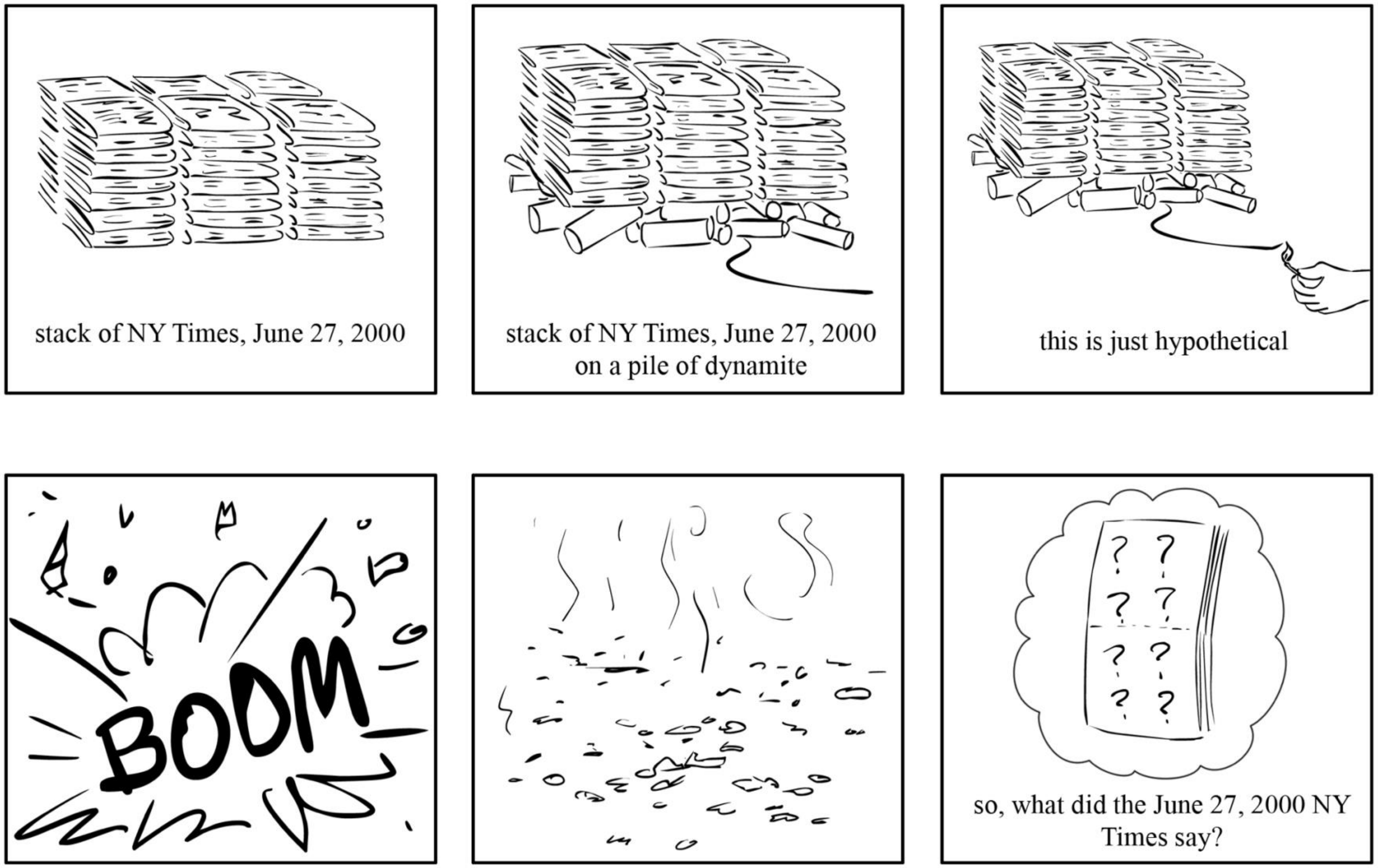
Genome Reconstruction: A Puzzle with a Billion Pieces

Phillip Compeau & Pavel Pevzner
Modified by Alexander Niema Moshiri
University of California, San Diego

The Newspaper Problem


stack of NY Times, June 27, 2000
stack of NY Times, June 27, 2000 on a pile of dynamite
this is just hypothetical
BOOM
so, what did the June 27, 2000 NY Times say?

Section 1: Introduction to Genome Sequencing

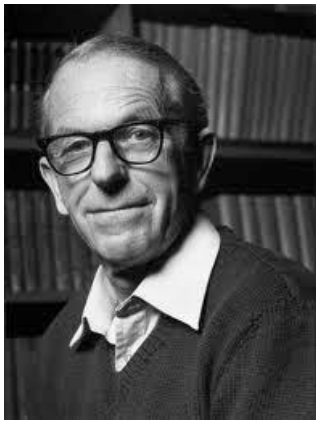
Brief History of Genome Sequencing

- **Late 1970s**: Walter Gilbert and Frederick Sanger develop independent sequencing methods.

- **1980**: They share the Nobel Prize in Chemistry.

- Still, their sequencing methods were too expensive for large genomes: with a $1 per nucleotide cost, it would cost $3 billion to sequence a human genome.

Walter Gilbert

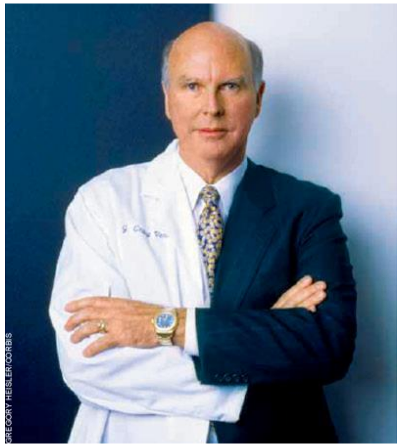

Frederick Sanger

Brief History of Genome Sequencing

- **1990**: The public Human Genome Project, headed by Francis Collins, aims to sequence the human genome.

Francis Collins

- **1997**: Craig Venter founds Celera Genomics, a private firm, with the same goal.



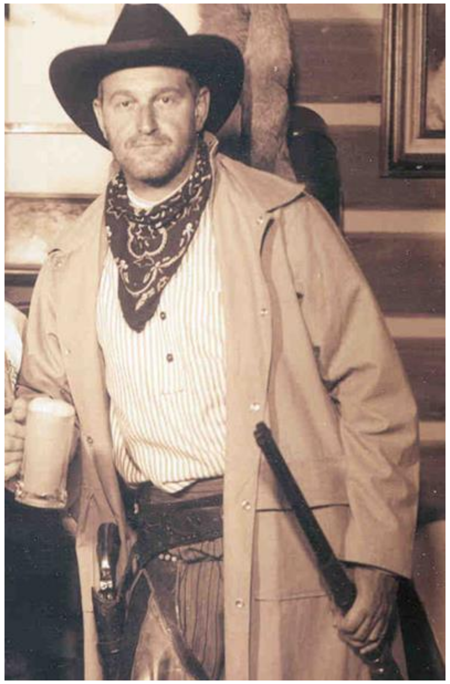
Craig Venter

Brief History of Mammalian Genome Sequencing

- **2000**: The draft of the human genome is simultaneously completed by the (public) Human Genome Consortium and (private) Celera Genomics.

The Eulerian Approach to DNA Sequencing

- **2001**: Pavel Pevzner, Haixu Tang, and Michael Waterman propose an Eulerian Path approach to Genome Assembly

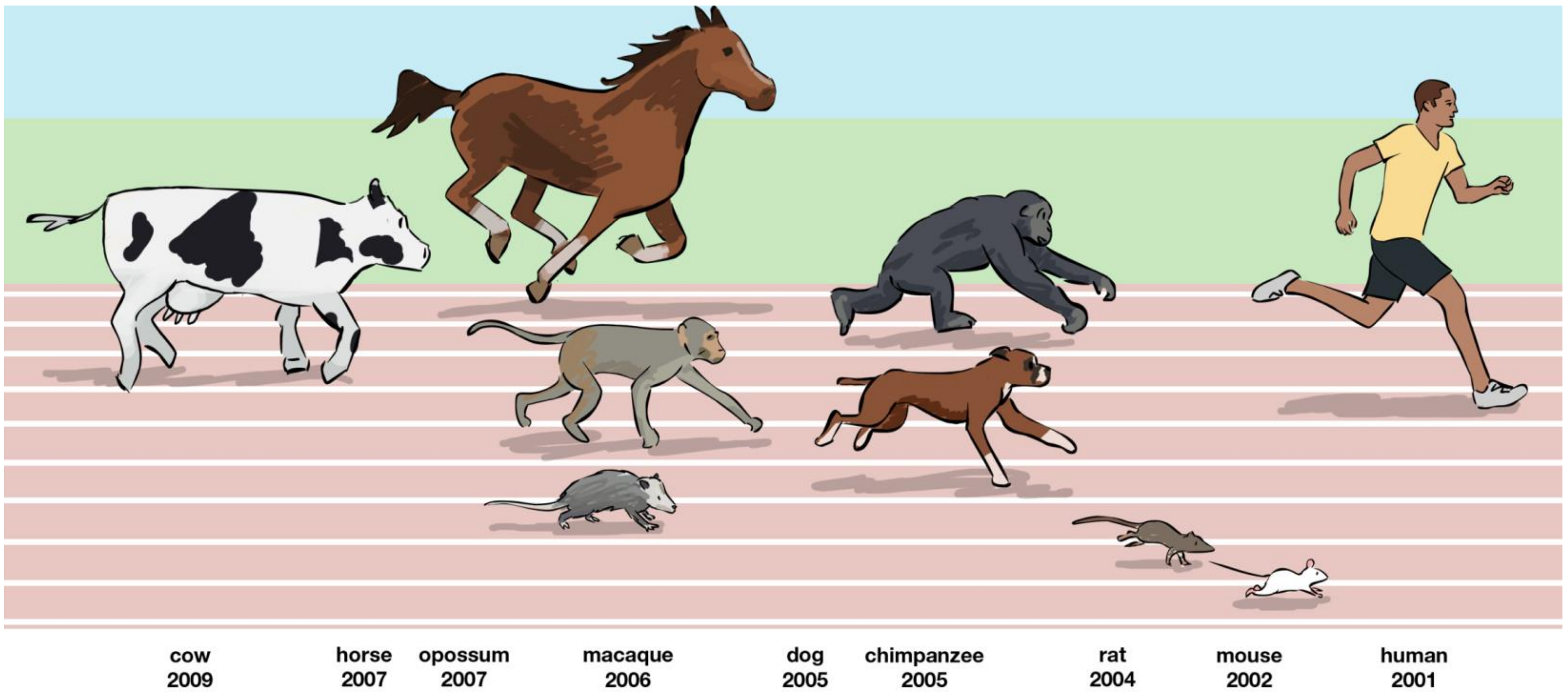

Pavel Pevzner

Haixu Tang

Michael Waterman

Brief History of Mammalian Genome Sequencing

- **2000s**: Many more mammalian genomes are sequenced.



The Arrival of Personal Genomics

- **2010s**: The market for sequencing machines takes off.
  - Illumina reduces the cost of sequencing an individual human genome from $3 billion to $1,000.
  - Complete Genomics builds a genomic factory in Silicon Valley that sequences hundreds of genomes per month.
  - Beijing Genome Institute orders hundreds of sequencing machines, becoming the world's largest sequencing center.
  - 23andMe offers partial genome sequencing for $499.

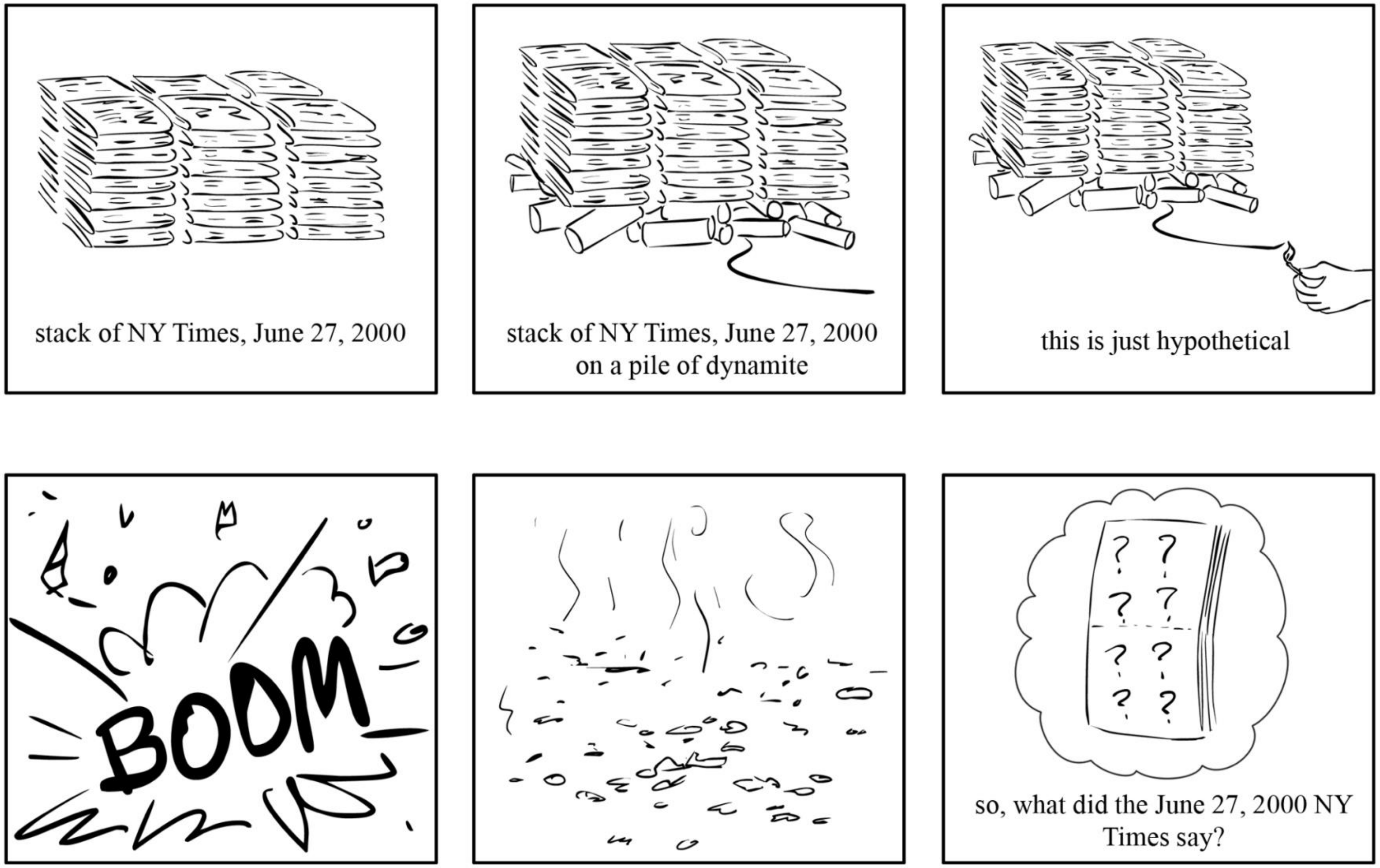
The Future of Genome Sequencing

- **2015+**: Hopefully, sequencing an individual genome will soon become as routine as an X-ray.

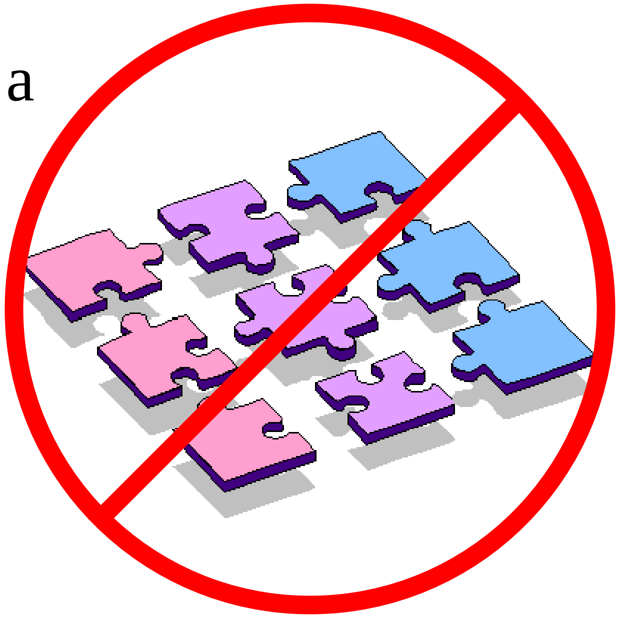
WHAT DOES IT MEAN THAT I HAVE A GENE THAT GIVES ME 'TYPICAL ODDS' FOR HAVING RED HAIR?
I HAVE CHOSEN NOT TO LEARN WHETHER I HAVE A GENE THAT INCREASES THE RISK OF ALZHEIMER'S DISEASE.
I CONSUME BEER, BROCCOLI AND BRUSSELS SPROUTS, EVEN THOUGH I HAVE THE GENE FOR TASTING THEIR BITTERNESS.

Section 2: The Newspaper Problem and Genome Sequencing

Returning to The Newspaper Problem


stack of NY Times, June 27, 2000
stack of NY Times, June 27, 2000 on a pile of dynamite
this is just hypothetical
BOOM
so, what did the June 27, 2000 NY Times say?

The Newspaper Problem as an "Overlap Puzzle"

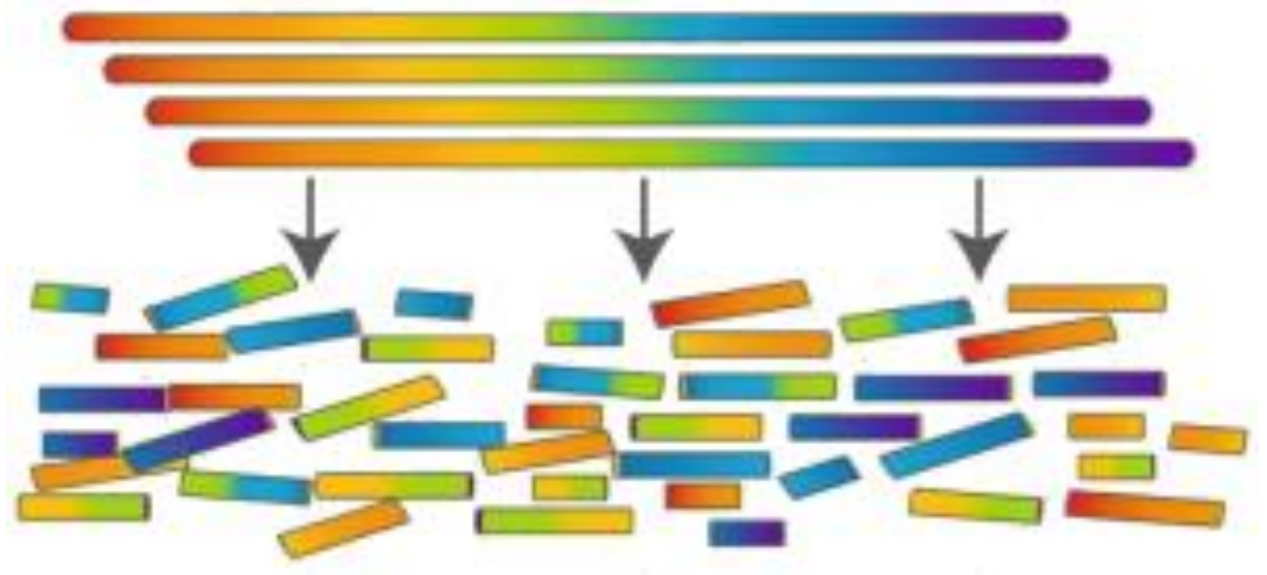


- The newspaper problem is not the same as a jigsaw puzzle:
  - We have multiple copies of the *same* edition of a newspaper.
  - Plus, some pieces of paper got blown to bits in the explosion.
- Instead, we must use *overlapping* shreds of paper to reconstruct what the newspaper said.
- This gives us a giant **overlap puzzle**.

What Makes Genome Sequencing So Difficult?

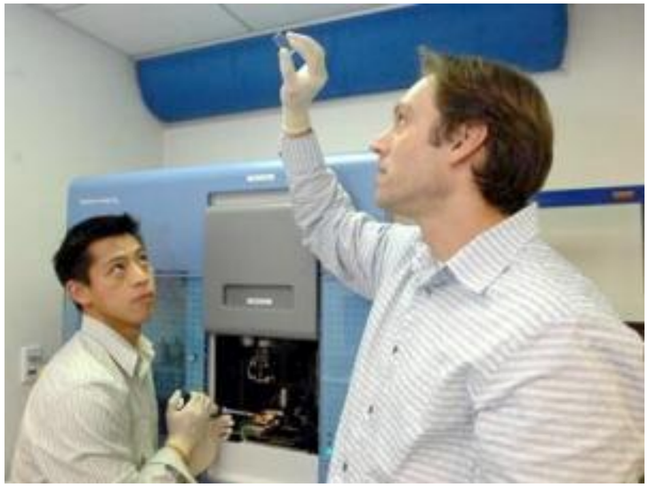
- When we read a book, we can read the entire book one letter at a time from beginning to end.

- However, modern sequencing machines can only read short pieces of DNA (~100 nucleotides long), called **reads**.

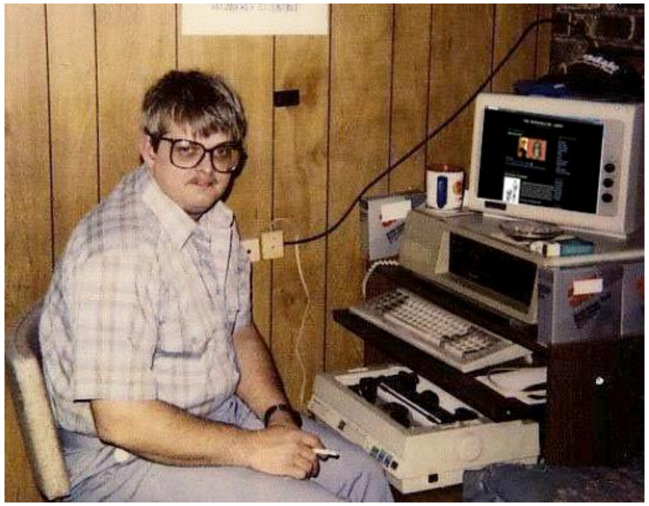


Sequencing a Genome: Lab + Computation

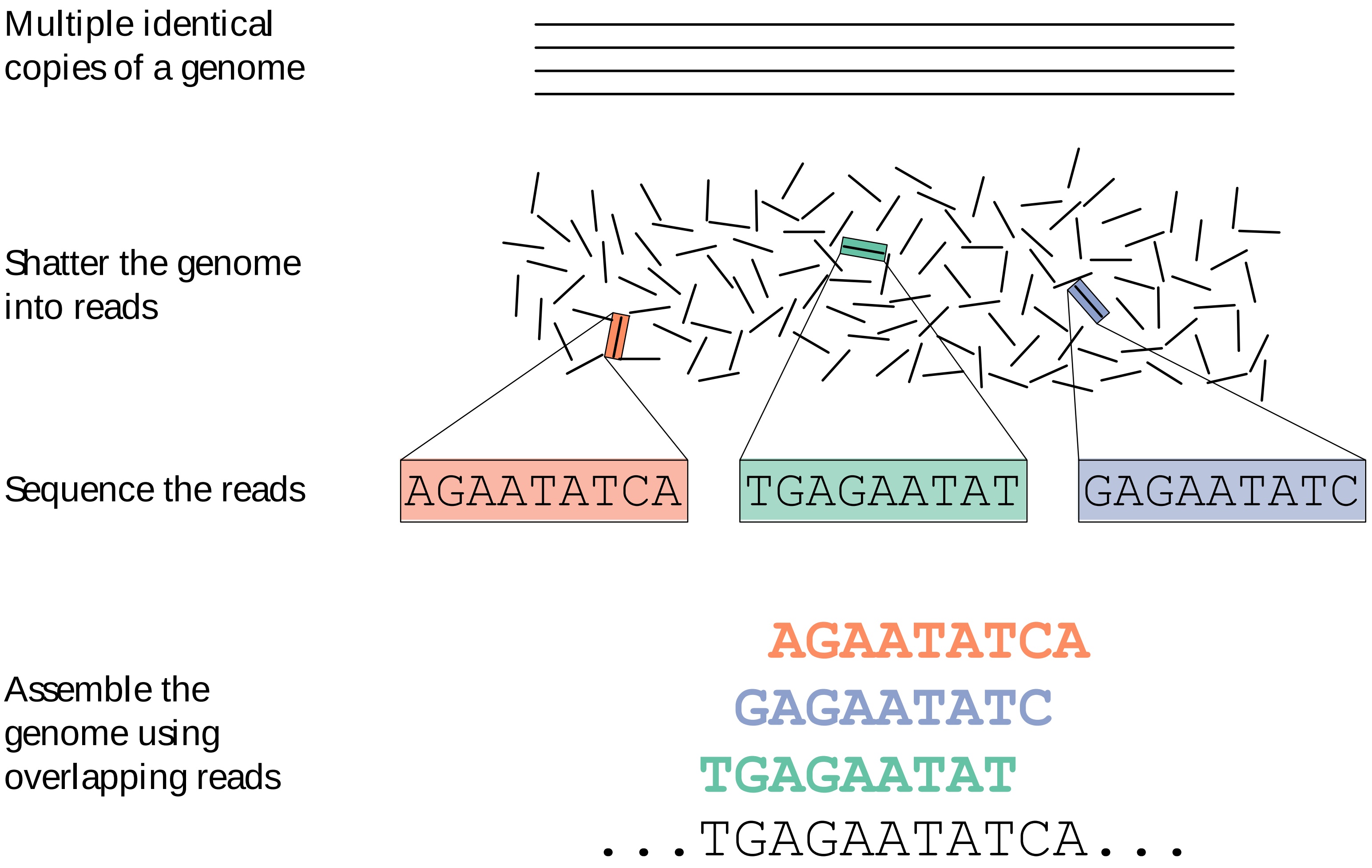
- **Read Generation**:
  Chemically blow multiple copies of a genome to bits to obtain many reads.



- **Fragment Assembly:** Use these reads to algorithmically
  put the genome back together.



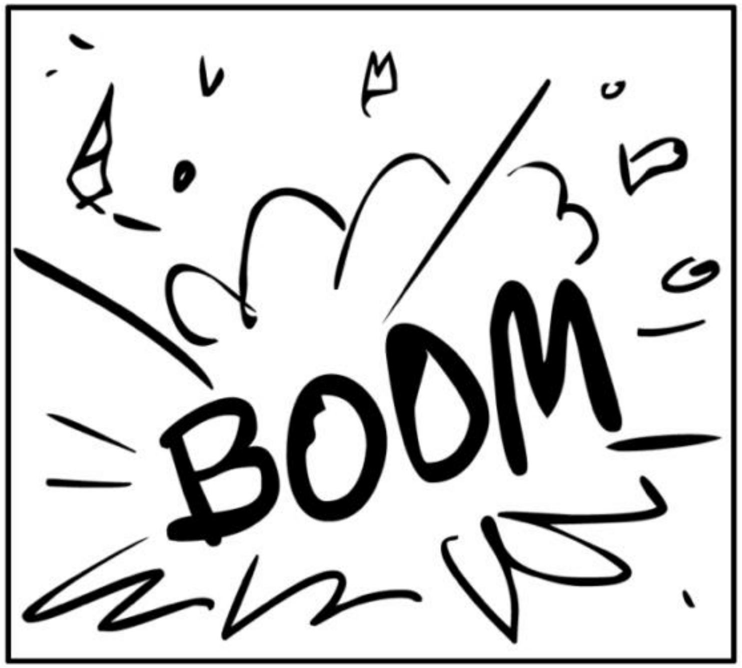
Sequencing a Genome: Illustration


Multiple identical copies of a genome
Shatter the genome into reads
Sequence the reads
AGAATATCA
TGAGAATAT
GAGAATATC
Assemble the genome using overlapping reads
AGAATATCA
GAGAATATC
TGAGAATAT
...TGAGAATATCA...

What does this process remind you of?

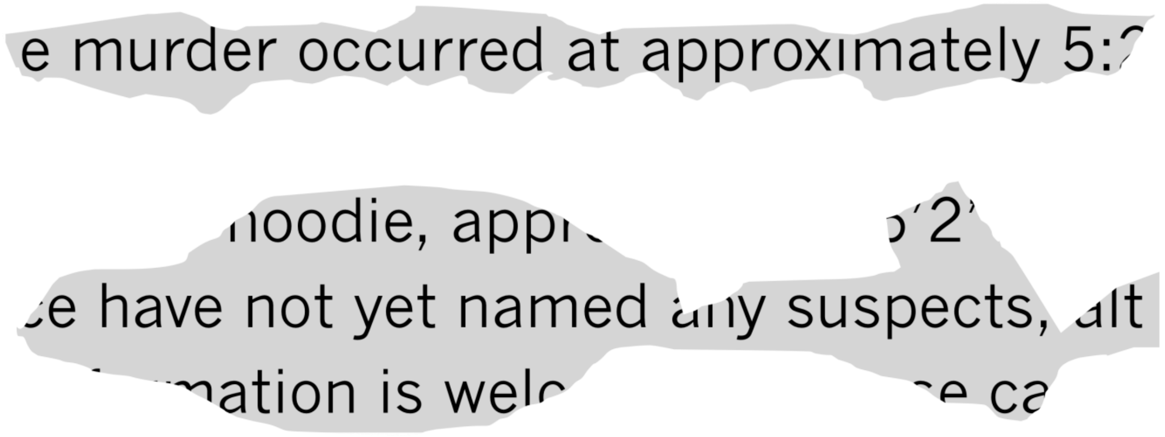
Sound Familiar?

- **Conclusion**: Fragment assembly reduces to an *overlap puzzle*!



Sequencing is Harder than Newspaper Problem

- In the newspaper problem, we have the rules of grammar and common sense (e.g. "**murder**" and "**suspect**" would often appear near each other in a newspaper.)



e murder occurred at approximately 5:

noodie, appr 5´2

e have not yet named any suspects, alt

mation is welc e ca

- However, the "grammar" of DNA remains largely unknown.

Sequencing is Harder than Newspaper Problem

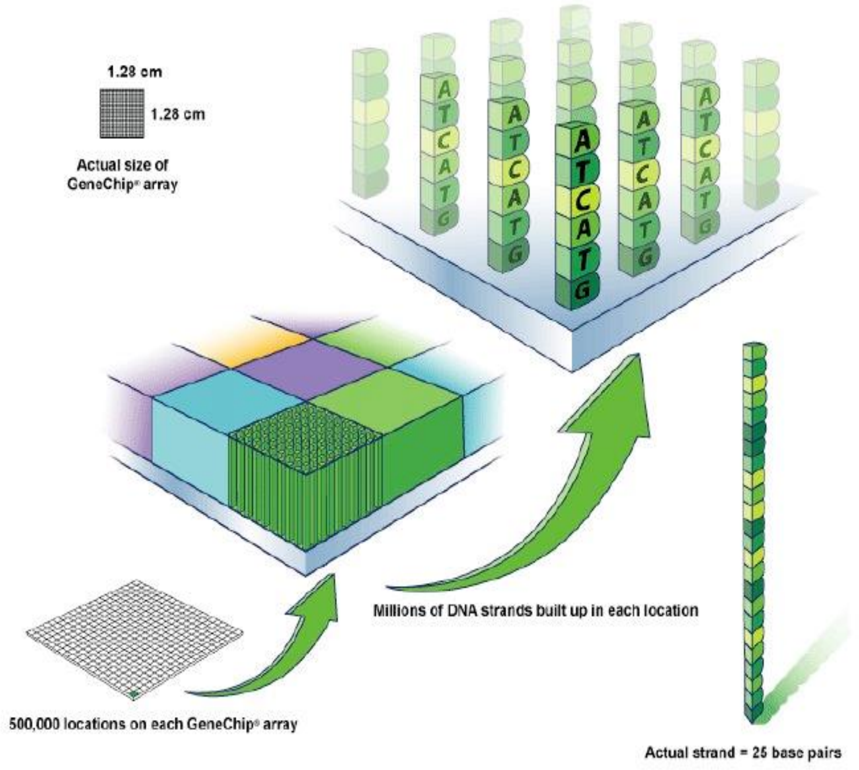
- 50% of the human genome is made up of **repeats**, or strings that appear multiple times with minor variations.

- **Analogy**: The “Triazzle” contains lots of repeated figures, which makes it difficult to solve (even with just 16 pieces).

Section 3: DNA Chips: A First Shot at Sequencing with Short Reads

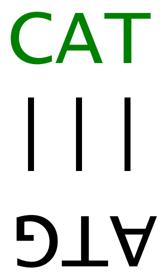
DNA Chips: Implementation

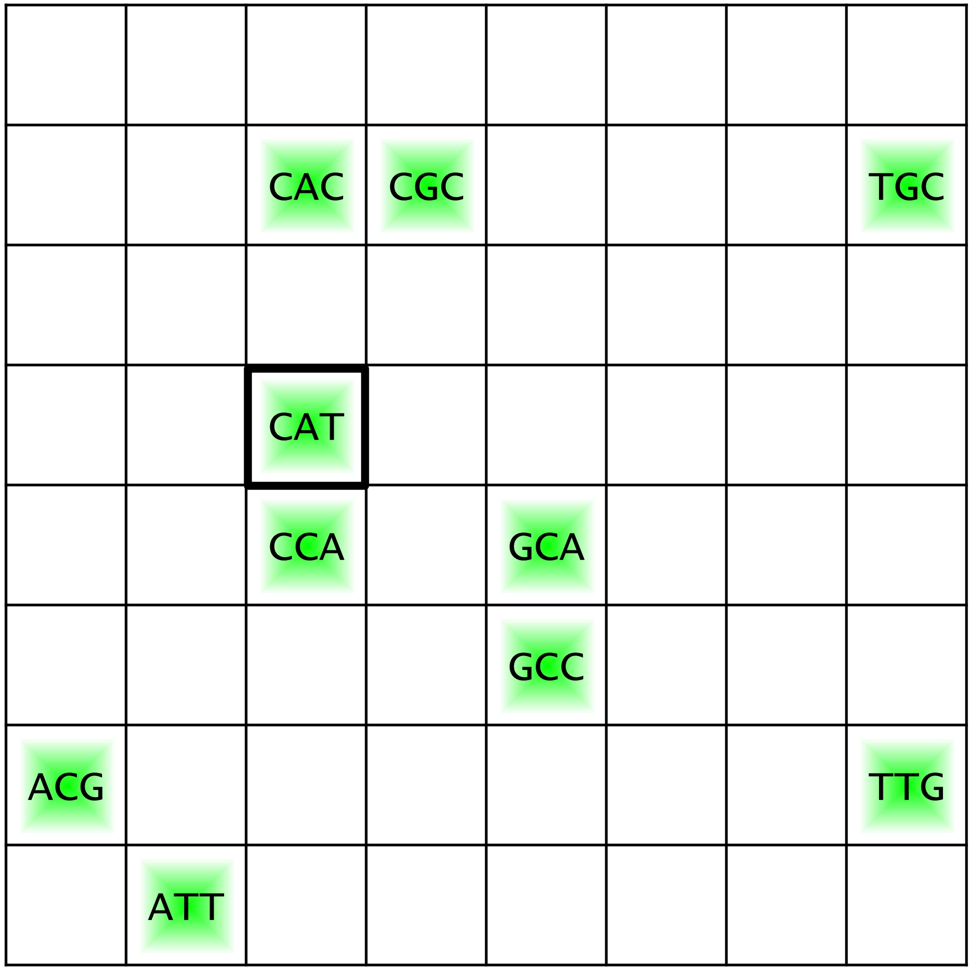
1. Synthesize a *distinct* read of length *k* in each cell of an array.
2. Cover the array with multiple copies of our fluorescently-labeled reads.
3. DNA will hybridize with a string if it contains its *reverse complement*.
4. Use a spectroscope to determine which sites emit light …the *complements* of these sites will reveal the reads within the unknown DNA fragment.


1.28 cm
1.28 cm
Actual size of GeneChip® array
500,000 locations on each GeneChip® array
Millions of DNA strands built up in each location
Actual strand = 25 base pairs
ATCATG

DNA Chips: Example

- What are our reads?



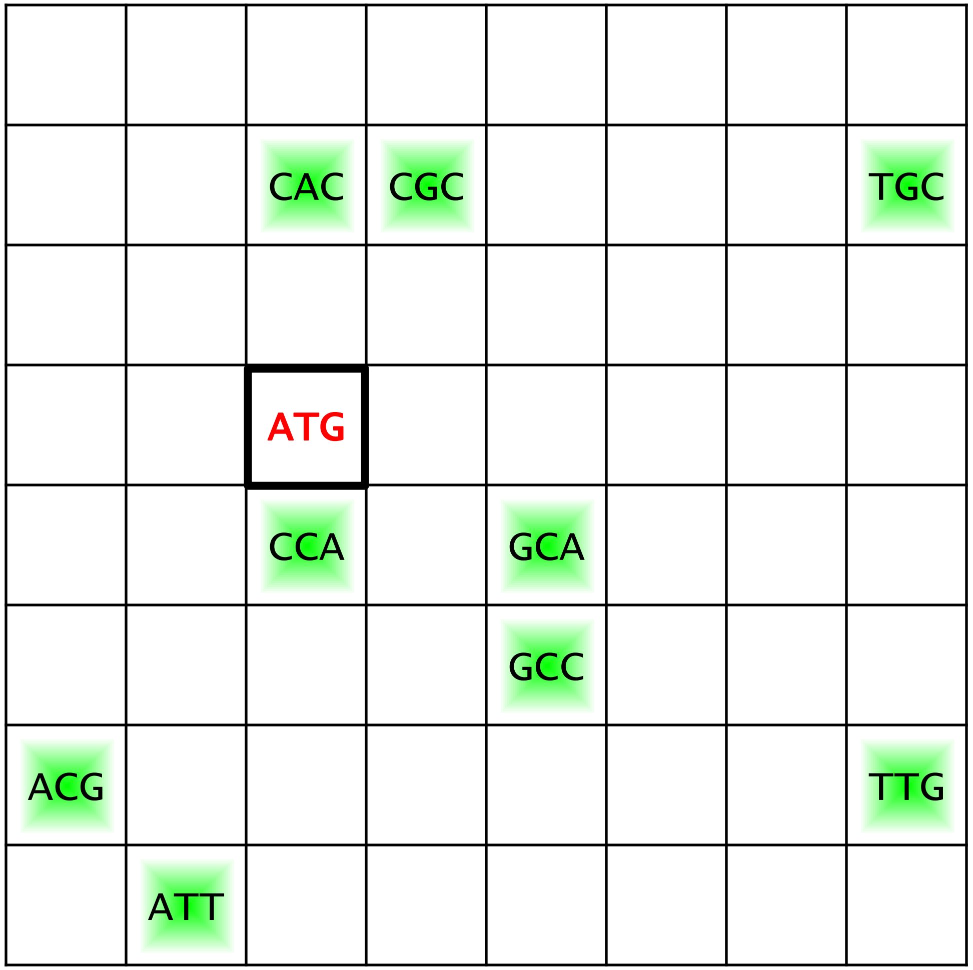


DNA Chips: Example

- What are our reads?

ATG

- So 3-mer **ATG** must occur in the genome!



Red Reads Must Occur in the Genome

- What are our reads?
  - CAC → GTG
  - CGC → GCG
  - CAT → ATG
  - TGC → GCA
  - ACG → CGT
  - ATT → AAT
  - CCA → TGG
  - GCA → TGC
  - GCC → GGC
  - TTG → CAA

		GTG	GCG				GCA
		ATG					
		TGG		TGC			
				GGC			
CGT							CAA
	AAT						

From Biological Data to Computational Problem

- **Aim**: Construct a shortest possible genome containing all our reads.

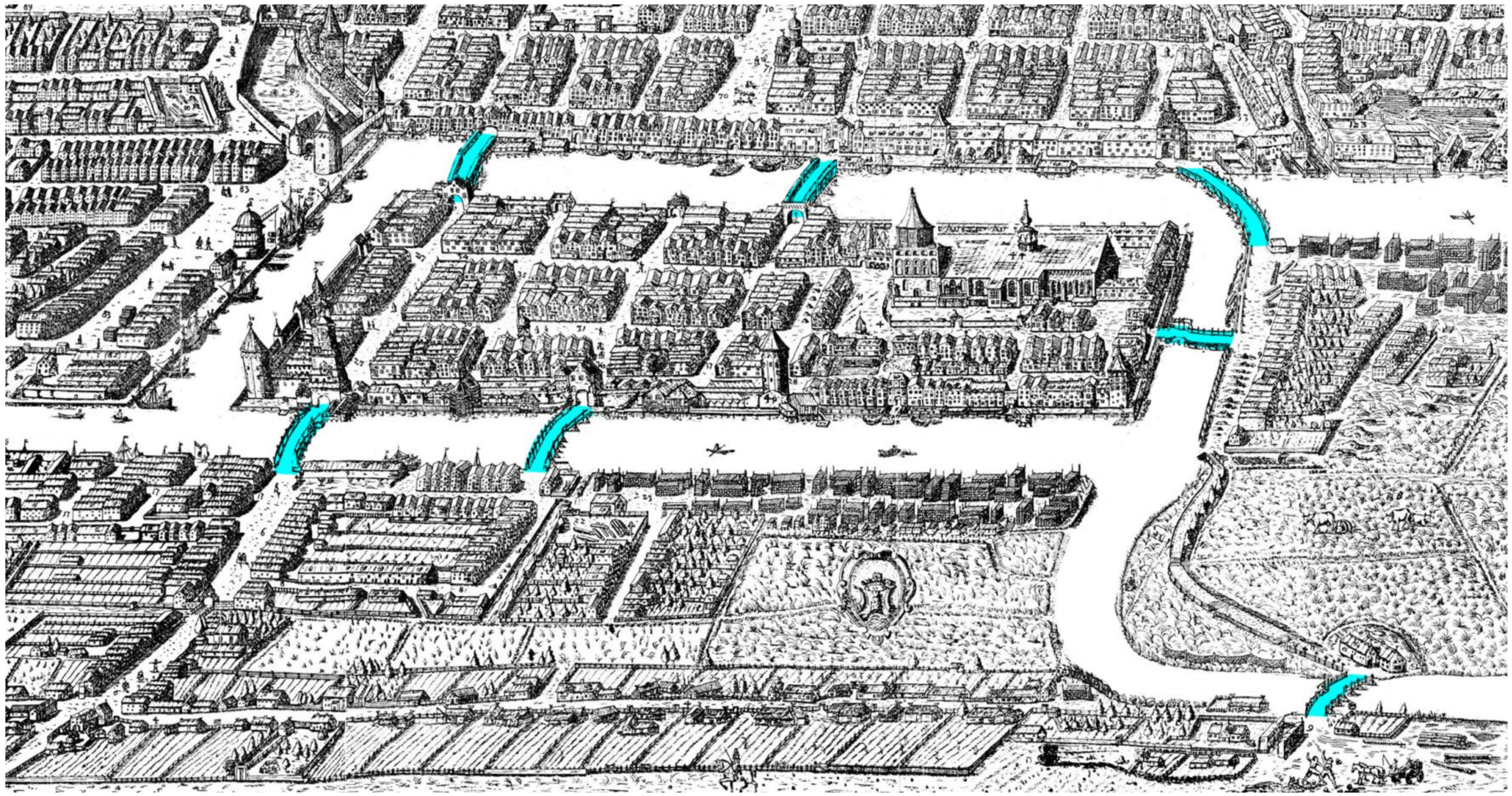
- How in the world would we solve this problem if we had *a billion* reads?

		GTG	GCG				GCA
		ATG					
		TGG		TGC			
				GGC			
CGT							CAA
	AAT						

Section 4: Two Mathematical Detours

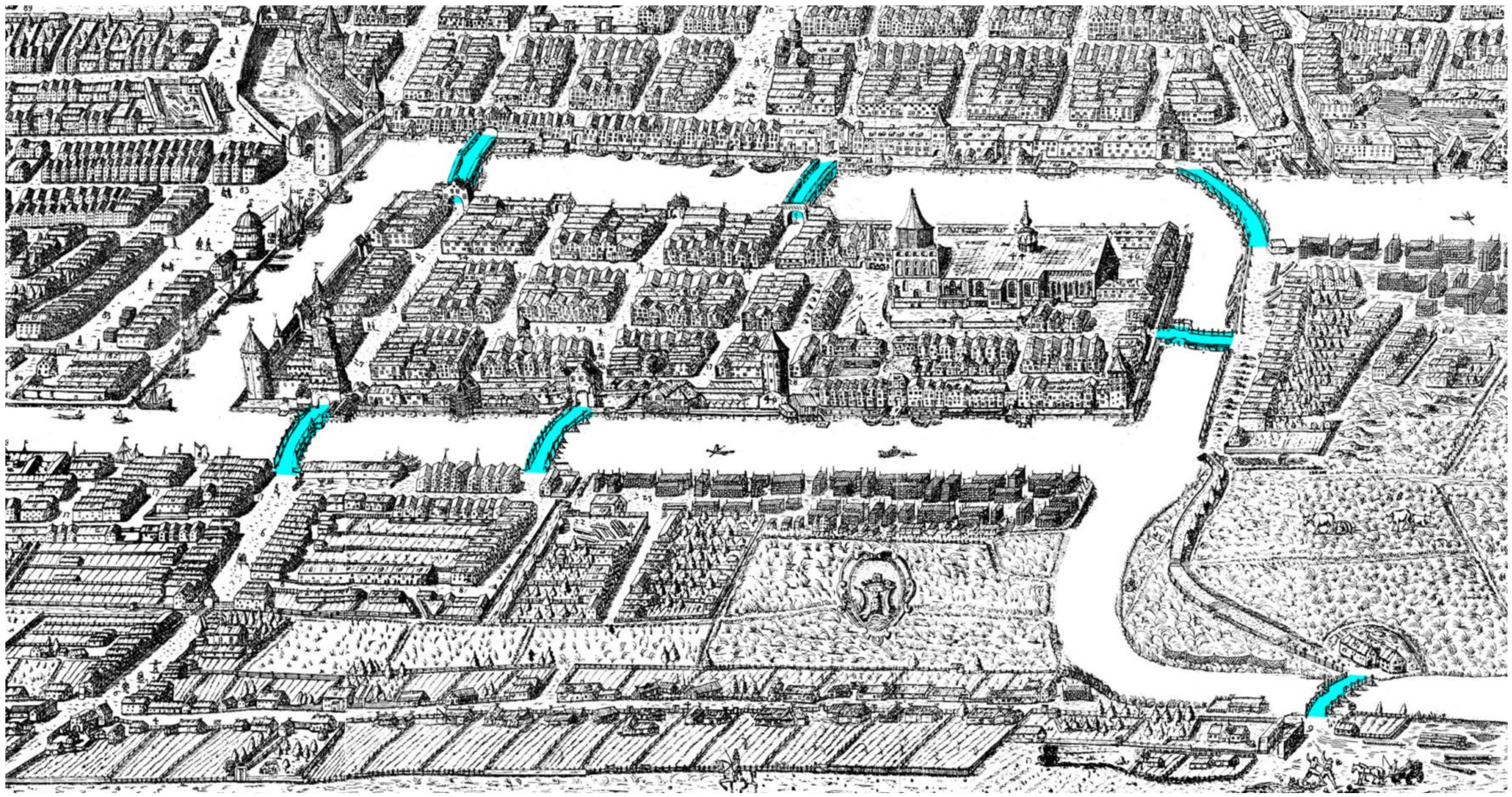
The Bridges of Königsberg

- The people of Königsberg, Prussia (present-day Kaliningrad, Russia) enjoyed taking walks.



The Bridges of Königsberg

- They wondered if they could walk through the city, cross each bridge (blue) **exactly once**, and return where they started.



The Bridges of Königsberg

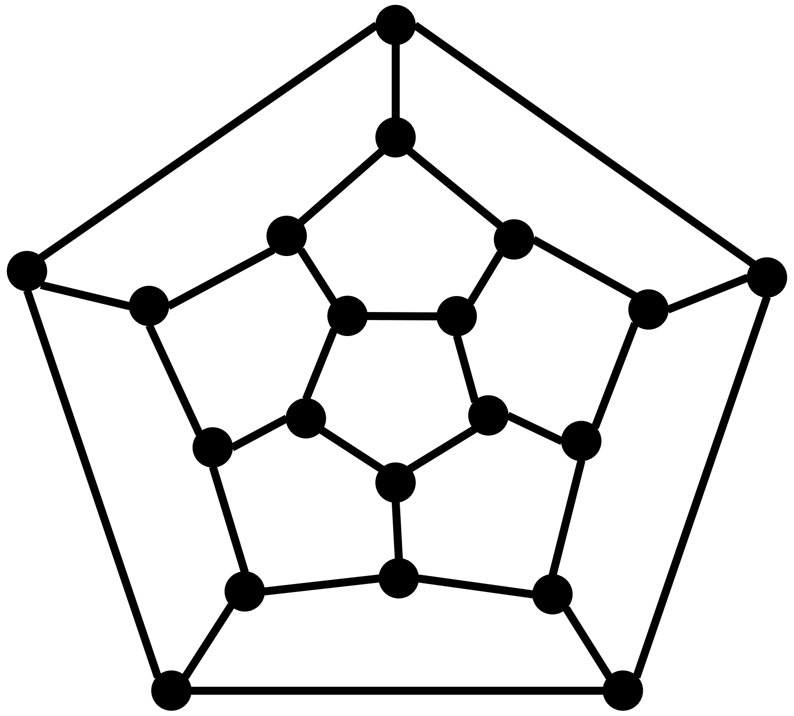
- **1735**: Leonhard Euler develops an approach to answer this question for *any* city, even for a "city" with a million islands.

- We will soon discuss Euler's approach.

Leonhard Euler

The Icosian Game

- Over a century passes…
- **1857**: Irish mathematician William Hamilton designs a game consisting of a board representing 20 "islands" connected by "bridges."
- **Goal**: find a walk that visits every island **exactly once** and returns back where it started.


Icosian Game

Similar Problems with Very Different Fates

- These two stories have something in common:
  - Find a walk that uses every ***bridge*** once and returns home (Konigsberg Bridge Problem)
  - Find a walk that visits every ***island*** once and returns home (Icosian game)

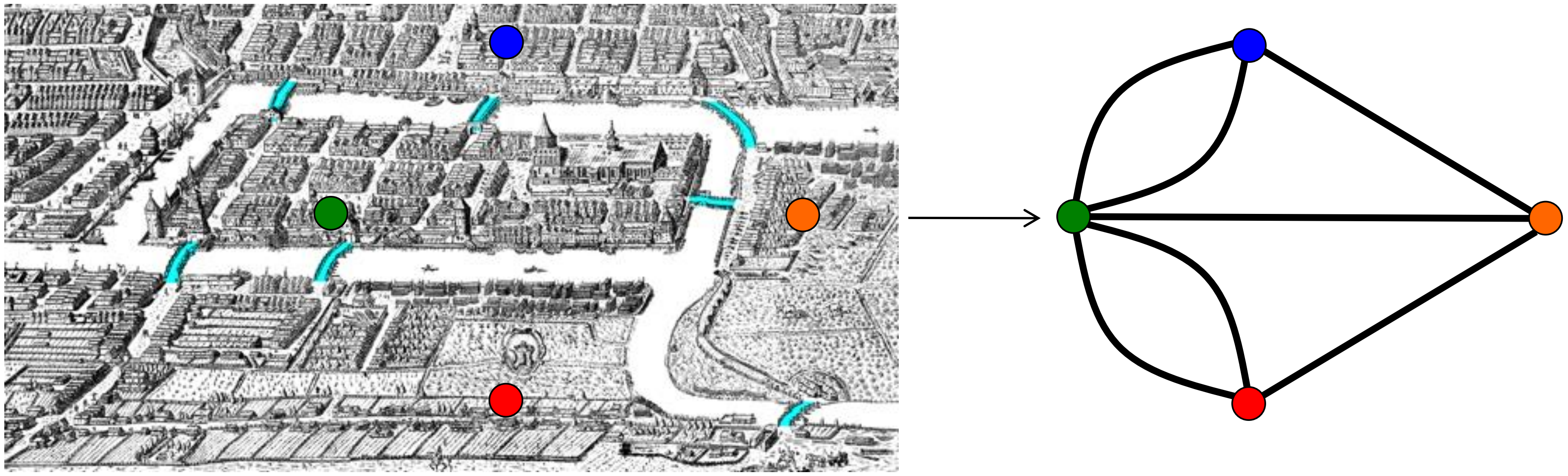
- However, while Euler solved the first problem (even for a city with a million ***bridges***), mathematicians still do not know how to solve the second problem, even for a city with just a thousand ***islands.***

- But where are the genomes???

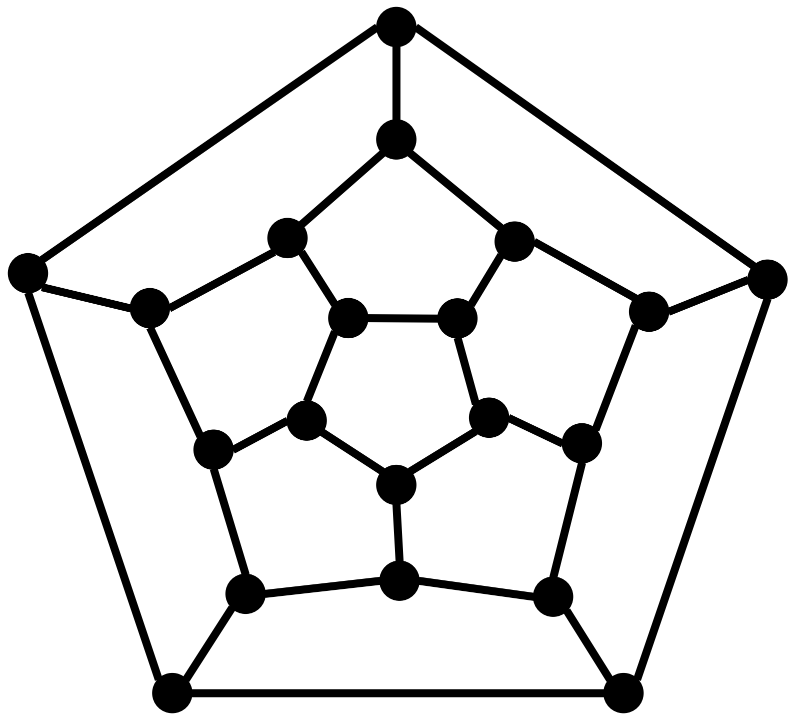
Section 5: Hamiltonian and Eulerian Cycles

Königsberg Bridges Network

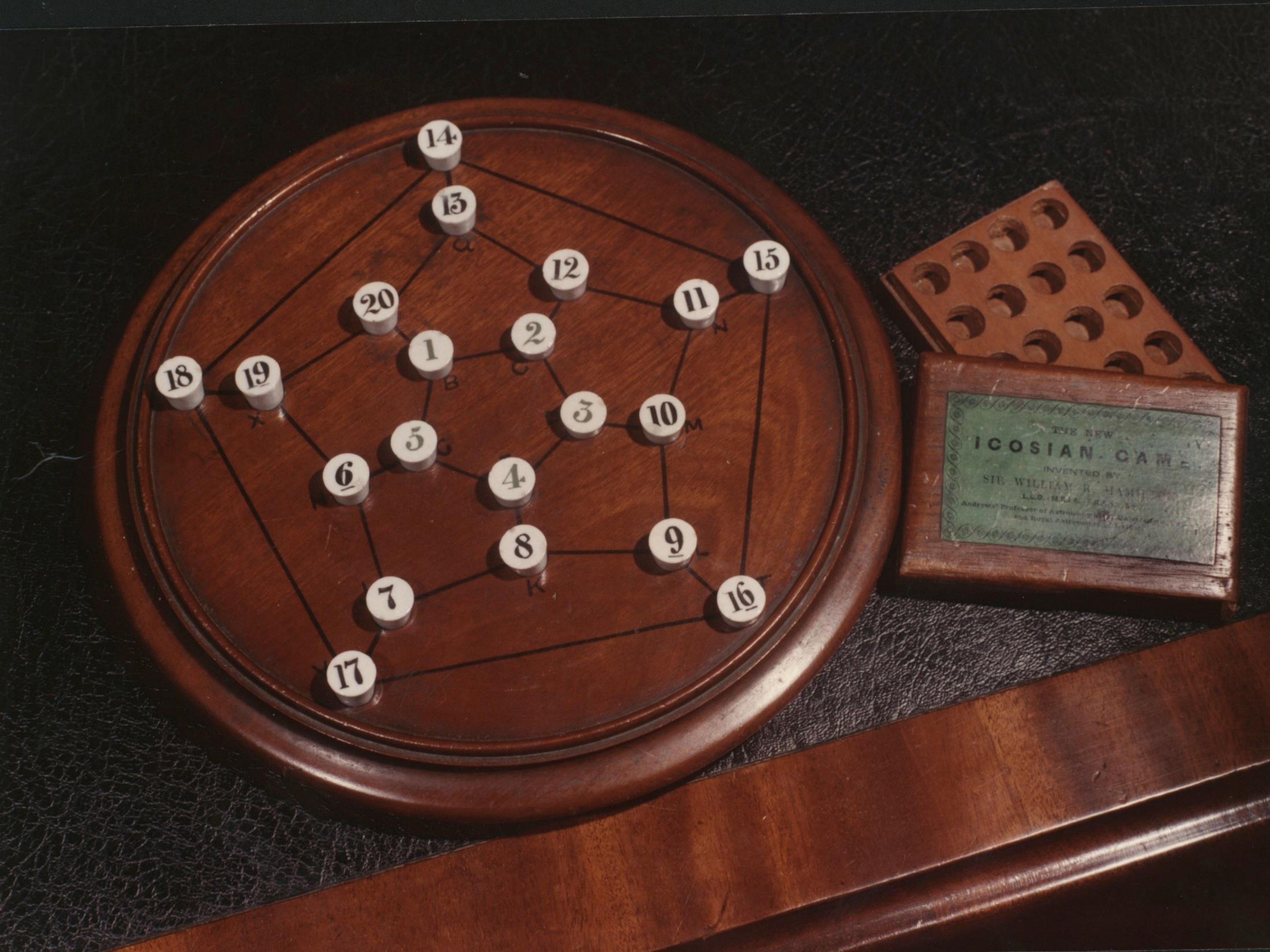
- For the Königsberg Bridge Problem, we create a **network**:
  - Nodes = 4 land masses of the city
  - Edges = 7 bridges connecting land masses



Icosian Game Network



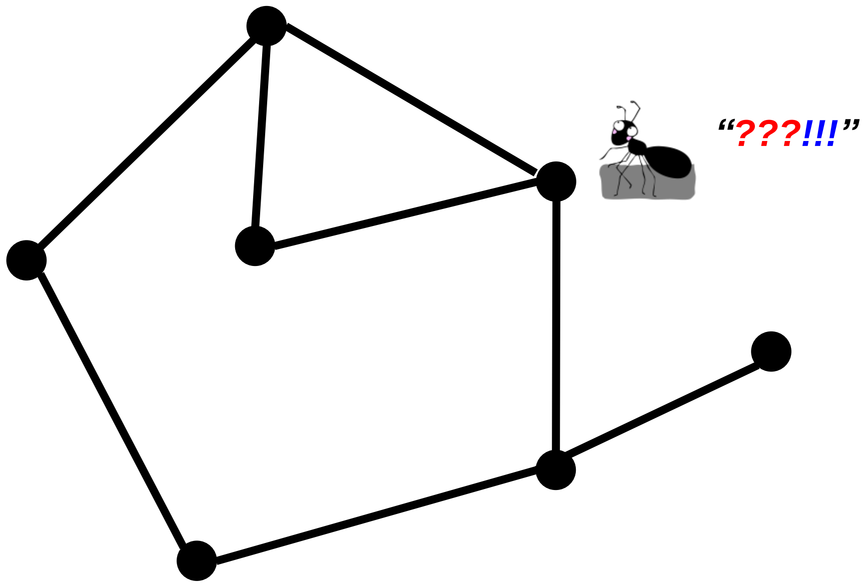
Can you see a solution?


14
13
12
15
20
11
2
1
18
19
3
10
5
6
4
8
9
7
16
17
THE NEW
ICOSIAN GAME
INVENTED BY
SIR WILLIAM R. HAMILTON
LL.D. M.R.I.A. F.R.A.S.
Andrews' Professor of Astronomy
and Royal Astronomer

Eulerian and Hamiltonian Cycles

- Two questions:

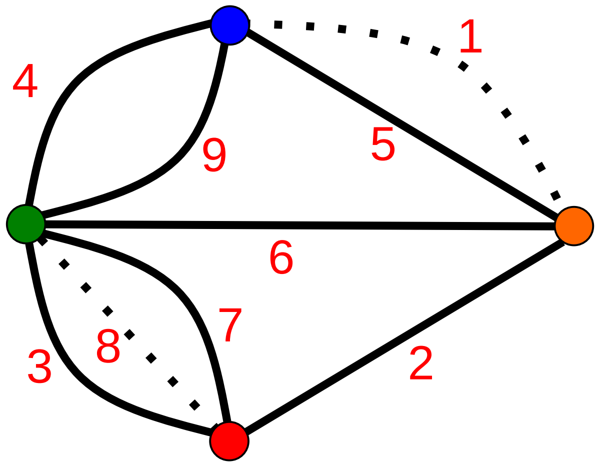
1. Can the ant walks through each ***edge*** exactly once and return to where it started? **Eulerian cycle**
2. Can the ant walk through each ***node*** exactly once and return to where it started? **Hamiltonian cycle**



Eulerian Cycles

- If there were a solution to the Königsberg Bridge Problem, then we could find an Eulerian cycle in this network.

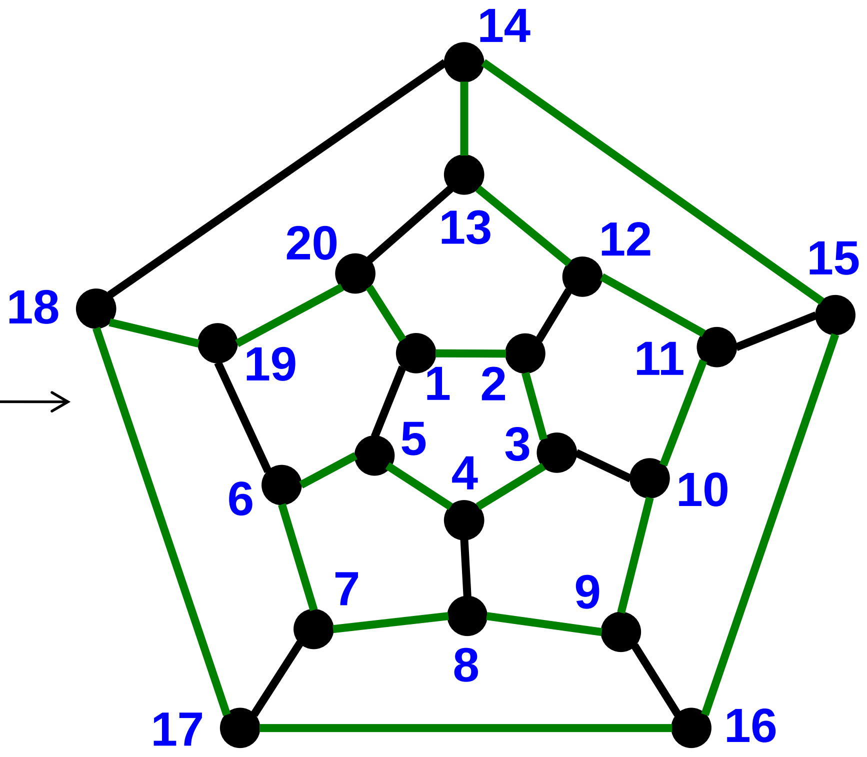
- However, no such cycle exists. Why?



- **If we add two more edges, there will be such a cycle.**

Hamiltonian Cycles

- A **Hamiltonian cycle** in a network uses each node exactly once and returns to its starting node.



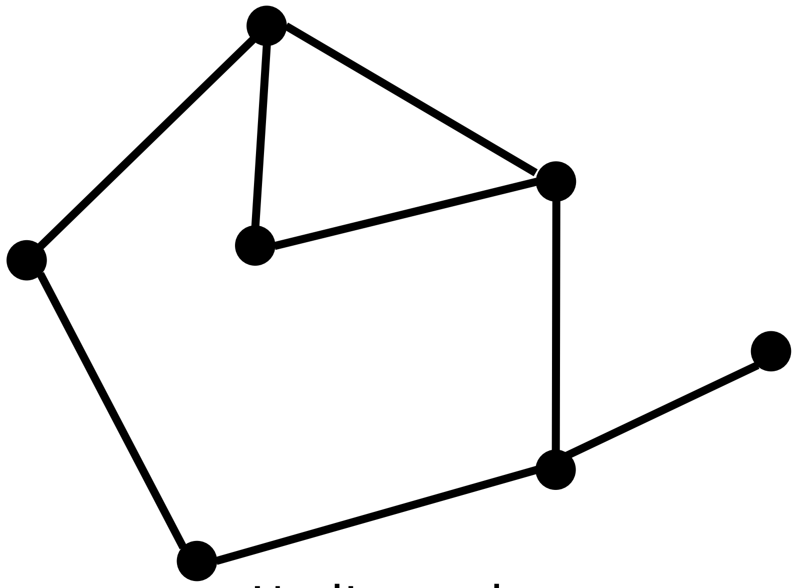
Finding Eulerian Cycles vs Hamiltonian Cycles

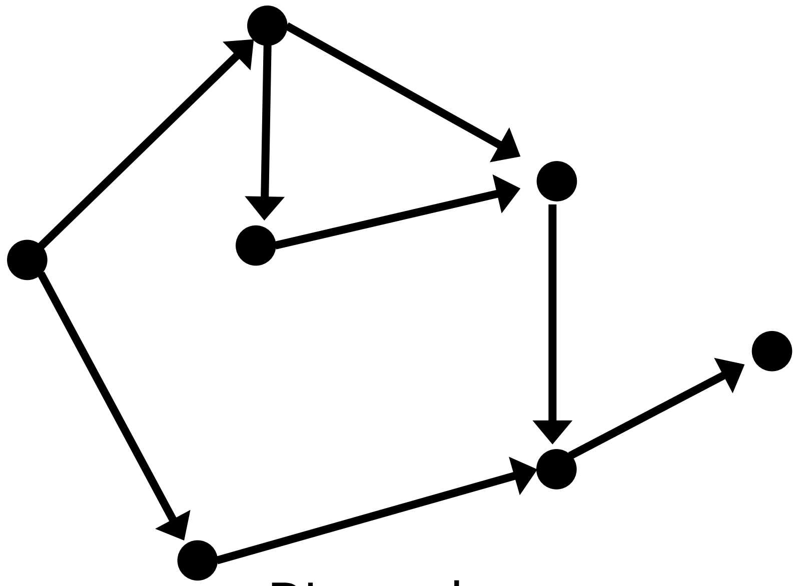
- Given a network *G*, we now have two questions that we can program a computer to answer about *G*.
- **Eulerian Cycle Problem (ECP)**: Find an Eulerian cycle in *G* or prove that *G* does not have an Eulerian cycle.
- **Hamiltonian Cycle Problem (HCP)**: Find a Hamiltonian cycle in *G* or prove that *G* does not have a Hamiltonian cycle.

Section 6: Euler’s Theorem

Directed Networks

- **Directed Network**: A network in which each edge has a *direction* (represented by an arrow).
  - You might like to think of directed edges as "one-way bridges."

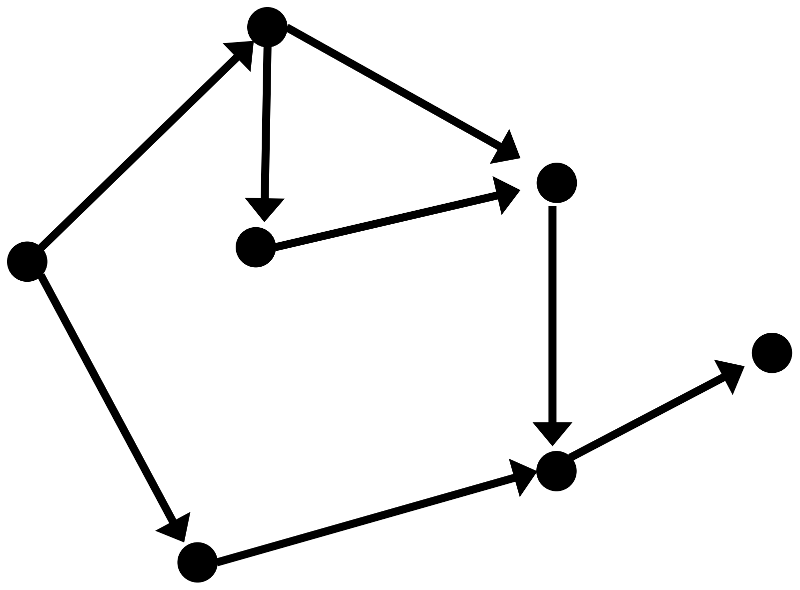

Undirected


Directed

Eulerian Cycles in Directed Networks

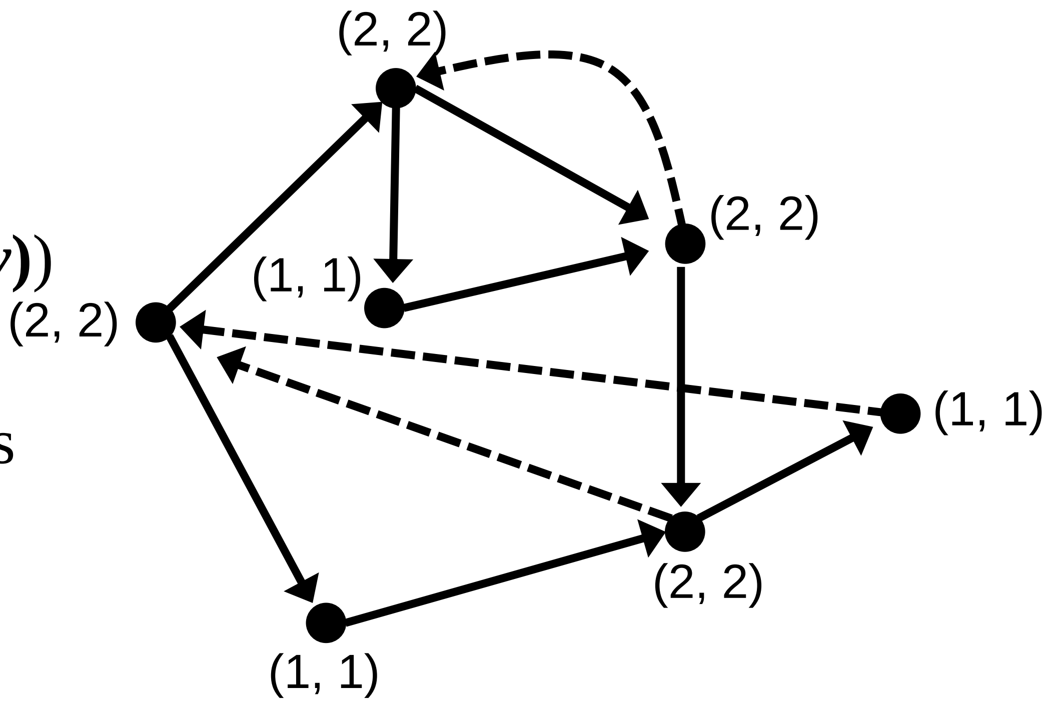
- An **Eulerian cycle** in a directed network must travel down all the edges in the correct direction.

- Does this graph have an Eulerian cycle?



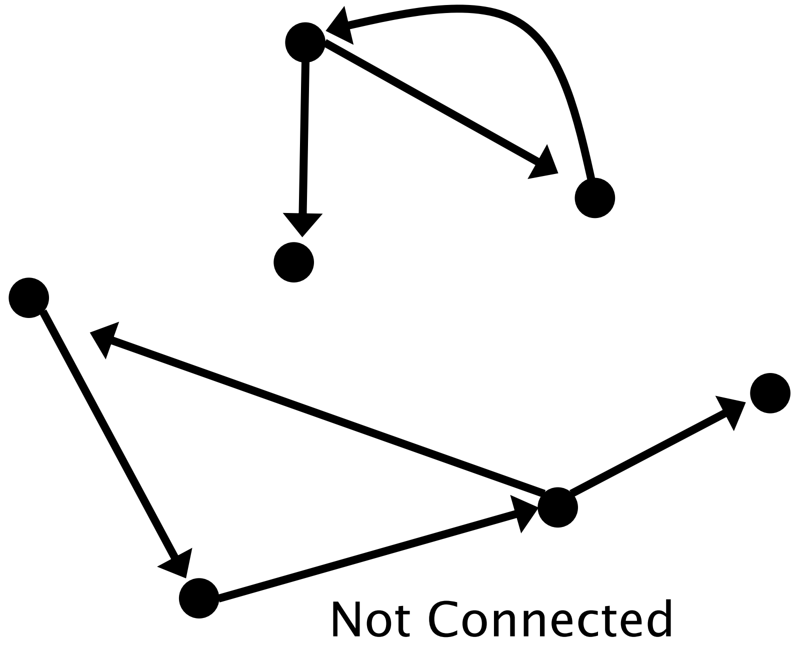
Balanced Graphs

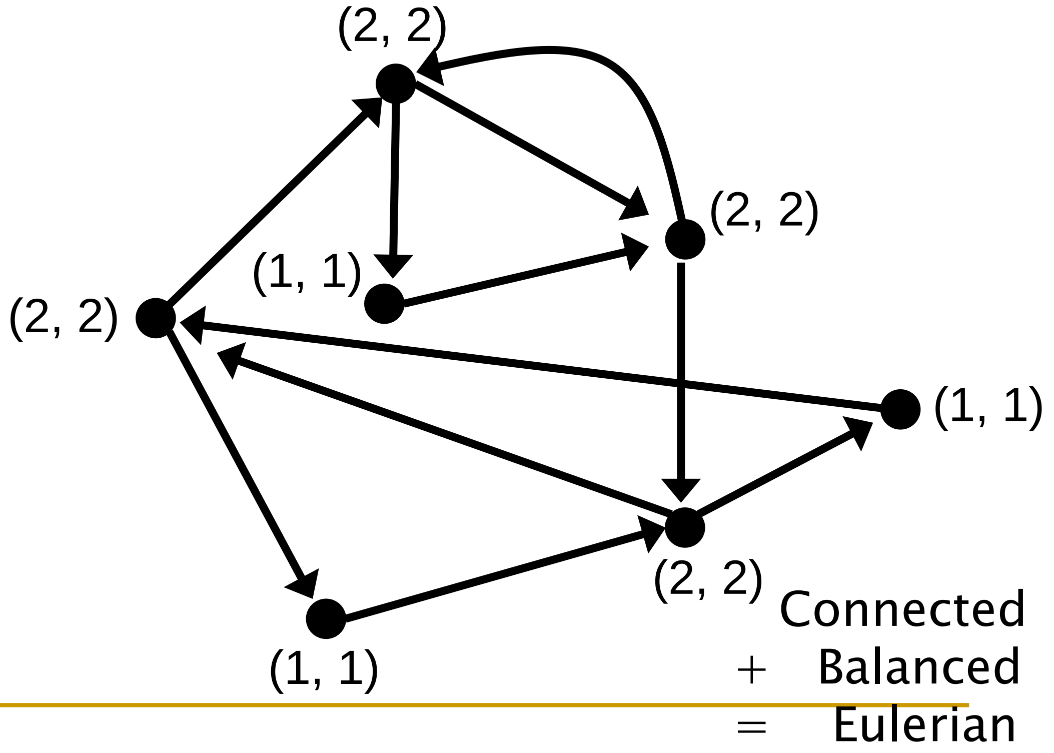
- **indegree(*v*)** = the number of edges leading into node *v*.
- **outdegree(*v*) =** the number of edges leading out of node *v*.
  - A graph is **balanced** if **indegree(*v*) = outdegree(*v*)** for every node *v*.
- Label each node *v* with (**indegree(*v*)**, **outdegree(*v*)**)
- Adding some edges makes the graph balanced.



Euler’s Theorem

- **Euler’s Theorem**: A directed network contains an Eulerian cycle when the network is connected and balanced.
    - A graph is **connected** if for every pair of vertices {*u*, *v*}, an ant can legally travel either from *u* to *v* or from *v* to *u*.





Section 7: ECP vs. HCP and Algorithmic Complexity

What’s the Big Deal?

- *“I thought computers were supermachines!”*
- *“Computers don’t need 300-year old mathematics to help them solve problems.”*
- *“Aren’t computers going to take over the world anyway?”*
- Let’s examine the case of finding a *Hamiltonian* cycle…

Searching for an Efficient Algorithm for HCP

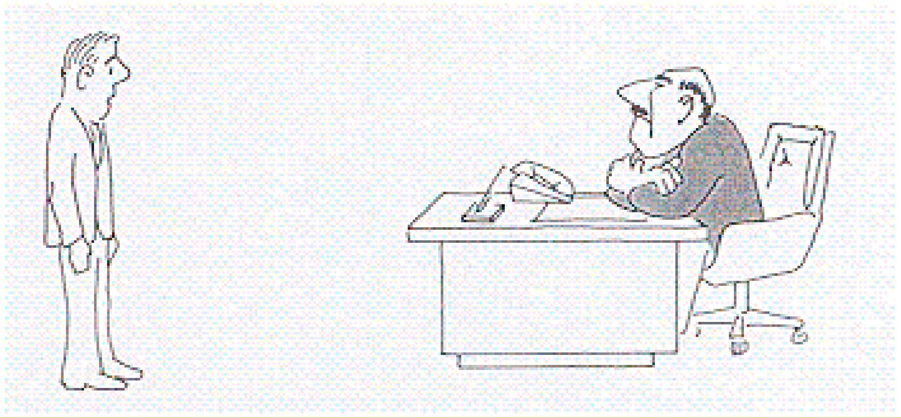
- **Key Point**: No one has ever found a similar efficient test if a network has a Hamiltonian cycle.
- Of course, we could examine every possible ant walk through the graph to solve the HCP.
- However, this **brute force** approach is just not *efficient*: there are more walks through the average network with just 1,000 nodes than there are atoms in the universe!

NP-Complete Problems

- In fact, the HCP has been classified as ***NP*-Complete**.
- This means that the HCP belongs to a collection containing thousands of computational problems that cannot be solved quickly for large data sets.
- *NP*-Complete problems are all **equivalent** to each other: find an efficient solution to one, and you have an efficient solution to them all.

NP-Complete Problems

- Attempting to solve any *NP*-Complete problem is difficult.



"I can't find an efficient algorithm, I guess I'm just too dumb."

From Garey and Johnson. *Computers and Intractability*. 1979

NP-Complete Problems

- Attempting to solve any *NP*-Complete problem is difficult.

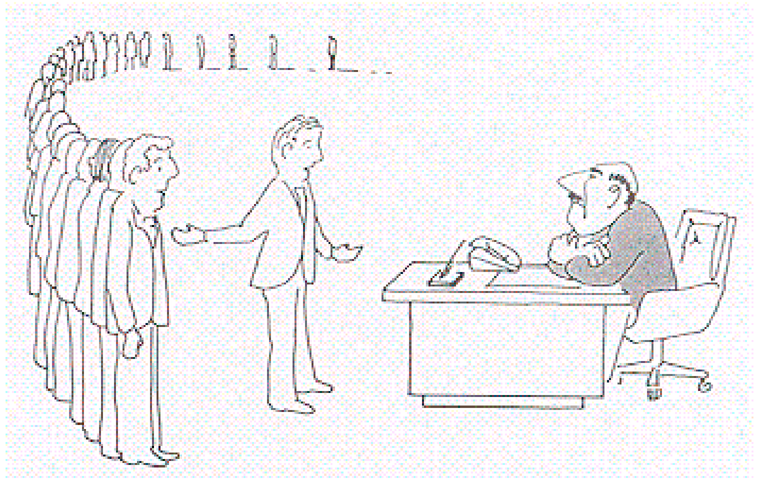
- The hope is that you could verify that you failed because an efficient algorithm to an *NP*-Complete problem doesn't exist.

"I can't find an efficient algorithm, because no such algorithm is possible."

From Garey and Johnson. *Computers and Intractability*. 1979

NP-Complete Problems

- Attempting to solve any *NP*-Complete problem is difficult.
- The hope is that you could verify that you failed because an efficient algorithm to an *NP*-Complete problem doesn't exist.
- The present state of affairs is somewhere in between.

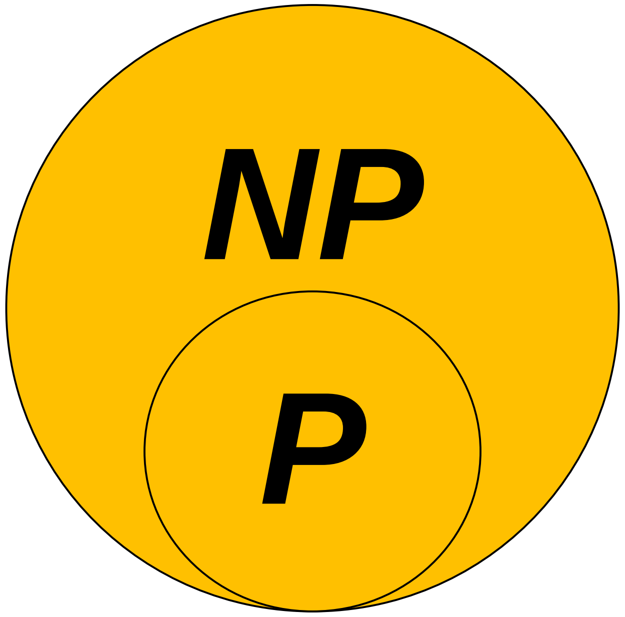


"I can't find an efficient algorithm, but neither can all these smart people."

From Garey and Johnson. *Computers and Intractability*. 1979

P vs. *NP*, *NP*-Complete vs. *NP*-Hard

- ***NP***: The set of problems that can be verified efficiently
- ***P***: The set of problems that can be solved efficiently
- As can be seen, *P* is a subset of *NP*



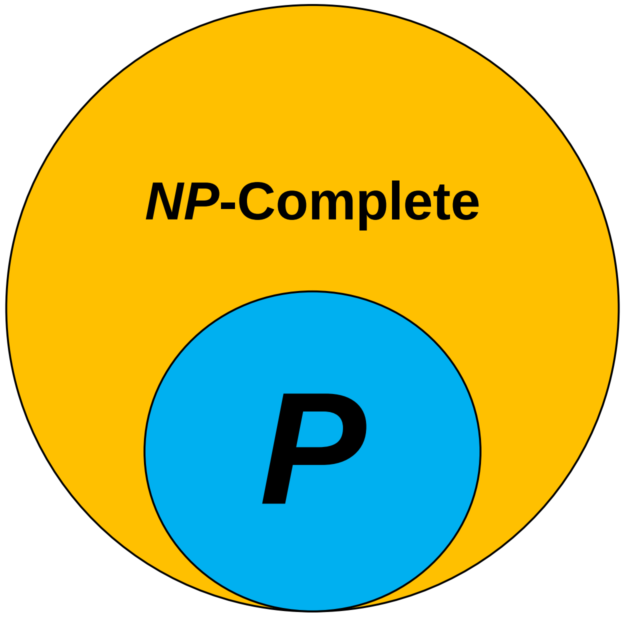
P vs. *NP*, *NP*-Complete vs. *NP*-Hard

- *NP*: The set of problems that can be verified efficiently

- *P*: The set of problems that can be solved efficiently

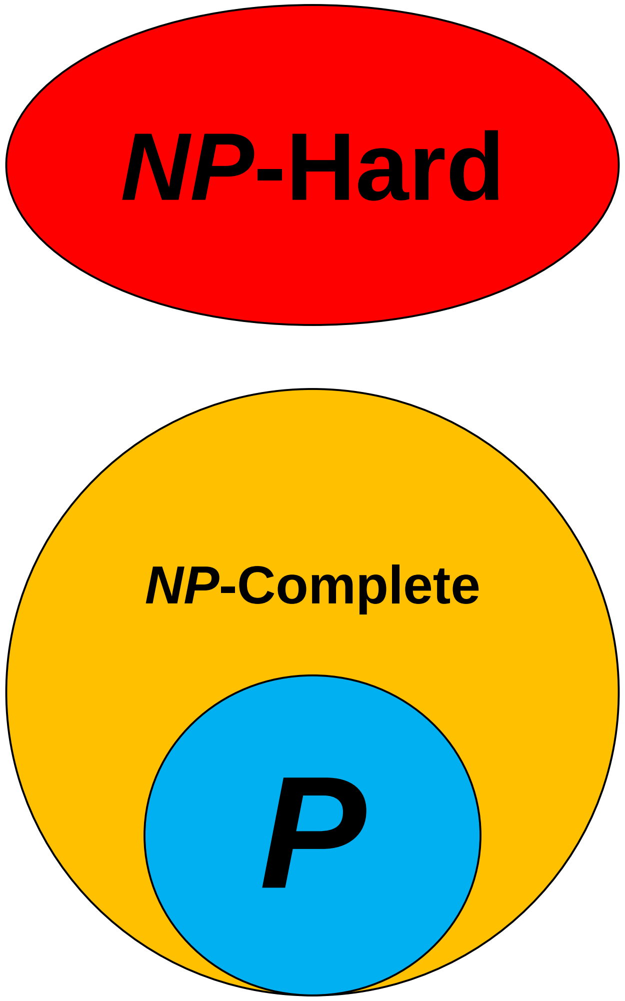
- As can be seen, *P* is a subset of *NP*

- Problems in *NP* that are not in *P* are called ***NP*-Complete**



P vs. *NP*, *NP*-Complete vs. *NP*-Hard

- ***NP*-Hard**: Problems that cannot be solved nor verified efficiently
- ***P* vs. *NP* Problem**: Can we prove that $P = NP$, or that $P \neq NP$?
- If $P = NP$, then ALL *NP* problems can be solved efficiently
- If $P \neq NP$, then *NP*-Complete problems can't be solved efficiently

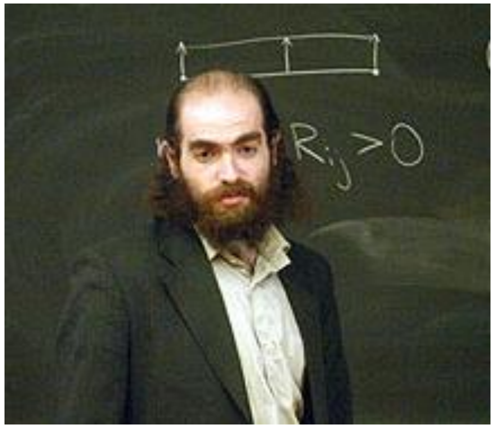


The *NP*-Completeness of the HCP

- The question of whether or not *NP*-Complete problems (including the HCP) can be solved efficiently is one of seven **Millennium Problems** in mathematics.

- Find an efficient algorithm for the HCP, or demonstrate that no such algorithm exists, and you will get $1 million.

- However, if you become a mathematician, odds are that you are not in it for the $$$...recently, Grigory Perelman solved one of these problems but turned down the prize.



Grigory Perelman

Section 8: From Euler and Hamilton to Fragment Assembly

First Try: The Network *H*

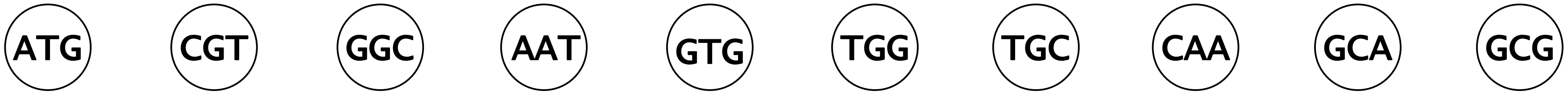
- Create a node for every read detected by our array.

		GTG	GCG				GCA
		ATG					
		TGG		TGC			
				GGC			
CGT							CAA
	AAT						

First Try: The Network *H*

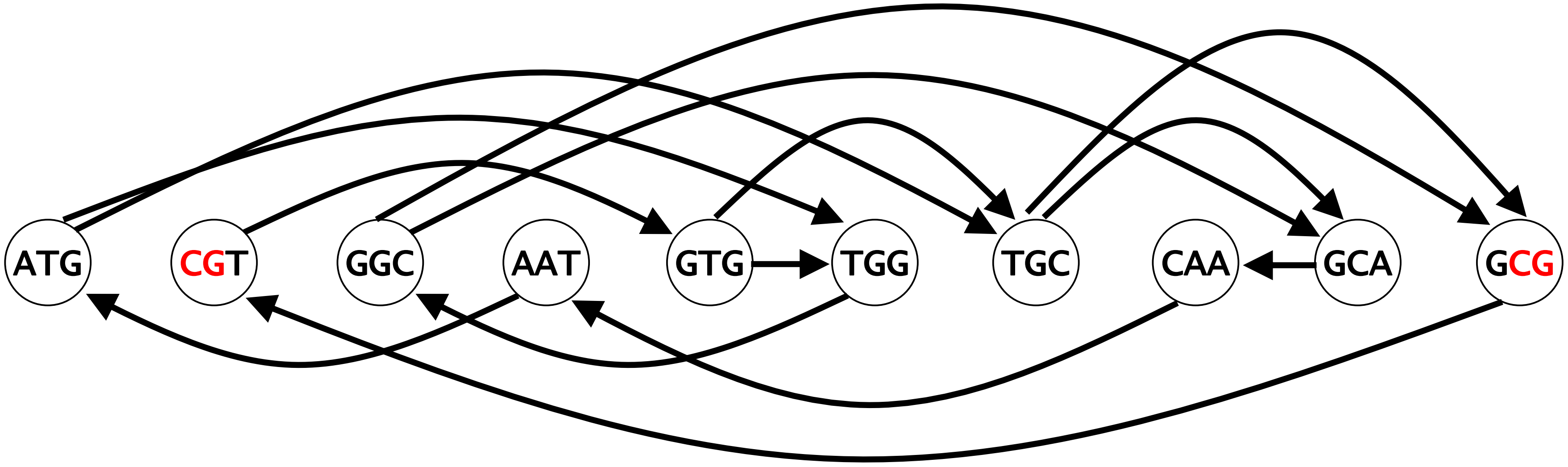
- Create a node for every read detected by our array.
  - **Prefix**: First 2 nucleotides of a read (CAA)
  - **Suffix**: Last 2 nucleotides of a read (CAA)

- Different 3-mers may share a prefix/suffix: ATG, TGA, CTG



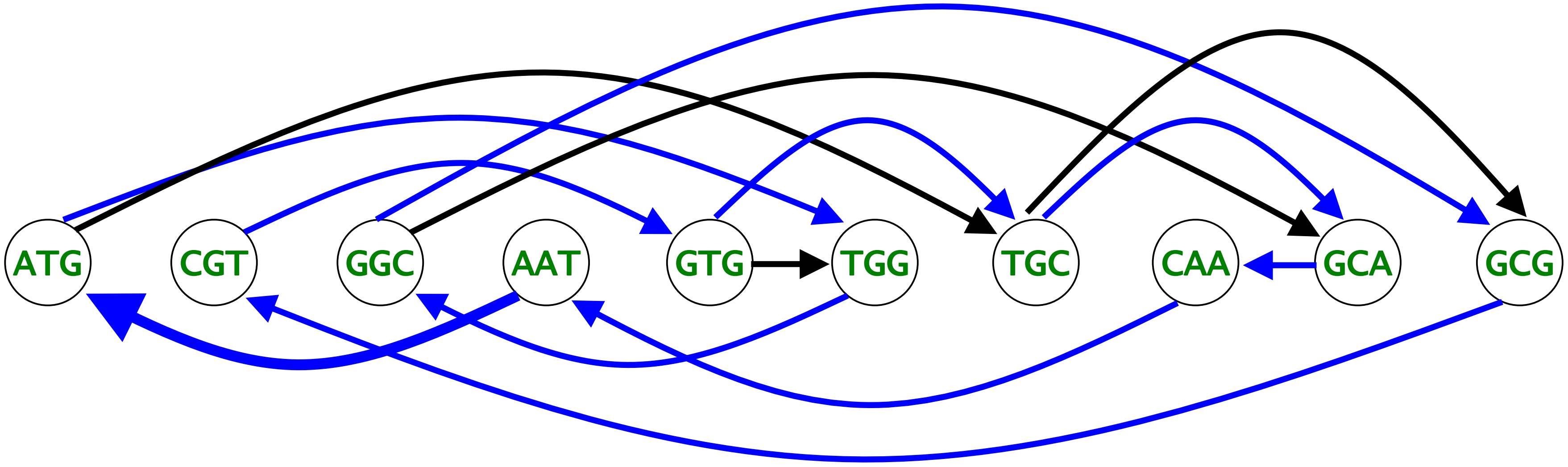
First Try: The Network *H*

- As for the edges of *H*, connect node *v* to node *w* with a *directed edge* if the suffix of *v* matches the prefix of *w*.



Hamiltonian Cycles in *H*

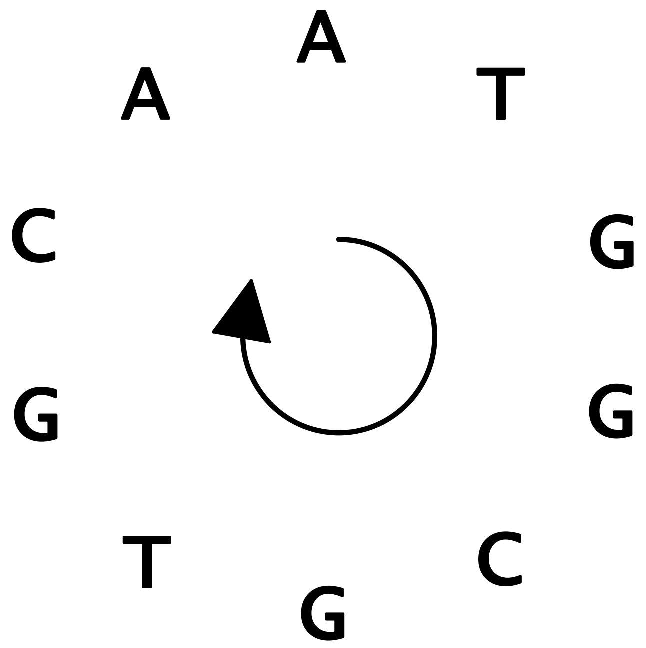
- Here we have a Hamiltonian cycle in *H*:
  - ATG → TGG → GGC → GCG → CGT → GTG → TGC → GCA → CAA → AAT → ATG



Hamiltonian Cycles in *H*

- Here we have a Hamiltonian cycle in *H*:
  - ATG → TGG → GGC → GCG → CGT → GTG → TGC → GCA → CAA → AAT → ATG

```
ATG
 TGG
  GGC
   GCG
    CGT
     GTG
      TGC
       GCA
        CAA
         AAT
          ATG
```



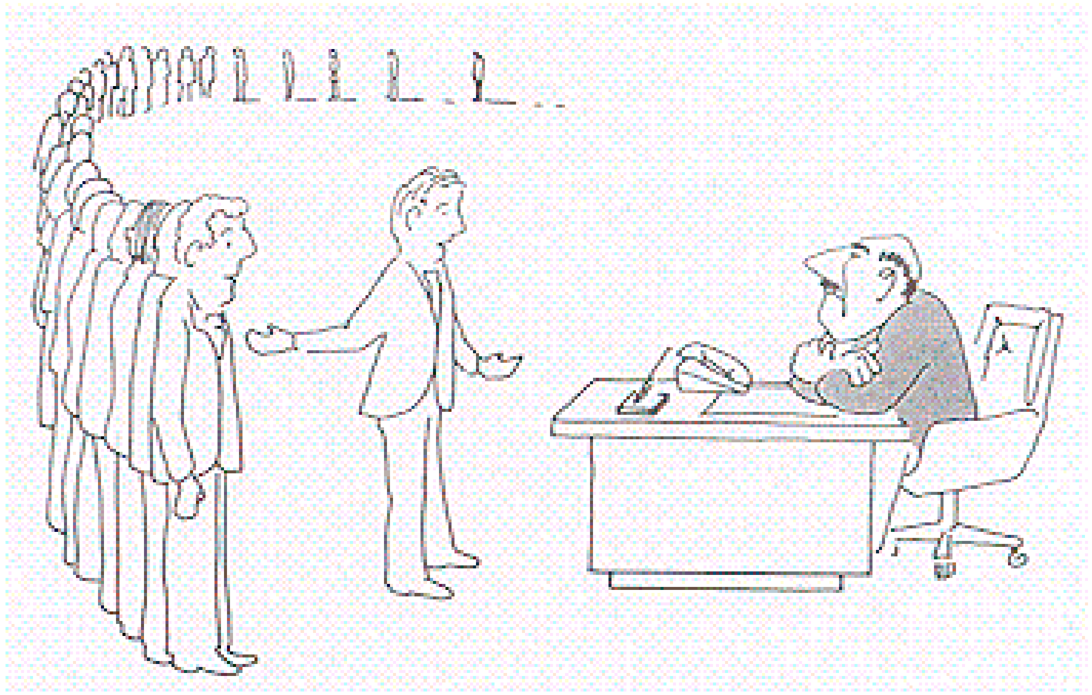
Genome: ATGGCGTGCA

Hamiltonian Cycles in *H*

- What is wrong with this approach?

Problem with *H*

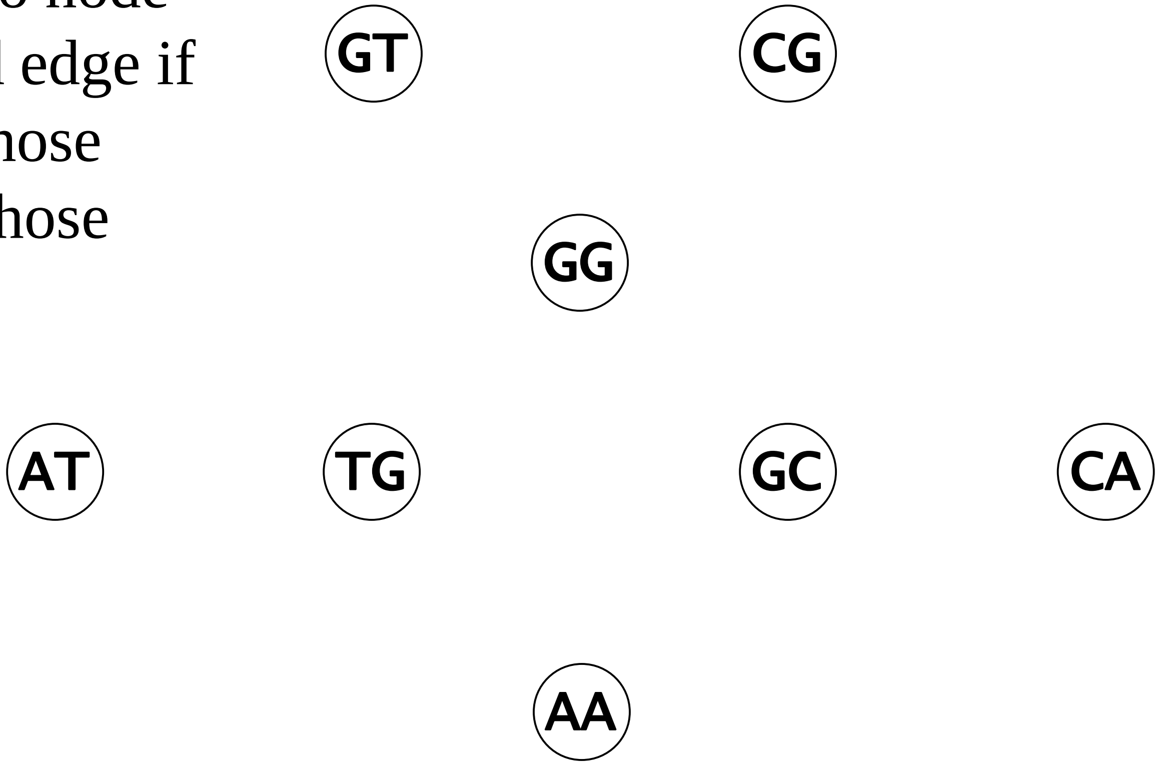
- Ultimately, we must solve the HCP on *H* (millions of nodes) in order to obtain a candidate genome …



Second Try: The Network *E*

- Form a different network *E* as follows:
  - Create a node for each *distinct* prefix/suffix from reads.
  - Connect node *v* to node *w* with a directed edge if there is a read whose prefix is *v* and whose suffix is *w*.

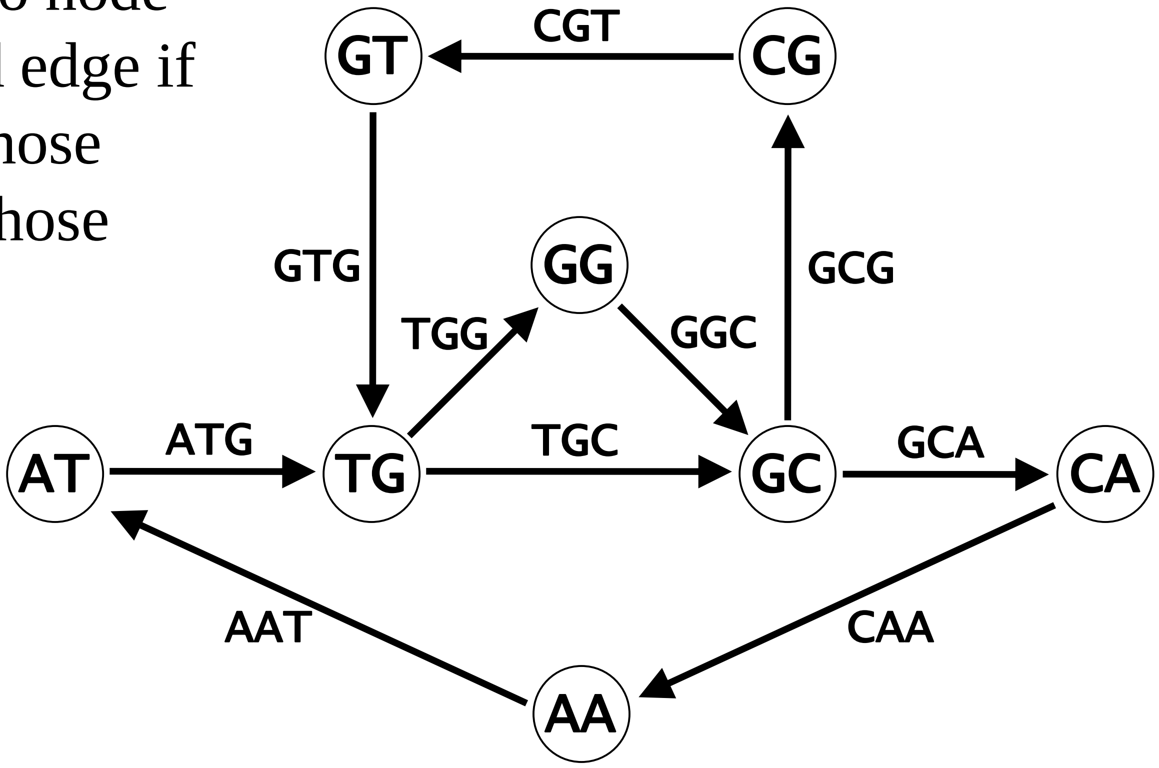
	GTG	TGC
	GCG	GGC
Reads	GCA	CGT
	ATG	CAA
	TGG	AAT



Second Try: The Network *E*

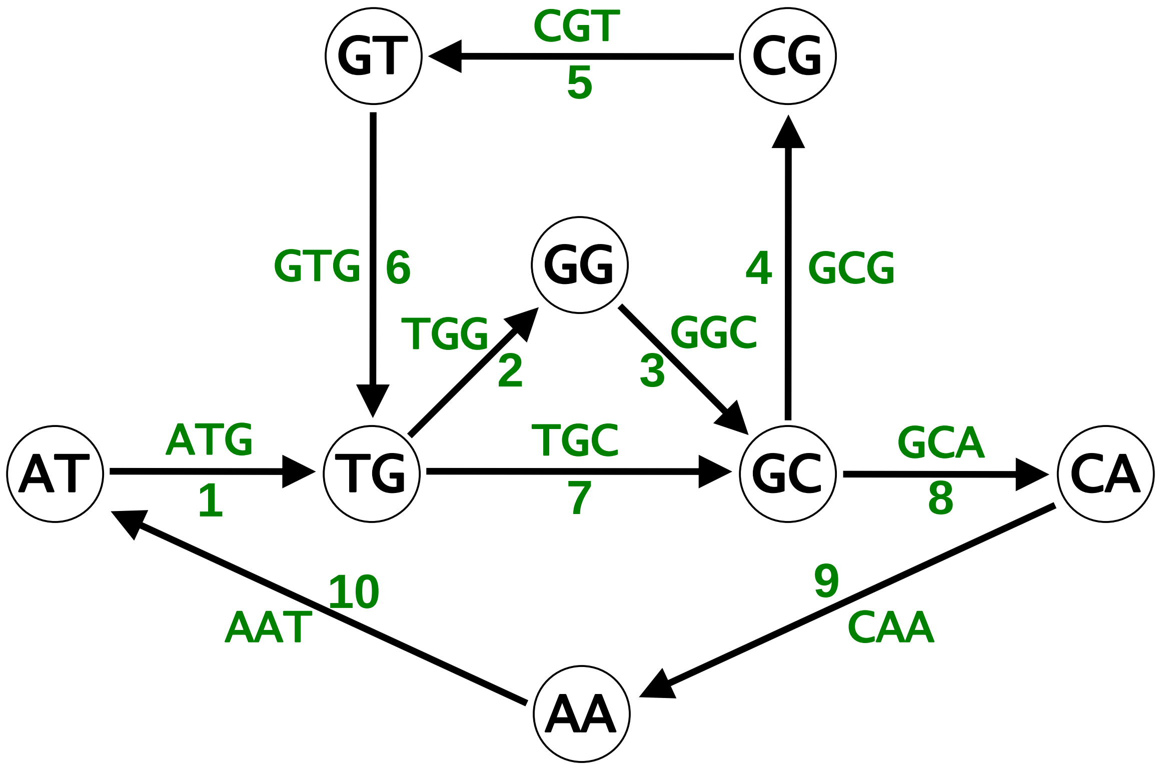
- Form a different network *E* as follows:
  - Create a node for each *distinct* prefix/suffix from reads.
  - Connect node *v* to node *w* with a directed edge if there is a read whose prefix is *v* and whose suffix is *w*.

	GTG	TGC
	GCG	GGC
Reads	GCA	CGT
	ATG	CAA
	TGG	AAT



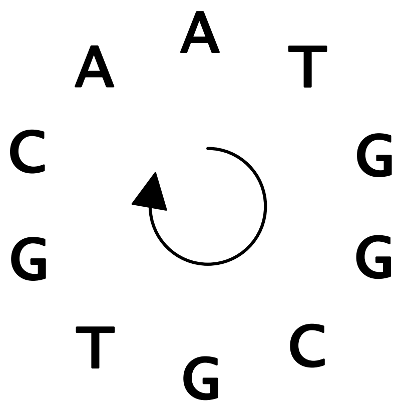
Eulerian Cycles in *E*

- We have an Eulerian cycle in *E*:
  - ATG → TGG → GGC → GCG → CGT → GTG → TGC → GCA → CAA → AAT



Eulerian Cycles in E

- We have an Eulerian cycle in *E*:
  - ATG → TGG → GGC → GCG → CGT → GTG → TGC → GCA → CAA → AAT
  - This is the same sequence of reads that we had in *H*!
  - Thus we will obtain the same sequenced genome as before.



```
ATG
 TGG
  GGC
   GCG
    CGT
     GTG
      TGC
       GCA
        CAA
         AAT
          ATG
```

Genome: ATGGCGTGCA

Eulerian Cycles in E

- We have an Eulerian cycle in *E*:
  - ATG → TGG → GGC → GCG → CGT → GTG → TGC → GCA → CAA → AAT
  - This is the same sequence of reads that we had in *H*!
  - Thus we will obtain the same sequenced genome as before.
  - **The only difference: a computer can find an Eulerian cycle quickly.**

```
ATG
 TGG
  GGC
   GCG
    CGT
     GTG
      TGC
       GCA
        CAA
         AAT
          ATG
```

Genome: ATGGCGTGCA

Example Problem

- What is the genome assembled from the following reads? Start with the read “GAT” when creating your Eulerian cycle

ACA

AGA

ATT

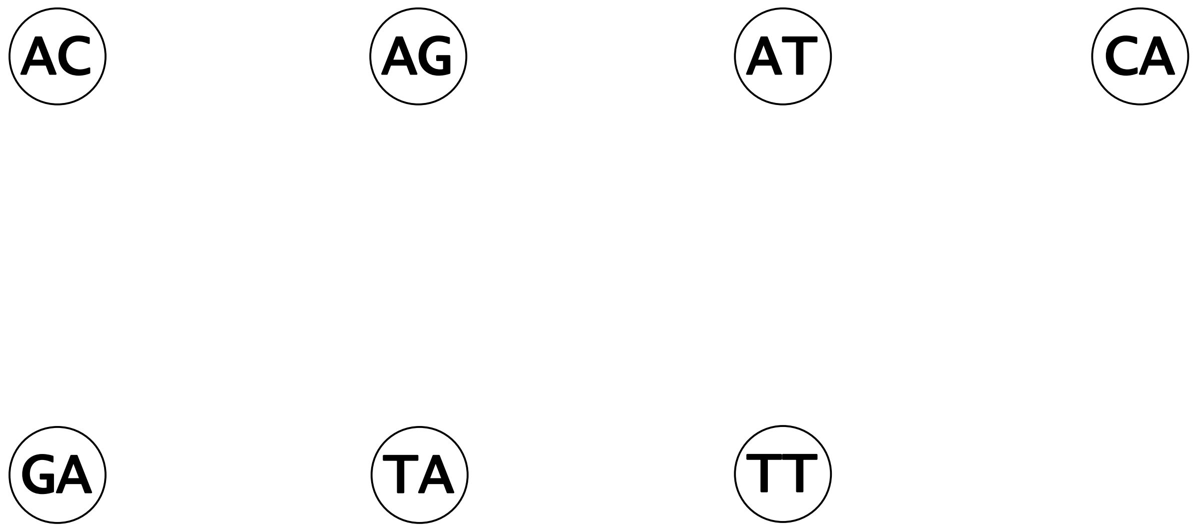
CAG

GAT

TAC

TTA

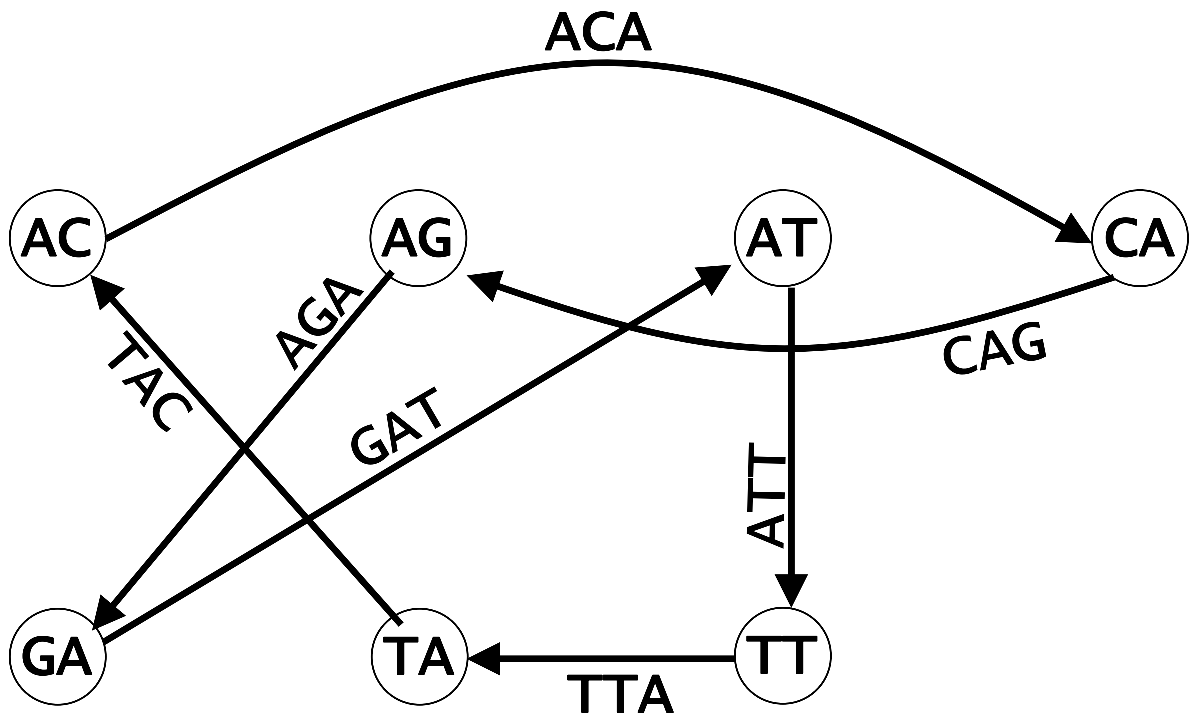
Example Problem



Reads:

ACA
AGA
ATT
CAG
GAT
TAC
TTA

Example Problem



<u>Reads:</u>

ACA

AGA

ATT

CAG

GAT

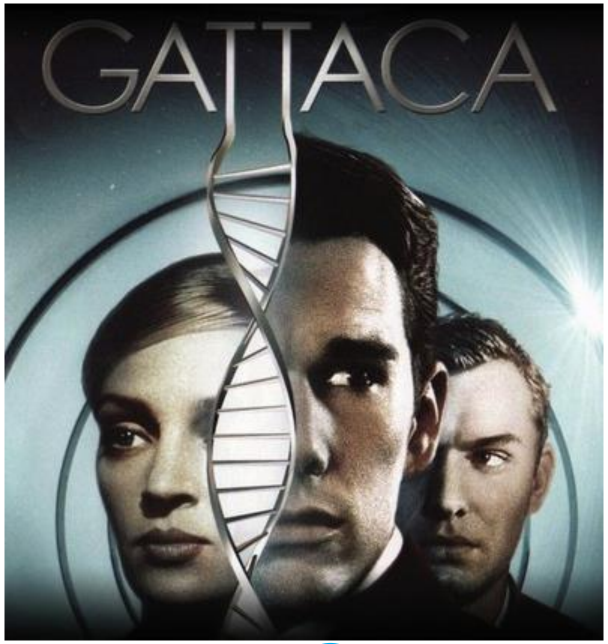
TAC

TTA

Example Problem

$$\begin{tikzcd}
AC \arrow[rrr, bend left, "ACA"] & AG \arrow[dl, "AGA"'] & AT \arrow[d, "ATT"] & CA \arrow[ll, bend left, "CAG"] \\
GA \arrow[urr, "GAT"] & TA \arrow[ul, "TAC"] & TT \arrow[l, "TTA"] &
\end{tikzcd}$$

GATTACA

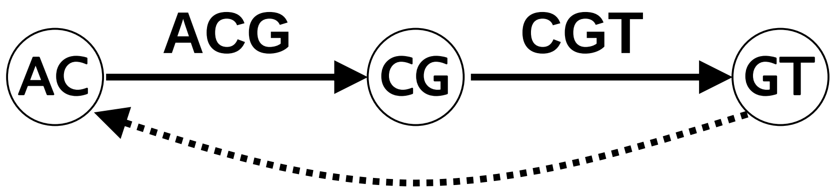


☺

Linear Genomes

- The previous example was for a circular genome, but what about for a linear genome?

- Example: “ACGT” (not circular)



- Now, we can use the exact same algorithm as before!

Section 9: Practical Complications

Analysis of *E*

- **Good News**: We now only have to find an Eulerian cycle in the network *E*.

- **Bad News**: We made some unrealistic assumptions.

1. In practice, reads are **error-prone**.
2. Reads have **imperfect coverage** (so we will not always be able to move from one read to the next).
3. Etc.

1st Unrealistic Assumption: Coverage Is Perfect

- Real reads capture only a small fraction of genome substrings.

```
atgccgtatggacaacgact
atgccgtatg
  gccgtatgga
     gtatggacaa
          gacaacgact
```

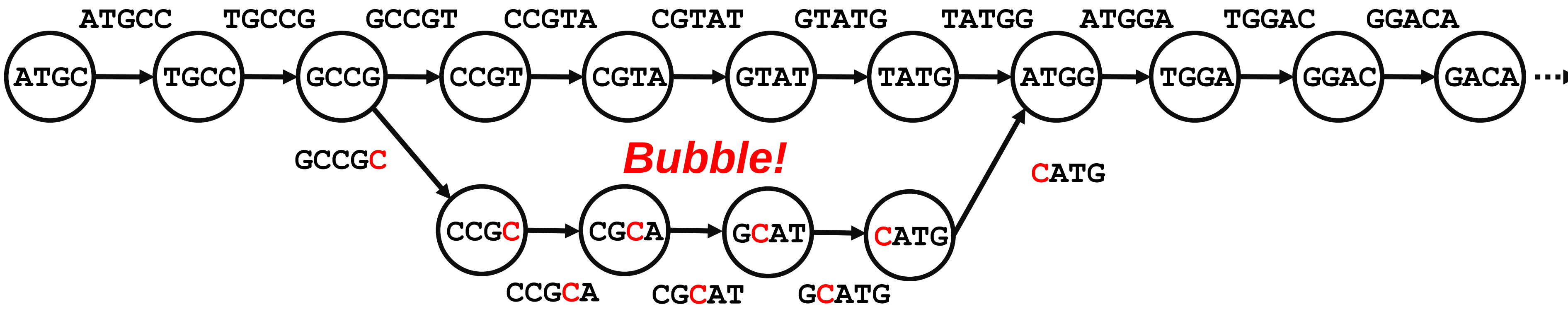
- What can we do?

Breaking Reads into Shorter Pieces

```
atgccgtatggacaacgact
atgccgtatg
  gccgtatgga
     gtatggacaa
          gacaacgact
```

```
atgccgtatggacaacgact
atgcc
 tgccg
  gccgt
   ccgta
    cgtat
     gtatg
      tatgg
       atgga
        tggac
         ggaca
          gacaa
           acaac
            caacg
             aacga
              acgac
               cgact
```

2nd Unrealistic Assumption: No Errors


ATGCC
TGCCG
GCCGT
CCGTA
CGTAT
GTATG
TATGG
ATGGA
TGGAC
GGACA
ATGC
TGCC
GCCG
CCGT
CGTA
GTAT
TATG
ATGG
TGGA
GGAC
GACA
GCCGC
Bubble!
CATG
CCGC
CGCA
GCAT
CATG
CCGCA
CGCAT
GCATG