Algorithmic Approaches for Biological Data, Lecture #17

Katherine St. John

City University of New York
American Museum of Natural History

6 April 2016

Outline

- Recap: Recursion

Outline

- Recap: Recursion
- Assembling Sequence Reads

Outline

- Recap: Recursion
- Assembling Sequence Reads
- Reframing Biology Questions

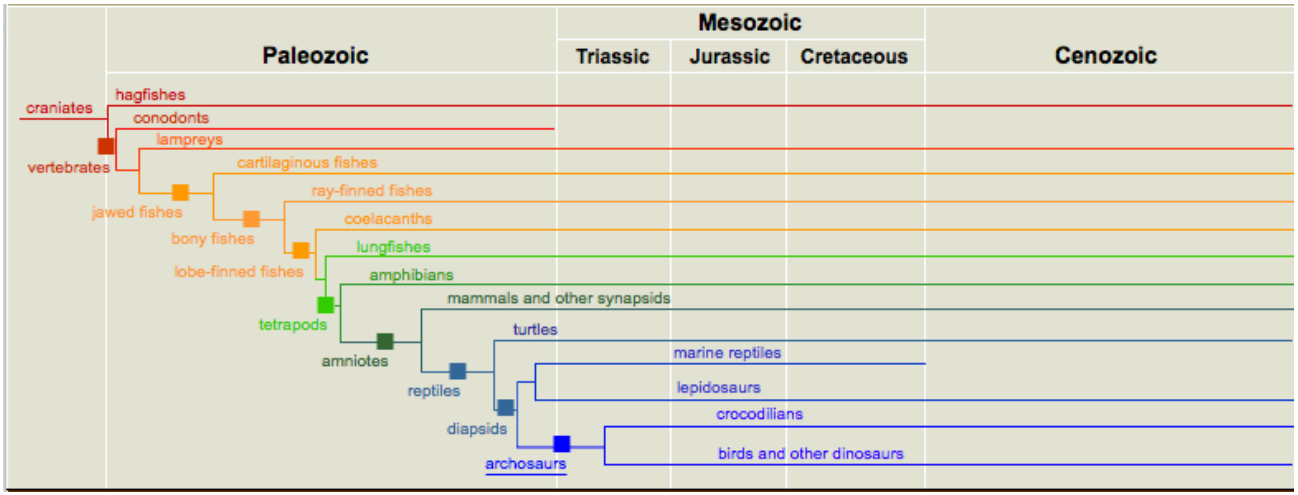
Outline

- Recap: Recursion
- Assembling Sequence Reads
- Reframing Biology Questions
- Overlap & Hamiltonian Graphs

Outline

- Recap: Recursion
- Assembling Sequence Reads
- Reframing Biology Questions
- Overlap & Hamiltonian Graphs
- Hamiltonian & Eulerian Paths

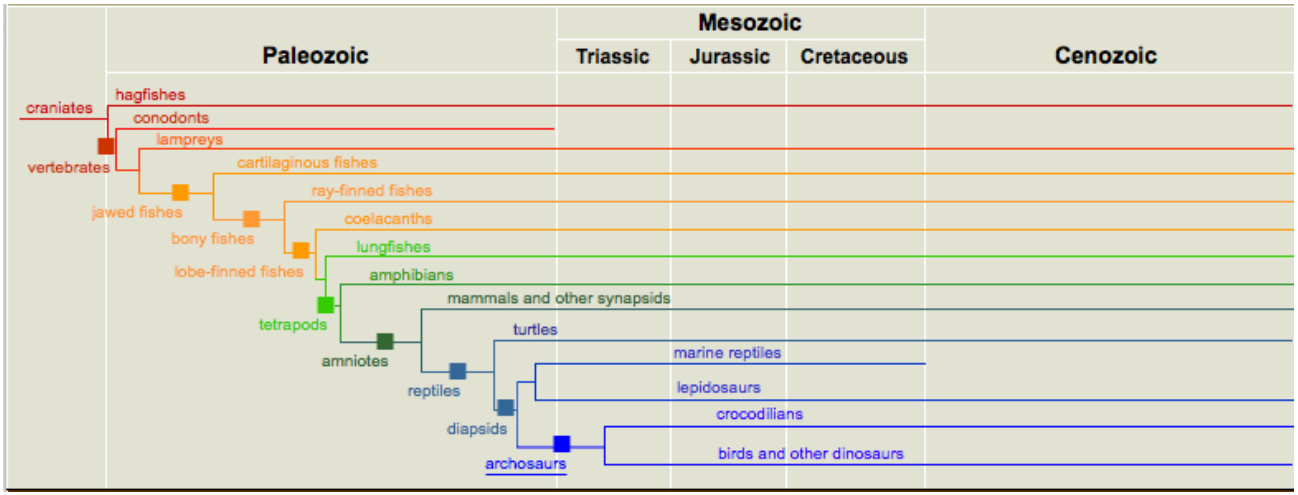
Recap: Recursion



Smithsonian Institute

- Works well for problems that break into pieces that can be solved independently

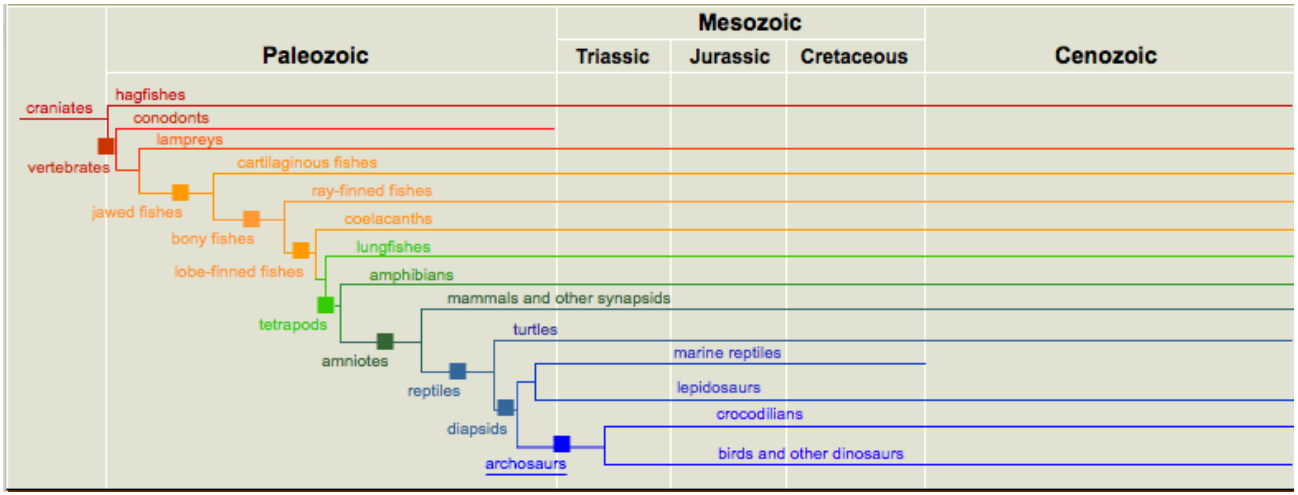
Recap: Recursion



Smithsonian Institute

- Works well for problems that break into pieces that can be solved independently
- Two parts:

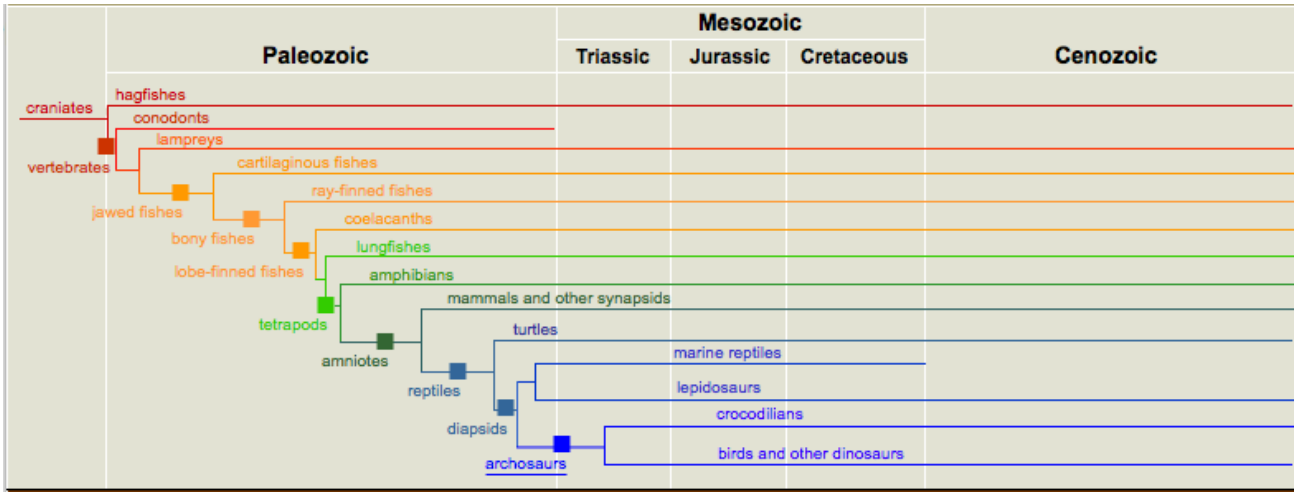
Recap: Recursion



Smithsonian Institute

- Works well for problems that break into pieces that can be solved independently
- Two parts:
  - Base Case: smallest possibility

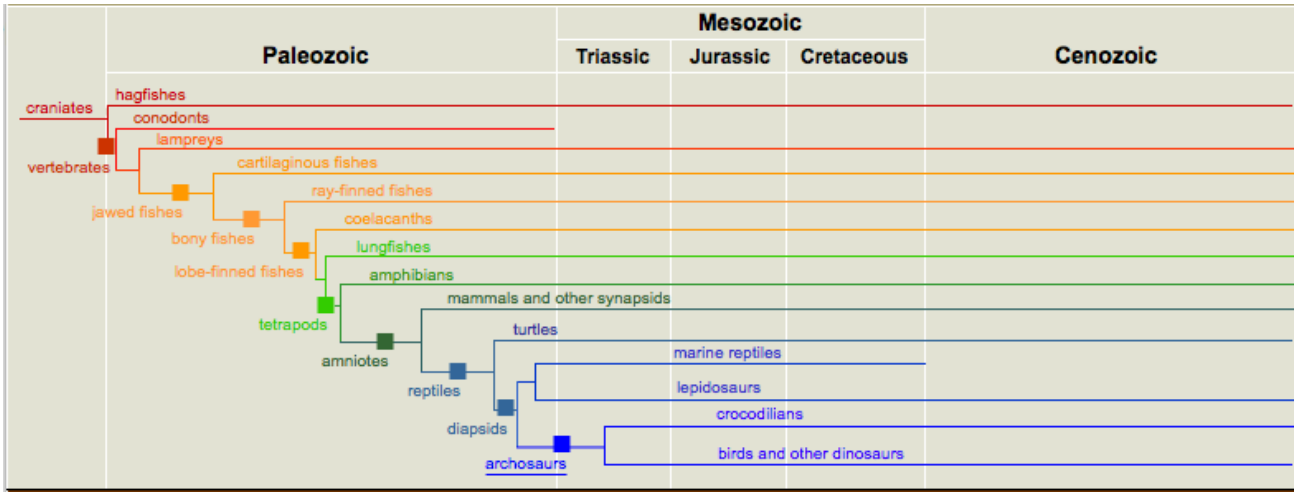
Recap: Recursion



Smithsonian Institute

- Works well for problems that break into pieces that can be solved independently
- Two parts:
  - Base Case: smallest possibility
    (for trees: almost always the leaves)

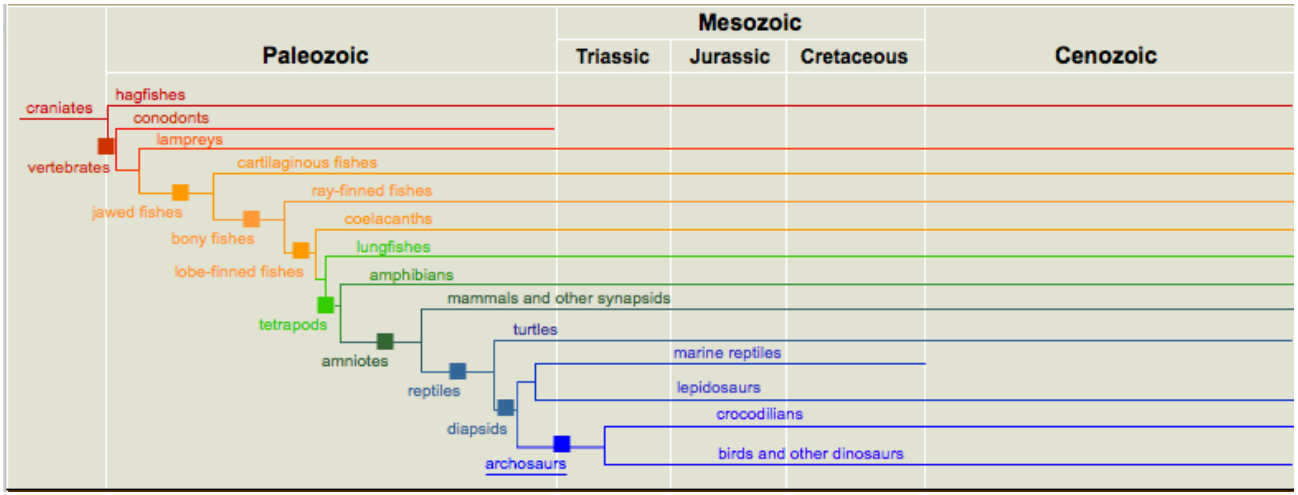
Recap: Recursion



Smithsonian Institute

- Works well for problems that break into pieces that can be solved independently
- Two parts:
  - Base Case: smallest possibility
    (for trees: almost always the leaves)
  - Recursive Call: calls the function on a "smaller" problem

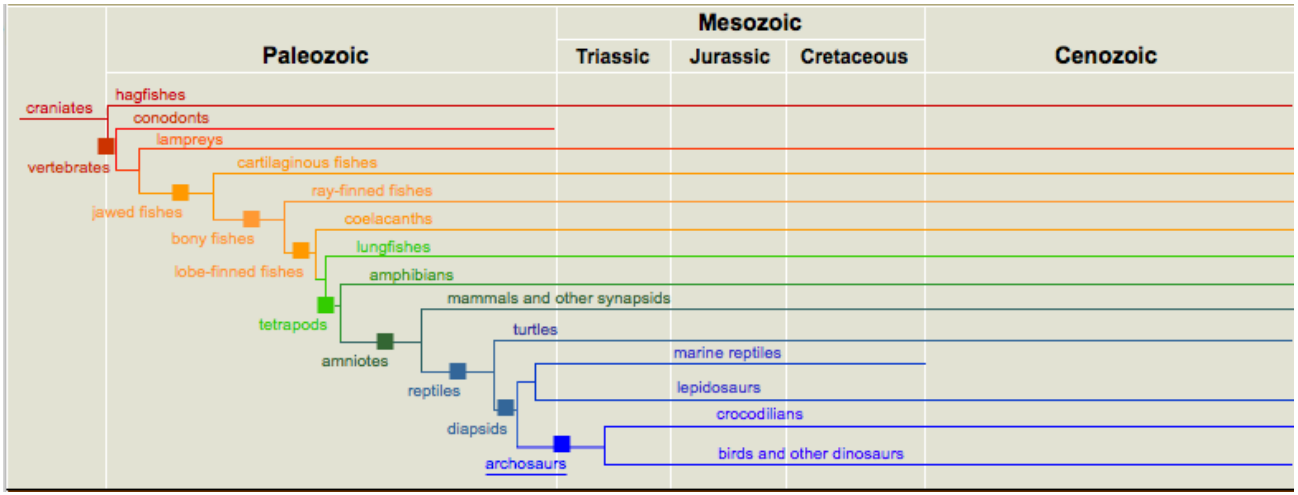
Recap: Recursion



Smithsonian Institute

- Works well for problems that break into pieces that can be solved independently
- Two parts:
  - Base Case: smallest possibility
    (for trees: almost always the leaves)
  - Recursive Call: calls the function on a "smaller" problem
    (for trees: internal nodes)

Recap: Recursion



Smithsonian Institute

- Works well for problems that break into pieces that can be solved independently
- Two parts:
  - Base Case: smallest possibility
    (for trees: almost always the leaves)
  - Recursive Call: calls the function on a "smaller" problem
    (for trees: internal nodes)
- *PythonTutor Demo*

Assembling Sequence Reads

Multiple identical copies of a genome

Shatter the genome into reads

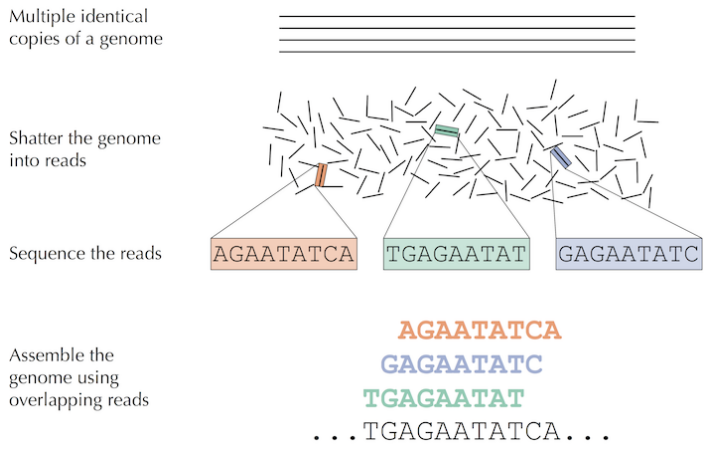
Sequence the reads

Assemble the genome using overlapping reads

AGAATATCA
GAGAATATC
TGAGAATAT
...TGAGAATATCA...

Compeau & Pevzner, Vol 1, Chapter 3

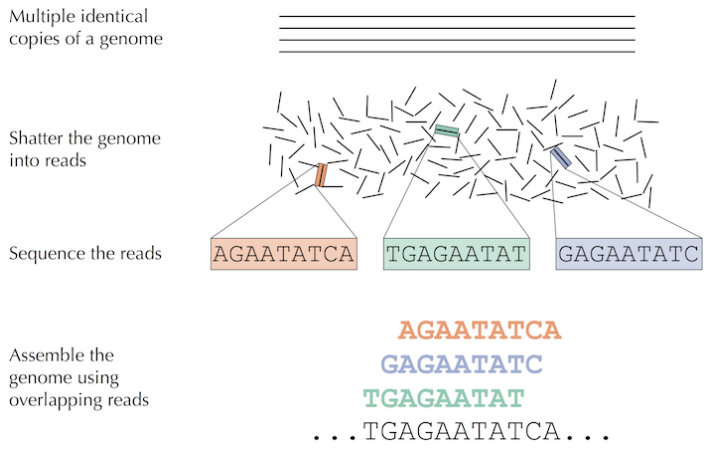
Assembling Sequence Reads



Compeau & Pevzner, Vol 1, Chapter 3

- Given a bunch of sequence reads (of length k), how do you assembly them?

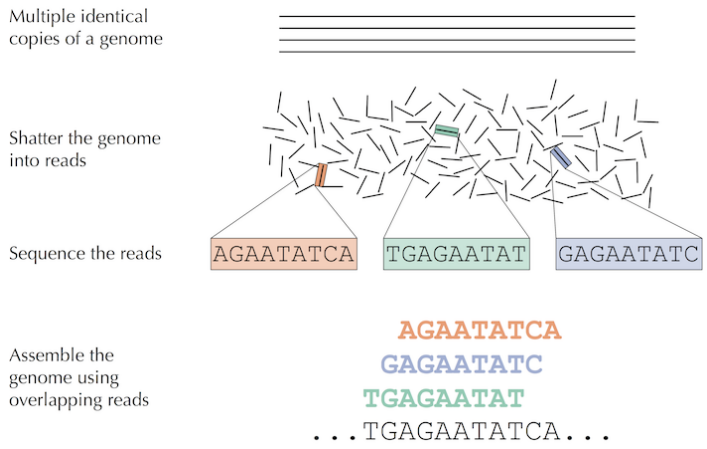
Assembling Sequence Reads



Compeau & Pevzner, Vol 1, Chapter 3

- Given a bunch of sequence reads (of length k), how do you assembly them?
- In pairs/triples:

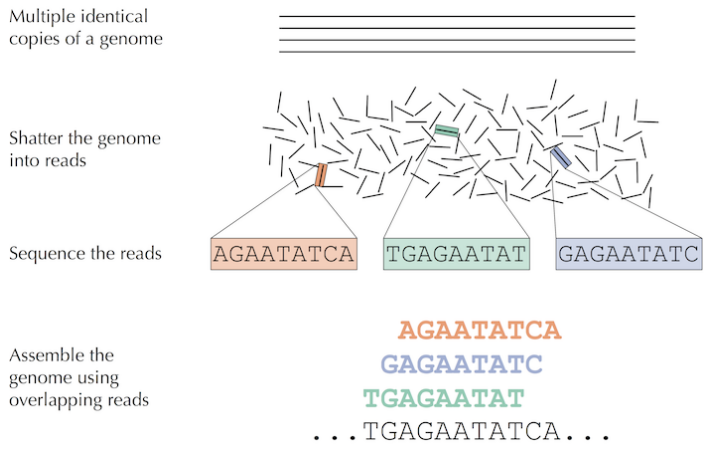
Assembling Sequence Reads



Compeau & Pevzner, Vol 1, Chapter 3

- Given a bunch of sequence reads (of length k), how do you assembly them?
- In pairs/triples:
  - 10 copies of a single-stranded sequence

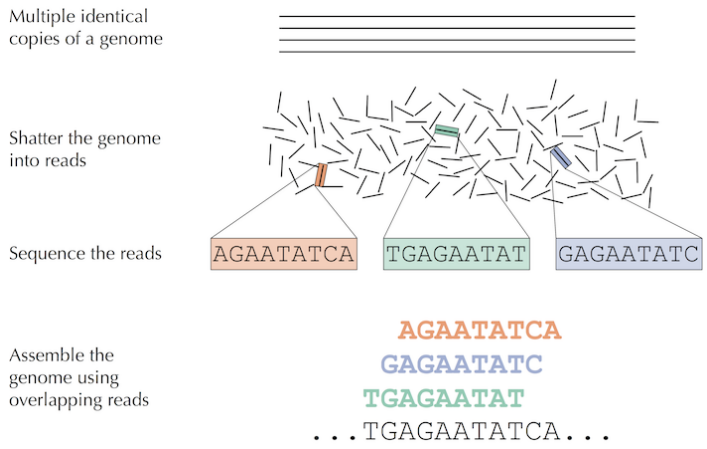
Assembling Sequence Reads



Compeau & Pevzner, Vol 1, Chapter 3

- Given a bunch of sequence reads (of length k), how do you assembly them?
- In pairs/triples:
  - 10 copies of a single-stranded sequence
  - k=4

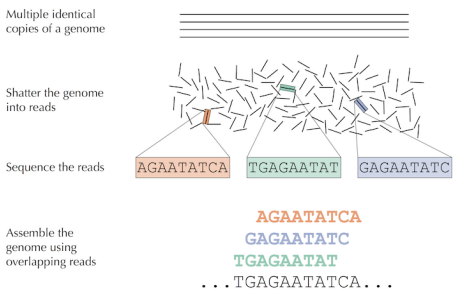
Assembling Sequence Reads



Compeau & Pevzner, Vol 1, Chapter 3

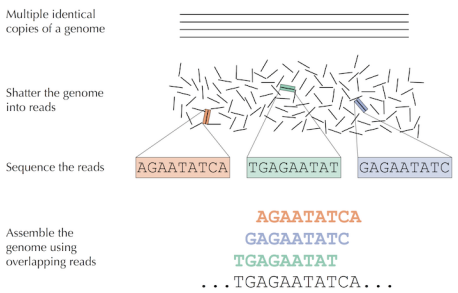
- Given a bunch of sequence reads (of length k), how do you assembly them?
- In pairs/triples:
  - 10 copies of a single-stranded sequence
  - k=4
  - Some errors in sequencing.

Reframing Biology Questions



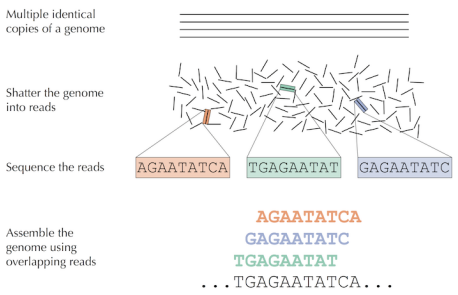
- How can we store the information in the computer?

Reframing Biology Questions



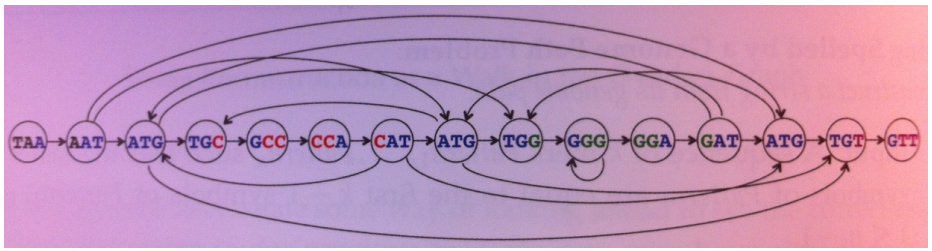
- How can we store the information in the computer?
- What additional information and what structures do we need?

Reframing Biology Questions

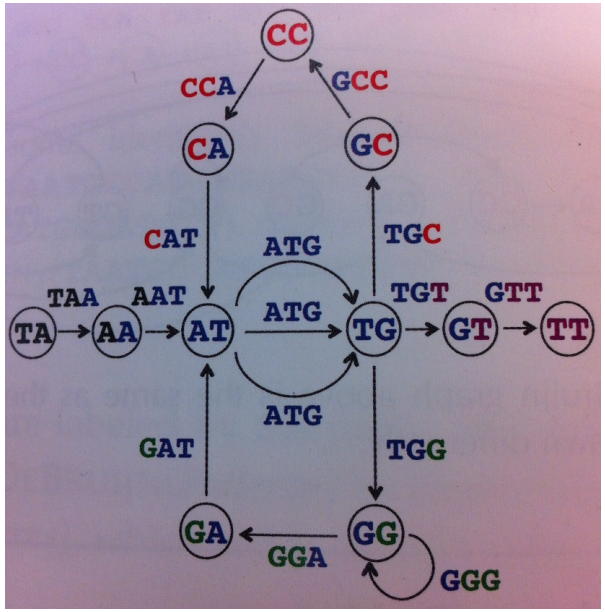


- How can we store the information in the computer?
- What additional information and what structures do we need?
- How do you represent overlaps in a graph?

Useful Graphs: Overlap & deBruijn



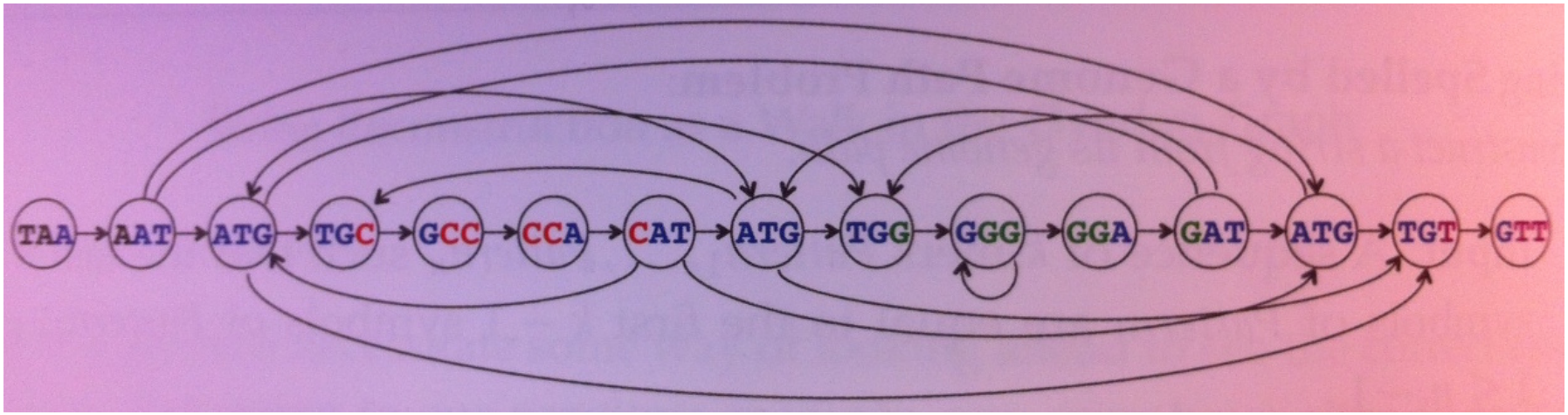
overlap graph



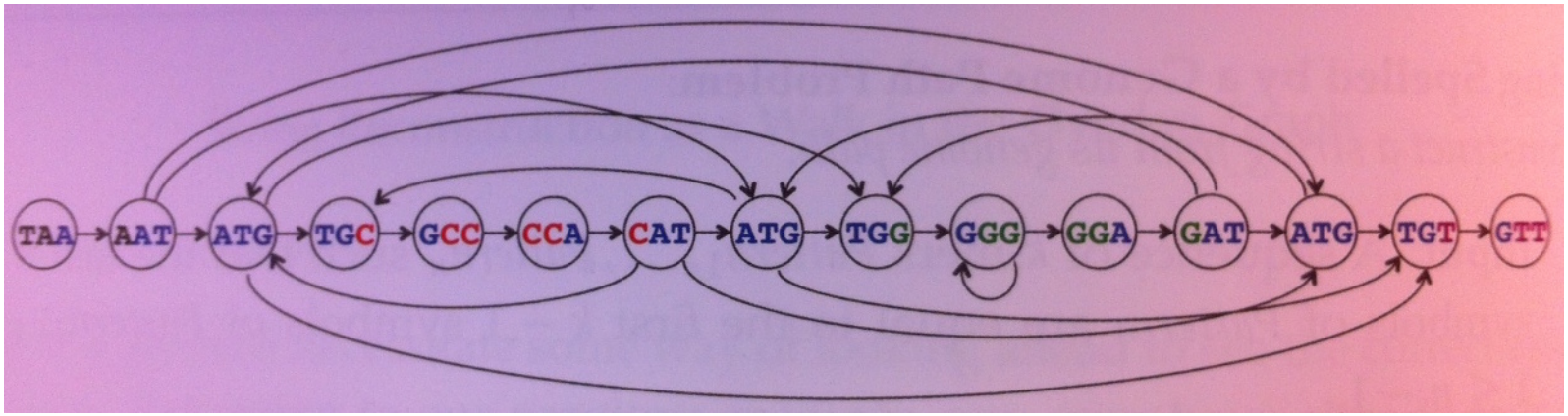
deBruijn graph

Compeau & Pevzner, Vol 1, Chapter 3

Overlap Graphs

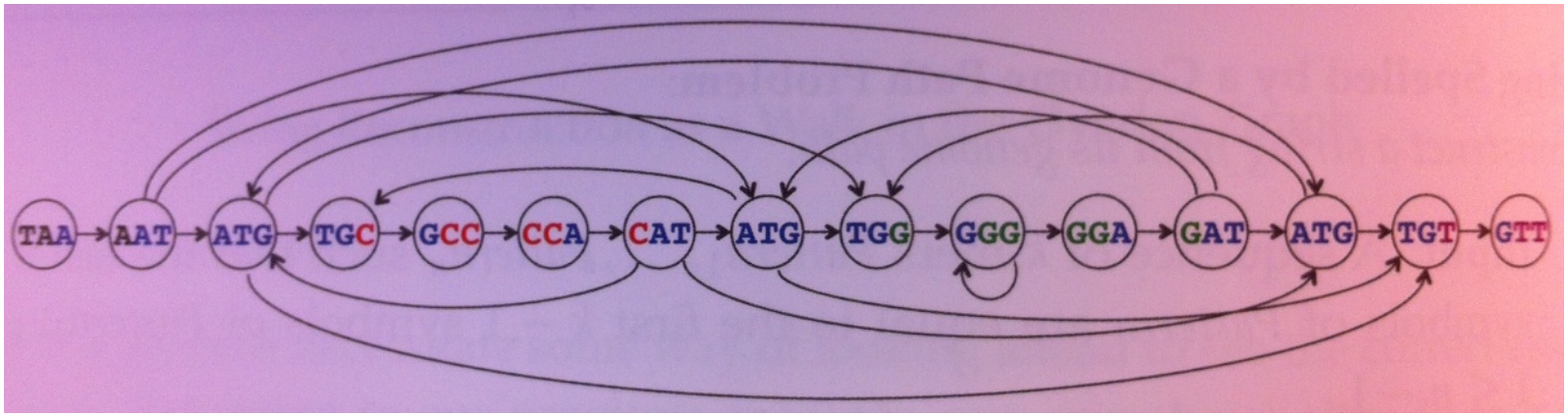


Overlap Graphs



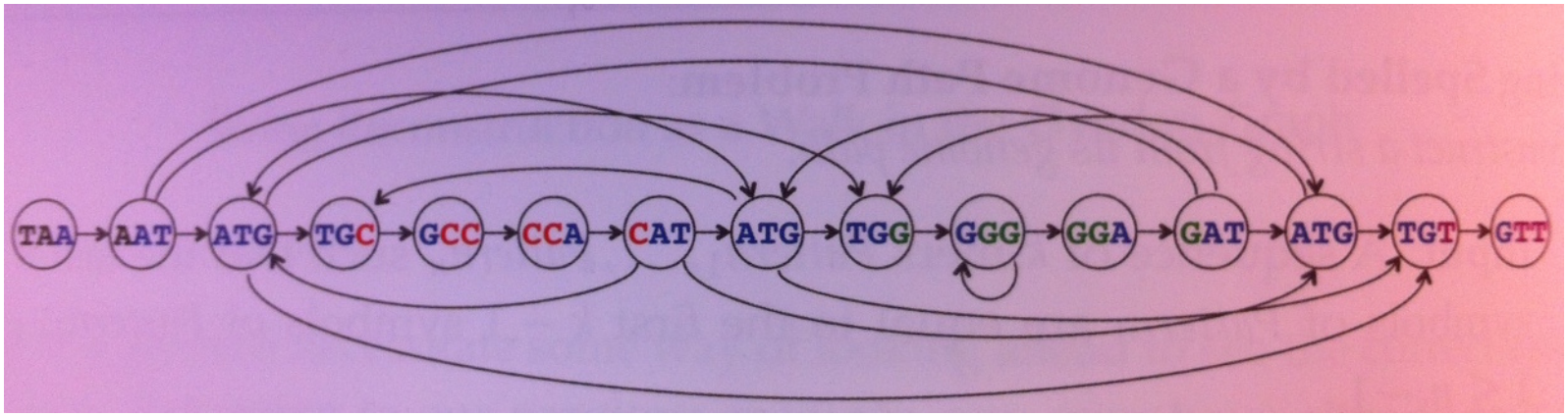
- The vertices are the reads.

Overlap Graphs



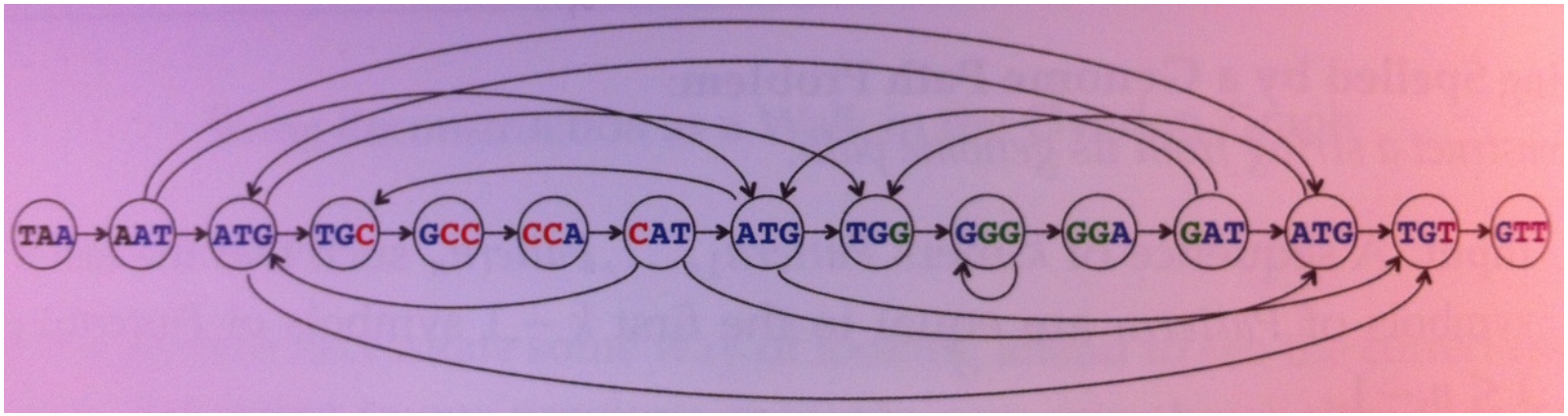
- The vertices are the reads.
- There's an edge from u to v if the $Suffix(u) = Prefix(v)$.

Overlap Graphs



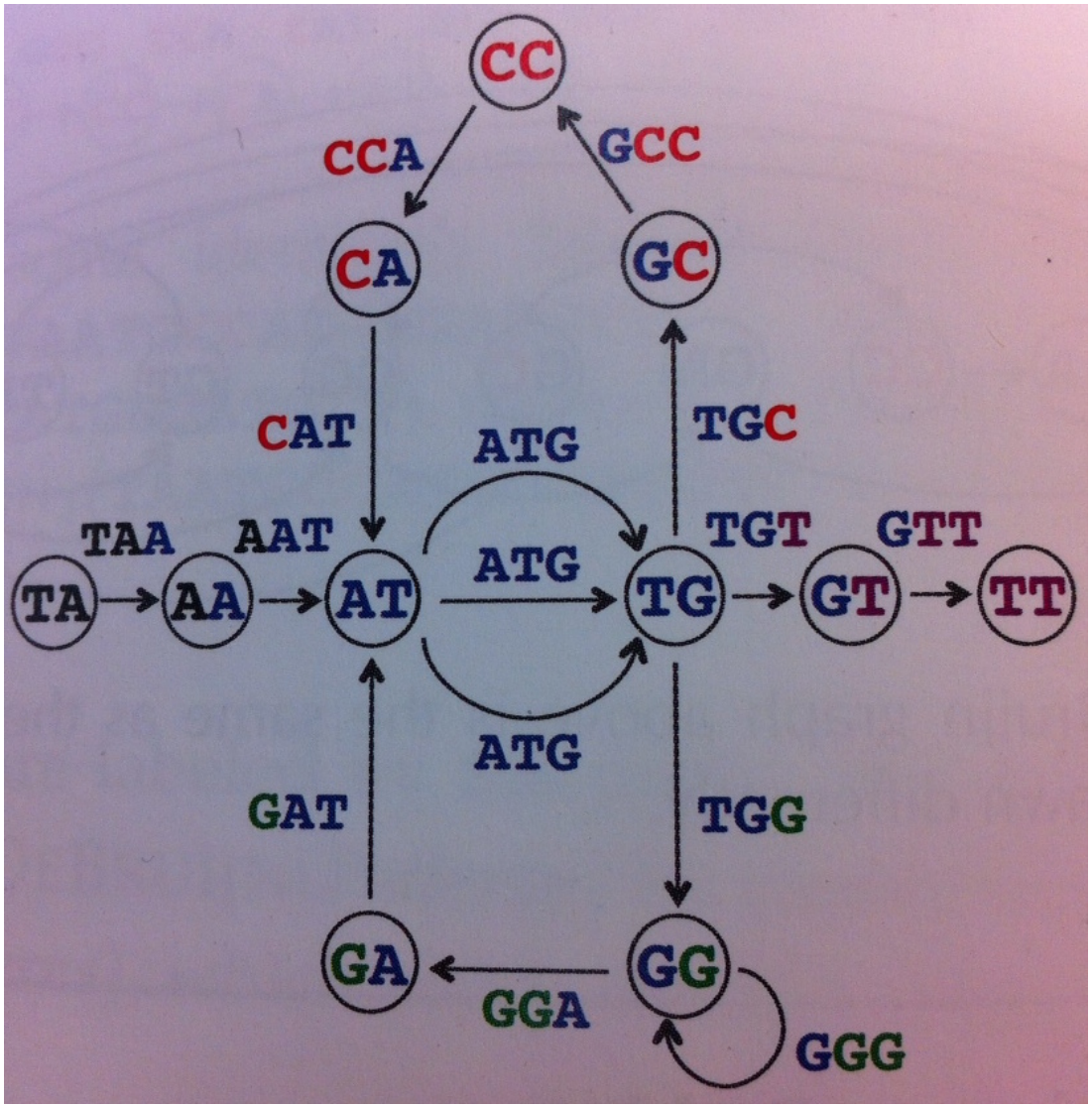
- The vertices are the reads.
- There's an edge from u to v if the $Suffix(u) = Prefix(v)$.
- Each read came from the original sequence, so, should be in the final sequence.

Overlap Graphs

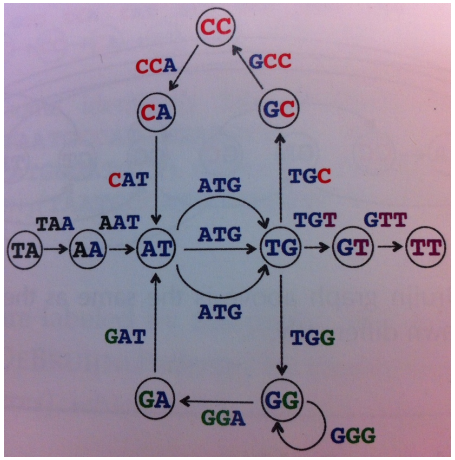


- The vertices are the reads.
- There's an edge from u to v if the $Suffix(u) = Prefix(v)$.
- Each read came from the original sequence, so, should be in the final sequence.
- Final sequence corresponds to a path that visits all the vertices (called Hamiltonian path).

deBruijn Graphs

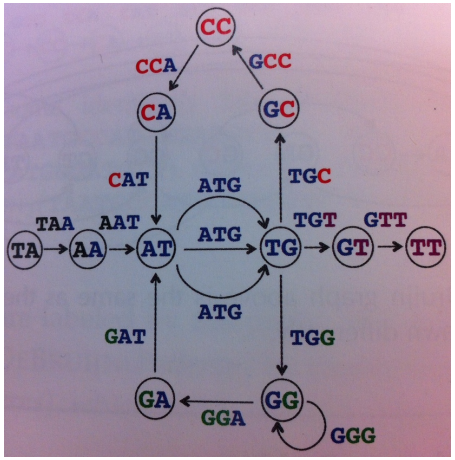


deBruijn Graphs



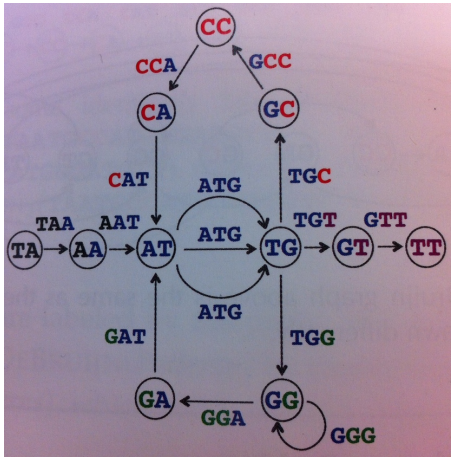
- Label edges by k-mers.

deBruijn Graphs



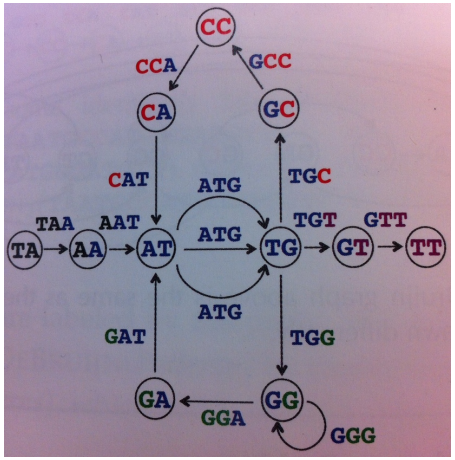
- Label edges by k-mers.
- Label vertices by the prefixes/suffixes.

deBruijn Graphs



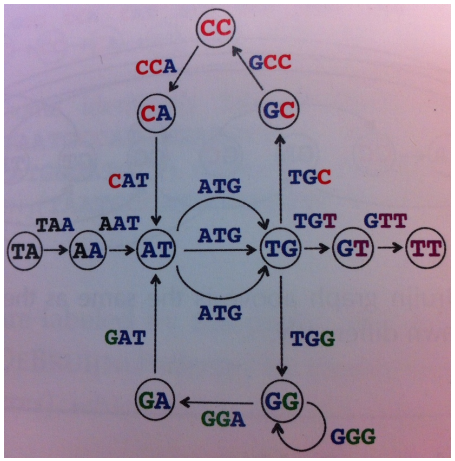
- Label edges by k-mers.
- Label vertices by the prefixes/suffixes.
- An Eulerian path through a graph visits every edge exactly once.

deBruijn Graphs



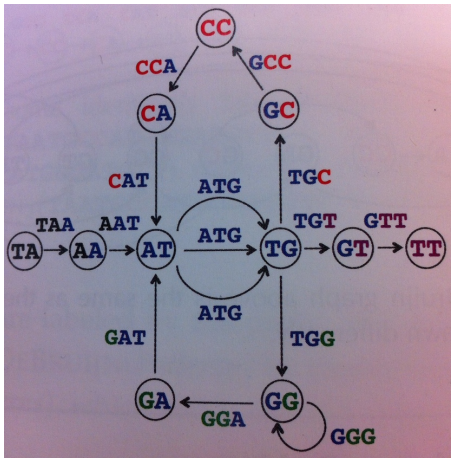
- Label edges by k-mers.
- Label vertices by the prefixes/suffixes.
- An Eulerian path through a graph visits every edge exactly once.
- The completed sequence is one of the Eulerian paths of the graph.

Computational Questions



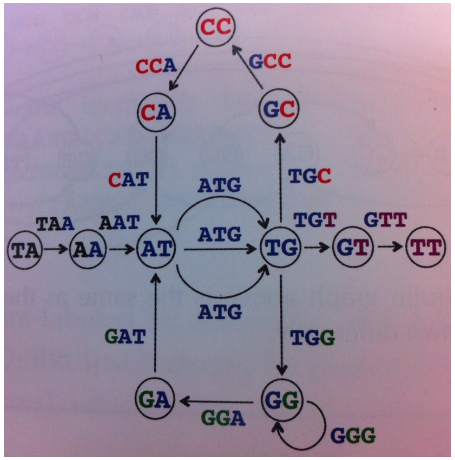
- Finding Hamiltonian paths is NP-hard.

Computational Questions



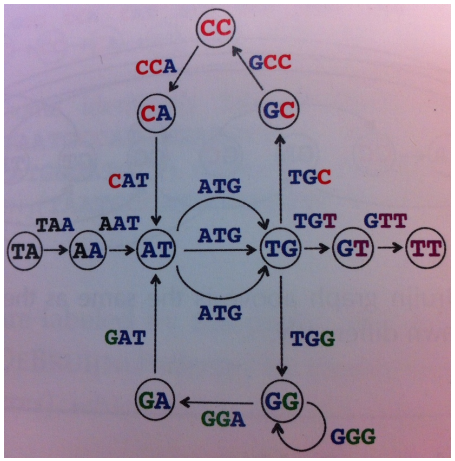
- Finding Hamiltonian paths is NP-hard.
- Eulerian paths exist if the out-degree = in-degree for every vertex.

Computational Questions



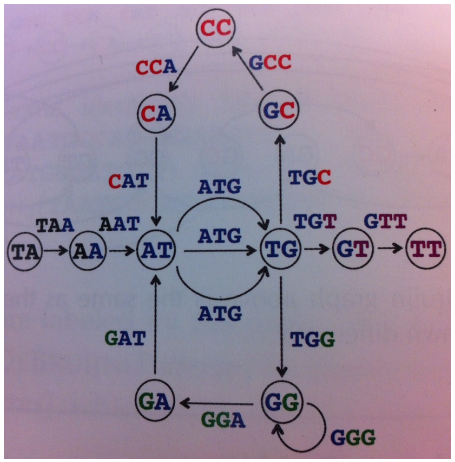
- Finding Hamiltonian paths is NP-hard.
- Eulerian paths exist if the out-degree = in-degree for every vertex.
- Since tractable, will compute Eulerian paths.

Computational Questions



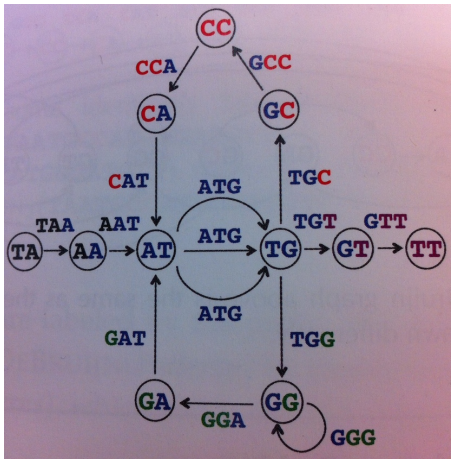
- Finding Hamiltonian paths is NP-hard.
- Eulerian paths exist if the out-degree = in-degree for every vertex.
- Since tractable, will compute Eulerian paths.
- And then, if more than one, choose (using additional information).

Building deBruijn Graphs



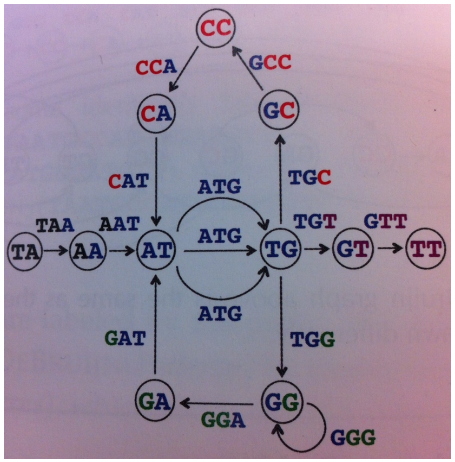
- Input: Reads of multiple sequences

Building deBruijn Graphs



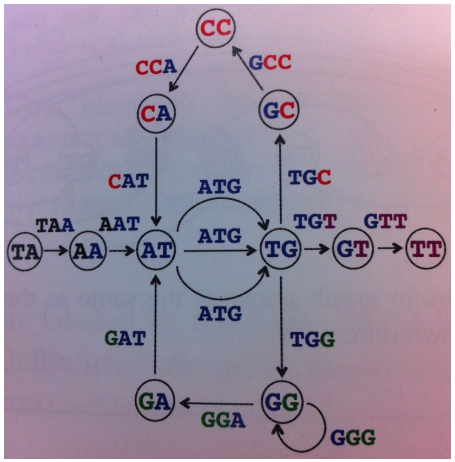
- Input: Reads of multiple sequences
- Make k-mers of the reads.

Building deBruijn Graphs



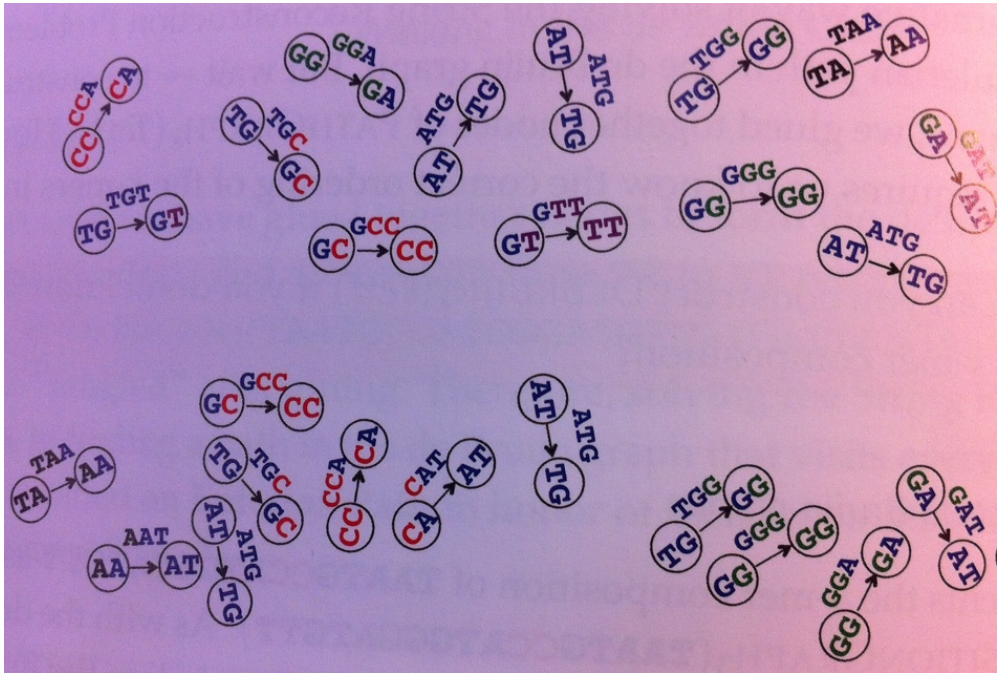
- Input: Reads of multiple sequences
- Make k-mers of the reads.
- Build deBruijn graphs from the reads.

Building deBruijn Graphs



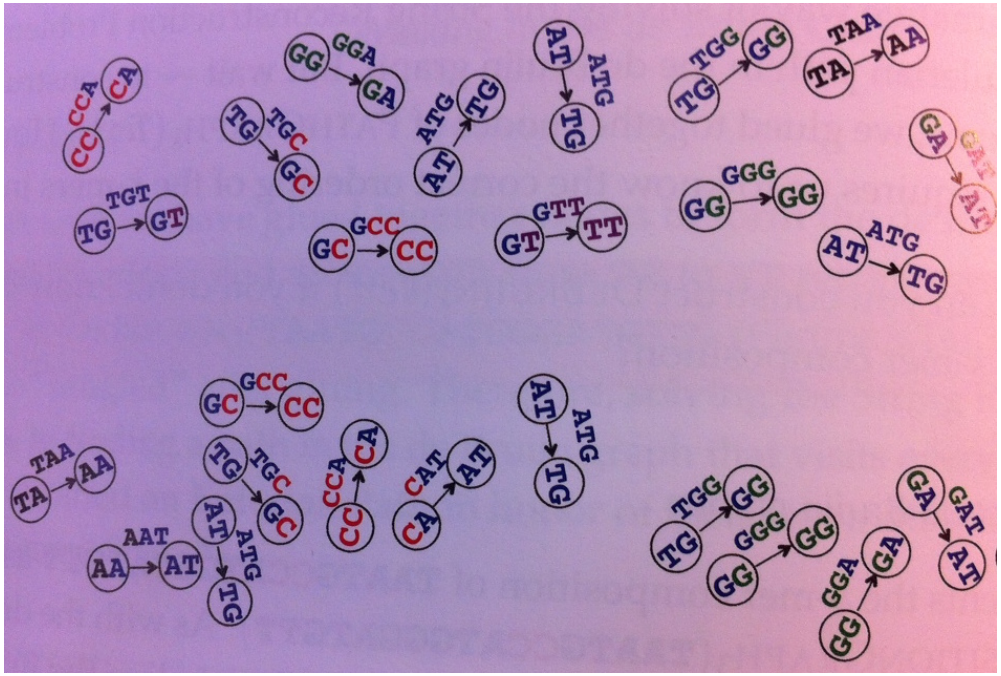
- Input: Reads of multiple sequences
- Make k-mers of the reads.
- Build deBruijn graphs from the reads.
- (Find Eulerian paths...)

Recap



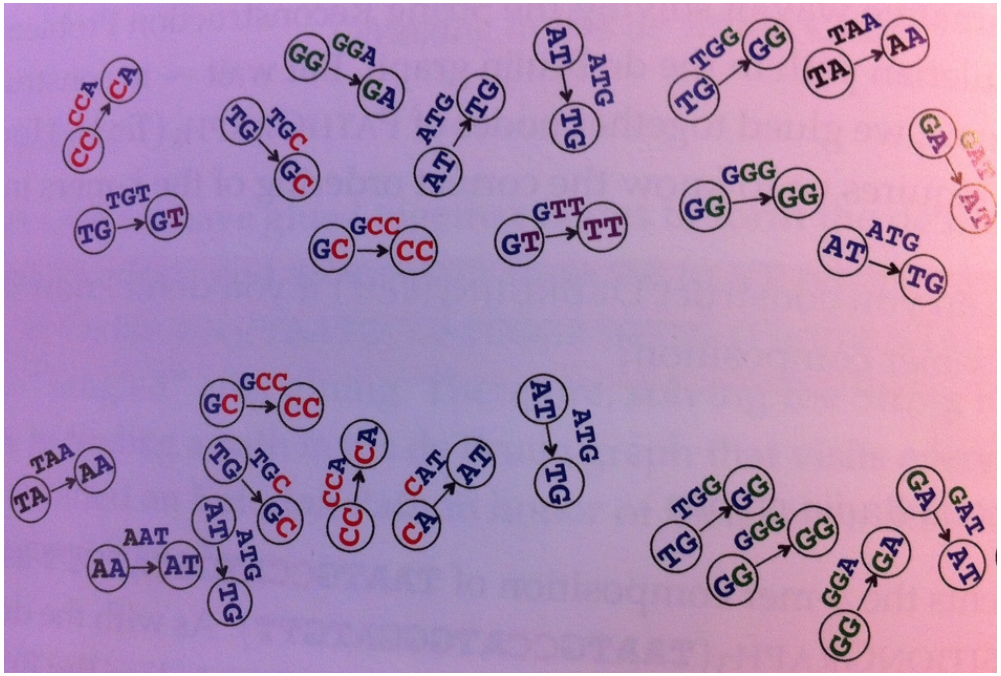
- Coding the deBruijn graph functions in lab today.

Recap



- Coding the deBruijn graph functions in lab today.
- Email lab reports to `kstjohn@amnh.org`

Recap



- Coding the deBruijn graph functions in lab today.
- Email lab reports to `kstjohn@amnh.org`
- Challenges available at `rosalind.info`