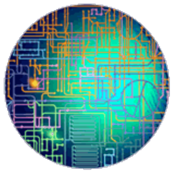


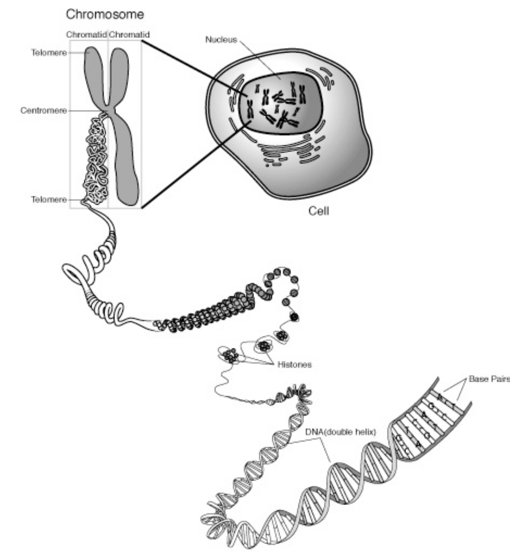
Computing with DNA

Can we really make a computer out of DNA?

- What is DNA?
- How to “compute” with DNA
- The traveling salesman: an important problem that could use some help
- Prospects for future DNA computers

DNA Carries Genetic Information

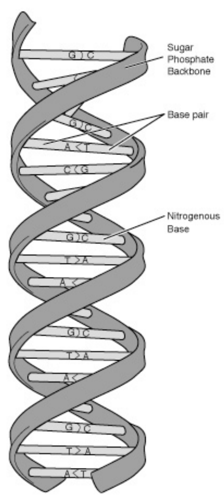
- DNA is found in the nucleus of a cell
- Very compact
  - in human cells over 8 billion “rungs” on this ladder fit into the nucleus
- In the terminology of information theory, each rung carries two “bits” of information
  - much more information per mm^3 than electronic memory chips



Close-up View

- This picture shows a DNA double helix in more detail
  - the four building blocks are known as “bases”
  - names abbreviated A, T, C, and G
  - the chemical structure of the bases allows A to connect to T and C to connect to G

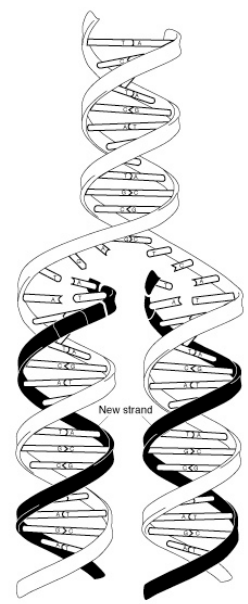
A ⇔ T
C ⇔ G



Replication

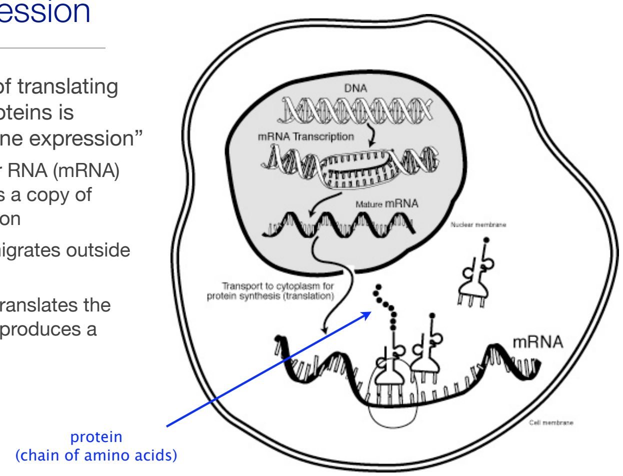
- When cells divide, the DNA is copied
  - each “daughter cell” has an exact copy of the information from the parent cell
  - the helix unwinds, and new strands form opposite the bases of the original strands
- When a strand of DNA connects to a complementary strand it is through a process known as “hybridization”

A ⇔ T
C ⇔ G



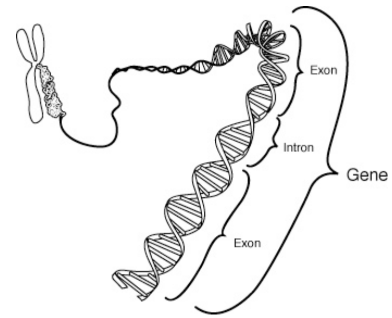
Gene Expression

- The process of translating genes into proteins is known as “gene expression”
  - a messenger RNA (mRNA) molecule has a copy of the information
  - the mRNA migrates outside the nucleus
  - a ribosome translates the information, produces a protein



Genes are Small Sections of DNA

- Very little of the DNA in human cells carries protein-coding information
  - 1% of the DNA has the “blueprints” used by the cell to make proteins
  - some of the rest (intergenic DNA) may has information used by other cellular processes
  - for the remainder -- ??

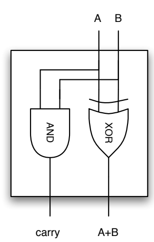


Can We Build a Computer from DNA?

- What is a “computer”?
  - something that stores data/information
  - something that is able to process information to produce new information
- Modern electronic computers store and process information represented as binary numbers

5 + 3 = 8

```
  1 1
  1 0 1
+ 0 1 1
-------
1 0 0 0
```



Bits Can Also Represent Text

- The ASCII code uses 8 bits to represent one letter
  - E = 01000101
  - u = 01100101
  - g = 01100111
  - e = 01100101
  - requires 6 x 8 = 48 bits to represent “Eugene”
- ASCII is limited to the Western (Roman) alphabet
  - 26 letters, 10 digits, some punctuation
  - UNICODE uses 16 bits per symbol, includes many European and Asian alphabets, special symbols, and more
  - α, β, ∃, ∞, é, å, あ, ...

DNA Can Encode Numbers and Text

- There are four bases (A, T, C, G)
- Each could represent two bits, e.g. A = 00, T = 01, C = 10, G = 11
  - E = 01000101 = TATT
  - u = 01100101 = TCTT
  - g = 01100111 = TCTG
  - e = 01100101 = TCTT
- Store the word "Eugene" in 6 x 4 = 24 bases

A Computation Using DNA

- So we know how to represent information with DNA
- Can we process information?
  - Is it possible to implement an algorithm that uses strands of DNA as input and produces new DNA representing the output?
- In 1994 computer scientist Leonard Adleman at USC used DNA to solve a problem known as the Hamiltonian Cycle problem
  - Adleman, L. "Molecular computation of solutions to combinatorial problems." *Science*, vol 266, pp 1021-1024, 1994.

Aside: Hamiltonian Cycles

- A Hamiltonian cycle is a path that connects all the nodes in a figure
  - visit each node exactly once
  - return to the starting point
- Not too hard to solve if
  - figure has a regular geometry, and
  - small number of nodes
- Very difficult in the general case
- A figure with n points can have up to n! paths
  - 5! = 120
  - 10! = 3,628,800
  - 15! = 1,307,674,368,000
  - 20! = 2.43×10^{18}

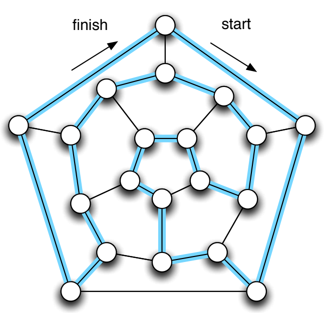
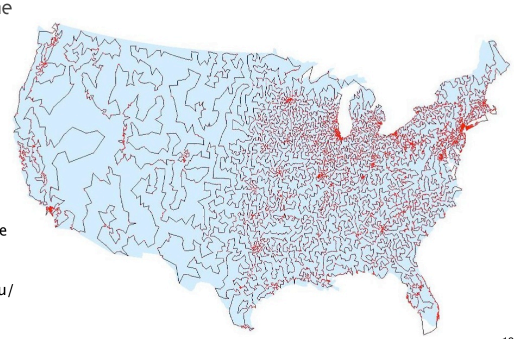


Figure with 20 nodes

A Similar Problem: The Traveling Salesman

- The traveling salesman problem (TSP) is very closely related to the Hamiltonian cycle problem
  - suppose we have a list of n cities
  - the goal is to define a tour that visits all n cities and returns to the starting place
  - visit each city only one time
- Here the goal is to find a *minimal cost* tour
  - at right: a tour of 13,500 US cities



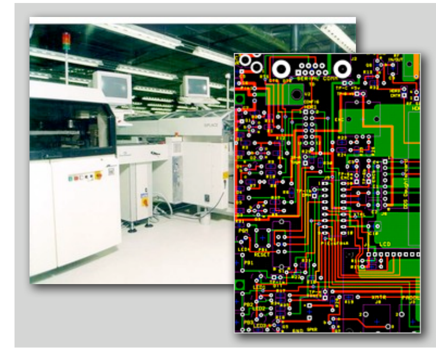
Images and examples from the Traveling Salesman Page at Georgia Tech:
http://www.tsp.gatech.edu/

Traveling Salesman (cont'd)

- Cost can be defined as driving time, distance, air fare, ...
  - assume the cost of going from X to Y is the same as going from Y to X
- Although it sounds simple this is a very hard problem to solve
  - some simplifications and transformations can reduce the number of steps to "only" 2^n
  - each time a city is added to the list the time to find a tour doubles
  - for a tour of 20 cities the program might have to check 1 billion combinations
- Some figures from TSPlib++ (an "industrial strength") solver:
  - 5.5 hours for 10,000 cities
  - estimate over 6,000 years for 25,000 cities

Do We Really Need to Solve This Problem?

- The idea that anyone would really plan a road trip to 13,000 cities is a bit silly
- But the TSP is identical to several important "real world" problems:
  - transportation: school bus routes, service calls, delivering meals, ...
  - manufacturing: an industrial robot that drills holes in printed circuit boards used in computers, video and stereo, almost anything electronic
  - communication: planning new telecommunication networks
  - biology: genetic markers on chromosomes / reassembly



TSP with DNA

- How did Adleman solve this sort of problem with DNA?
- Step 1: use DNA to represent names of cities
  - Eugene = TATTTCTTTCTGTCTTTCGCTCTT
  - Corvallis = TATGTCTTTGACTGTCTCACTCGA...
  - make many copies of strands of DNA for each city

A = 00
T = 01
C = 10
G = 11

- Step 2: make "roads" by making new strands
  - road DNA is the *complement* of city DNA
  - a road connecting A to B has the second half of A and the first half of B

last half of "Eugene"   first half of "Corvallis"

```
...TCTTTCGCTCTT     TATGTCTTTGAC...
   ||||||||||||     ||||||||||||
   AGAAAGCGAGAA ... ATACAGAAACTG
```

"road" matching last part of Eugene and first part of Corvallis

TSP with DNA (cont'd)

- After making the city and road DNA (and lots of copies of each) mix them all together
  - a tour will consist of a long chain of cities connected by roads
  - you will get lots of tours -- most of them wrong
  - may be too short -- come back to the starting point too soon
  - may be too long or include the same city more than once
- After all the DNA is hybridized filter out the incorrect tours and you'll be left with long strands that represent the correct solution(s)
- Adleman was able to find a Hamiltonian cycle in a 7-node graph

The Good News

- Adleman’s paper caused a lot of excitement and raised expectations
- DNA is very compact, and it was relatively easy to make a beaker full of DNA with lots of copies of the “roads” and “cities”
- DNA hybridizes very quickly
- Biggest advantage: all tours are considered in parallel
- Because there were so many copies of the roads and cities, Adleman’s DNA computer did the equivalent of 10^{14} calculations per second
  - aka “100 teraflops”
  - the fastest supercomputer in 1994 could do 35 teraflops
  - the fastest supercomputer in the last “Top 500” list (Nov 2006) does 280 teraflops
  - these supercomputers are very big -- the DNA computer sat on a lab bench

The Bad News

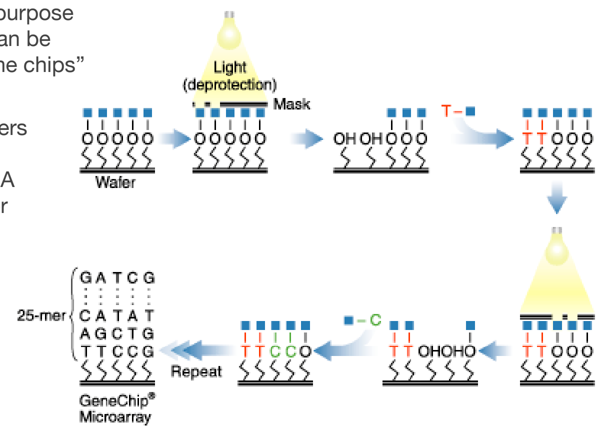
- Adleman’s method is not “scalable” -- like an electronic computer, a DNA computer will have a hard time with a 200-node graph
- By one estimate, to find a tour of 200 cities would require an amount of DNA that would weigh more than the Earth
  - the problem: we need enough copies of the city and road DNA to allow all combinations to form
- This is a classic tradeoff in computer science --
  - a solution always balances time vs space
  - it is often possible to find a fast algorithm if one is allowed to use an infinite amount of space
- Bottom line: the difficulty in TSP is mathematical complexity
  - silicon and DNA are two ways of solving the math problem
  - DNA is not capable of changing the mathematical properties of the problem

More Bad News

- Turning the promise of DNA computing into a reliable and useful technology has proven to be very difficult
- Some of the issues:
  - errors in creating the DNA strands
  - errors in hybridization (i.e. mismatches)
    - these sorts of errors occur all the time in nature -- they are examples of genetic mutations
    - natural systems have many ways of dealing with these errors, but how can we detect them and deal with them in a DNA computer?
  - it takes a long time to set up the calculation and another long time to interpret the results
    - the actual “computing” goes very quickly
    - I/O is another matter altogether
- Bottom line: DNA computers are not likely to be used for general purpose computing any time soon

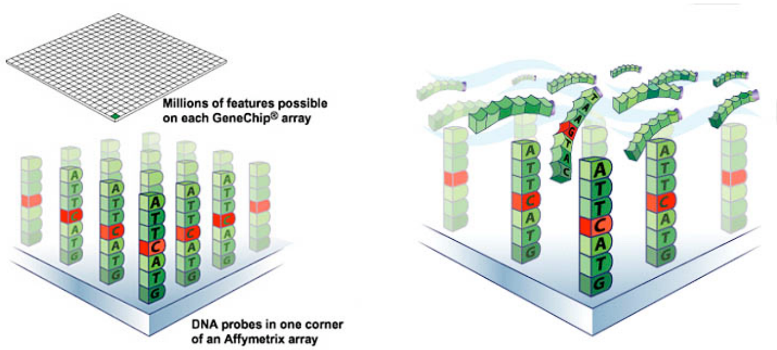
To End on a Positive Note...

- There are special-purpose calculations that can be carried out by “gene chips”
- Companies like Affymetrix and others manufacture chips with strands of DNA attached to a wafer



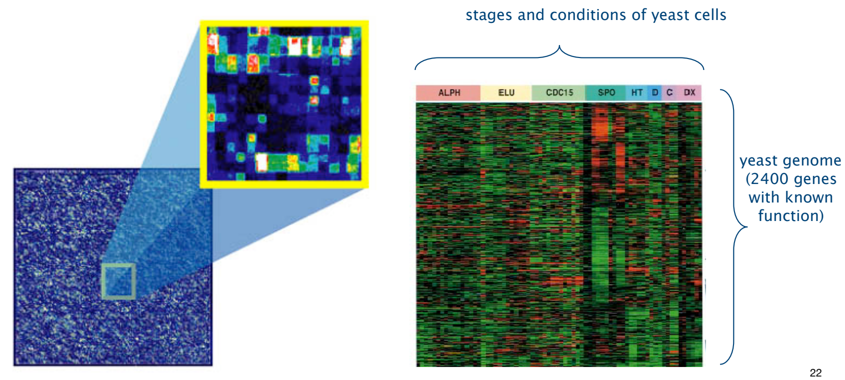
Gene Chip

- The result of the manufacturing process is a “chip” with thousands of strands of specifically engineered DNA
- cDNA (copied from mRNA in a cell) can be “washed over” the chip

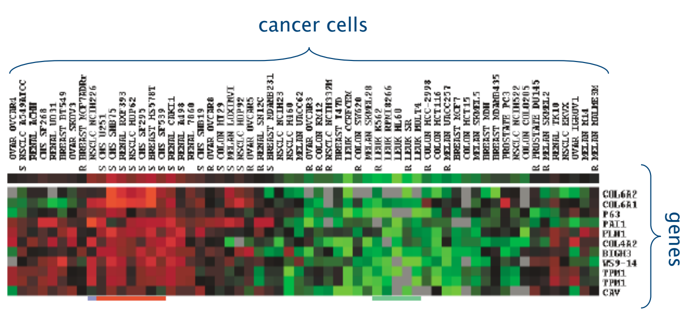


Analyzing the Data

- When the experiment is done, matching cDNA has stuck to the chip
- By looking at which chip cells have cDNA attached one can get an idea of what the cell is doing -- i.e. which genes are active



Finding Genes Active in Cancerous Cells

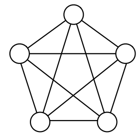


Recap

- DNA is a “polymer” -- long strand made up from smaller building blocks
- A strand of DNA can attach itself to a matching strand of DNA
  - a process called *hybridization*
- It is possible to build artificial strands of DNA using any sequence we want
- These strands can represent numbers, names of cities, ...
- There has been some success in implementing algorithms through hybridization (e.g. connecting two cities by binding complementary DNA)
- DNA computing is far from “ready for prime time” for use in general purpose computing

Questions

- Using the simple code introduced here (A = 00, T = 01, C = 10, G = 11) and a table of ASCII codes (search for "ASCII" on Wikipedia) show how your name would be stored on a gene chip
- Consider the graph at right, with 5 nodes and a connection between each pair of nodes. Are there really 5! = 120 different Hamiltonian cycles in this graph? Do you think a graph with 6 nodes will have 6! = 720 paths?



- Did Adleman's DNA computer solve the Traveling Salesman Problem or the Hamiltonian Cycle Problem?
  - hint: what's the difference between the two problems?