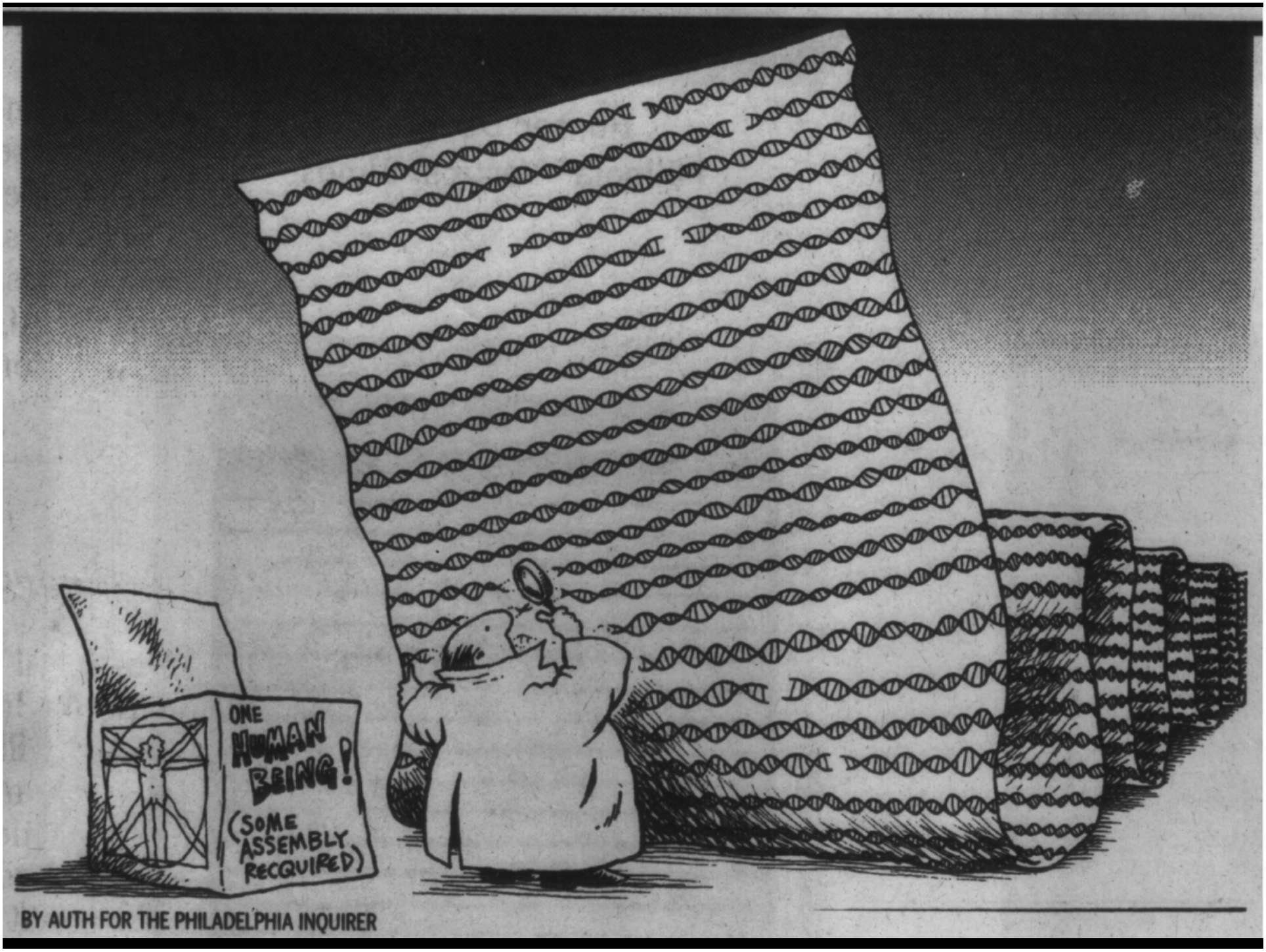
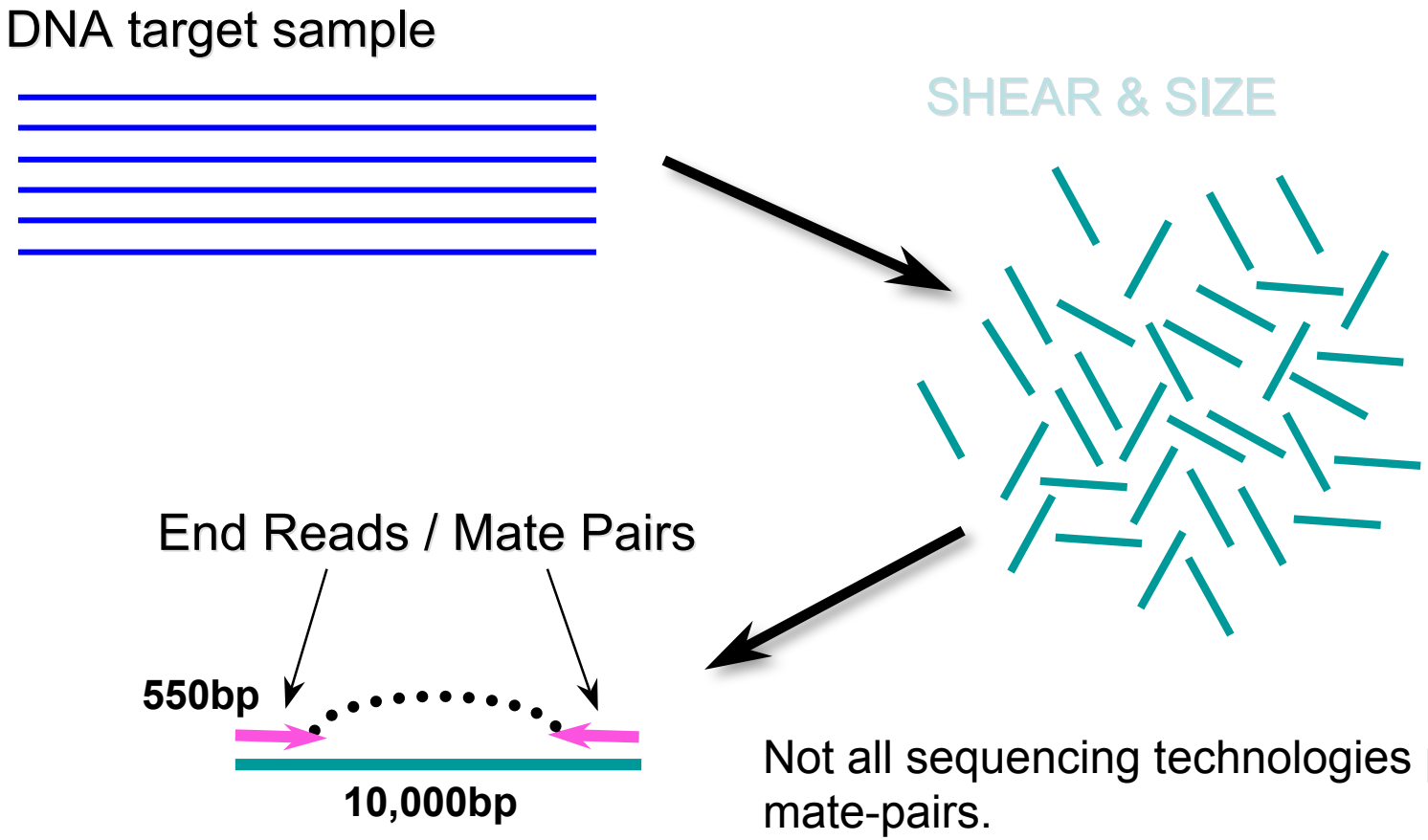
CS 394C
March 19, 2012

Tandy Warnow


ONE
HUMAN
BEING!
(SOME
ASSEMBLY
RECQUIRED)
BY AUTH FOR THE PHILADELPHIA INQUIRER

Shotgun DNA Sequencing



Not all sequencing technologies produce mate-pairs.
Different error models
Different read lengths

Basic Challenge

- Given many (millions or billions) of reads, produce a linear (or perhaps circular) genome
- Issues:
  - Coverage
  - Errors in reads
  - Reads vary from very short (35bp) to quite long (800bp), and are double-stranded
  - Non-uniqueness of solution
  - Running time and memory

Simplest scenario

- Reads have no error
- Read are long enough that each appears exactly once in the genome
- Each read given in the same orientation (all 5’ to 3’, for example)

De novo vs. comparative assembly

- *De novo* assembly means you do everything from scratch

- Comparative assembly means you have a "reference" genome. For example, you want to sequence your own genome, and you have Craig Venter's genome already sequenced. Or you want to sequence a chimp genome and you have a human already sequenced.

Comparative Assembly

Much easier than *de novo*!

Basic idea:

- Take the reads and map them onto the reference genome (allowing for some small mismatch)
- Collect all overlapping reads, produce a multiple sequence alignment, and produce consensus sequence

Comparative Assembly

Fast

Short reads can map to several places (especially if they have errors)

Needs close reference genome

Repeats are problematic

Can be highly accurate even when reads have errors

De Novo assembly

- Much easier to do with long reads
- Need very good coverage
- Generally produces fragmented assemblies
- Necessary when you don’t have a closely related (and correctly assembled) reference genome

De Novo Assembly paradigms

- Overlap Graph:

  overlap-layout-consensus methods
  - greedy (TIGR Assembler, phrap, CAP3...)
  - graph-based (Celera Assembler, Arachne)

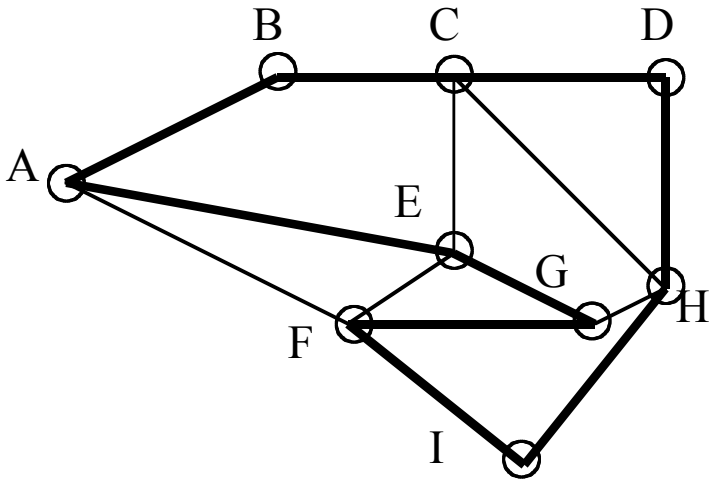
- k-mer graph (especially useful for assembly from short reads)

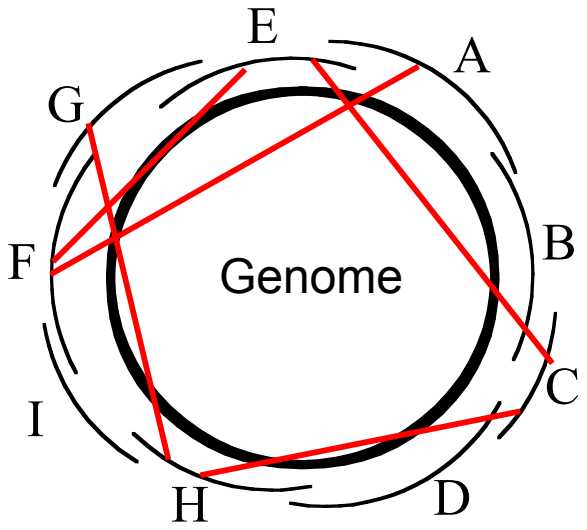
Overlap Graph

- Each read is a node
- There is a directed edge from u to v if the two reads have sufficient overlap
- Objective: Find a Hamiltonian Path (for linear genomes) or a Hamiltonian Circuit (for circular genomes)

Paths through graphs and assembly

- Hamiltonian circuit: visit each node (city) exactly once, returning to the start





Hamiltonian Path Approach

- Hamiltonian Path is NP-hard (but good heuristics exist), and can have multiple solutions
- Dependency on detecting overlap (errors in reads, overlap length)
- Running time (all-pairs overlap calculation)
- Repeats
- Tends to produce fragmented assemblies (contigs)

Example

Reads:

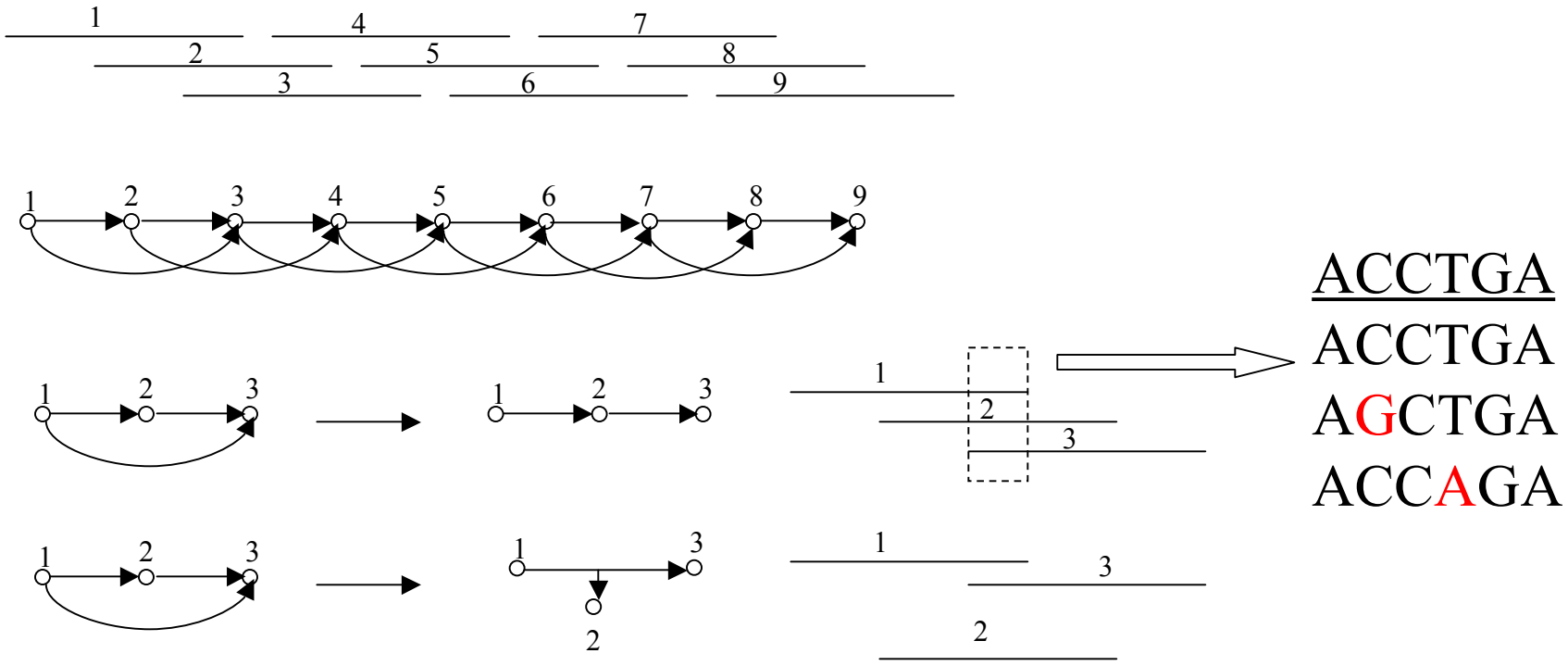
- TAATACTTAGG
- TAGGCCA
- GCCAGGAAT
- GAATAAGCCAA
- GCCAATTT
- AATTTGGAAT
- GGAATTAAGGCAC
- AGGCACGTTTA
- CACGTTAGGACCATT
- GGACCATTTAATACGGAT

If minimum overlap is 3, what graph do we get?
If minimum overlap is 4, what graph do we get?
If minimum overlap is 5, what graph do we get?

Overlap-layout-consensus

Main entity: read

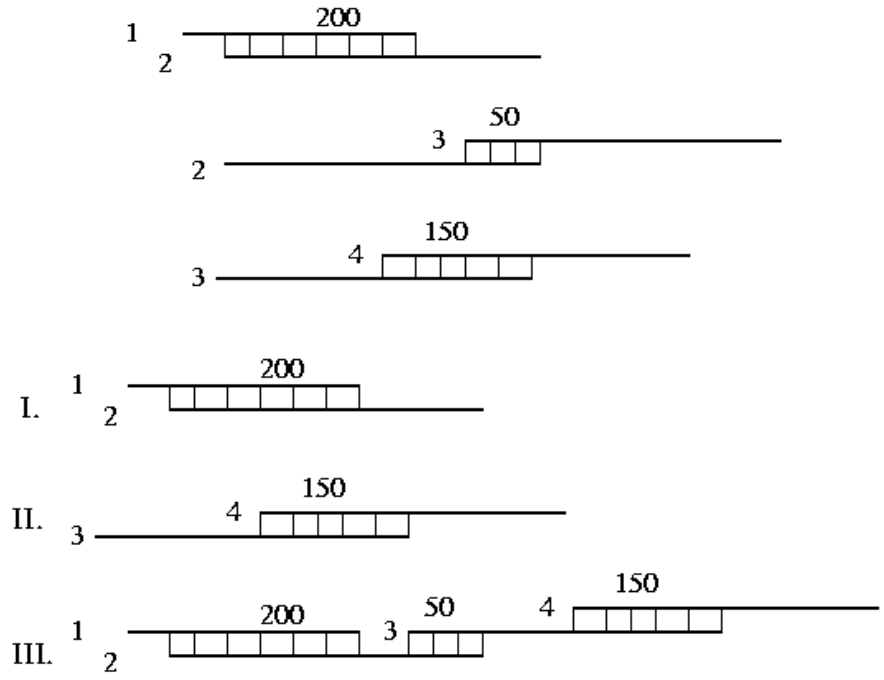
Relationship between reads: overlap



TIGR Assembler/phrap

Greedy

- Build a rough map of fragment overlaps
- Pick the largest scoring overlap
- Merge the two fragments
- Repeat until no more merges can be done



Challenges for Overlap-Layout-Consensus

- Computing all-pairs overlap is computationally expensive, especially for NGS datasets, which can have millions of short reads.
- (The Hamiltonian Path part doesn't help either)
- This computational challenge is increased for large genomes (e.g., human)
- Need something faster!

k-mer graph (aka de Bruijn graph)

- Vertices are k-mers that appear in some read, and edges defined by overlap of k-1 nucleotides
- Small values of k produce small graphs
- ***Does not require all-pairs overlap calculation!***
- But: loss of information about reads can lead to "chimeric" contigs, and incorrect assemblies
- Also produces fragmented assemblies (even shorter contigs)

Eulerian Paths

- An Eulerian path is one that goes through every edge exactly once
- It is easy to see that if a graph has an Eulerian path, then all but 2 nodes have even degree. The converse is also true, but a bit harder to prove.
- For directed graphs, the cycle will need to follow the direction of the edges (also called “arcs”). In this case, a graph has an Eulerian path if and only if the indegree(v)=outdegree(v) for all but 2 nodes (x and y), where indegree(x)=outdegree(x)+1, and indegree(y)=outdegree(y)-1.

de Bruijn Graphs are Eulerian

- If the k-mer set comes from a sequence and every k-mer appears exactly once in the sequence,
- then the de Bruijn graph has an Eulerian path!

de Bruijn Graph

- Create the de Bruijn graph for the following string, using k=5
  - ACATAGGATTCAC

- Find the Eulerian path
- Is the Eulerian path unique?
- Reconstruct the sequence from this path

Using de Bruijn Graphs

Given: set of k-mers from a DNA sequence

Algorithm:

- Construct the de Bruijn graph
- Find an Eulerian path in the graph
- The path defines a sequence with the same set of k-mers as the original

Using de Bruijn Graphs

Given: set of k-mers from a set of reads for a sequence

Algorithm:

- Construct the de Bruijn graph
- Try to find an Eulerian path in the graph

No matter what

- Because of
  - Errors in reads
  - Repeats
  - Insufficient coverage

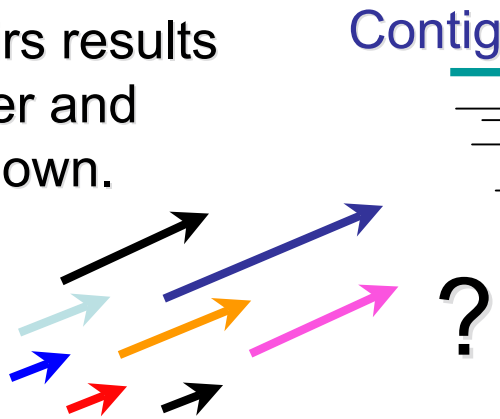
  the overlap graphs and de Bruijn graphs generally don't have Hamiltonian paths/circuits or Eulerian paths/circuits
- This means the first step doesn't completely assemble the genome

Reads, Contigs, and Scaffolds

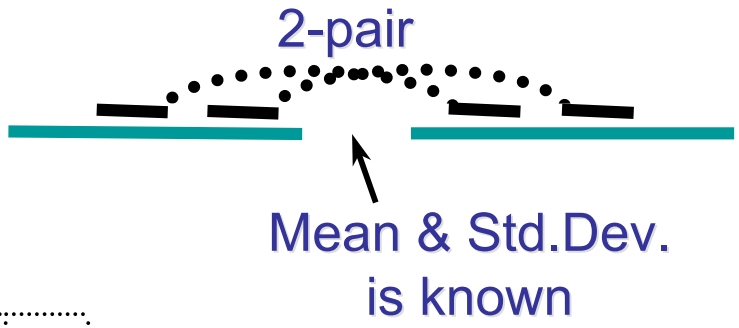
- Reads are what you start with (35bp-800bp)
- Fragmented assemblies produce contigs that can be kilobases in length
- Putting contigs together into scaffolds is the next step

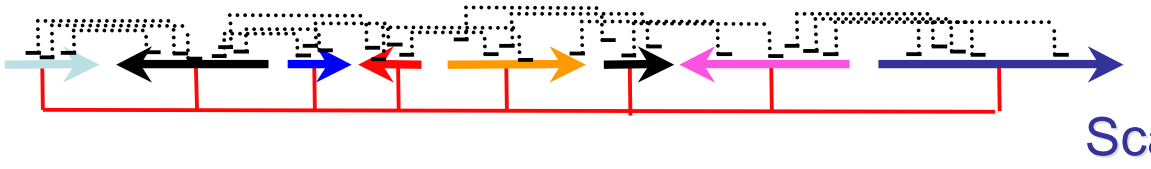
Mate Pairs Give Order & Orientation

Assembly without pairs results in contigs whose order and orientation are not known.



Pairs, especially groups of corroborating ones, link the contigs into scaffolds where the size of gaps is well characterized.

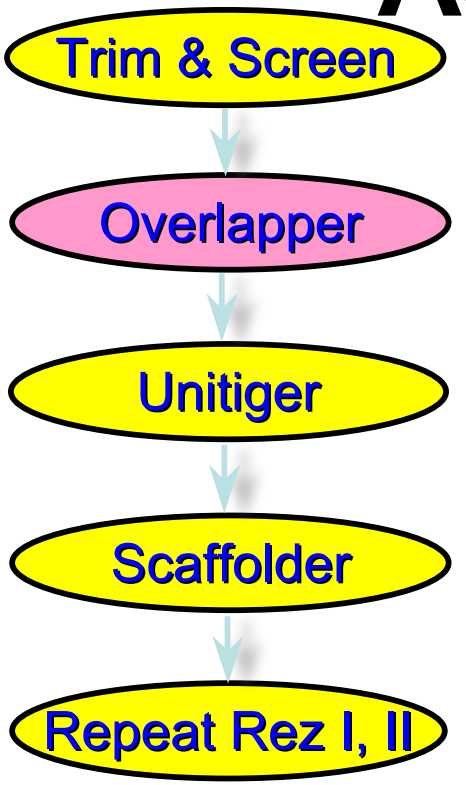




Handling repeats

1. Repeat detection
   - **pre-assembly:** find fragments that belong to repeats
     - statistically (most existing assemblers)
     - repeat database (*RepeatMasker*)
   - **during assembly:** detect "tangles" indicative of repeats (Pevzner, Tang, Waterman 2001)
   - **post-assembly:** find repetitive regions and potential mis-assemblies.
     - *Reputer, RepeatMasker*
     - "unhappy" mate-pairs (too close, too far, mis-oriented)
2. Repeat resolution
   - find DNA fragments belonging to the repeat
   - determine correct tiling across the repeat

Assembly Pipeline



Find all overlaps ≥ 40bp allowing 6% mismatch.

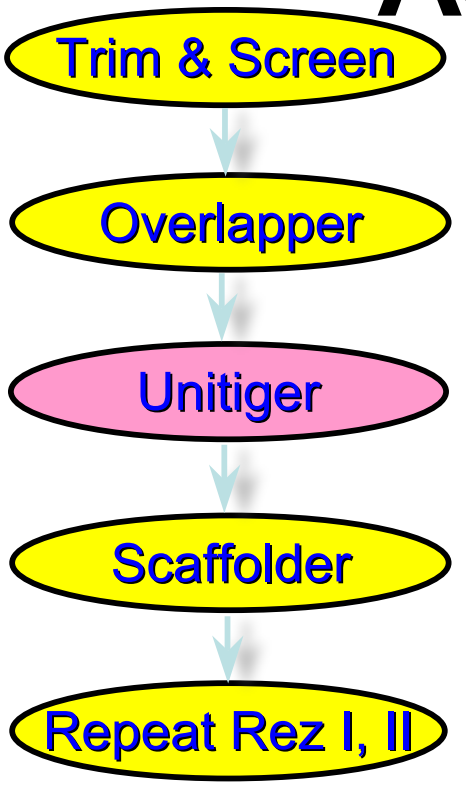
implies

TRUE

OR

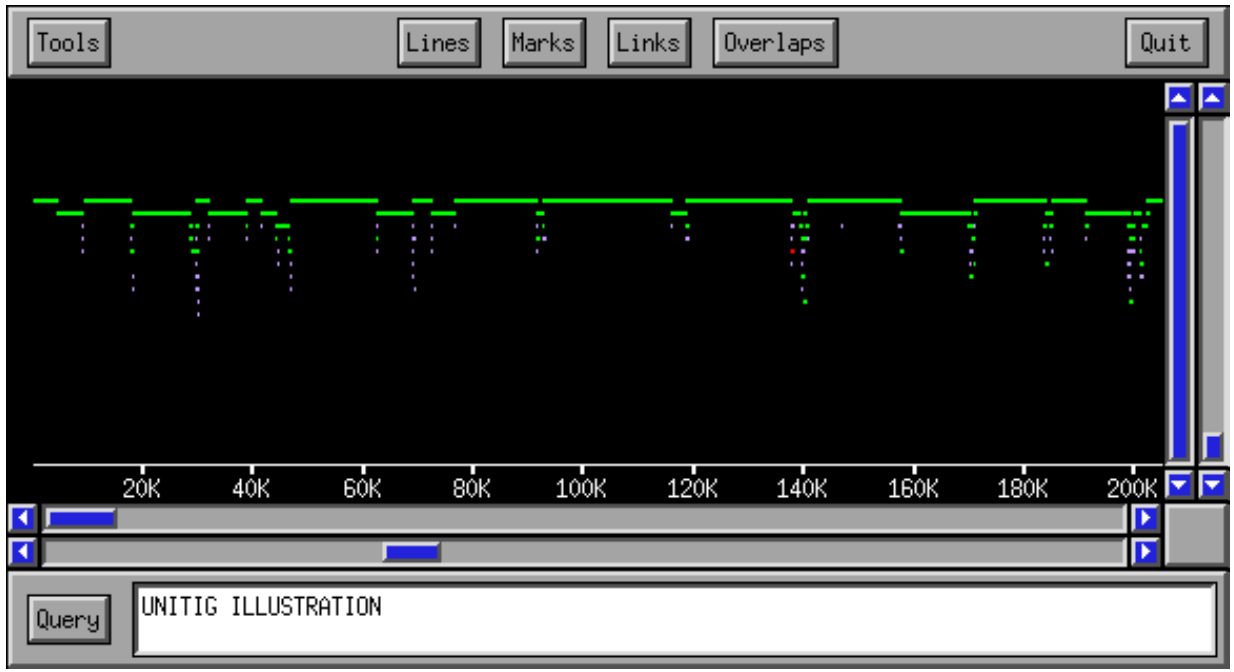
REPEAT-INDUCED

Assembly Pipeline



Compute all overlap consistent sub-assemblies:

Unitigs (Uniquely Assembled Contig)



Assembly Pipeline

Trim & Screen

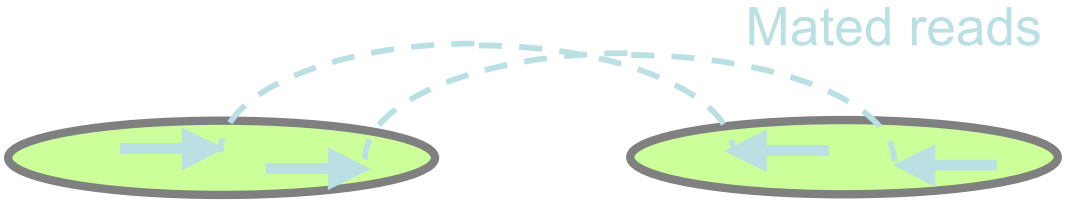
Overlapper

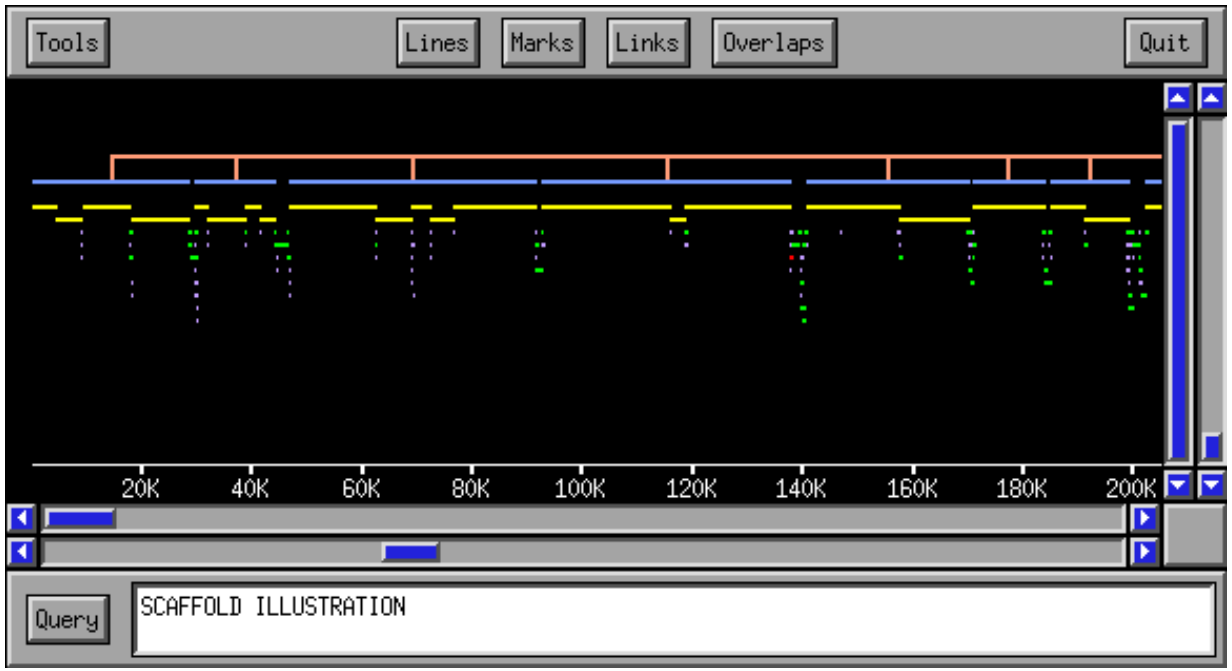
Unitiger

Scaffolder

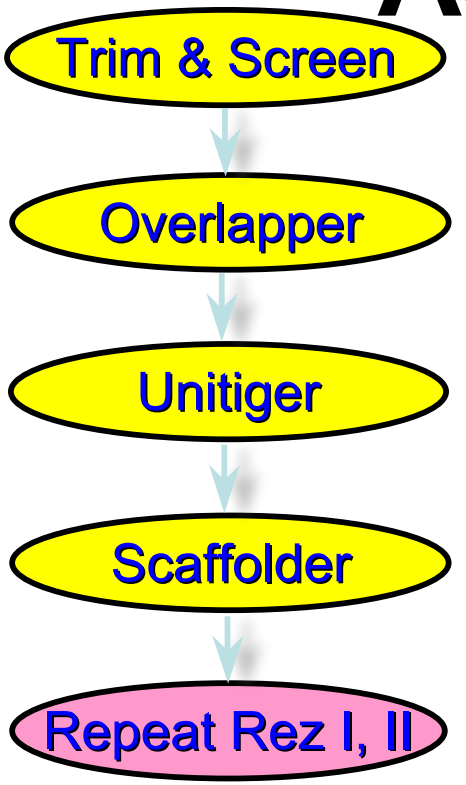
Repeat Rez I, II

Scaffold U-unitigs with confirmed pairs





Assembly Pipeline



Fill repeat gaps with doubly anchored positive unitigs