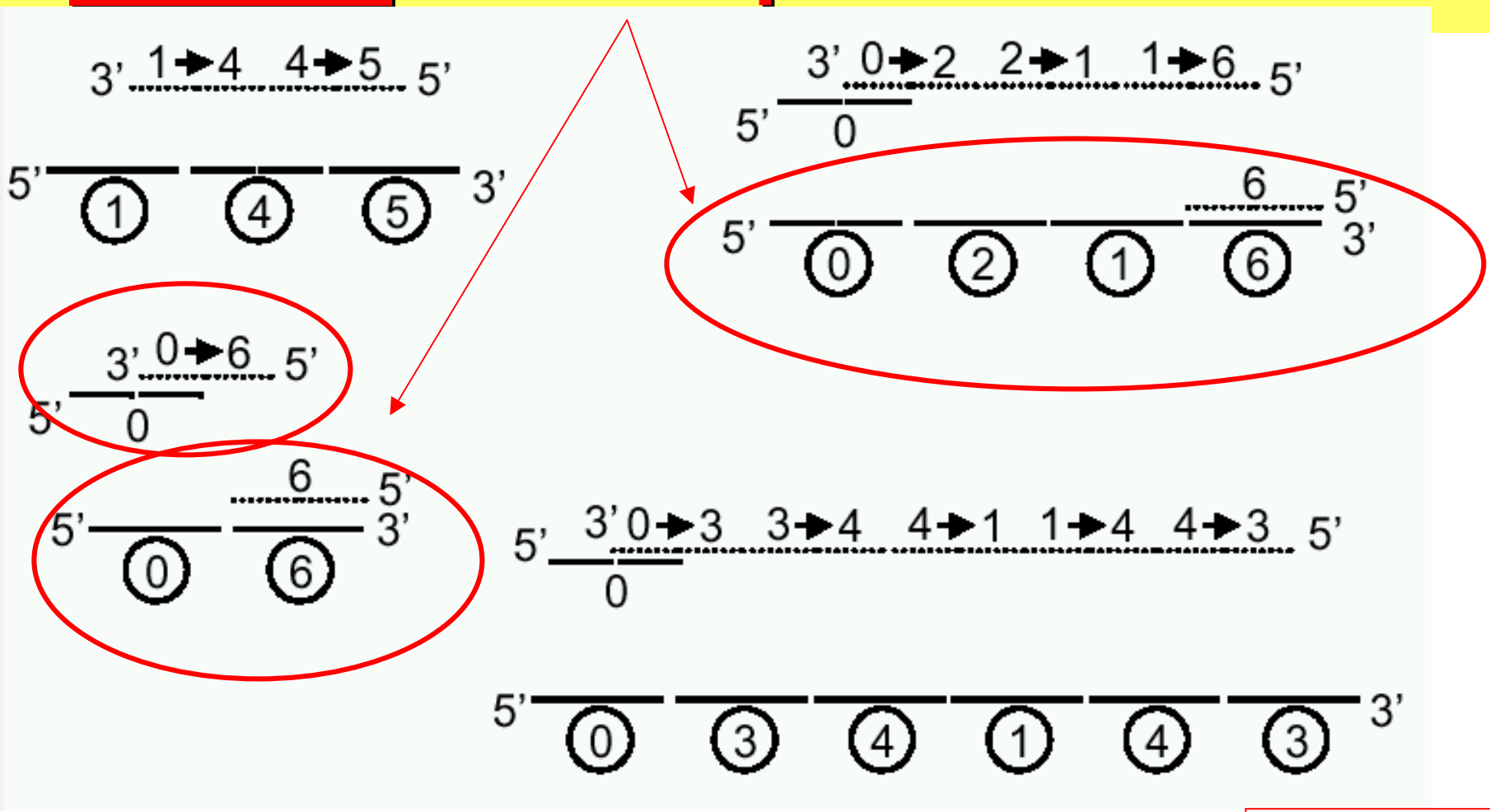


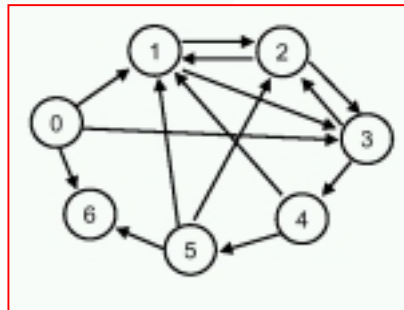
Recall: denature = separate strands

….more precisely…..



Step2b: PCR amplifies 0-6 strands



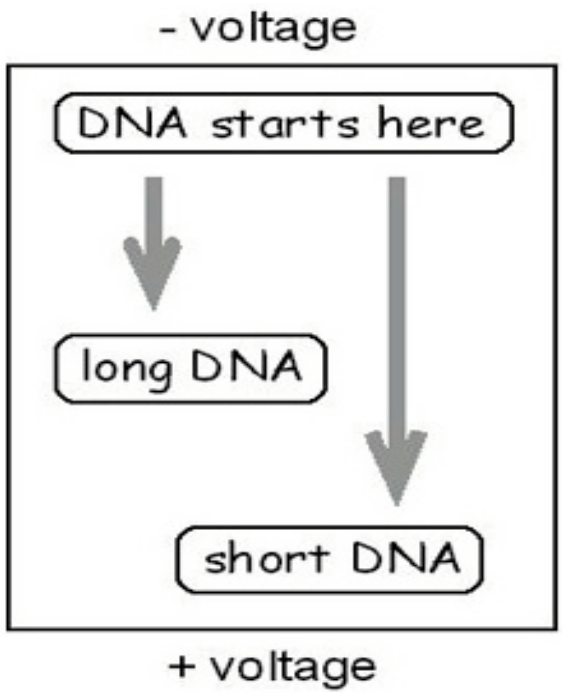


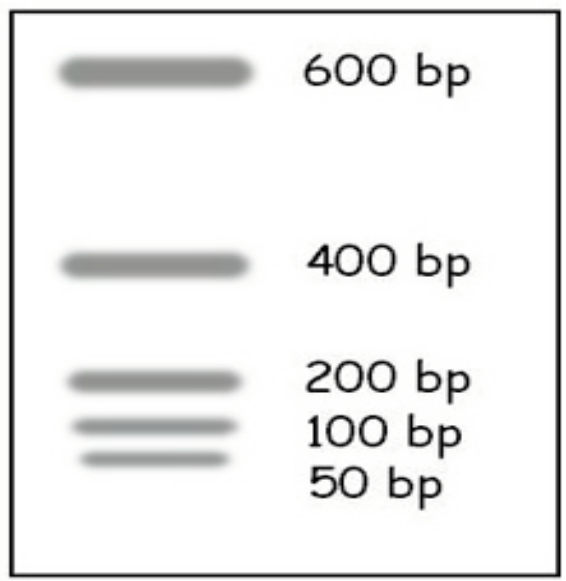
Step 3:
Use Gel Electrophoresis to remove too long (and too short) paths

- At this point, we already have a bunch of DNA strands that:
  - have the correct beginning,
  - have the correct end,
  - but we don't know anything about what is in the middle.
- **Agarose Gel Electropheresis** works on the idea that DNA strands have electric charge.
  - When we insert the DNA strands into it, it causes the DNA to move through the gel.

Gel Electropheresis (Continued)

- DNA is negatively charged
- Place DNA in a gel matrix at the negative end. (Gel Electrophoresis)
- The shorter DNA strands move farther in the gel and the longer ones don't move as far.
- This means that the DNA strands spread according to the length.
- Then we will compare the a DNA strand that is known has the correct length to the DNA strands that are being tested to produce the DNA strand that has the correct length.





Step3 continued: Find paths with 7 nodes

- The DNA with 140 base pairs (corresponding to double-stranded DNA encoding paths entering exactly 7 nodes) was:
  - extracted,
  - PCR amplified,
  - subjected to electrophoresis a few times to purify a sample

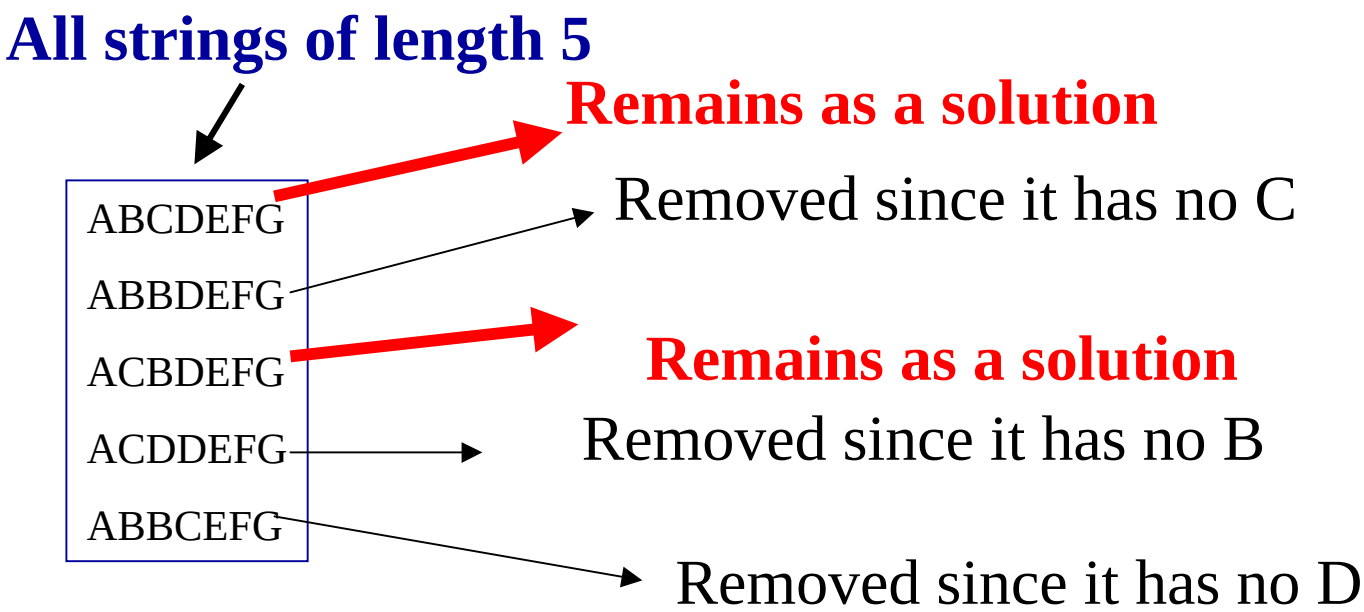
# bp	30	20	20	20	20	30	Σ =140
$O_{i \to j}$	$0 \to a$	$a \to b$	$b \to c$	$c \to d$	$d \to e$	$e \to 6$	

Step 4: Affinity Purification.

Filter the DNA searching for one city at a time.
Do this by using a technique called Affinity Purification. (*remind the magnetic beads!*)

- 1. The product of step 3 was denatured.
- **2. DNA strands that match various nodes visited (the complements of 1 -- 5) are attached to iron balls (magnetic beads) .**
- 3. The product was successively filtered by annealing with solutions containing single complement node beads

Use magnetic beads to extract paths that contain no repeats - let us explain the principles first.



Magnetized Balls

- First, we drop in the balls Bloomington attached and head up the mixture of DNA so that the strands break apart and attach themselves to the iron balls in the test tube.

Magnetized Balls (Continued)

- All the strands that do not visit New York are not attached to the metal balls and can be filtered out.
- The process is then repeated with iron balls representing all the cities that we are considering.
- After the last ball is tests the strands and the bad strands have been siphoned off, you are left with the answer.

- Keep only paths that enter all vertices:
  - Produce single stranded DNA
  - Affinity separate
    - Conjugate $\underline{O_i}$ to metal beads
    - $\underline{O_i}$ anneals to path containing O_i
    - Magnetic field retains $\underline{O_i}$
    - Drop other paths
    - Repeat for all $\underline{O_i}$

Step5: PCR amplify remaining product

- The final step is to determine the sequence of the paths through DNA sequencing and report all of the correct paths.
- There might be more that one answer. It depends on the number of cities involved and the number of ways you can visit the cities.

- Conduct a “graduated PCR” using a series of PCR amplifications.
- Use primers for the start, New York, and the n^{th} item in the path.
- So to find where Minneapolis lies in the path you would conduct a PCR using the primers from New York and Chicago.

Electrophorese

If 140 base-pair band is present
Hamiltonian path exists

TSP: von Neumann Machine

- von Neumann Implementation:
  - Set up a search tree
  - Measure each branch sequentially
  - Keep shortest one
- For 20 cities: 50,000 TB memory!
  - Using a 100 MIPS machine: 2 years!
    - (each city in every path generated per instruction)

Adleman Experiment: Results

- The DNA computer successfully found a solution to the Hamiltonian path
- Step #1 produced approximately 10^{14} operations.
- -8 kcal/mol for ligation ≈ 1 joule sufficient for $2x10^{19}$ operations!
- Theoretical limit: $34x10^{19}$ irreversible operations / joule (at 300 degrees Kelvin)

Adleman Experiment: Results

- DNA storage allows 1 bit / cubic nm
- Downside #1: PCR, oligonucleotide synthesis requires a lot of energy right now
- Downside #2: Inflexibility – current protocols and enzymes limit the complexity of operations.
- Downside Example: It would take tremendous effort to multiply numbers

Adleman's Algorithm: Summary

- First molecular computation
  - Brute force
  - Massively parallel
- Attempt to repeat experiment initially failed
  - Ambiguous results
    - Protocols error prone
    - Sensitive to impurities

- **Operations done manually in the lab.**
- **Natural tools are what they are…**
  - **Formation of a library (statistic way)**
  - **Operations problems**

•This work took Adleman (the inventor of DNA computing, 1994) a week.

Complexity

Linear complexity in bio steps

Exponential complexity in DNA strands

70 vertices require 10^{25} kg DNA

•As the number of nodes increases, the quantity of DNA needed rises exponentially, so the DNA approach does not scale well. The problem is NP-complete.

•But for N nodes, where N is not too large, the 10^{15} DNA molecules offer the advantages of **massive parallelism**.

Problems with Adelman's Approach to DNA computing

- Solving a Hamiltonian graph problem with 200 nodes would require a weight of DNA larger than the earth!
- What algorithms can be profitably implemented using DNA?
- What are the practical algorithms?
- Can errors be controlled adequately?

New generation of computers?

- The Adleman experiment is not the single application case of DNA computing…
- Theoretical models based on it has been created.
- It is proven through language theory that DNA computing "guarantees universal computations".
- Many architectures have been invented since then for DNA computations.

Why don't we see DNA computers everywhere?

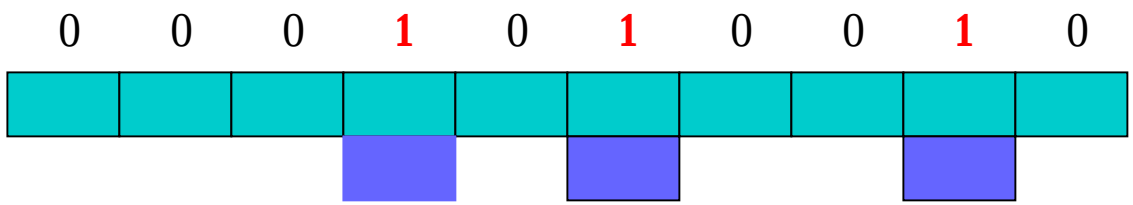
- DNA computing has wonderful possibilities:
  - Reducing the time of computations* (parallelism)
  - Dynamic programming !
- However one important issue is to find "the killer application".
- Great hurdles to overcome…

Sticker Model

- **What needs to be defined?**
  - How is data represented on DNA?
  - How do we set/clear bits?
  - How do we perform separation?
- **Data Representation**
  - Memory complex = Strand of DNA (single or semi-double).
  - ***Example : Single DNA strand*** with N bases (ATCG)
  - M bases represent a bit, storage becomes N/M bits, *for instance m=2*
    - Trade off space for reliability by changing M
- Stickers are segments of DNA, that are composed of a certain number of DNA bases.
  - Create "sticker" strands that bind to a particular region of M bases
    - Memory strand needs to be configured so that each M-base region sufficiently unique
  - To use correctly the stickers model, each sticker must be able to anneal only at a specific place in the memory complex
  - Presence of sticker = 1, absence = 0

Visualization of the sticker idea:

- Memory complex = Strand of DNA (single or semi-double).
- Stickers are segments of DNA, that are composed of a certain number of DNA bases.
- To use correctly the stickers model, each sticker must be able to anneal only at a specific place in the memory complex.



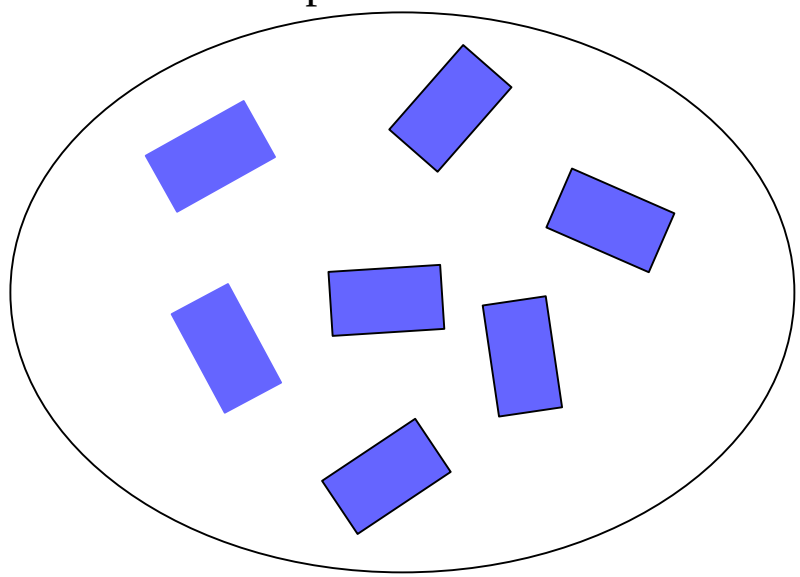
=

A G C A T G A T

Zoom

Memory complex:
Semi-double

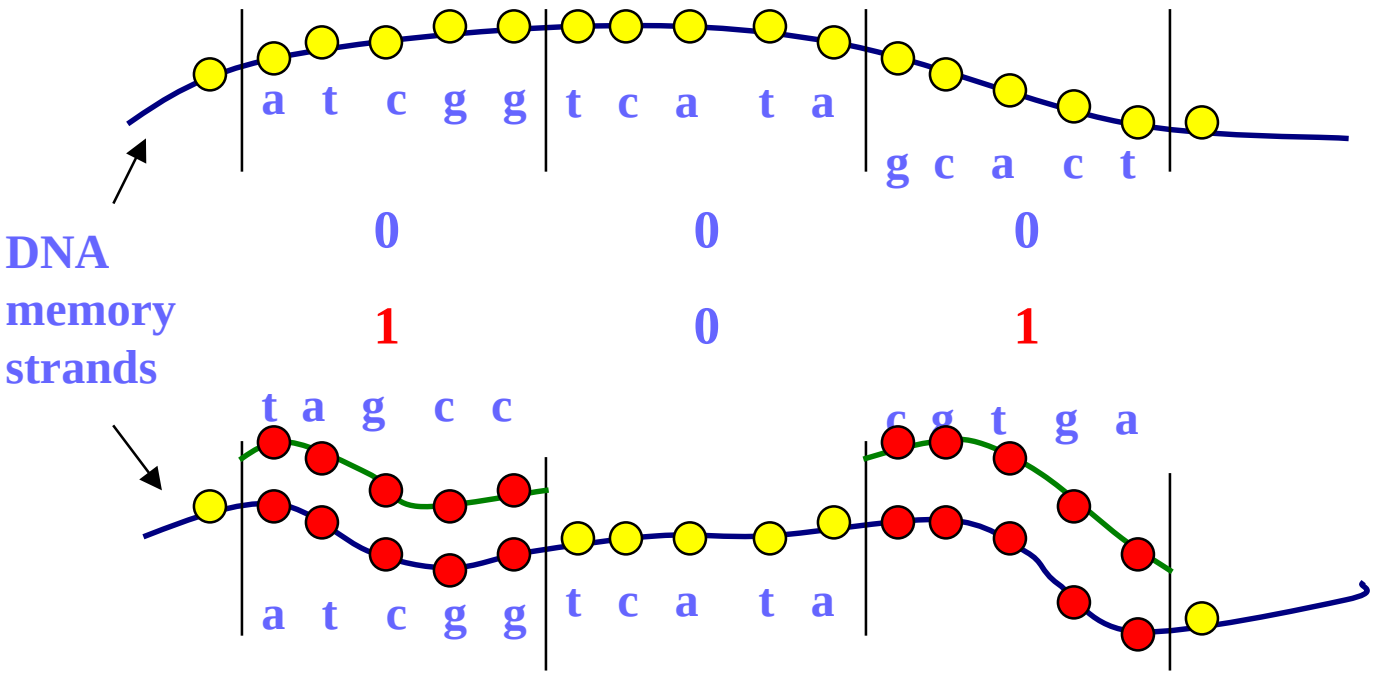
Soup of stickers:



Positional Representation similar to Cube Calculus Machine

Writing and Reading Operations on DNA Memory

Writing : make DNA sequences



Reading : hybridization and readout

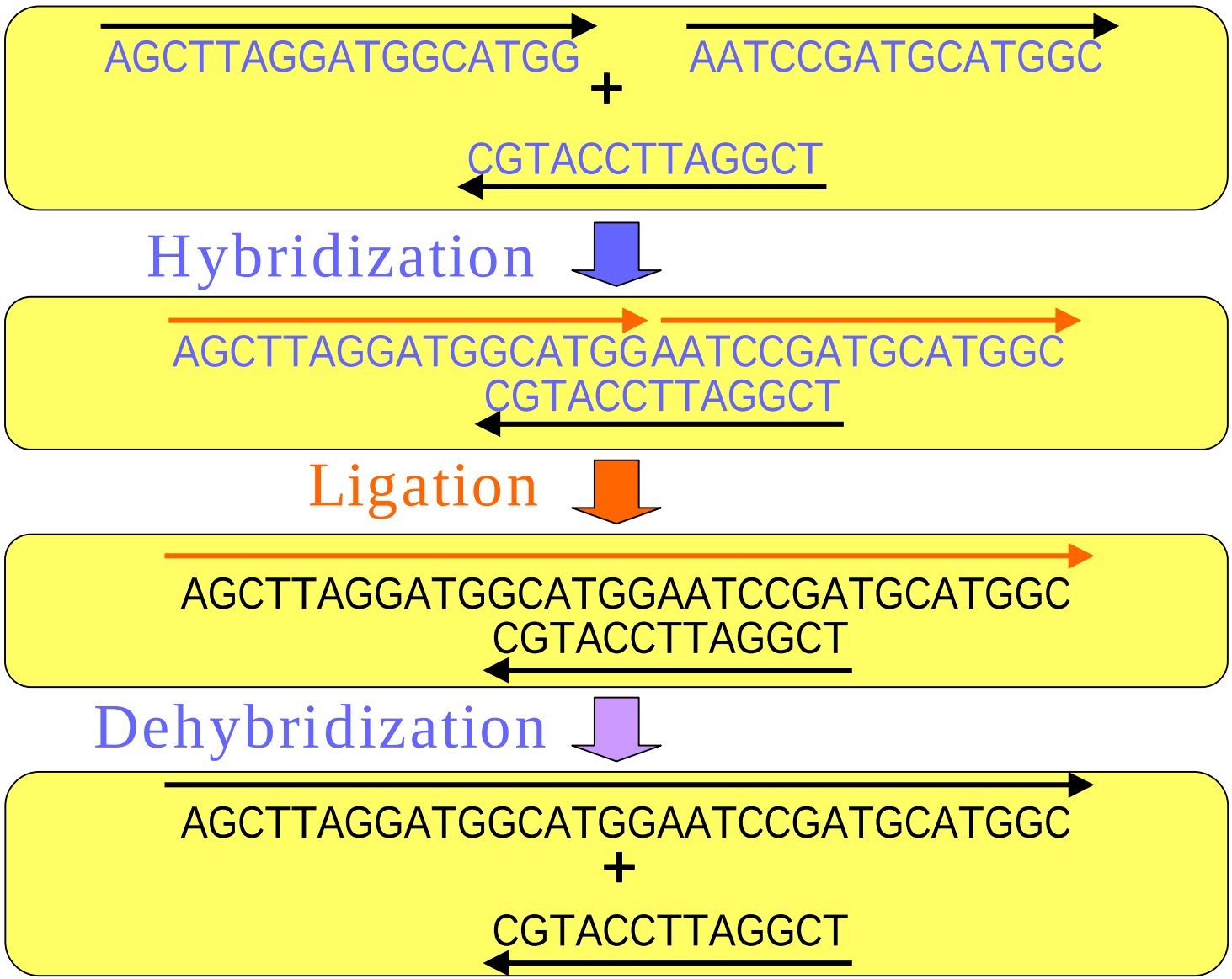
Sticker Model

- **Setting Bits**
  - Just add solution of the appropriate sticker to the tube
- **Clearing Bits**
  - Use chemical process to remove the appropriate sticker (still somewhat undefined)
- **Separation Operation**
  - Construct “probes” that bind to a region if no sticker present, one end of probe is fixed
  - Mix DNA solution with probes, binds all strands with that bit 0 to a probe
  - Remove probes from solution, separate bound strands from probes.
    - Now have two samples, one with bit set and one with bit clear

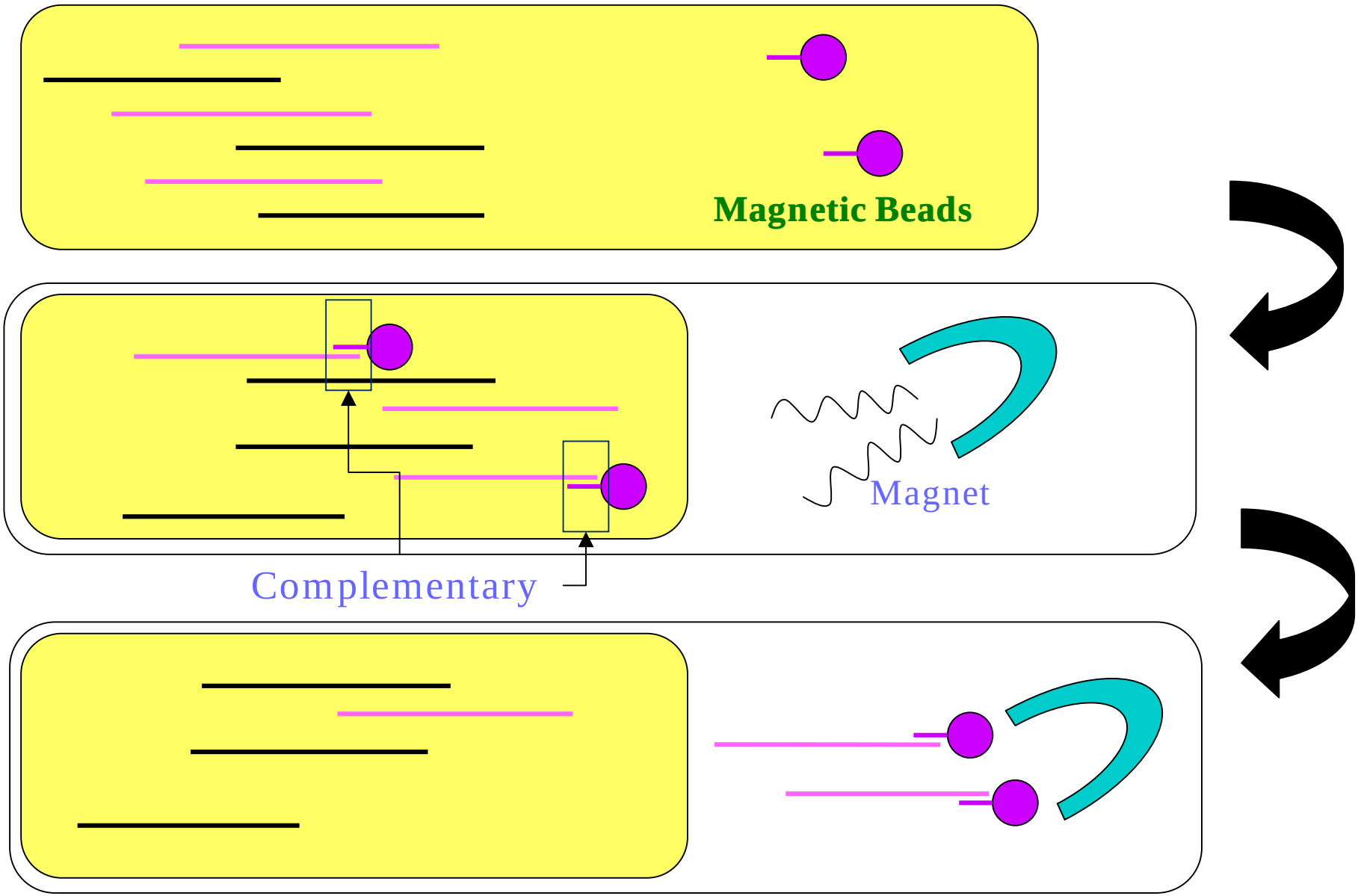
Initialization

- Want to create input set with every possible combination
- Start with memory strands with no bits set
- For each of the input bits:
  - Separate DNA into equal portions
  - Mix excess sticker solution with one portion
  - Combine separated portions
  - Cause stickers to release from memory strands
  - Allow to recombine randomly
- If done perfectly with 2^L (L=N/M) strands, creates any given combination with probability 63%
  - Probability decreases if operations not perfect
  - Can use more DNA to increase probabilities

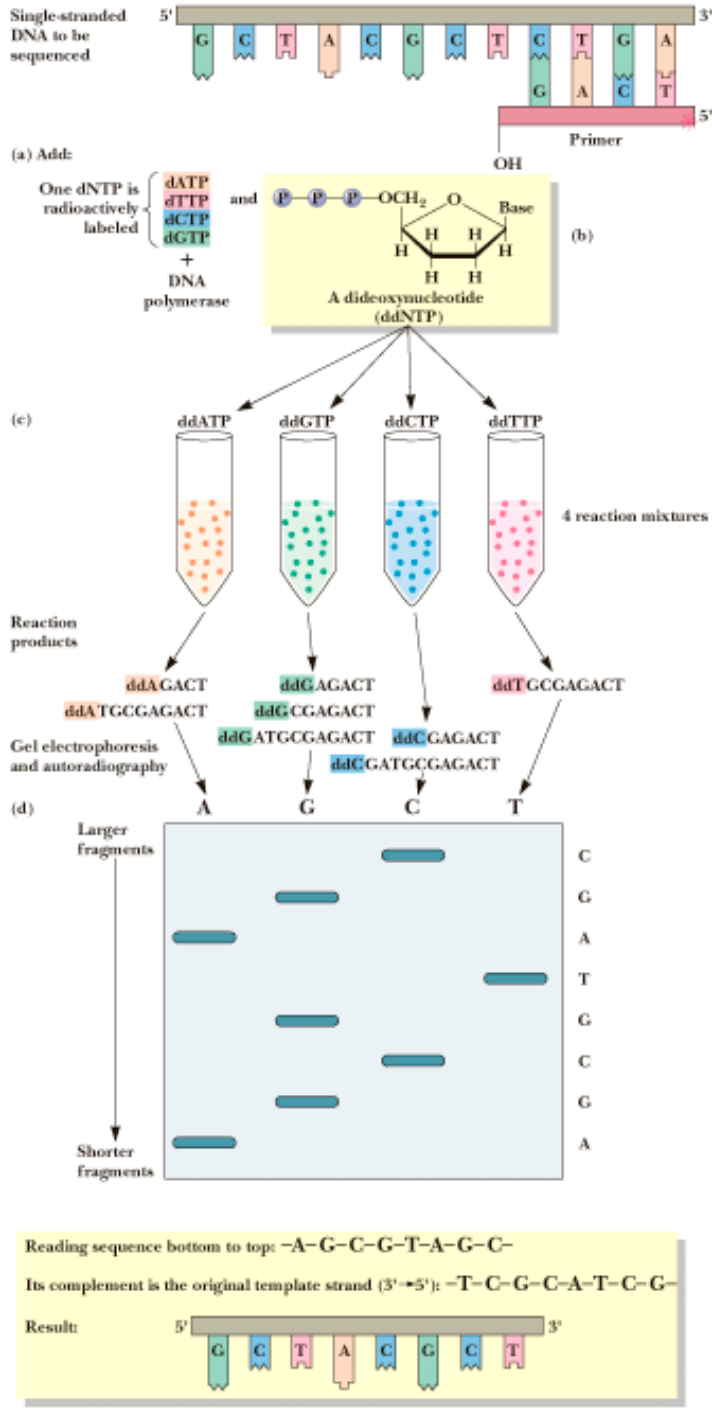
DNA Hybridization & Ligation

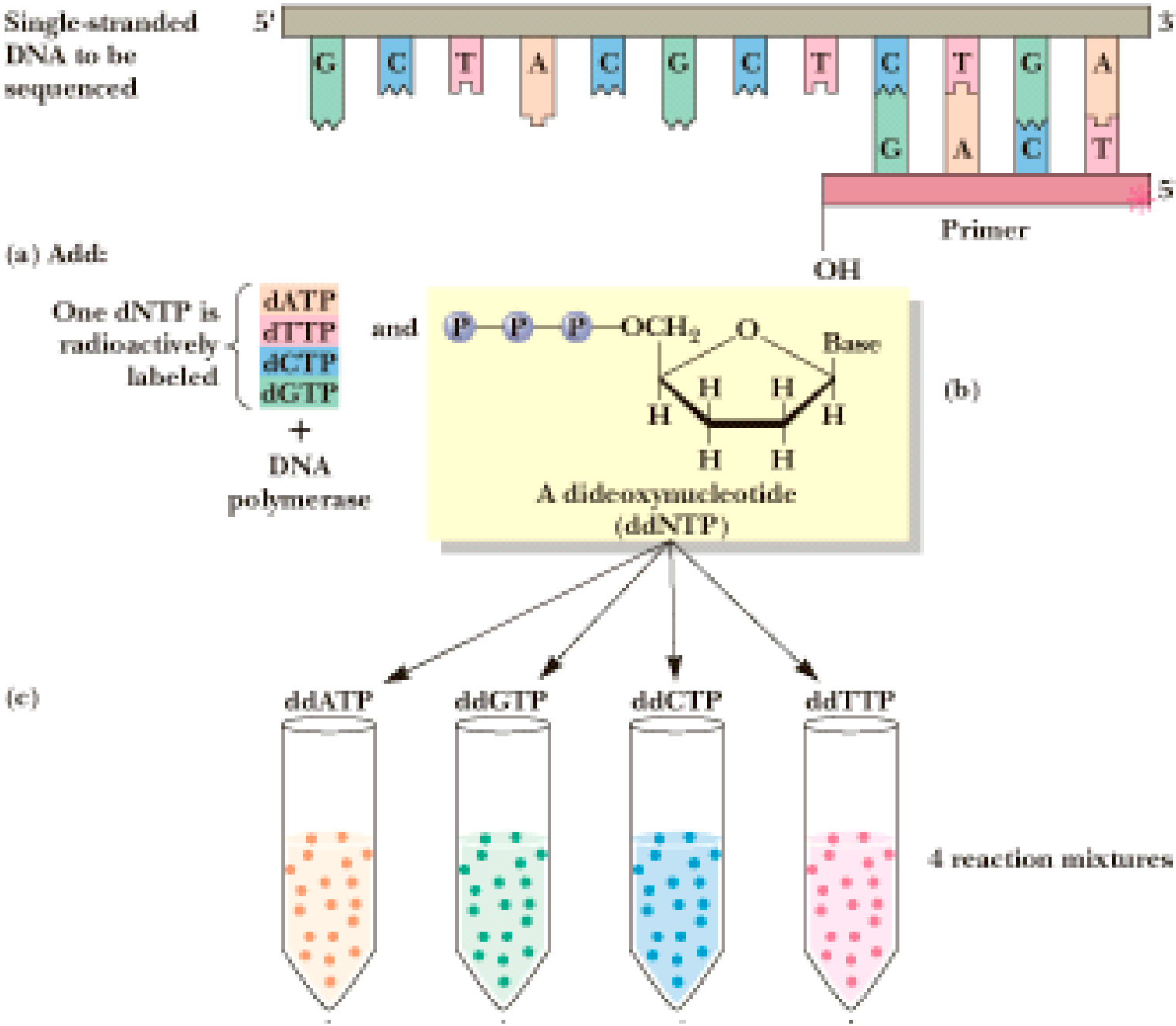


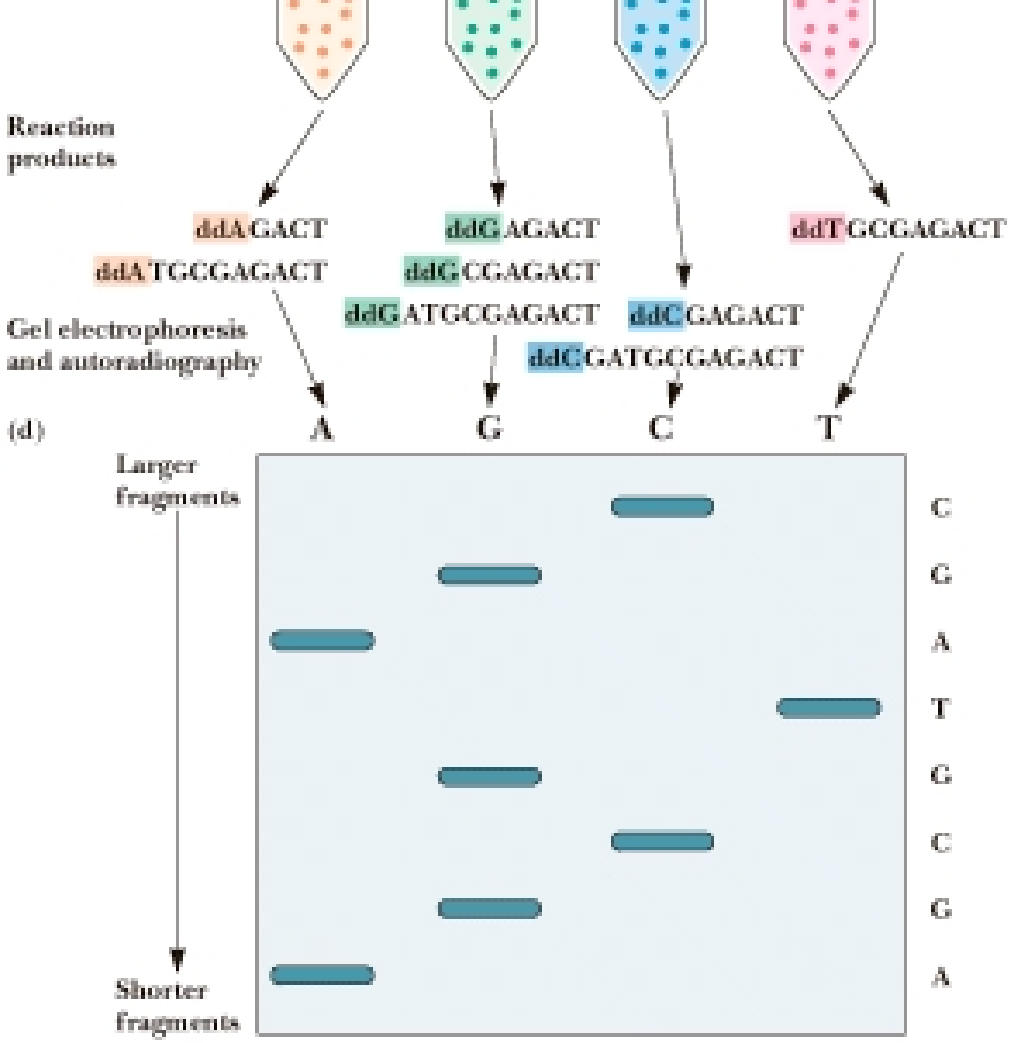
Bead Separation



DNA Sequencing



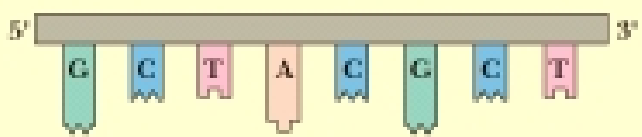

Single-stranded DNA to be sequenced
5'
3'
G C T A C G C T C T G G A
G A C T
5'
Primer
OH
(a) Add:
One dNTP is radioactively labeled
dATP
dTTP
dCTP
dGTP
+
DNA polymerase
and
P P P OCH2 O Base
H H H H H H
A dideoxynucleotide (ddNTP)
(b)
(c)
ddATP
ddGTP
ddCTP
ddTTP
4 reaction mixtures



Reading sequence bottom to top: –A–G–C–G–T–A–G–C–

Its complement is the original template strand (3'→5'): –T–C–G–C–A–T–C–G–

Result:



Operations on a stickers machine

- Simple operations:
  - *merge*,
  - *select*,
  - *detect*,
  - *clean*.
  - **Separate:** Extract all DNA with a particular bit set from a sample
  - **Merge:** Mix two samples
  - **Detect:** Determine if at least one molecule of DNA left in a tube
  - **Amplify:** Replicate a tube of DNA exactly
- → Tubes are considered (cylinders with two entries)

Future stickers machine

- **Implementable models**
  - Branching Programs (Nondeterministic as well if we have amplify)
  - Turing machines are equivalent to branching programs, modulo size
    - Poly-size BP = log-space non-uniform TM
  - Big concern: constants and orders of growth
    - For conventional computer, "polynomial" is often good enough
    - But here we need a much tighter bound when each step takes a long time
- However for a mere computation (DES):
  - Great number of tubes is needed (1000).
  - Huge amount of DNA needed as well.
- Practically no such machine has been created….

  → Too much engineering issues.

Work is ongoing

DNA Computing

Why Try This New Stuff ?

- **We will need a dramatically new technology**:
  - to overcome some CMOS limitations,
  - to offer new opportunities.
- Problems like:
  - learning,
  - pattern recognition,
  - fault-tolerant system,
  - large set search algorithms

  are intrinsically very difficult to solve even with fast evolution of CMOS.
- Hope of achieving ***massive parallelism***.

Why Try DNA Computing?

- 6.022×10^{23} molecules / mole
- Massively Parallel Search of All Possibilities
  - Desktop: 10^9 operations / sec
  - Supercomputer: 10^{12} operations / sec
  - **1 μmol of DNA:** 10^{26} reactions
- Favorable Energetics: Gibb's Free Energy

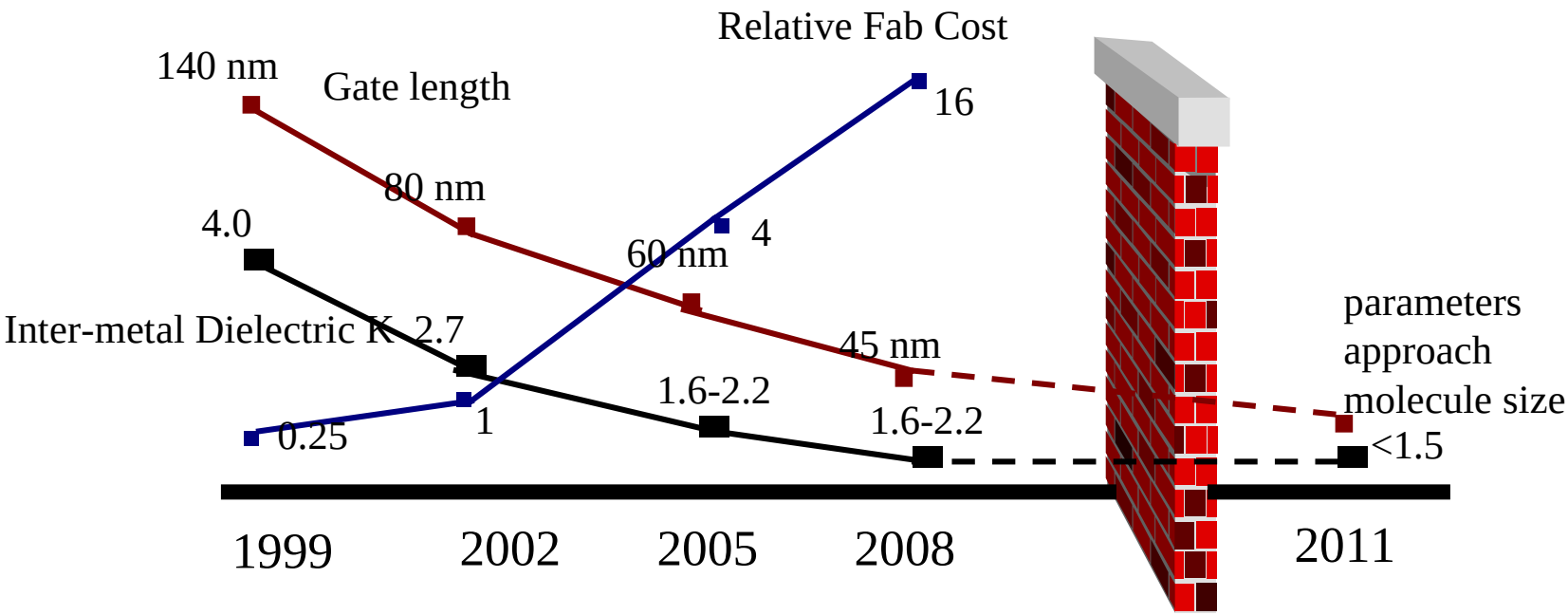
  $\Delta G = -8 kcal \text{ mol}^{-1}$

  - 1 *J* for 2×10^{19} operations
- **Storage Capacity:** 1 bit per cubic nanometer

Why Try DNA Computing?

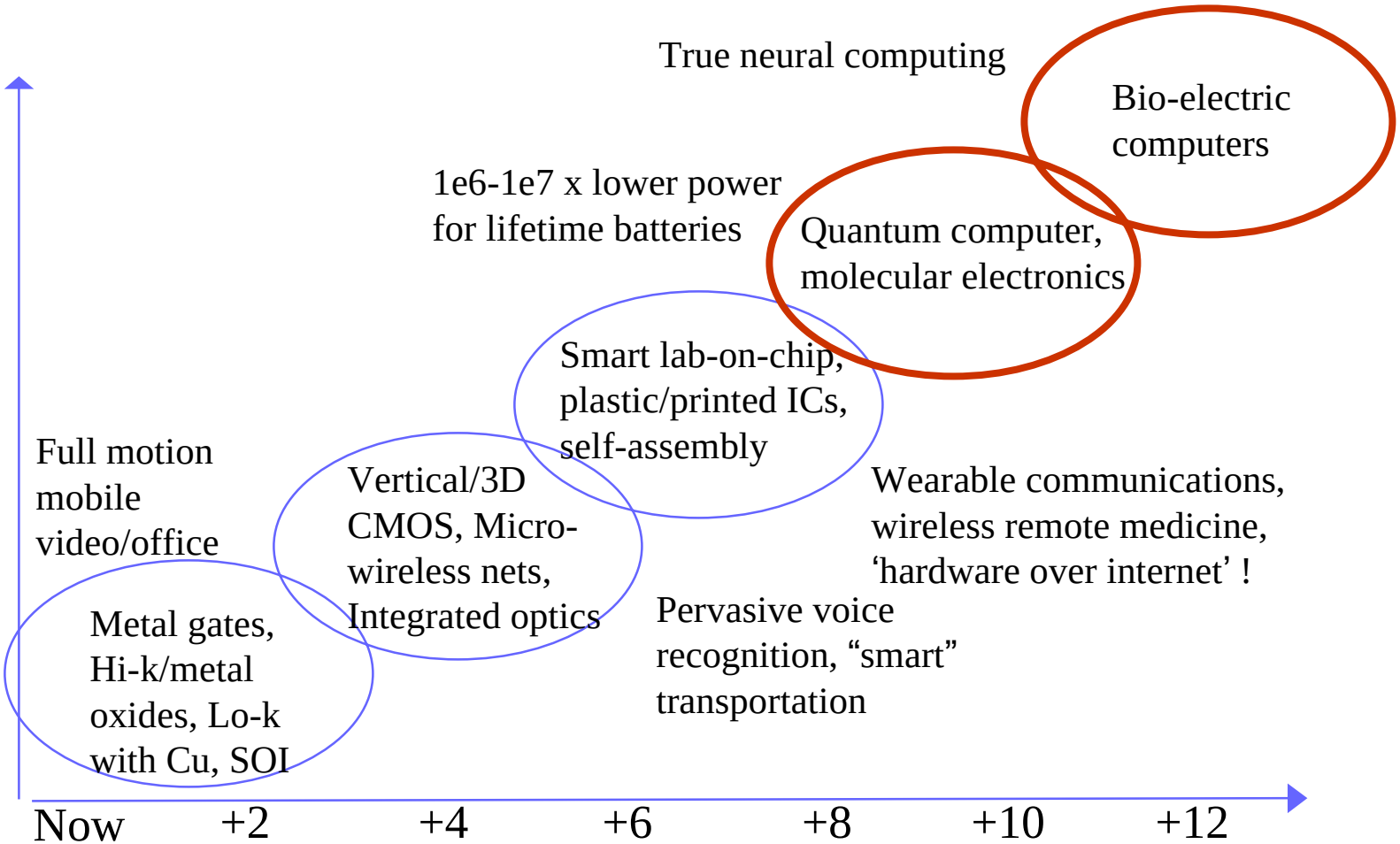
- The fastest supercomputer **versus** DNA computer:
  - 10^6 op/sec **vs**. **10^{14} op/sec**
  - 10^9 op/J **vs. 10^{19} op/J** (in ligation step)
  - 1bit per 10^{12} nm^3 **vs**. **1 bit per 1 nm^3** (video tape **vs**. molecules)

Known CMOS limitations



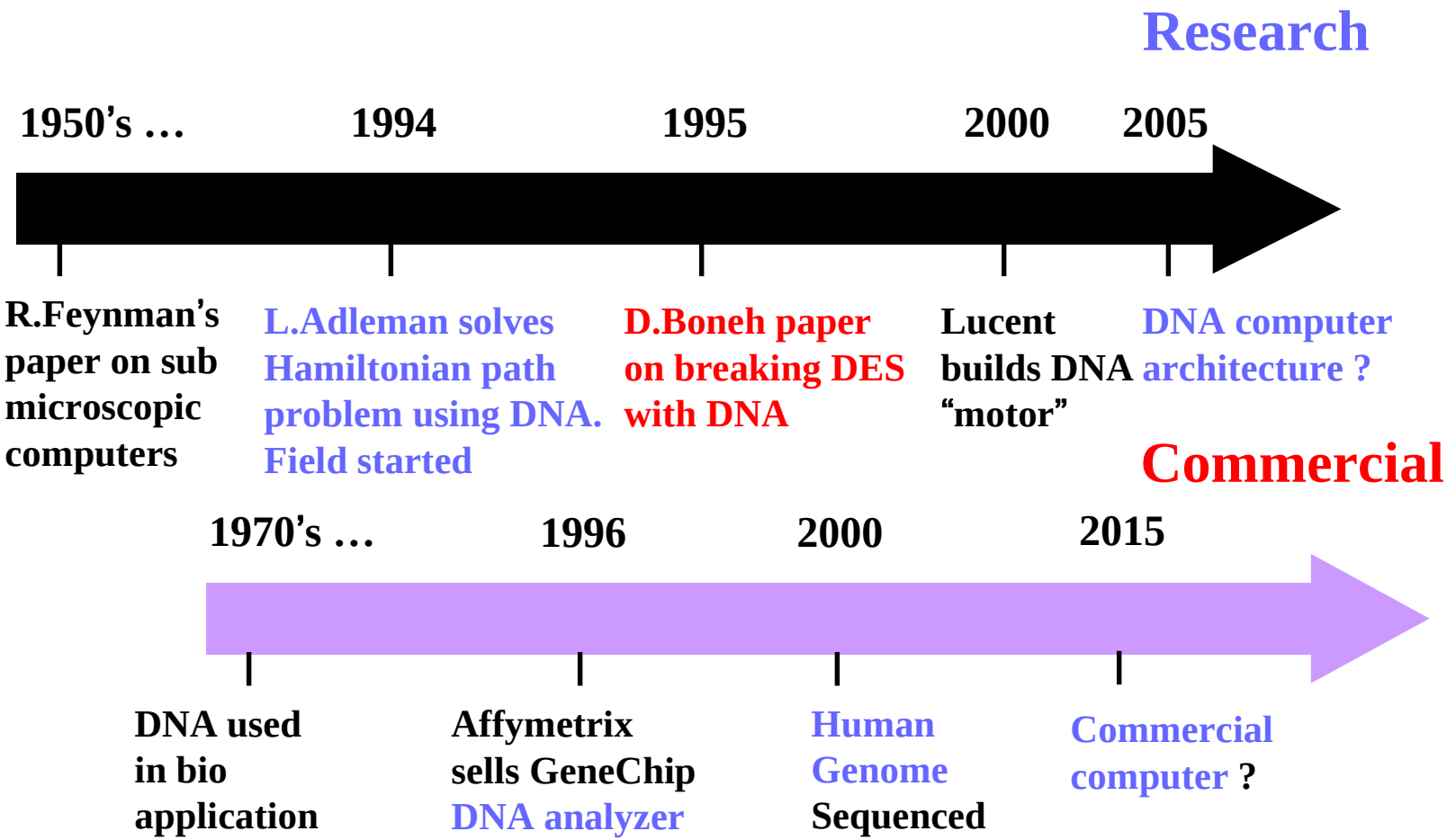
Source: Texas Instruments and ITRS IC Design Technology Working Group

Future Technology Enablers



Source: Motorola, Inc, 2000

Historical Timeline of DNA computing



What's Up to Date

- Research still in **very early stages** but promise is big
- First groundbreaking work by Adleman at USC only in 1994.
- No commercialization in sight within **~10 years**
- Main work on settling for a logic **abstraction model**, ways of using DNA to compute
- **Research sponsored** by universities (Princeton, MIT, USC, Rutgers, etc) and NEC, Lucent Bell Labs, Telcordia, IBM
- One way or another, DNA computing will have a significant impact

DNA Computers vs. Conventional Computers

DNA-based computers	Microchip-based computers
slow at individual operations	fast at individual operations
can do billions of operations simultaneously	can do substantially fewer operations simultaneously
can provide huge memory in small space	smaller memory
setting up a problem may involve considerable preparations	setting up only requires keyboard input
DNA is sensitive to chemical deterioration	electronic data are vulnerable but can be backed up easily

Practical Issues

- **Execution time?**
  - 6719 steps
  - At 1 day/step (grad student), takes 18 years
  - At 1 hour/step (simple machine), takes 9 months
  - At 1 minute/step (really snazzy machine), takes 5 days
- **Error issues**
  - Consider 63% as a “reasonable” chance of getting a correct result
  - With error rates of 10^{-4} per operation, need 1.4 grams DNA, expect .008 distractors/run
  - With error rates of 10-3 per operation, need 580 grams DNA, expect 3.2 distractors
  - With error rates of 10-2/operation, need 1.5×10^{29} grams DNA, 8.3×10^{26} distractors
    - Something like 23x the mass of the Earth

Practical Issues: reliability

- Mostly relate to the fact that DNA processing is unreliable
  - Authors suggest techniques to improve reliability
  - Can use redundancy in the initial sample to reduce the consequences of errors
  - One approach is to view the biological computing part as a "filter" that reduces the number of possibilities
    - For NP-complete problems, can then use electronic techniques to check results of the biological step
- Authors propose prototype **DNA computing workstation**
  - Mainly intended to explore concepts, not to be built

Is Biological Computing Practical?

- **Still really early**
  - Couple real experiments have been run
- **Issues**
  - Reliability – current processing technologies are nowhere near as reliable as required
  - Automation – need to design machines that can perform these computations
  - Basic operation time – makes a big difference, current systems are too slow for practicality
- **Applications**
  - Best suited for NP-complete problems that are amenable to brute force
  - Issue in that the size of the problem is limited by physical issues
    - For example, can probably defeat DNA-based code breaking by using longer keys

DNA versus Silicon

- Sequentially, DNA can be replicated at 500 base pairs/sec = 1000 bits/sec
- Replication enzymes can start making 2^{nd} copy of DNA even before 1^{st} is done
- Replication rates rises by 1000bits/sec or 2^N (N:#iterations)
  - N=10: 1Mbit/sec; N=30: 1000Gbit/sec

DNA versus Silicon

- von Neumann machines are basically sequential—sequence of fetches, decodes and executes
  - *“The inside of a computer is as dumb as hell, but it goes like mad!"*

  -Richard Feynman
- DNA computers
  - Stochastic
  - Not von Neumann

DES Example.

- **Data Encryption Standard**
  - Produces 64-bit ciphertext from 64-bit plaintext using 56-bit key
  - For a while, government was backing for cryptography
    - I believe they've recently backed off from this
  - No known attack much more efficient than brute-force trying of all possible keys
- **Basic Approach**
  - Assumes that we have one plaintext-ciphertext pair, want to determine key
  - Use vast parallelism to try all 2^{56} keys in parallel
  - Initialize DNA strands to represent all keys
  - In parallel, encrypt the plaintext using all possible keys
  - Find and output the key whose encrypted ciphertext matches the ciphertext from the pair

Implementation

- **Memory strands**
  - 11580 nucleotides/strand
  - 20 nucleotides/block = 579 blocks = 579 bits of data
  - 56 blocks hold key, 64 hold ciphertext, rest are temporary storage
- **Initialization**
  - Random initialization of key region, rest set to 0
- **DES Encryption**
  - Series of stages
  - Each stage made up of logic operations on fixed sets of bits
    - Implement logic operations by separating out all combinations of the bits, setting desired bit to one for those with output = 1
  - Stage generates 2 bits of output data, many bits of scratch
    - Clear memory after each stage

Minimal Set Cover Example

- Given B bags, containing some objects of A types, what is the smallest set of bags that contains all types

- **Approach**
  - Create memory strands with K= B+A regions, initialize B regions randomly
    - First B regions represent bags, later A bits will represent objects.
  - For I = 1 to B
    - Separate solution based on bit I
    - Set bits in the A region that of strands with bit I set, corresponding to the objects in bag B
    - Recombine
  - For J = B+1 to B+A
    - Separate out and discard strands that don't have bit J set
  - Perform counting loop to sort remaining strands by how many bits in B region set

DNA Computing: Applications

- Solving NP Complete Problems
- Solving problems that require enumerating an exponential number of paths
- Disease notification?
- Possible use in bio-chips?

DNA Computing: Opinion

- Currently infeasible for large scale applications
  - Need to automate process
  - Time-expensive encoding
  - Cost also high (PCR equipment,etc)…
  - Noisy – uncertain results. Larger strings more noisy.
- Need DNA-chip architecture
- Possible future in disease detection

Pros

- DNA processors take much shorter time to perform computations too large to be run on electronic supercomputers.
- can solve more types of problems than electronic supercomputers.
- the potential for information storage in molecular computers follows the same trend as speed and efficiency; DNA processors has an information storage density of 1 bit per cubic nanometer-a trillion times less space compare to 1 bit per 1012 cubic nanometers information storage density with storage media of today, such as videotapes.

Pros (Continued)

- In energy efficiency, DNA processors can perform 2*1019 power operations using one joule of energy compare to 1010 operations with supercomputer.
- In speed, DNA computers can perform 1000 operations per second more than the fastest supercomputers(1012 operations per second) which means thousand million times slower than the DNA computers.

Pros (Continued)

- DNA computers use cheap, clean and readily available biomaterials rather than costly and often toxic materials that go into traditional microprocessors.

Cons

- DNA processors take longer time to multiply two 100 digit integers(or other simple problems) than electronic supercomputers do.
- every operation with DNA computer, is somewhat random, that is, unlike the transistors in pentium, which reliably compute what they're supposed to

Cons (Continued)

- The components in the DNA computer are probablistic. for example, if the answer produced by Pentium is 1, with DNA the answer is 1 90% of the time and 0 10% of the time.
- DNA computing is cumbersome because it is not entirely mechanized.
- It can be costly to make longer strands of DNA that encode more information, and programming DNA computers themselves can prove difficult because of the problems still inherent in manipulating DNA.

The Future of DNA Computing

- Problems:
  - Controlling DNA is not as easy as controlling electrons
  - Errors (stochastic method)
    - Error rate for 10 iterations: 1%
    - Error rate for 100 iterations: 63%
- 3-Sat problem was a huge step forward comparing to Adleman's.

- DNA Manipulation technology has rapidly improved in recent years, and future advances may make DNA computers more efficient.
- The University of Wisconsin is experimenting with chip-based DNA computers.
- Promising applications include areas of encryption, genetic programming, language systems, and algorithms
- DNA "fingerprinting" may be one of the most promising applications for DNA computing.

The Future of DNA Computing

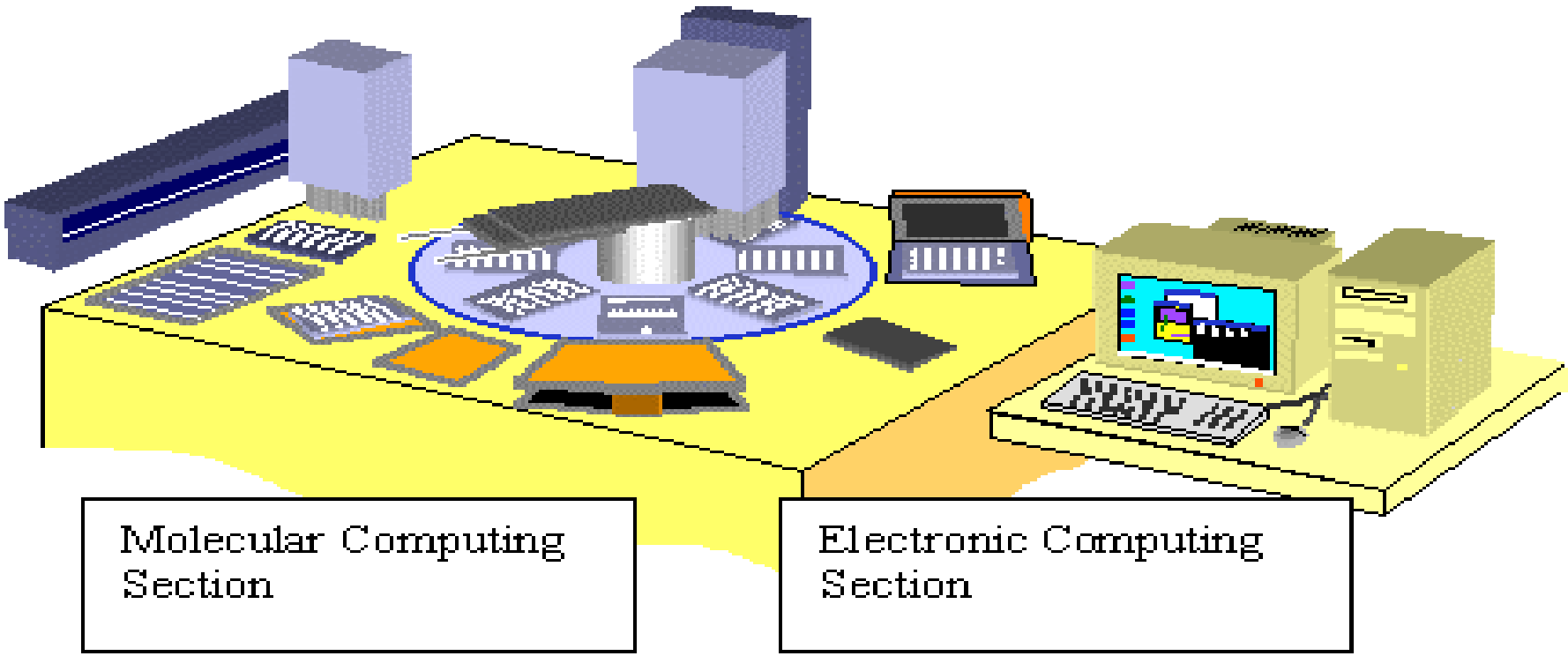
- DNA computing started only in 1994.
- **Celera** can now do in 1 day what a person used to take 5 years to do
- Government and industry (pharmaceutical) spending

  on research has been increasing over the years
- **DNA chips** are out in the market today:
  - Affymetrix Genechip
- **Olympus Inc.** has created a working **DNA computer**

The Olympus DNA Computer

- Solves genetic problems
- Problems that otherwise take 3 days can be solved in 6 hours
- 96 wells with 100 DNA strands each
- Molecular Computing Unit:
  - DNA used for input, output and internal operations
- Electronic Computing Unit
  - Interpretation of results of Molecular Computing Unit

The Olympus DNA Computer



Molecular Computing Section

Electronic Computing Section

Computations performed with DNA as input/output data; DNA reactions, capture of DNA results and DNA detection all performed automatically.

Information processing program performed; output includes DNA reaction calculations and an analysis of results.

The Olympus DNA Computer

Conclusions

- DNA computing uses DNA molecules as storage material and their liquid-phase biochemical reactions for processing.
- DNA computing technology has many interesting properties, including
  - Massively parallel, solution-based, biochemical
  - Miniaturized, nano-scale, biocompatible
  - High energy efficiency
  - High memory storage density
- DNA computing is in a very early stage of development, but seems very promising, especially for solving the class of problems which are inherently difficult for solid-state silicon computers.

Conclusions

- Currently, molecular computing is a field with a great deal of potential, but few results of practical value.
- In the wake of Adleman's solution of the Hamiltonian path problem, there came a host of other articles on computation with DNA.
- However, most of them were purely theoretical.

Conclusions

- Currently, a functional DNA "computer" of the type most people are familiar with lies many years in the future.
- But work continues: in his article Speeding Up Computation via Molecular Biology <ftp://ftp.cs.princeton.edu/pub/people/rjl/bio.ps> Lipton shows how DNA can be used to construct a Turing machine.
- While it currently exists only in theory, it's possible that in the years to come computers based on the work of Adleman, Lipton, and others will come to replace traditional silicon-based machines.

Conclusions

Research Projects/Groups

- MIT, Caltech, Princeton University, Bell Labs
- EMCC (European Molecular Computing Consortium)
  is composed of national groups from 11 European countries
- BioMIP Institute (BioMolecular Information Processing)
  at the German National Research Center for
  Information Technology (GMD)
- Molecular Computer Project (MCP) in Japan
- Leiden Center for Natural Computation (LCNC)
- Molecular Evolutionary Computing (MEC) Project in Korea, Seoul National Univ.

References

- Leonard M. Adleman, *Molecular Computation of Solutions to Combinatorial Problems*, Science 226, 1994
- Leonard M. Adleman, *Solution of a 20 Variable 3-SAT Problem on a DNA Computer,* Science1069 197, 2002
- Will Ryu, *DNA Computing: A Primer*
- Michael Conrad, *Molecular Computing: The Lock-Key Paradigm*, IEEE Computer November 1992
- Yali Friedman, *DNA Based Computers*, www.dna2z.com

Books

- Cristian S, Calude and Gheorghe Paun, ***Computing with Cells and Atoms: An introduction to quantum, DNA and membrane computing***, Taylor & Francis, 2001.
- Pâun, G., Ed., ***Computing With Bio-Molecules: Theory and Experiments***, Springer, 1999.
- Gheorghe Paun, Grzegorz Rozenberg and Arto Salomaa, ***DNA Computing, New Computing Paradigms***, Springer, 1998.
- C. S. Calude, J. Casti and M. J. Dinneen, ***Unconventional Models of Computation***, Springer, 1998.
- Tono Gramss, Stefan Bornholdt, Michael Gross, Melanie Mitchell and thomas Pellizzari, ***Non-Standard Computation: Molecular Computation-Cellular Automata-Evolutionary Algorithms-Quantum Computers***, Wiley-Vch, 1997.

Other References

- Boneh, Dan, Christopher Dunworth, Richard J. Lipton and Jiří Sgall. "Making DNA Computers error resistant." <http://crypto.stanford.edu/~dabo/papers/bioerror.ps.gz> (24 Feb 2002)
- "Error Analysis." <http://www.mills.edu/ACAD_INFO/MCS/CS/S00MCS125/DNAComputing/error.html> (24 Feb 2002)
- Fufa, Duretti. "*DNA Replication and Repair.*" *SparkNotes*. <http://www.sparknotes.com/biology/molecular/dnareplicationandrepair> (24 Feb 2002)
- Horton, H. Robert, et.al.. Principles of Biochemistry. Upper Saddle River, NJ: Prentice Hall, 1996
- Langohr, Kris. "Sources of Error in DNA Computation." <http://www.csd.uwo.ca/~jamie/.Refs/Courses/CS881/presentation.html> (23 Feb 2002)
- Malacinski, Geroge M., and David Frefelder. Essentials of Molecular Biology. Boston, MA: Jones and Bartlett Publishers, 1998
- Raven, Peter H., and George B. Johnson. Understanding Biology. St. Louis: Mosby Year Book,

Other References

- Ryu, Will. “DNA Computing: A Primer.” ArsTechnica April 2000 <http://arstechnica.com/reviews/2q00/dna/dna-1.html> (19 Feb 2002)

- Silberberg, Martin. Chemistry: The Molecular Nature of Matter and Change. St. Louis: Mosby, 1996

- Watson, J.D., and F.H.C. Crick. “A Structure for Deoxyribonucleic Acid.” Nature 2 Apr 1953 <http://biocrs.biomed.brown.edu/Books/Chapters/Ch 8/DH-Paper.html> (24 Feb 2002).

- Watson, James D. The Double Helix. New York: Atheneum, 1968.

- Wisz, Mike. “DNA Computing.” SciTech Online Winter 1996 <http://www.englib.cornell.edu/scitech/w96/DNA.html> (19 Feb 2002)

- Zumdahl, Steven S. Chemistry. Lexington, MA: D.C. Heath and Co., 1993.

Web Resources

- European Molecular Computing Consortium (EMCC): http://www.csc.liv.ac.uk/~emcc/
- BioMolecular Information Processing (BioMip): http://www.gmd.de/BIOMIP
- Leiden Center for Natural Computation (LCNC): http://www.wi.leidenuniv.nl/~lcnc/
- Biomolecular Computation (BMC): http://bmc.cs.duke.edu/
- DNA Computing and Informatics at Surfaces: http://www.corninfo.chem.wisc.edu/writings/DNAcomputing.html
- SNU Molecular Evolutionary Computing (MEC) Project: http://bi.snu.ac.kr/Research/

http://cbit.snu.ac.kr/

- http://www.cs.wayne.edu/~kjz/KPZ/NaturalComputing.html
- http://bi.snu.ac.kr/
- http://dna2z.com/dnacpu/dna.html
- http://www.intermonde.net/adn/liens.html