Genome Assembly Paradigms

CMSC 423

Carl Kingsford

Shortest Common Superstring

Def. Given strings s_1, ..., s_n, find the shortest string T such that each s_i is a sub**string** of T.

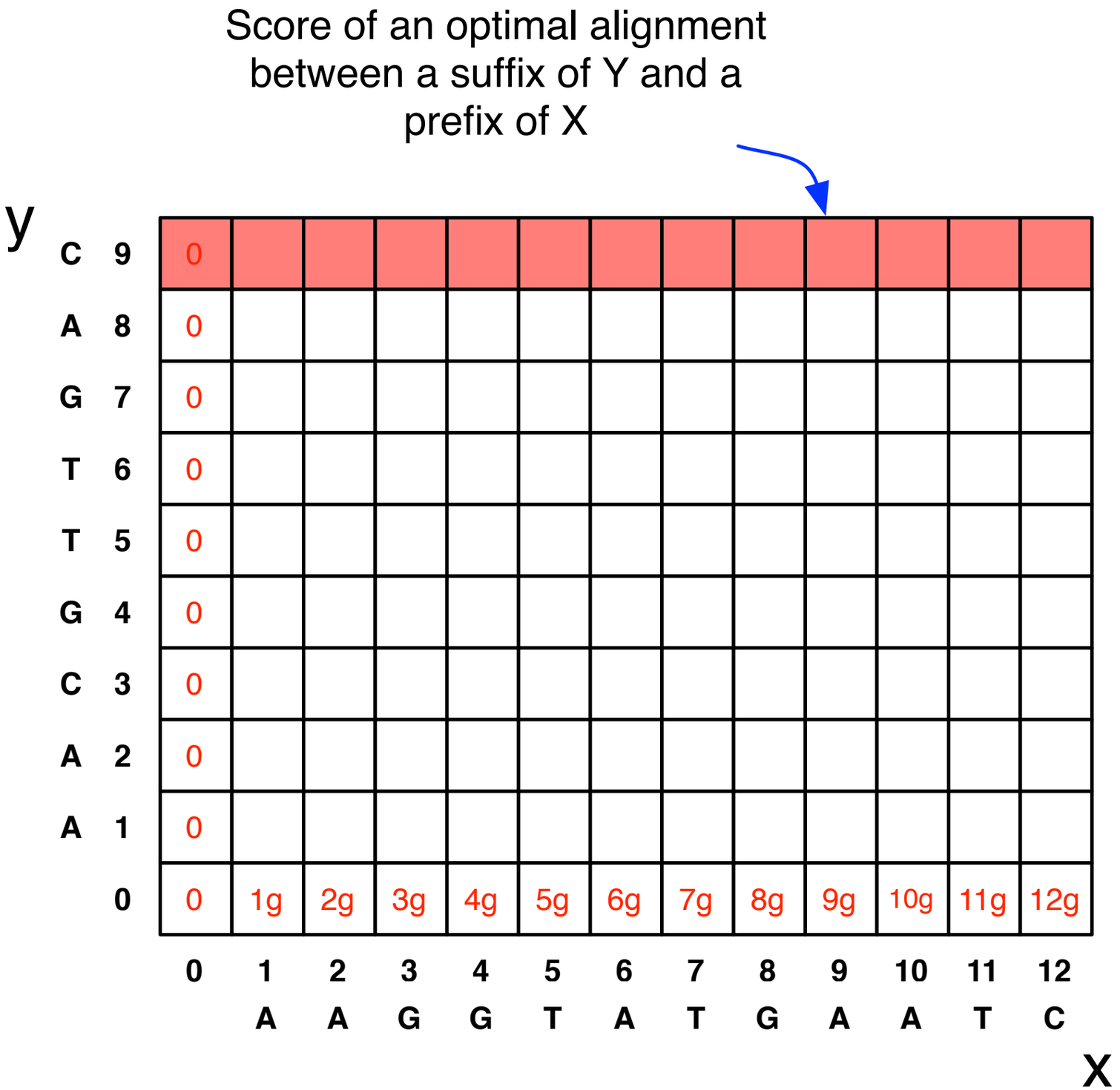
- NP-hard (contrast with case when requiring s_i to be sub**sequences** of T)

- Approximation algorithms exist with factors: 4, 3, 2.89, 2.75, 2.67, 2.596, 2.5, ...

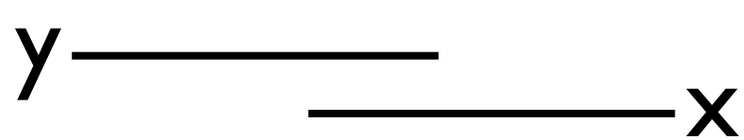
- Basic greedy method: find pair of strings that overlap the best, merge them, repeat (4 approximation):

Given match, mismatch, gap costs, how can we compute the score of the best overlap?

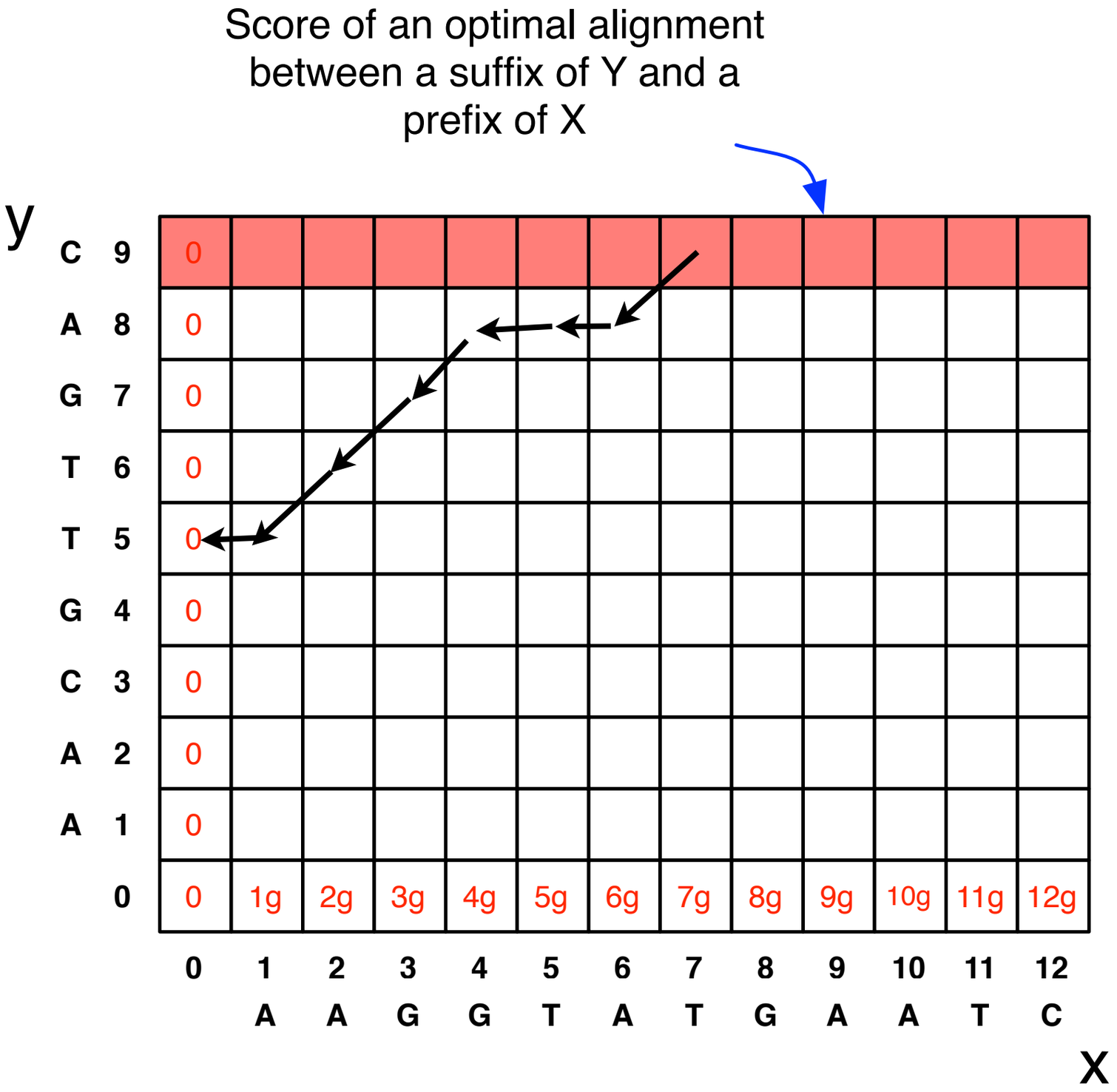
Overlap Alignment



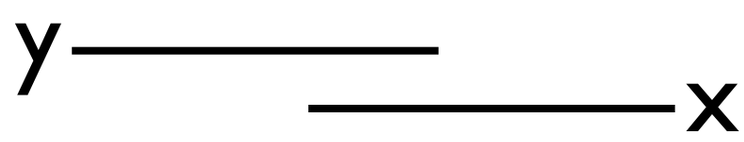
- Initialize first column to 0s
- Answer is maximum score in top row (traceback starts from there until it falls off left side)



Overlap Alignment

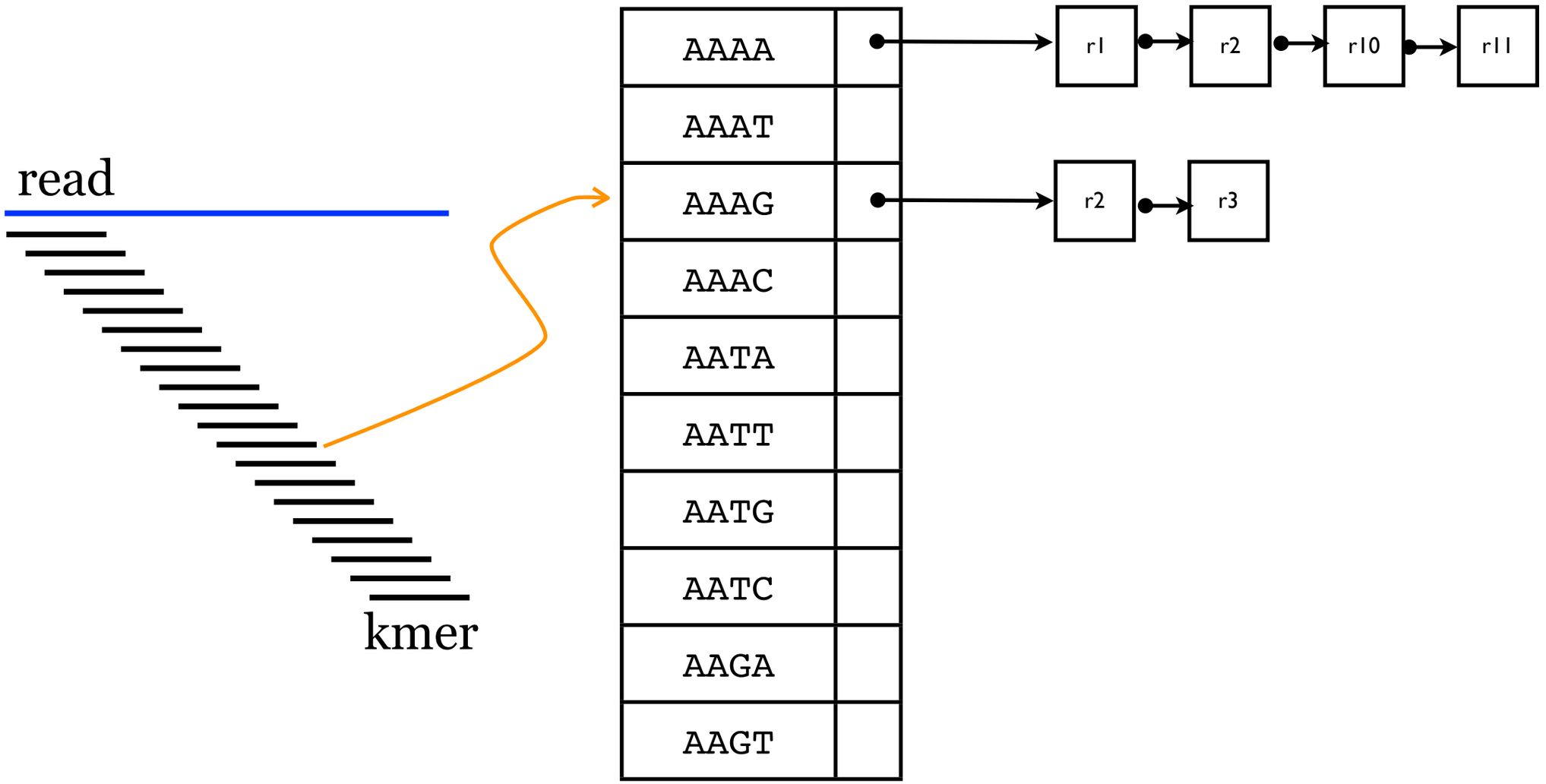


- Initialize first column to 0s
- Answer is maximum score in top row (traceback starts from there until it falls off left side)



K-mer Hashing

Only compute overlap alignment
between reads that share a kmer:



The problem with Shortest Common Superstring (SCS): Repeats

Truth:

AAAAAAAAAAAAAAAAAAAAA

AAAAA
  AAAAA
    AAAAA
      AAAAA
        AAAAA
          ...

SCS:

AAAAA
AAAAA
AAAAA
AAAAA
AAAAA

More complex example:

ACCGCCT ACCGCCT ACCGCCT

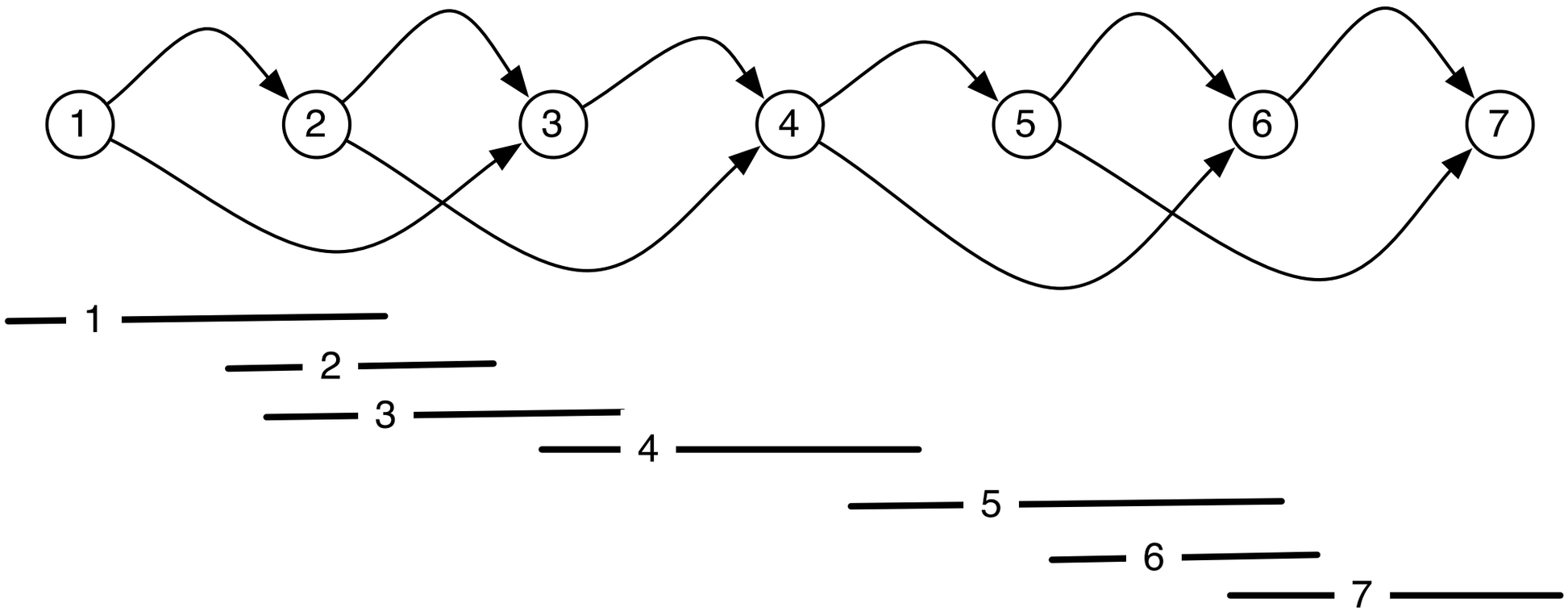
2 or 3 copies?

Overlap Graph

Overlap graph:

Nodes = reads

Edges = overlaps



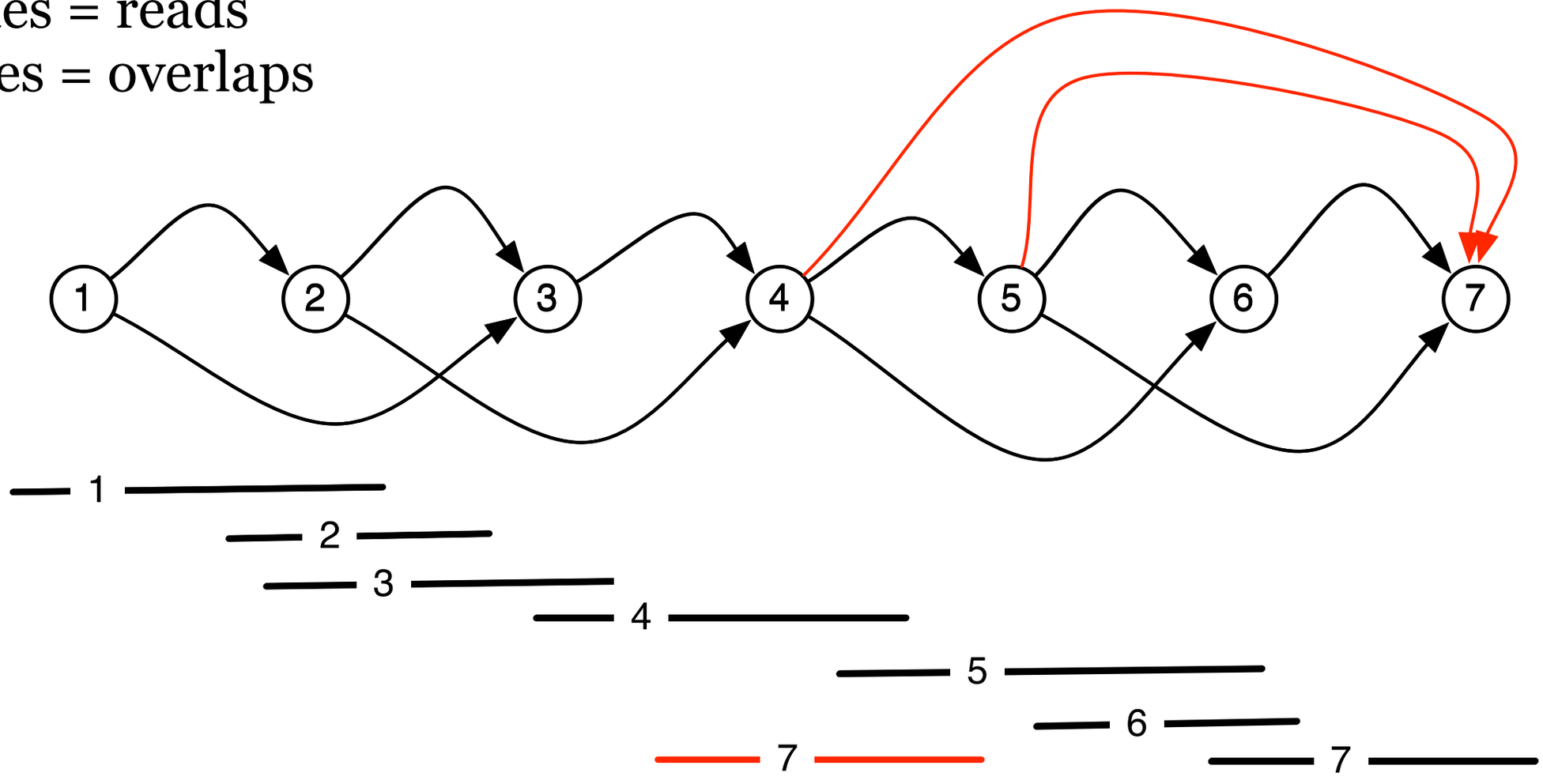
Given overlap graph, how can we find a good candidate assembly?

Overlap Graph

Overlap graph:

Nodes = reads

Edges = overlaps



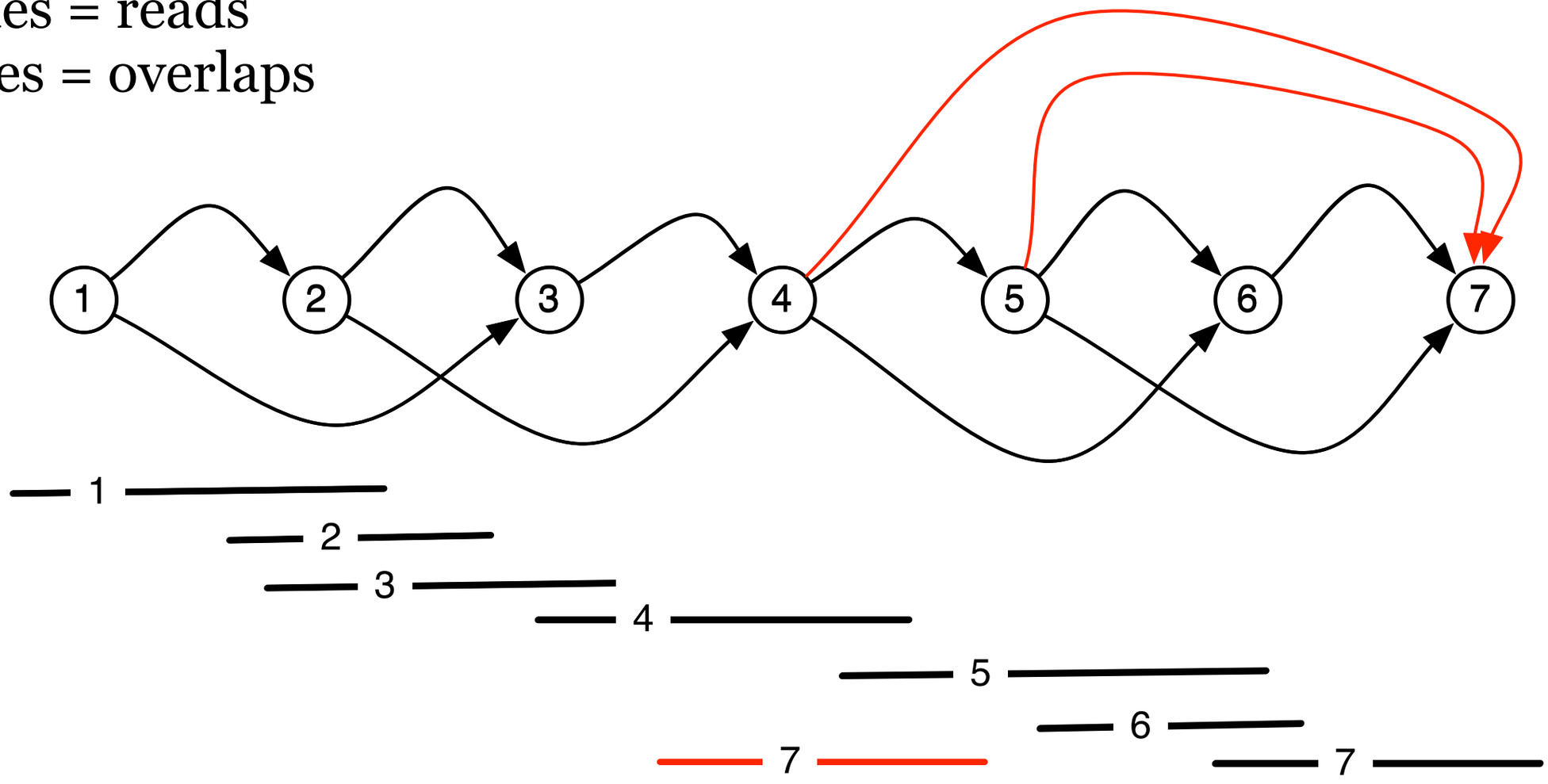
Given overlap graph, how can we find a good candidate assembly?

Overlap Graph

Overlap graph:

Nodes = reads

Edges = overlaps



Given overlap graph, how can we find a good candidate assembly?

Hamiltonian Path (aka Traveling Salesman Path): visit every node in the graph exactly once.

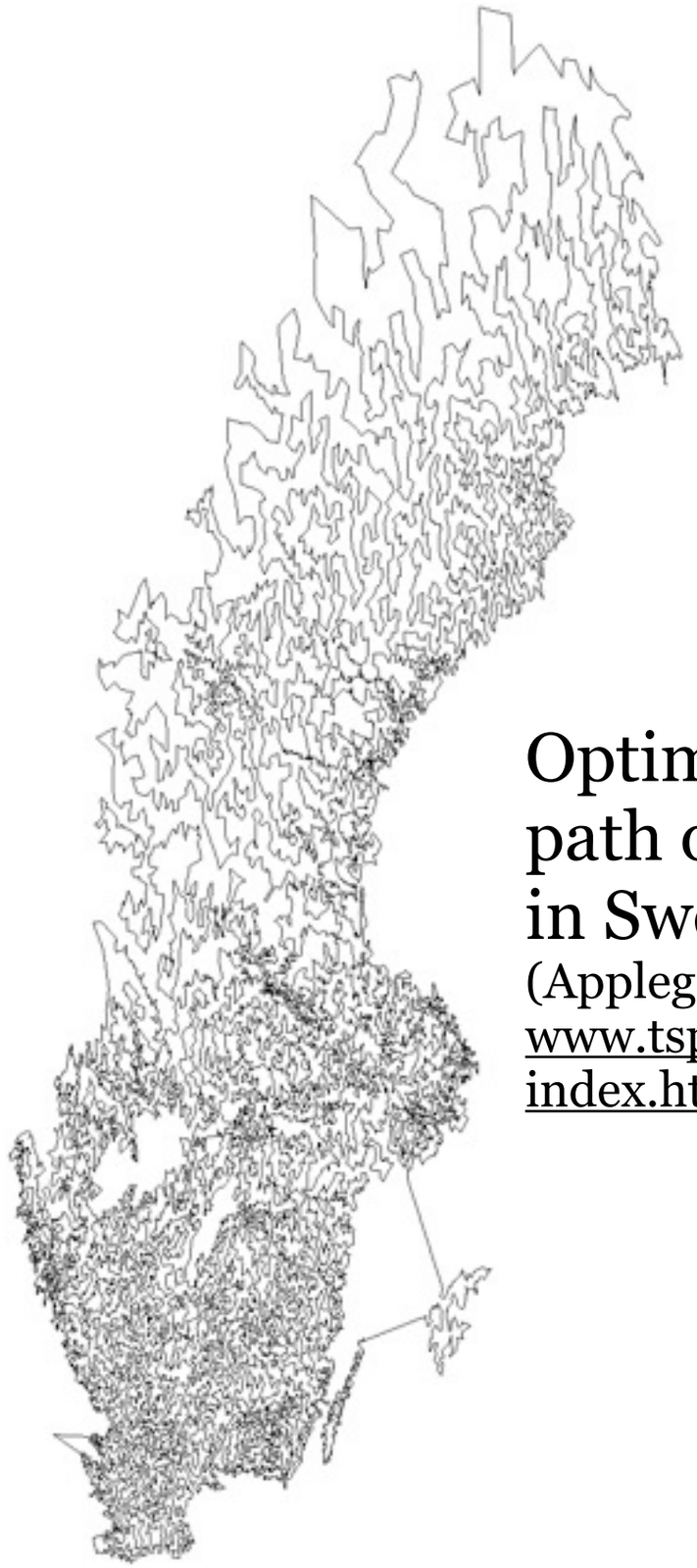
Hamiltonian Path

- Motivation: Every read must be used in exactly one place in the genome.

- Hamiltonian Path is NP-hard.

- Though good solvers exist, they can't operate on the millions of reads from a sequencing project.

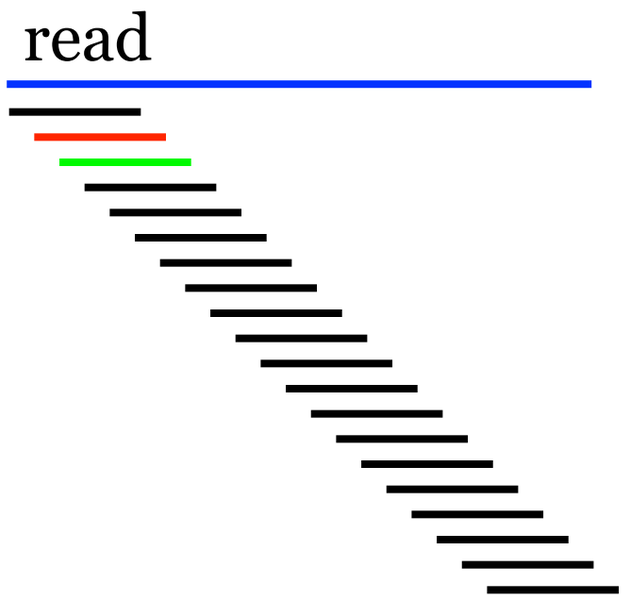
- Solution: greedy walk along the graph.

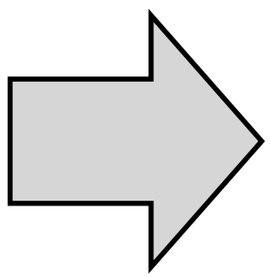


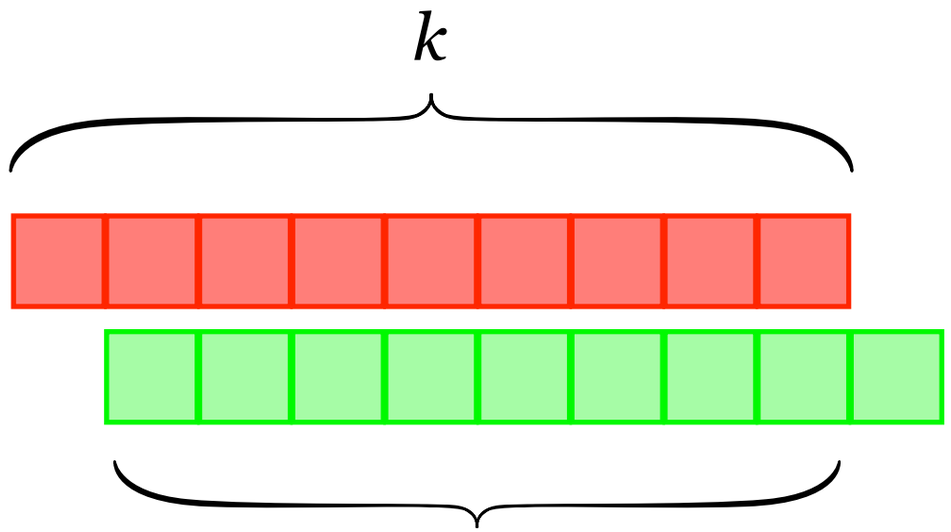
Optimal Hamiltonian path of 24,978 cities in Sweden
(Applegate et al, 2004, www.tsp.gatech.edu/sweden/index.html).

Assembly via Eulerian Path

de Bruijn graph

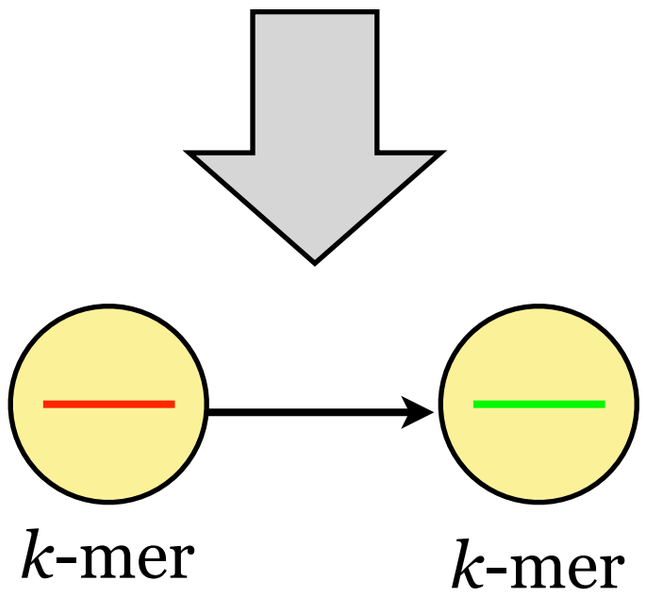




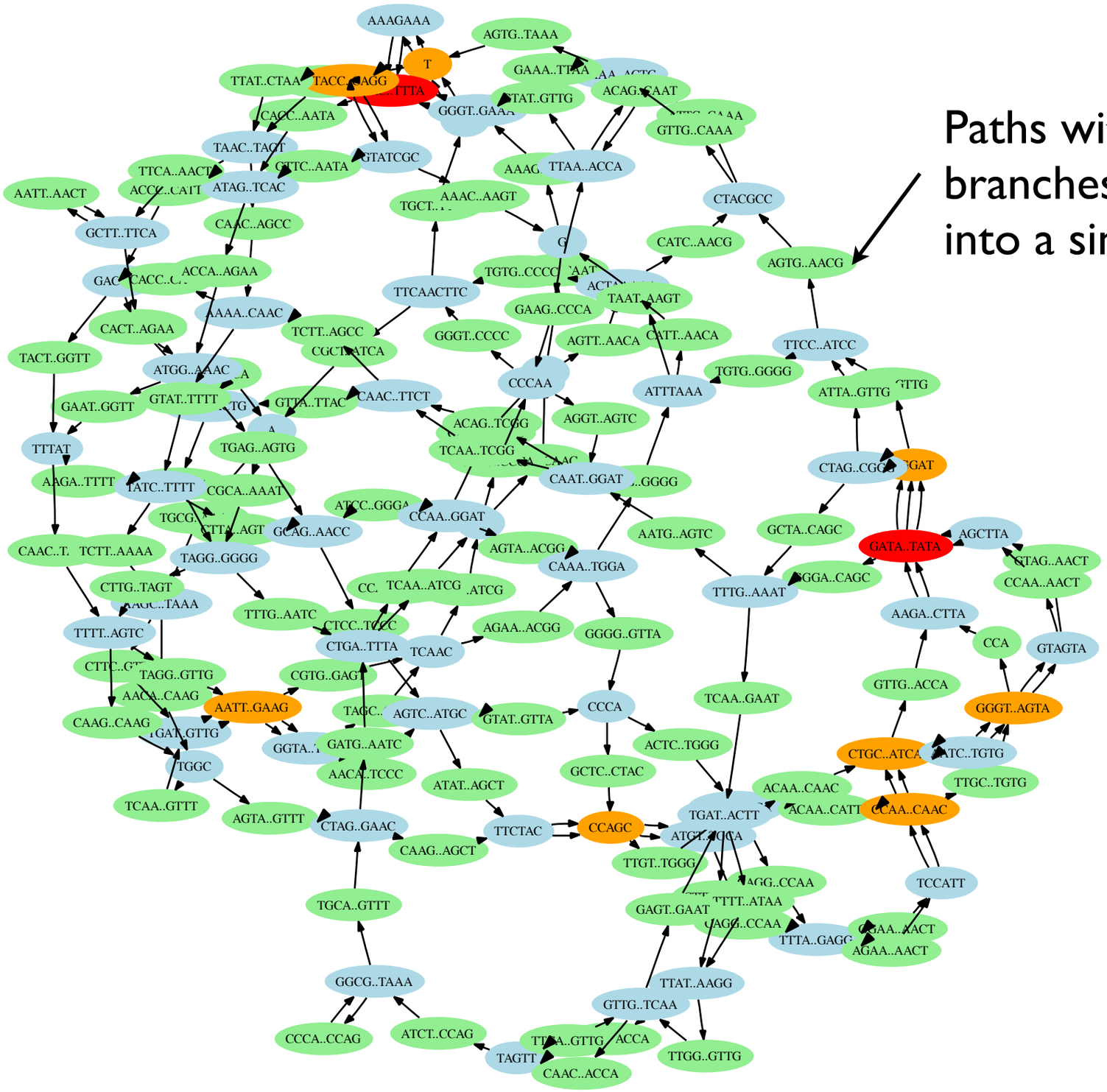


de Bruijn graph: nodes represent kmers, edges connect k-mers that are known to follow each other based on an observed read.

Can have > 1 edge between nodes.



Example bacterial de Bruijn graph

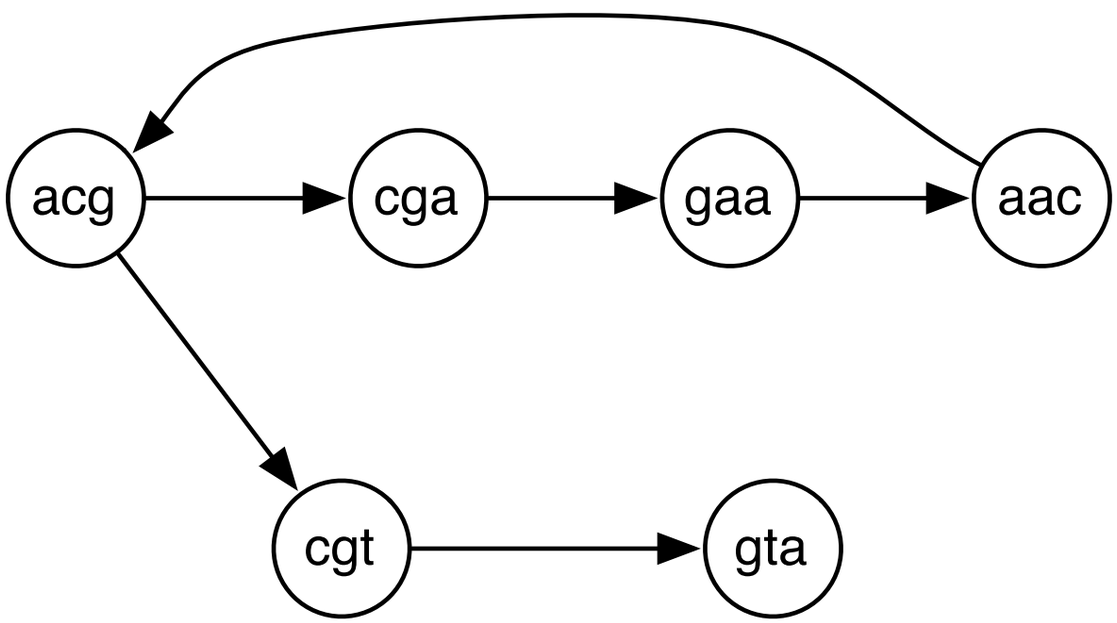


Paths with no branches compressed into a single node

Eulerian path = use every edge exactly once.

With perfect data, the genome can be reconstructed by some Eulerian path through this graph

Assembly via Eulerian Path



acgaacgta

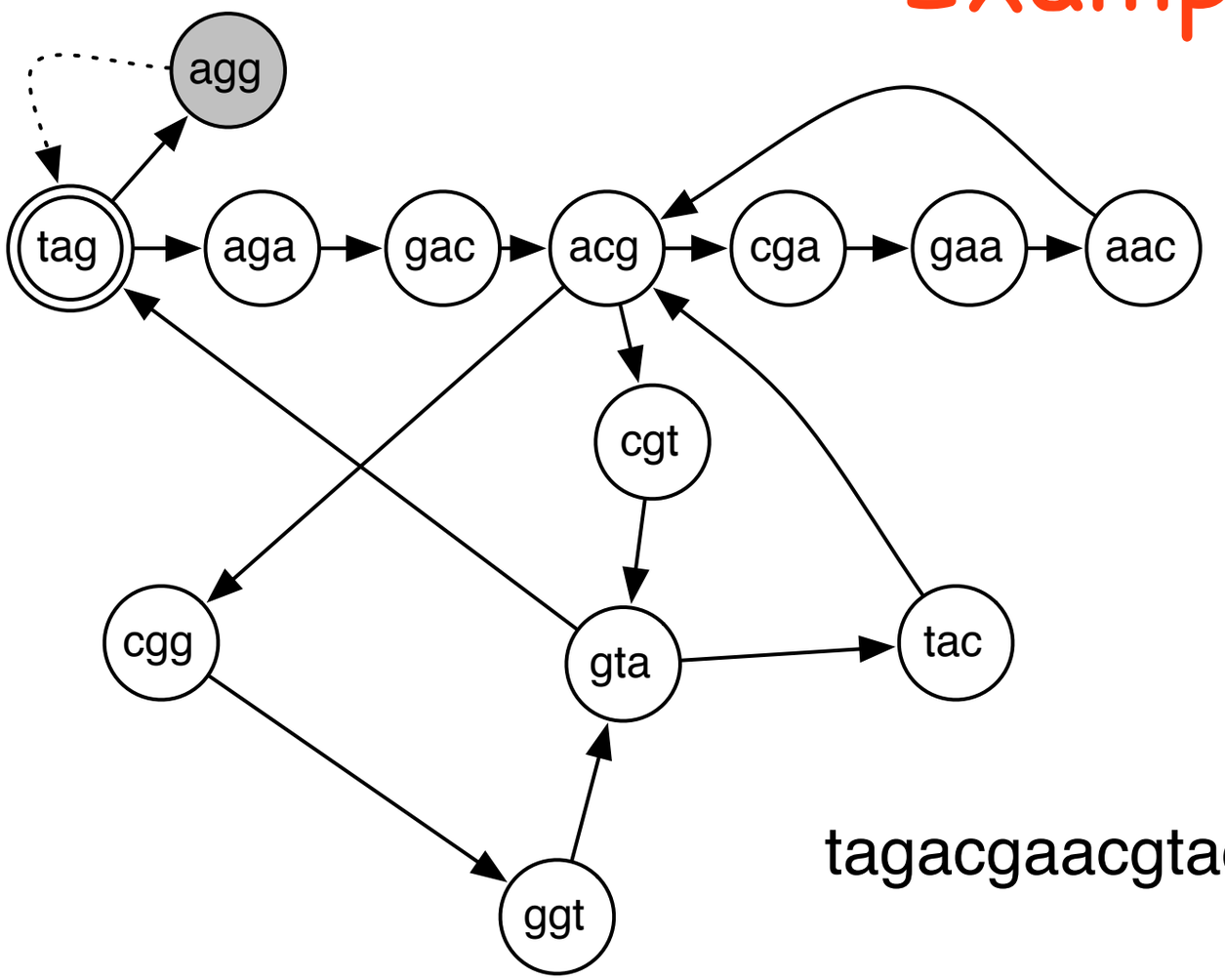
Let dG(s) be the de Bruijn graph of string s. Then s corresponds to some Eulerian path in dG(s).

A directed graph has an Eulerian path if and only if:

- One node has one more edge leaving it than entering
- One node has one more edge entering than leaving
- All other nodes have the same number of edges entering and leaving

How can we find such a path?

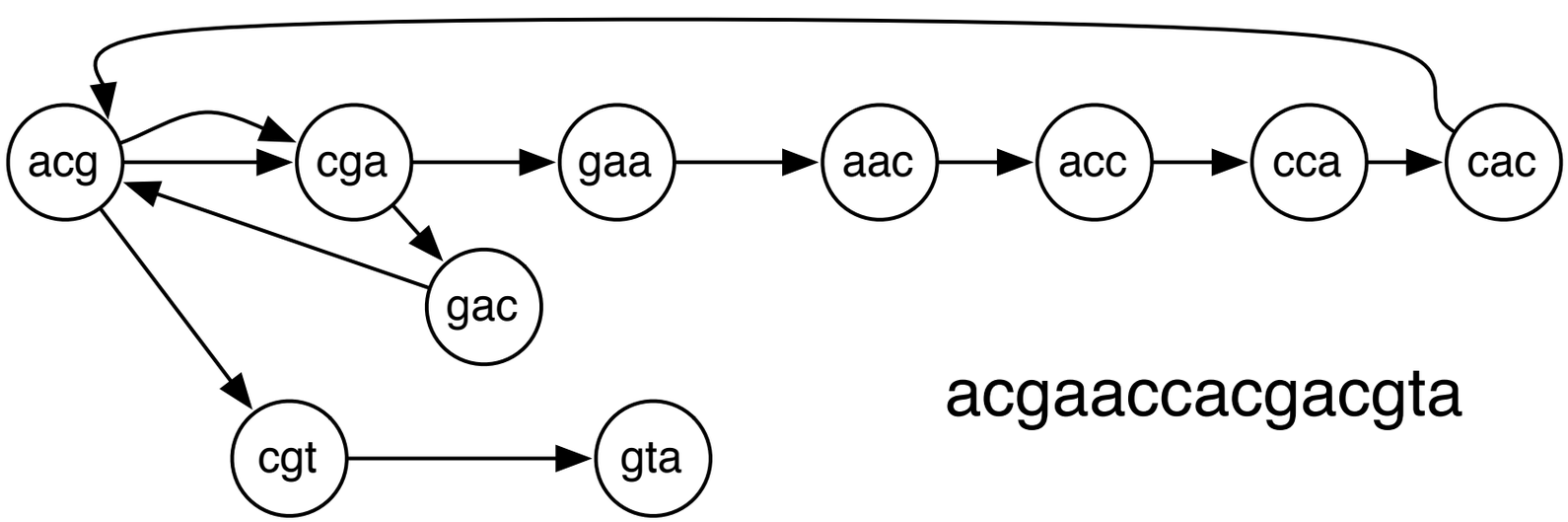
Examples



A directed graph has an Eulerian **cycle** if and only if:

- All nodes have the same number of edges entering and leaving

tagacgaacgtacggtagg



acgaaccacgacgta

Eulerian Path Algorithm

Connect node with out-degree < in-degree to node with out-degree < in-degree. So that we will have an Eulerian cycle.

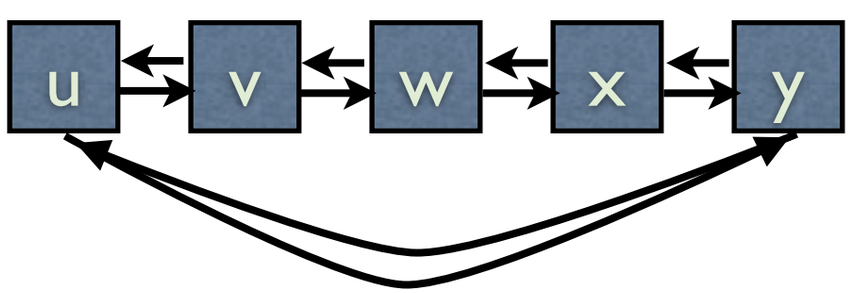
Why will you return to *u*?

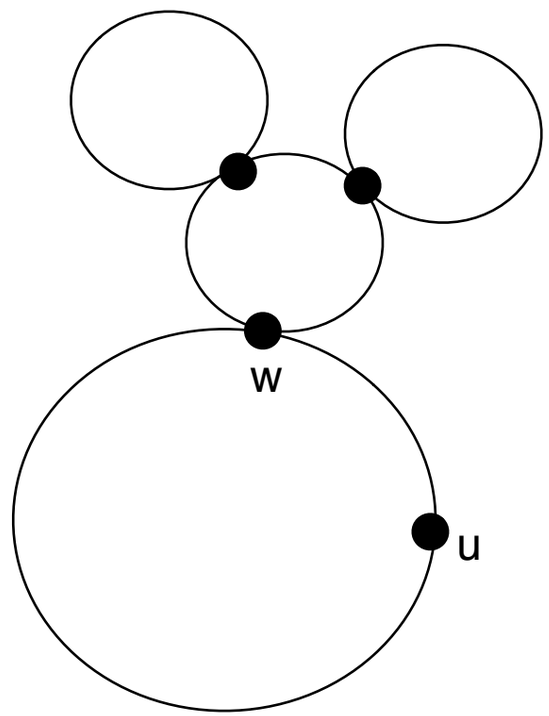
Walk from some arbitrary node *u* until you return to *u*, creating a doubly liked list of the path you visit.

Repeat until all edges used:

*How can find such a node quickly?

- Start from some node *w* on the current tour with unused edges*.
- Walk along unused edges until you return to *w*, inserting the visited nodes after *w* into the current tour list.





Eulerian Path Algorithm

Connect node with out-degree < in-degree to node with out-degree < in-degree. So that we will have an Eulerian cycle.

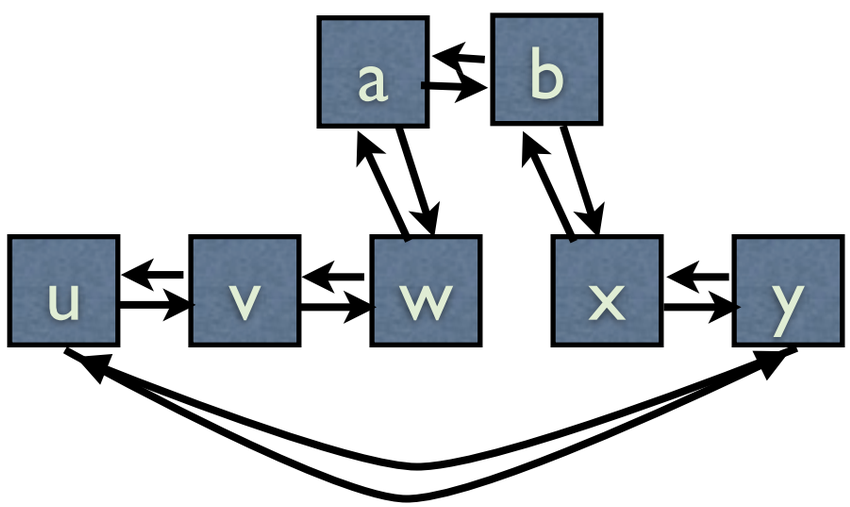
Why will you return to *u*?

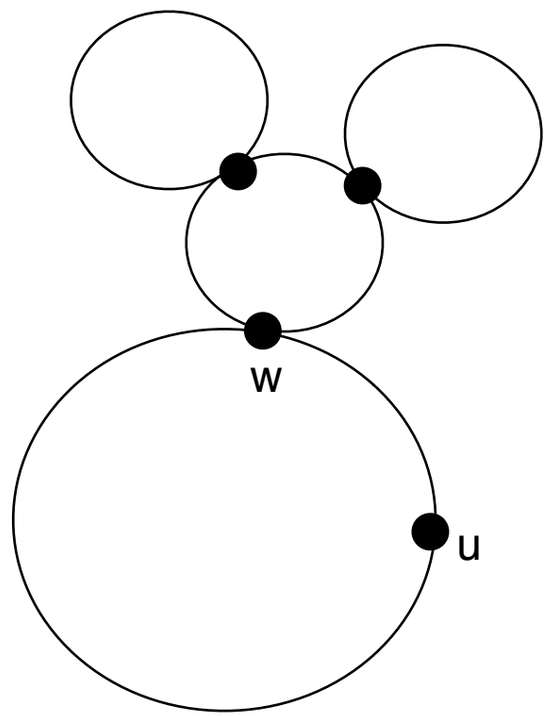
Walk from some arbitrary node *u* until you return to *u*, creating a doubly liked list of the path you visit.

Repeat until all edges used:

*How can find such a node quickly?

- Start from some node *w* on the current tour with unused edges*.
- Walk along unused edges until you return to *w*, inserting the visited nodes after *w* into the current tour list.



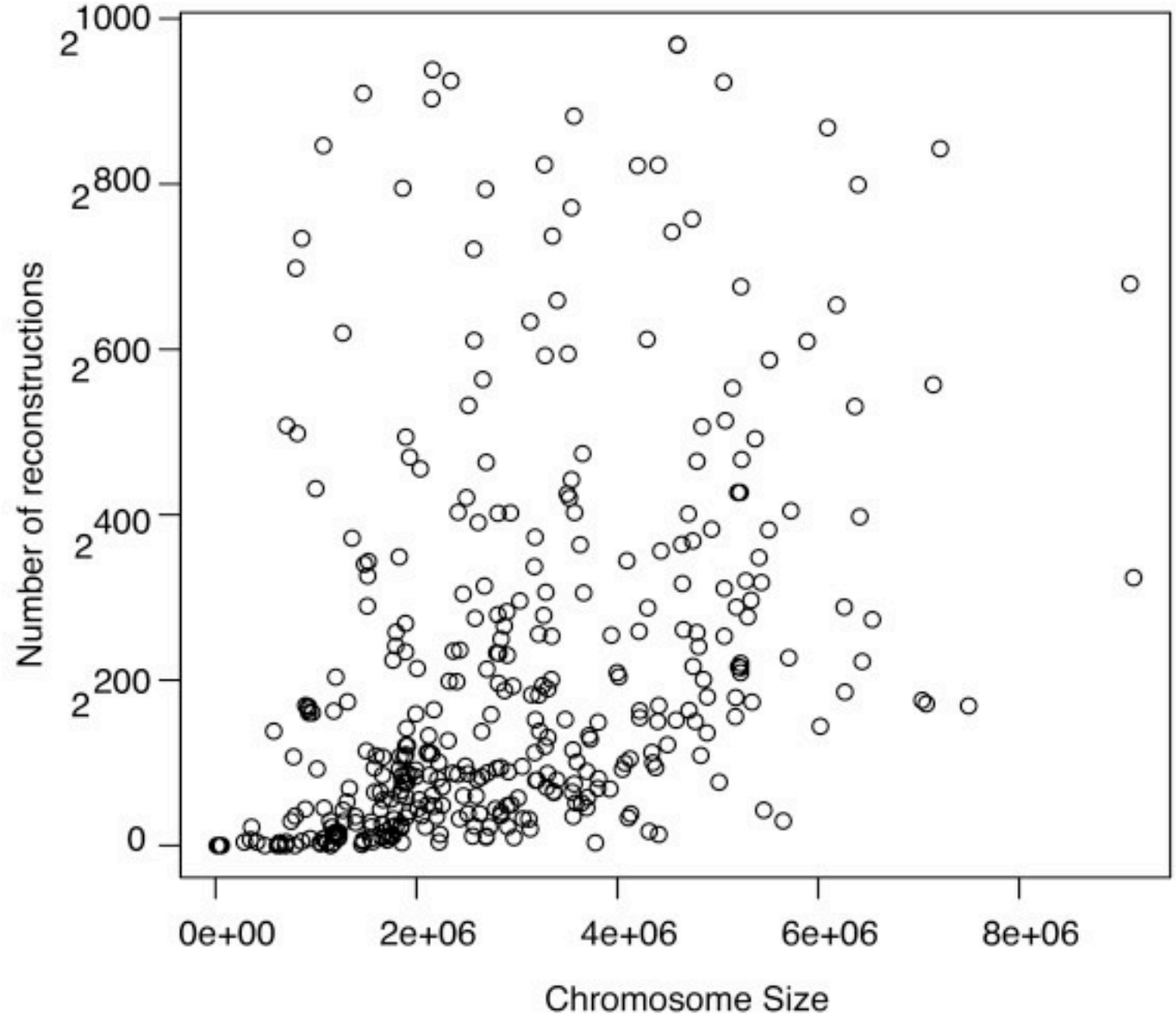


The Problem with Eulerian Paths

There are typically an astronomical number of possible Eulerian tours with perfect data.

Adding back constraints to limit # of tours leads to a NP-hard problem.

With imperfect data, there are usually NO Eulerian tours.

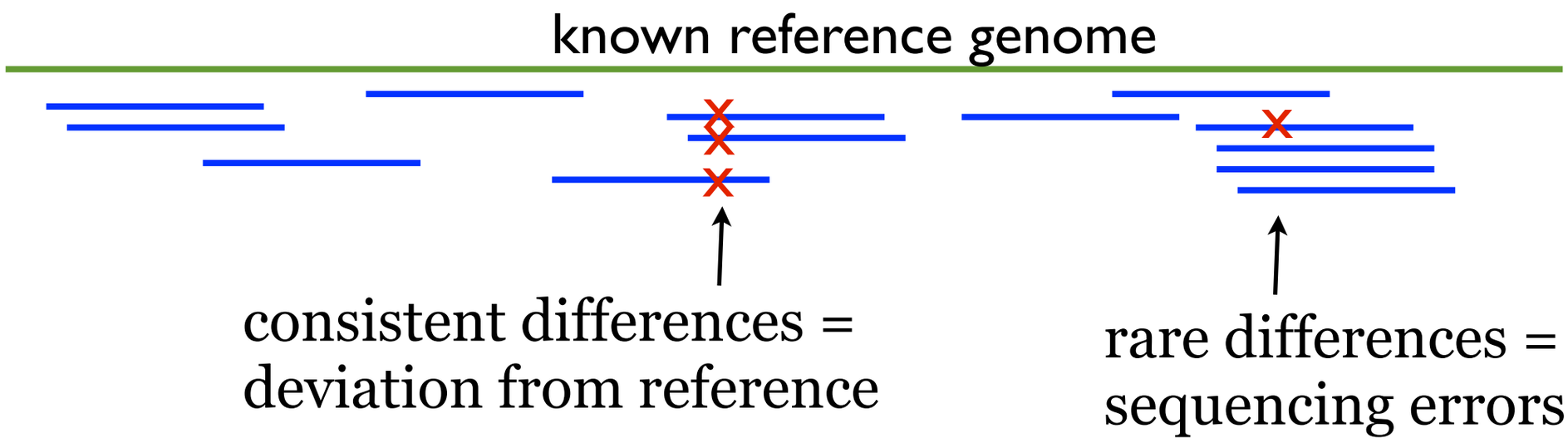


(Kingsford, Schatz, Pop, 2010)

Aside: counting # of Eulerian tours in a directed graph is easy, but in an undirected graph is #P-complete (hard).

Comparative Assembly

Align reads to known genome:



Can use much lower coverage
(e.g. 4X coverage instead of 20-30X for *de novo* assembly).

Aligning a large # of short sequences to one large sequence is an important special case of sequence alignment.

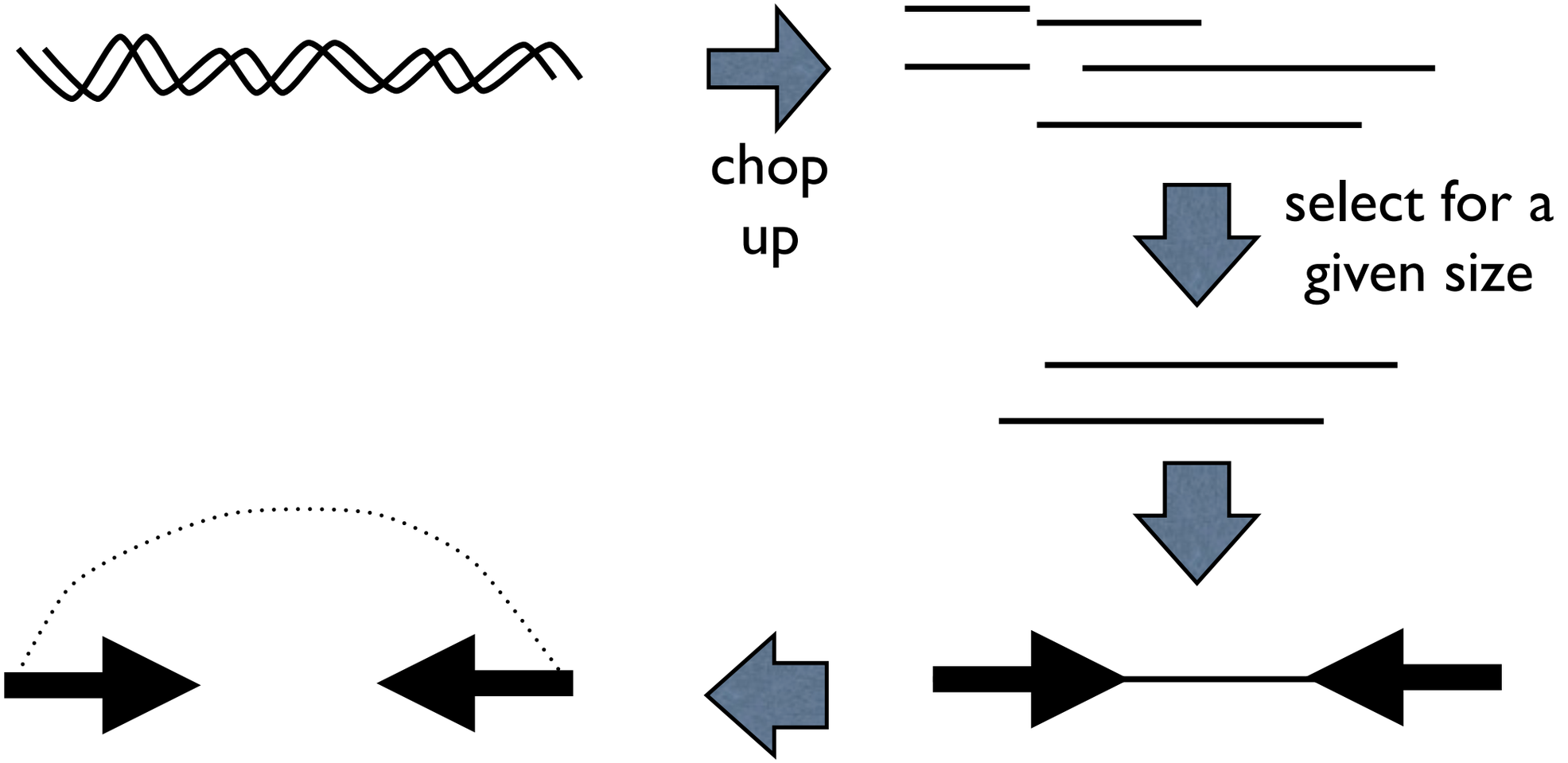
"1000" Genomes Project

find variants that occur in > 1% of the population: sequence ≈2500 genomes at 4X coverage, align them to reference.

1000 Genomes Samples						6-Sept-11		
Population	Status	Available to research community (dates approx)	DNA sequenced from blood	Offspring samples from trios	First set	Second set	Third set	Total
Utah residents (CEPH) with Northern and Western European ancestry (CEU)	Available	Available	no	yes	100			100
Toscani in Italia (TSI)	Available	Available	no	no	100			100
British from England and Scotland (GBR)	Available	Available	no	no	96	4		100
Finnish from Finland (FIN)	Available	Available	no	no	100			100
Iberian populations in Spain (IBS)	Available to project	Available	no	yes	30	70		100
Total European ancestry					**426**	**74**		**500**
Han Chinese in Beijing, China (CHB)	Available	Available	no	no	100			100
Japanese in Toyko, Japan (JPT)	Available	Available	no	no	100			100
Han Chinese South (CHS)	Available	Available	most	yes	100			100
Chinese Dai in Xishuangbanna (CDX)	Available to project	Oct-Dec 2011	some	no		100		100
Kinh in Ho Chi Minh City, Vietnam (KHV)	Available to project	Oct-Dec 2011	yes	some		100		100
Chinese in Denver, Colorado (CHD) (pilot 3 only)	Available	Available	no	no				0
TOTAL East Asian ancestry					**300**	**200**		**500**
Yoruba in Ibadan, Nigeria (YRI)	Available	Available	no	yes	100			100
Luhya in Webuye, Kenya (LWK)	Available	Available	no	no	100			100
Gambian in Western Division, The Gambia	Collecting samples	Mar-May 2012	no	yes			100	100

http://www.1000genomes.org/about#ProjectSamples

Mate Pairs



mate pair: 2 reads, of opposite orientation, separated by an approximately known distance

sequence ≈ 1000 bases from each end

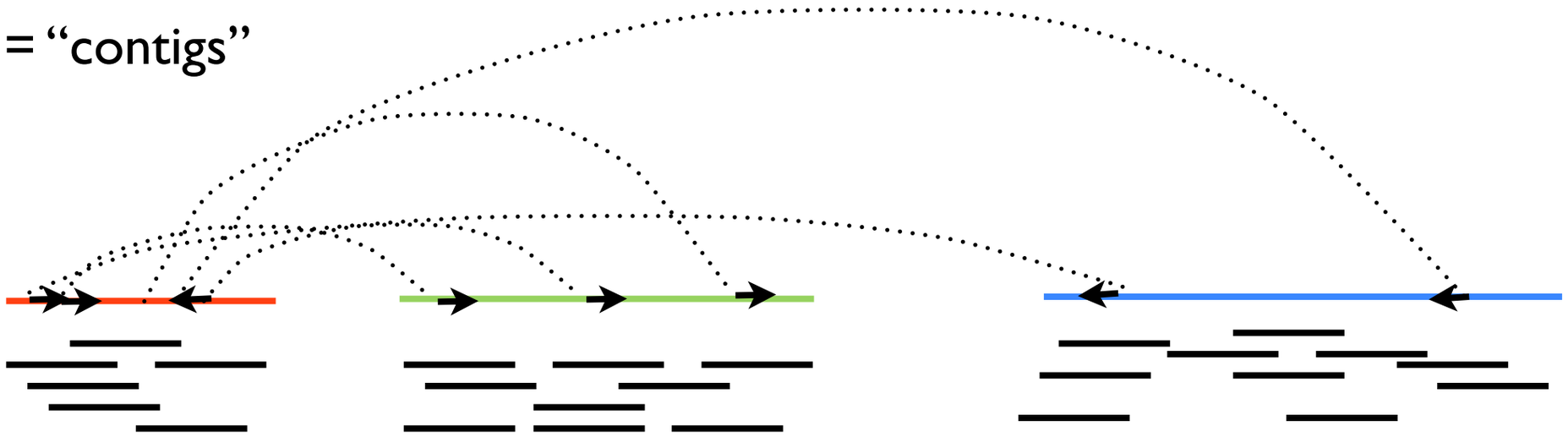
⇒ long range information

Scaffolding

Islands = “contigs”

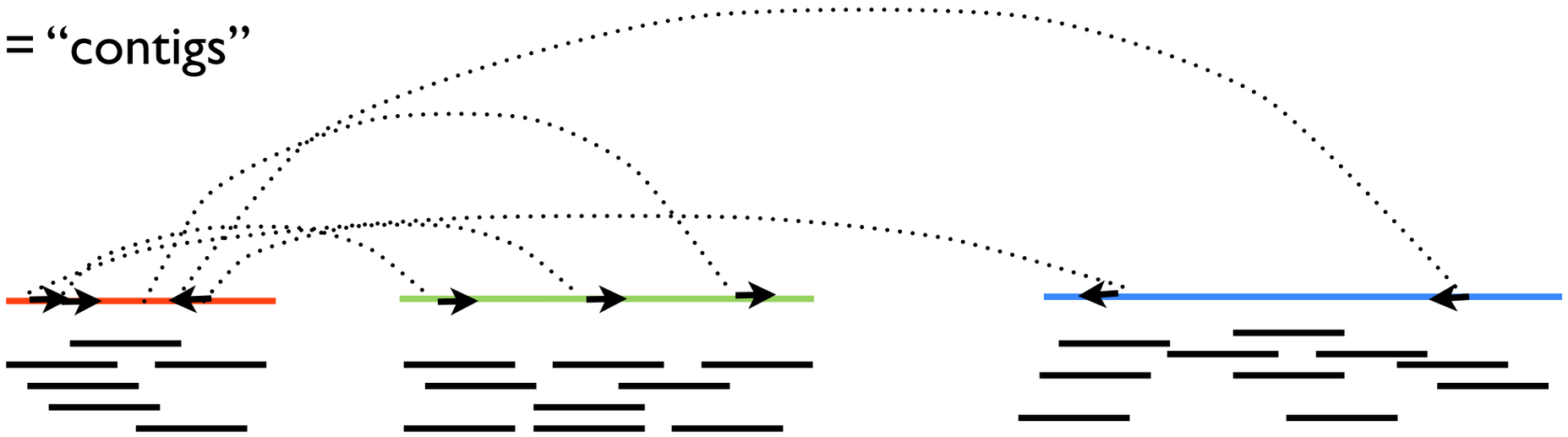
Scaffolding

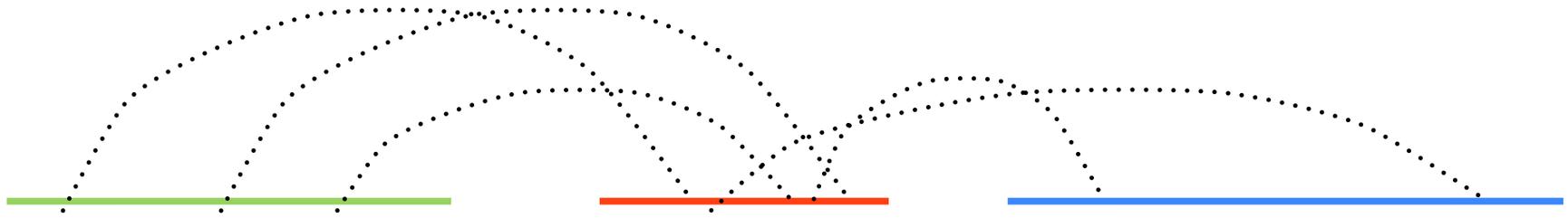
Islands = “contigs”



Scaffolding

Islands = “contigs”





Summary

- Sanger sequencing reads DNA via synthesis; 800-1000bp.
- Assembly Paradigms:
  - Shortest Common Superstring (NP-hard; sensitive to repeats)
  - Hamiltonian cycle in overlap graph (NP-hard)
  - Eulerian cycle in de Bruijn graph (polynomial in basic form, but large # of solutions)
- Overlap alignment can be computed with slight variant of sequence alignment DP.
  - K-mer hashing technique avoids all pairs overlap alignment

Hard vs. Easy

- Eulerian path – visit every edge once (easy)
- Hamiltonian path – visit every node once (hard)

- Shortest common supersequence (easy)
- Shortest common superstring (hard)

- Counting Eulerian tours in directed graphs (easy)
- Counting Eulerian tours in undirected graphs (hard)

- Aligning 2 sequences (easy)
- Aligning $k > 2$ sequences (hard)

- Shortest path (easy)
- Longest path (hard)