Sequence Assembly

Graphs and fragment assembly

Outline

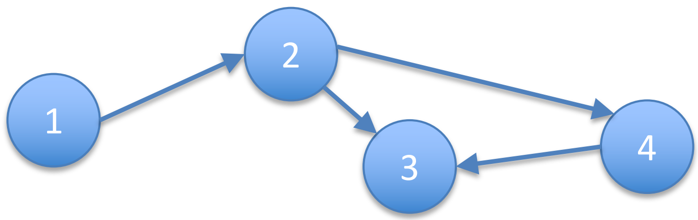
- Graphs
- Fragment assembly as a task on the overlap graph
- Greedy algorithms
- Operations on the overlap graph

Graph Basics

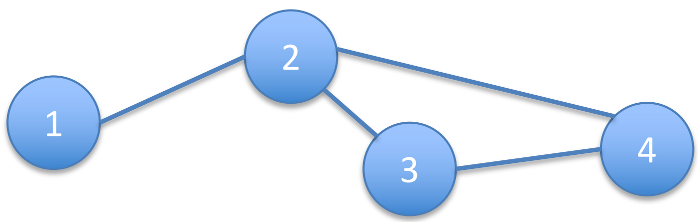
- A graph (*G*) consists of vertices (*V*) and edges (*E*)

$$G = (V,E)$$

- Edges can either be *directed* (*directed graphs)*

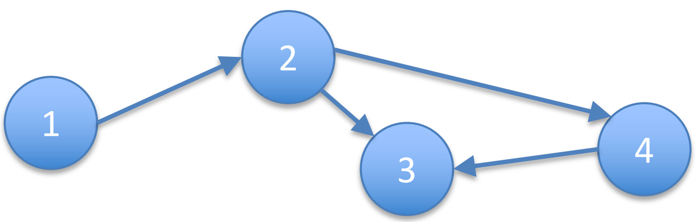


- or *undirected* (*undirected graphs)*



Vertex degrees

- The *degree* of a vertex: the # of edges incident to that vertex
- For directed graphs, we also have the notion of
  - *indegree:* The number incoming edges
  - *outdegree:* The number of outgoing edges



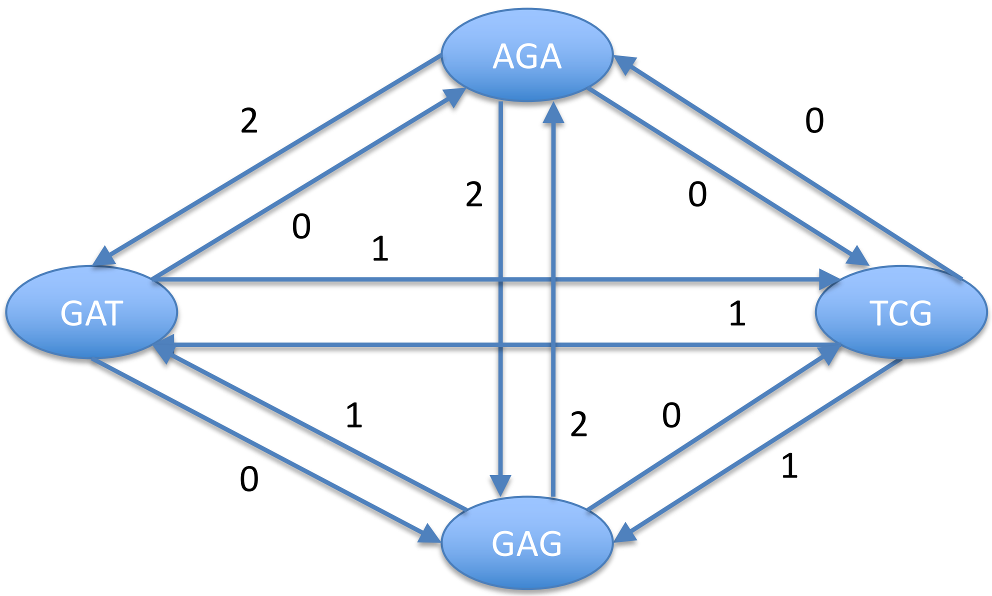
degree$(v_2) = 3$
indegree$(v_2) = 1$
outdegree$(v_2) = 2$

Overlap graph

- For a set of sequence reads S, construct a directed weighted graph $G = (V,E,w)$
  - with one vertex per read (v_i corresponds *to* s_i)
  - edges between all vertices (a *complete* graph)
  - $w(v_i,v_j) = overlap(s_i,s_j) =$ length of longest suffix of s_i that is a prefix of s_j

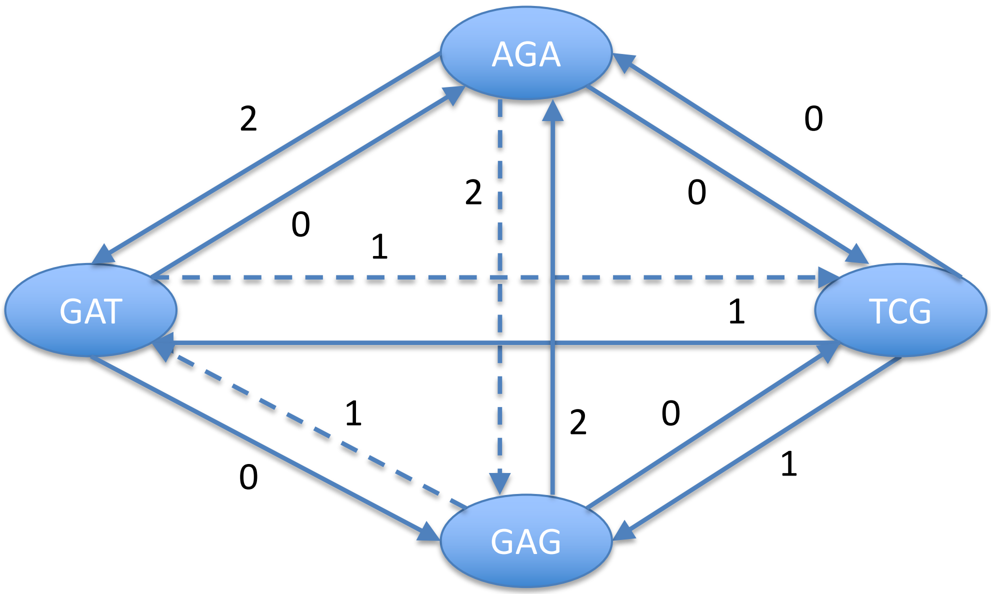
Overlap graph example

- Let S = {AGA, GAT, TCG, GAG}



Assembly as Hamiltonian Path

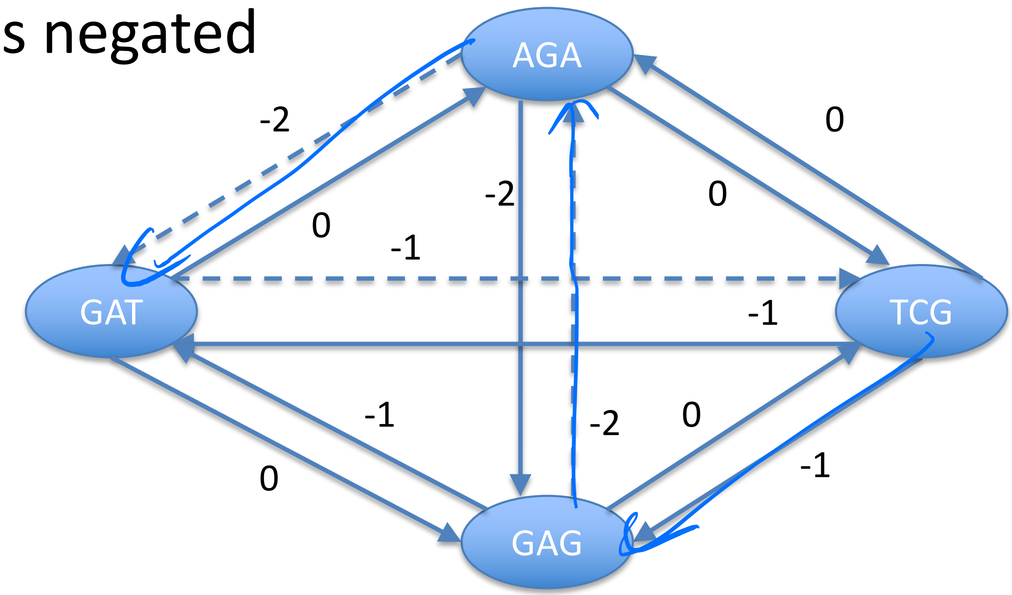
- *Hamiltonian Path*: path through graph that visits each vertex exactly once



Path: AGAGATCG

Shortest superstring as TSP

- minimize superstring length ➔minimize hamiltonian path length in overlap graph with edge weights negated



Path: GAGATCG
Path length: -5
String length: 7

- This is essentially the Traveling Salesman Problem (also *NP*-complete)

The Greedy Algorithm

- Let *G* be a graph with fragments as vertices, and no edges to start
- Create a queue, Q, of overlap edges, with edges in order of increasing weight
- While *G* is disconnected
  - Pop the next possible edge *e* = (*u*,*v*) off of Q
  - If *outdegree(u) = 0* and *indegree(v) = 0* and *e* does not create a cycle
    - Add *e* to *G*

Greedy Algorithm Example

Q:

AGA → GAG	-2
GAG → AGA	-2
AGA → GAT	-2
TCG → GAG	-1
GAG → GAT	-1
GAT → TCG	-1
TCG → GAT	-1
.	
.	0
.	
	0

ATCGA $\xrightarrow{-3}$ CGAAG $\xrightarrow{-2}$ AGAG $\xrightarrow{-1}$ GGG

Greedy Algorithms

- **Definition**: An algorithm that always takes the best immediate, or local, solution while finding an answer.
- Greedy algorithms find the overall, or globally, optimal solution for some optimization problems, but may find less-than-optimal solutions for some instances of other problems.

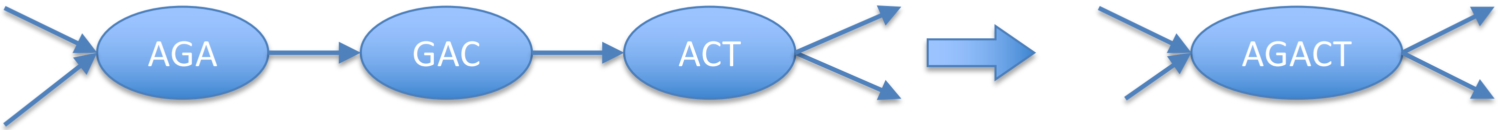
Paul E. Black, "greedy algorithm", in Dictionary of Algorithms and Data Structures [online], Paul E. Black, ed., U.S. National Institute of Standards and Technology. 2 February 2005. http://www.itl.nist.gov/div897/sqg/dads/HTML/greedyalgo.html

Greedy Algorithm Examples

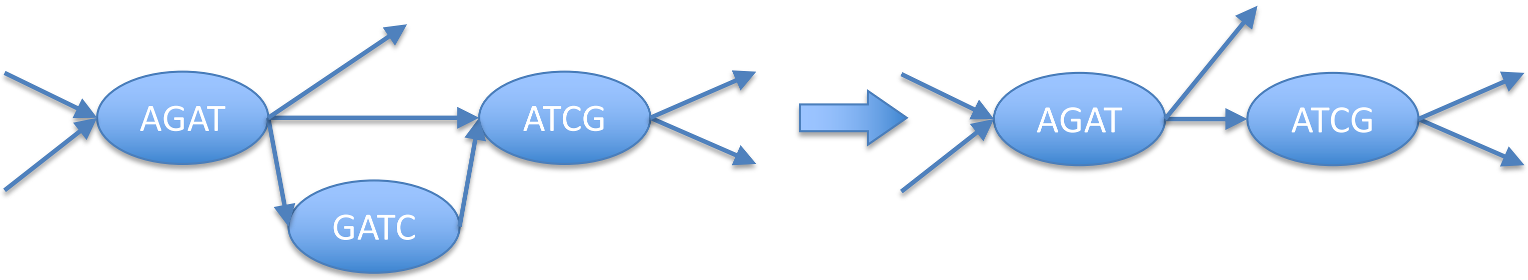
- Kruskal's Algorithm for Minimum Spanning Tree
  - *Minimum spanning tree*: a set of n-1 edges that connects a graph of n vertices and that has minimal total weight
  - *Kruskal's algorithm* adds the edge that connects two components with the smallest weight at each step
    - Proven to give an optimal solution
- Traveling Salesman Problem
  - Greedy algorithm chooses to visit closest vertex at each step
  - Can give far-from-optimal answers

Simplifications of overlap graph

- Require minimum length for overlap
- Linear chain compression



- Transitive edge removal



Summary

- Fragment assembly algorithms often use graphs
  - Overlap graph: vertices = reads, edges = overlaps
  - Shortest superstring = minimum weight Hamiltonian path
- Greedy algorithms are often simple and intuitive but are not guaranteed to give the optimal solution (except for certain problems)