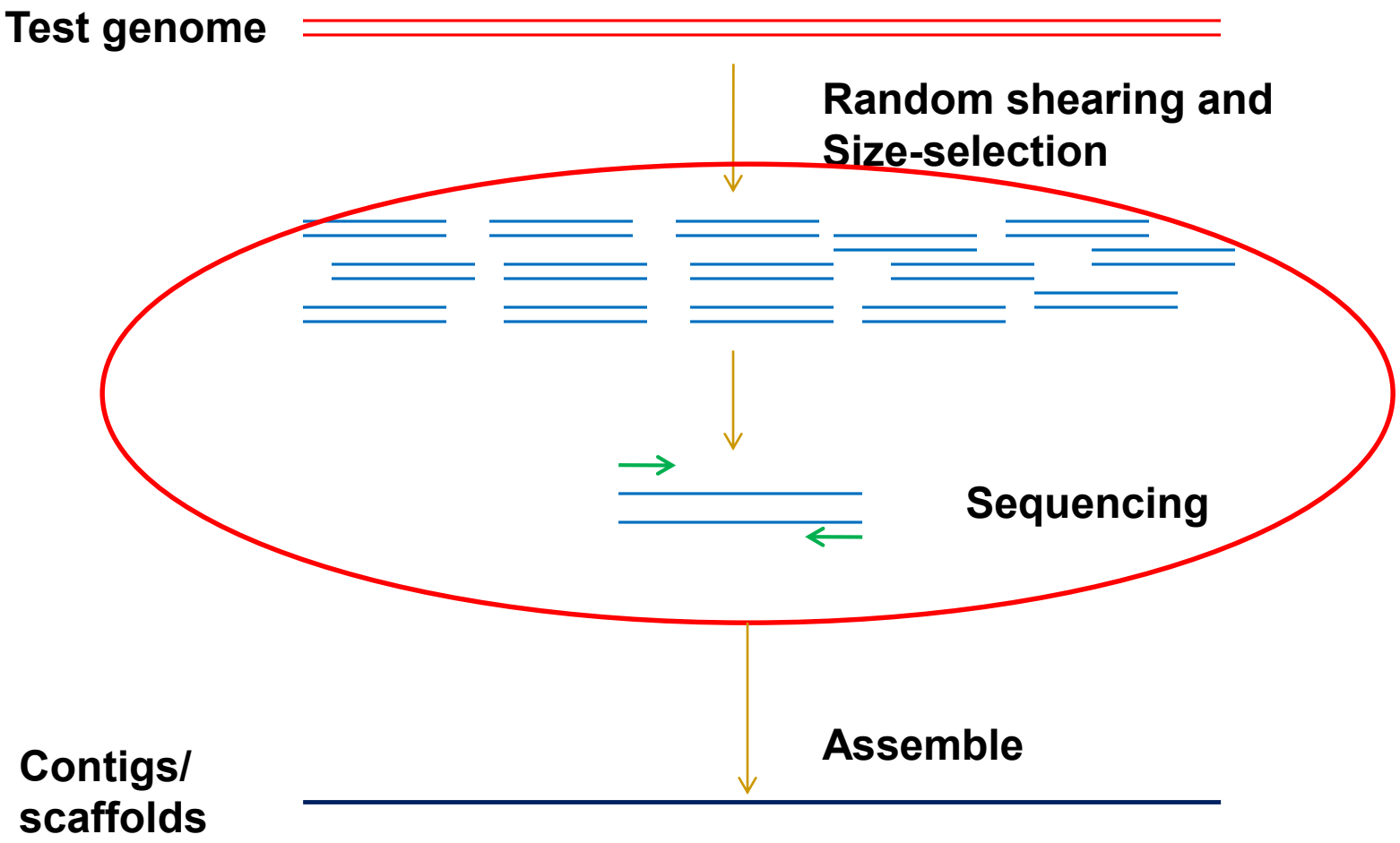
Genome Assembly

- Given a set of sequence reads (Sanger, NGS single end, NGS paired end, NGS strobe, etc.) reconstruct the genomic sequence
  - Reference guided: When a reference genome (same species or highly similar) is available
  - *de novo*: No apriori information needed

Genome Assembly



Challenges

- DNA is double stranded; assemblers must consider 2 versions for each read
- Sequencing errors
- Repeats & duplications
- Heterozygosity
  - Diploid genomes: 2 alternates of each locus
  - Polyploid plant genomes are harder to deal with!

Challenges (cont'd)

- Large genomes require
  - More computational power
  - More memory (most algorithms >300 GB for mammalian genomes)
- Contamination:
  - Quite common to have DNA from other sources in the dataset
    - Eg. yeast, E. coli, other bacteria, etc.
      - Initial dataset from the bonobo genome was contaminated even with tomato and corn!
- Big data
  - Billions of reads to work with

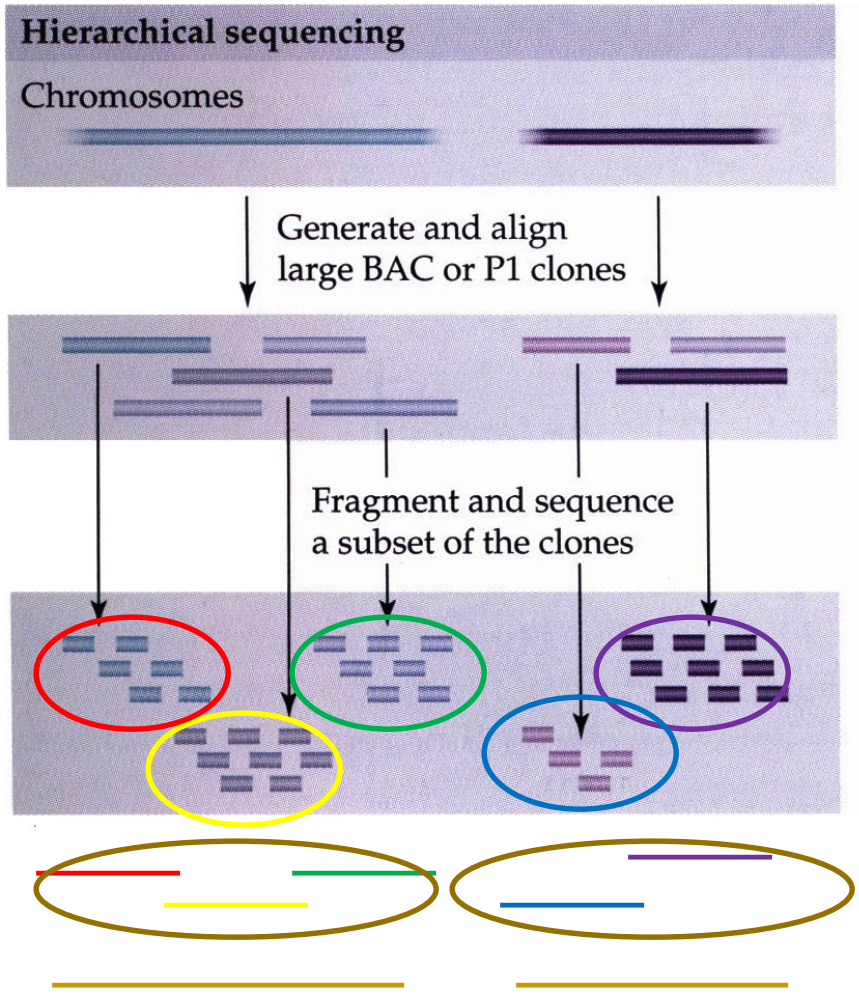
Parameters for assembly

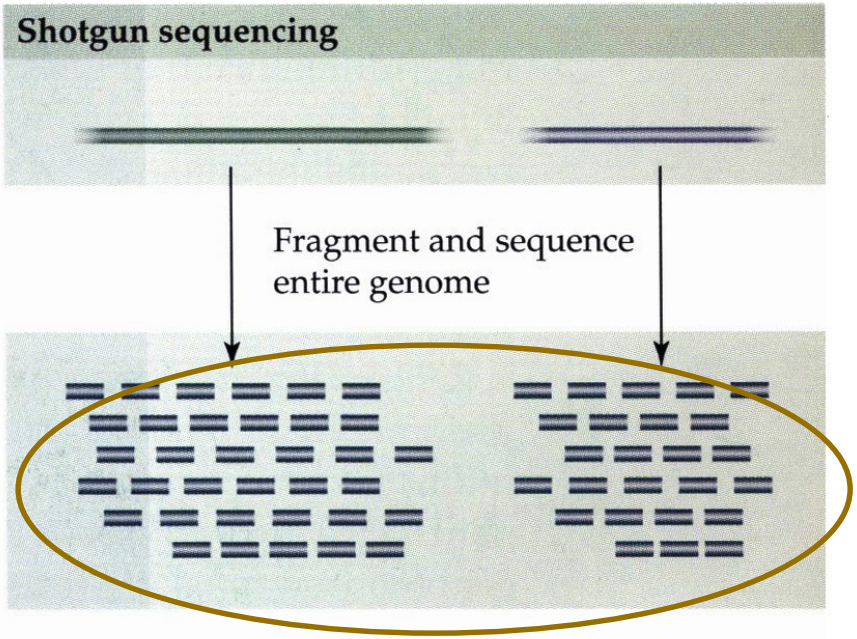
- Coverage
  - GC% biases can be ameliorated a little by increasing overall coverage
- Read length
- Insert size
- Better with multiple libraries with different insert sizes
- Better with multi-platform data
- Better with additional information
  - Physical fingerprinting (if clones available)
  - STS mapping (needs some *a priori* information)

Basics

- No technology can read a chromosome from start to finish; all sequencers have limits for read lengths
- Two major approaches
  - Hierarchical sequencing (used by the human genome project)
    - High quality, very low error rate, little fragmentation
    - Slow and expensive!
  - Whole genome shotgun (WGS) sequencing
    - Lower quality, more errors, assembly is more fragmented
    - Fast and cheap(er)

Hierarchical vs. shotgun sequencing





Assemble all

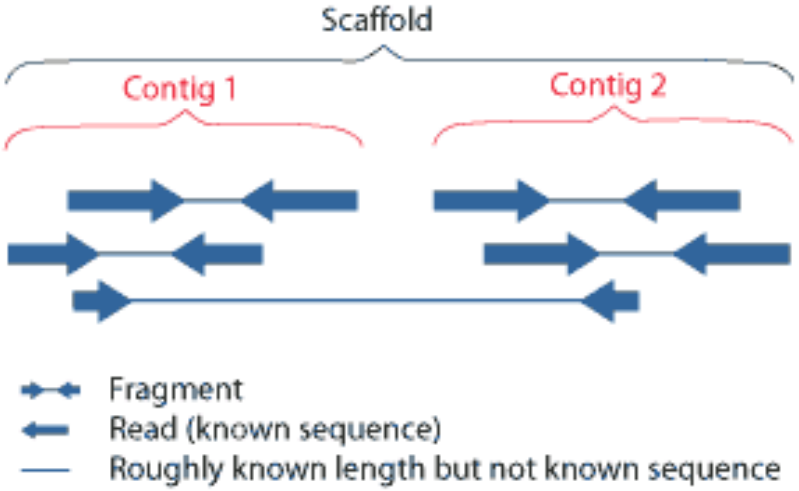
Assemble step by step

Cloning vectors

- Plasmids: carry 3-10 kbp of DNA
- Fosmids: carry ~40 kbp of DNA
- Cosmids: carry ~35-50 kbp of DNA
- BACs (bacterial artificial chromosomes): ~150-200 kbp of DNA
- YACs (yeast artificial chromosomes): 100 kbp – 3 Mbp of DNA

Assembly terminology

- Contig: contiguous segments of DNA sequences generated by the assembler using the reads
- Scaffold: Ordering of contigs separated by gaps
- Draft assembly: Includes many contigs and scaffolds, most sequence remains unassigned to chromosomes
- Finished assembly: most sequence assigned to chromosomes, most gaps are closed
  - Typically involves manual intervention & costly and slow methods



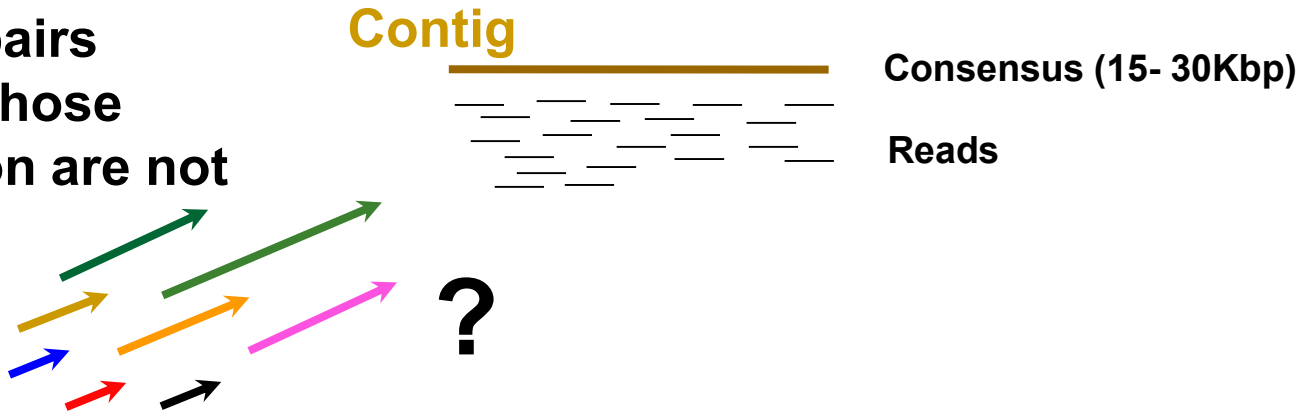
http://genome.jgi.doe.gov/help/scaffolds.html

Assembly quality assessment

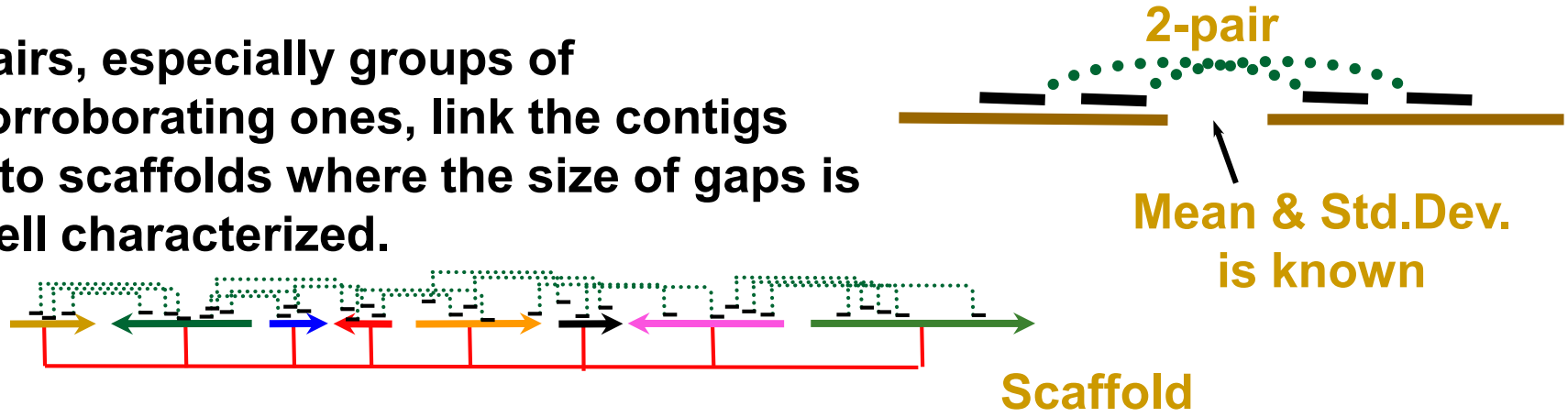
- Assembly size: is the summation of contig/scaffold lengths similar to what is expected from the genome of interest?
- Number of contigs/scaffolds: lower is better
  - Ideally equal to # of chromosomes
- N50: contig length such that using equal or longer contigs produces half the bases of the genome
  - L = Sum of all contig lengths c[1..n]
  - Sort contigs in descending order by length
  - X = 0, I = 0
  - X = X + c[i]
    - If X >= L/2; N50 = c[i]

Scaffolding with read pairs

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

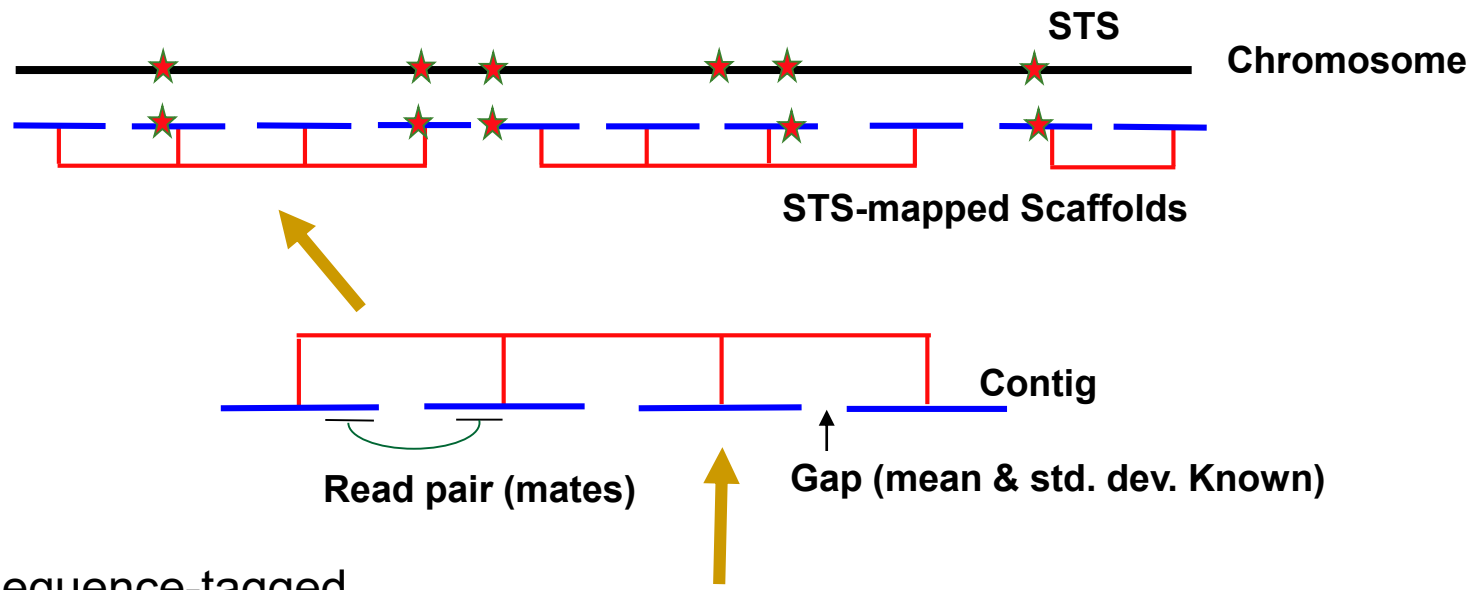


Pairs, especially groups of corroborating ones, link the contigs into scaffolds where the size of gaps is well characterized.



Slide from Mihai Pop

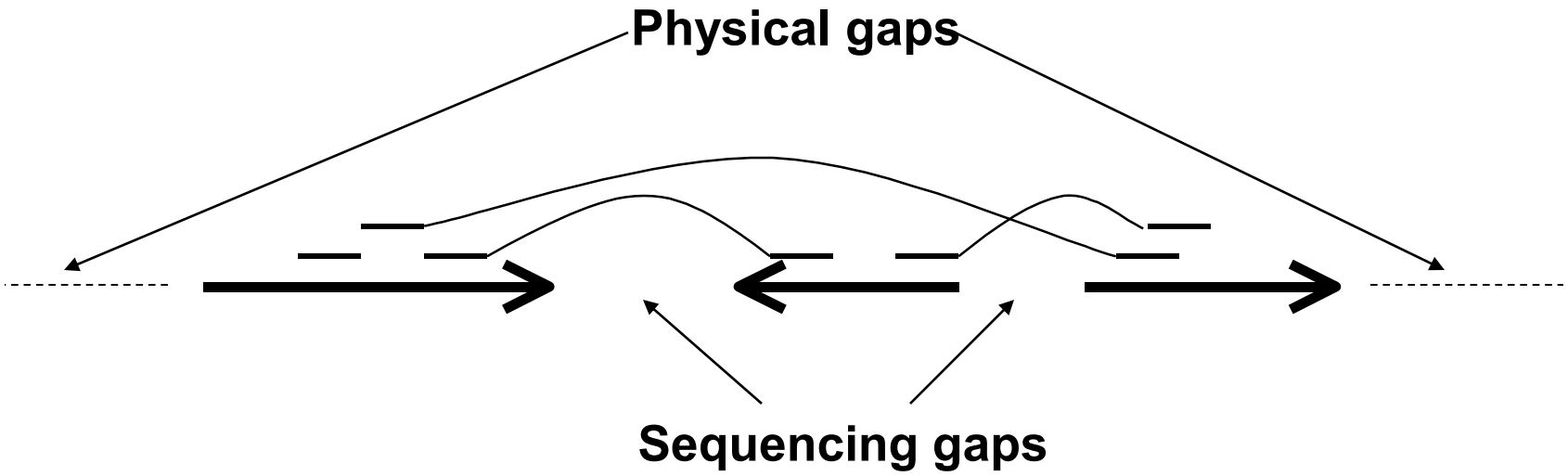
WGS Assembly



STS: sequence-tagged sites = 200-500 bp of sequence that is unique In the genome

Slide from Mihai Pop

Assembly gaps

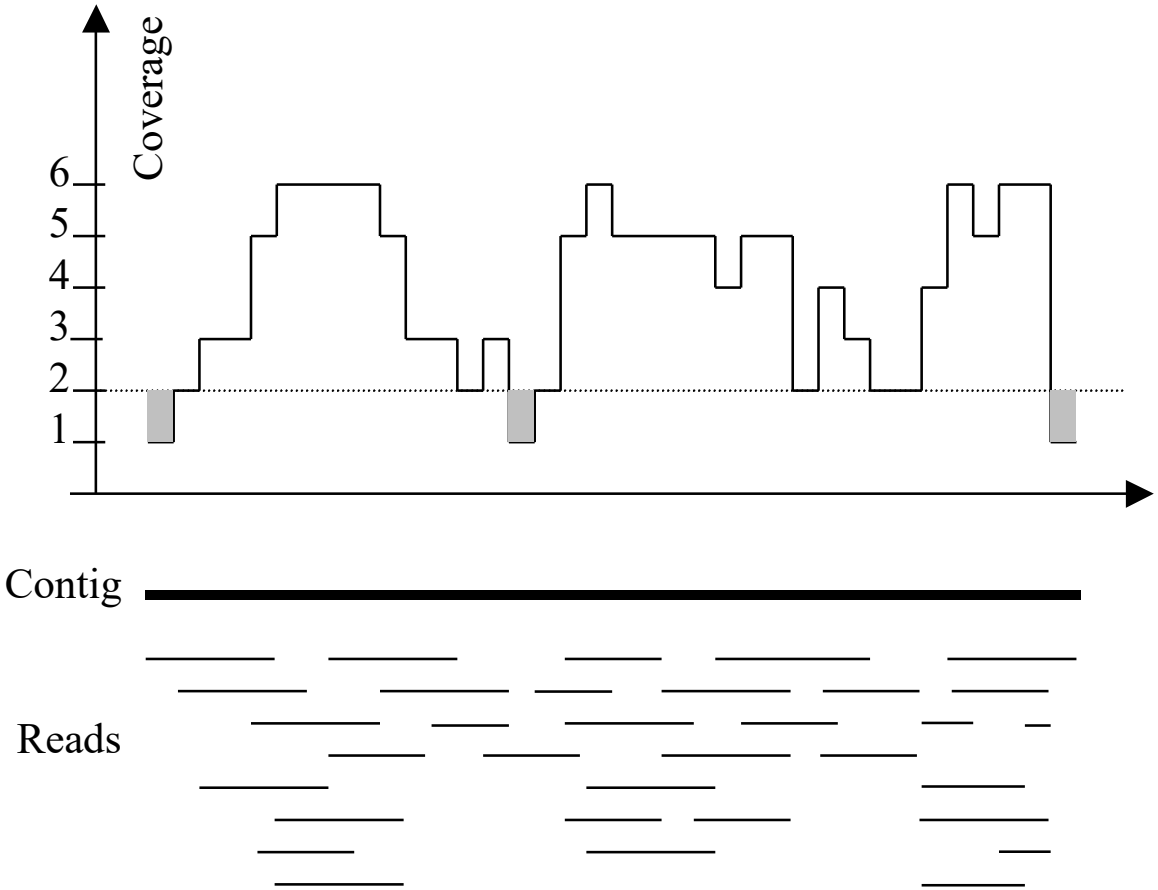


sequencing gap - we know the order and orientation of the contigs and have at least one clone spanning the gap

physical gap - no information known about the adjacent contigs, nor about the DNA spanning the gap

Slide from Mihai Pop

Typical contig coverage

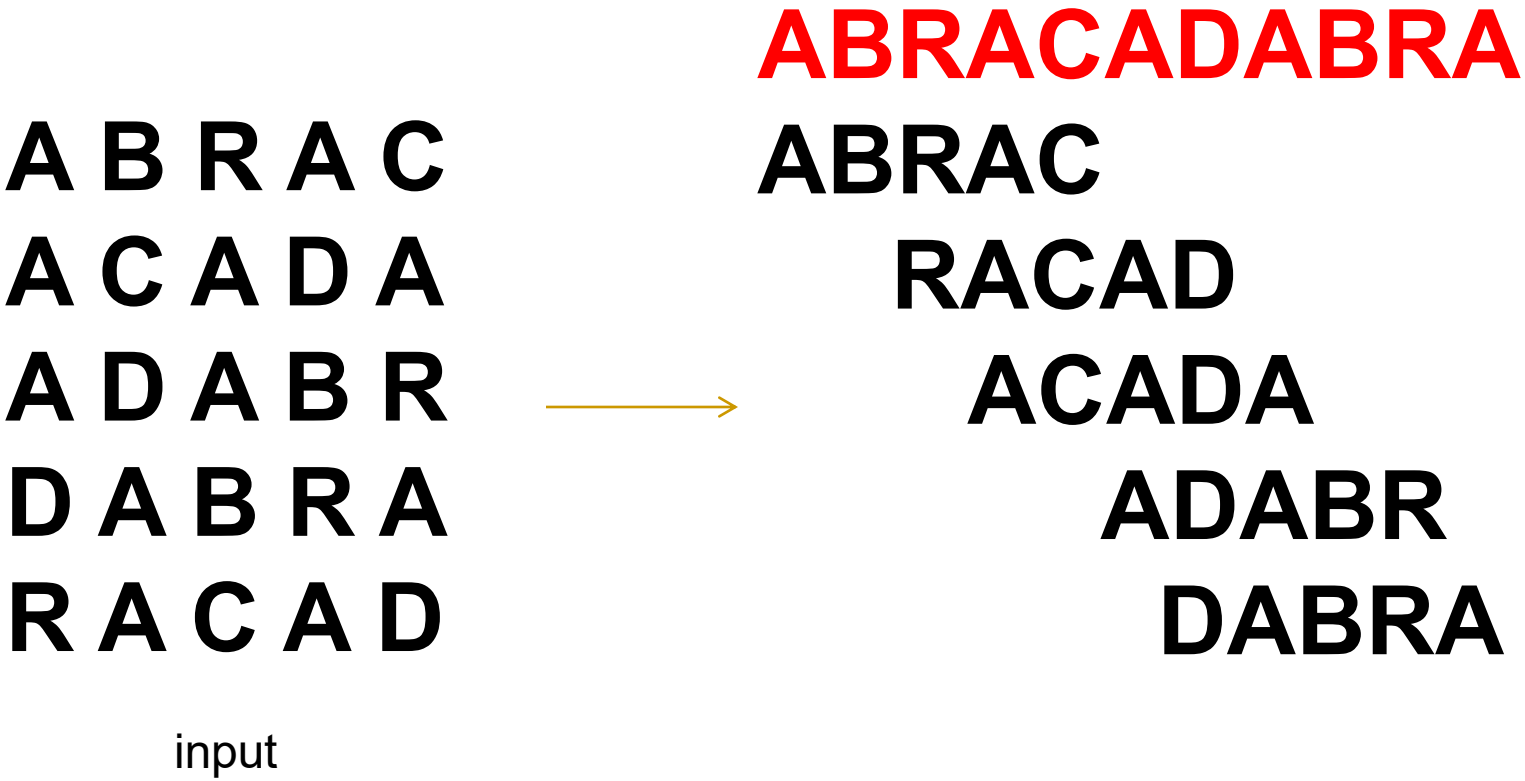


Slide from Mihai Pop

Basic algorithmic definition

- Genome assembly problem is finding **shortest common superstring** of a set of sequences (reads):
  - Given strings $\{s_1, s_2, \ldots, s_n\}$; find the superstring T such that every s_i is a substring of T
  - NP-hard problem
  - Greedy approximation algorithm
    - Works for simple (low-repeat) genomes

Shortest superstring problem



Assembly paradigms

- Overlap-layout-consensus
  - greedy (TIGR Assembler, phrap, CAP3...)
  - graph-based (Celera Assembler, Arachne)
    - SGA for NGS platforms
- Eulerian path on de Bruijn graphs(especially useful for short read sequencing)
  - EULER, Velvet, ABySS, ALLPATHS-LG, Cortex, etc.

Slide from Mihai Pop

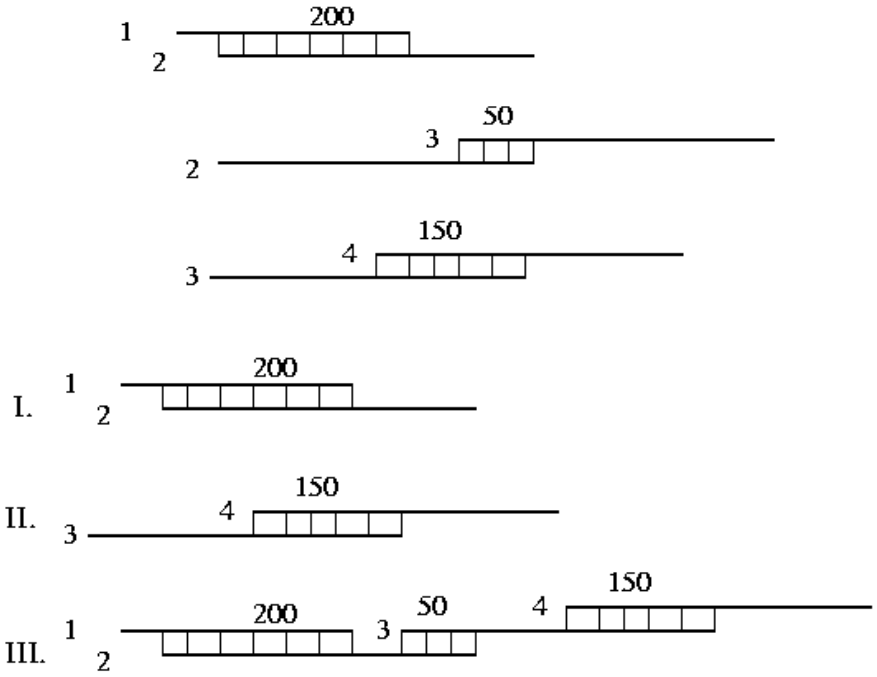
Greedy Algorithms

- The greedy solution to shortest common superstring problem
- Good for small genomes with no or low repeat/duplication content
- First assembly algorithms used greedy methods

TIGR Assembler/phrap

Greedy method

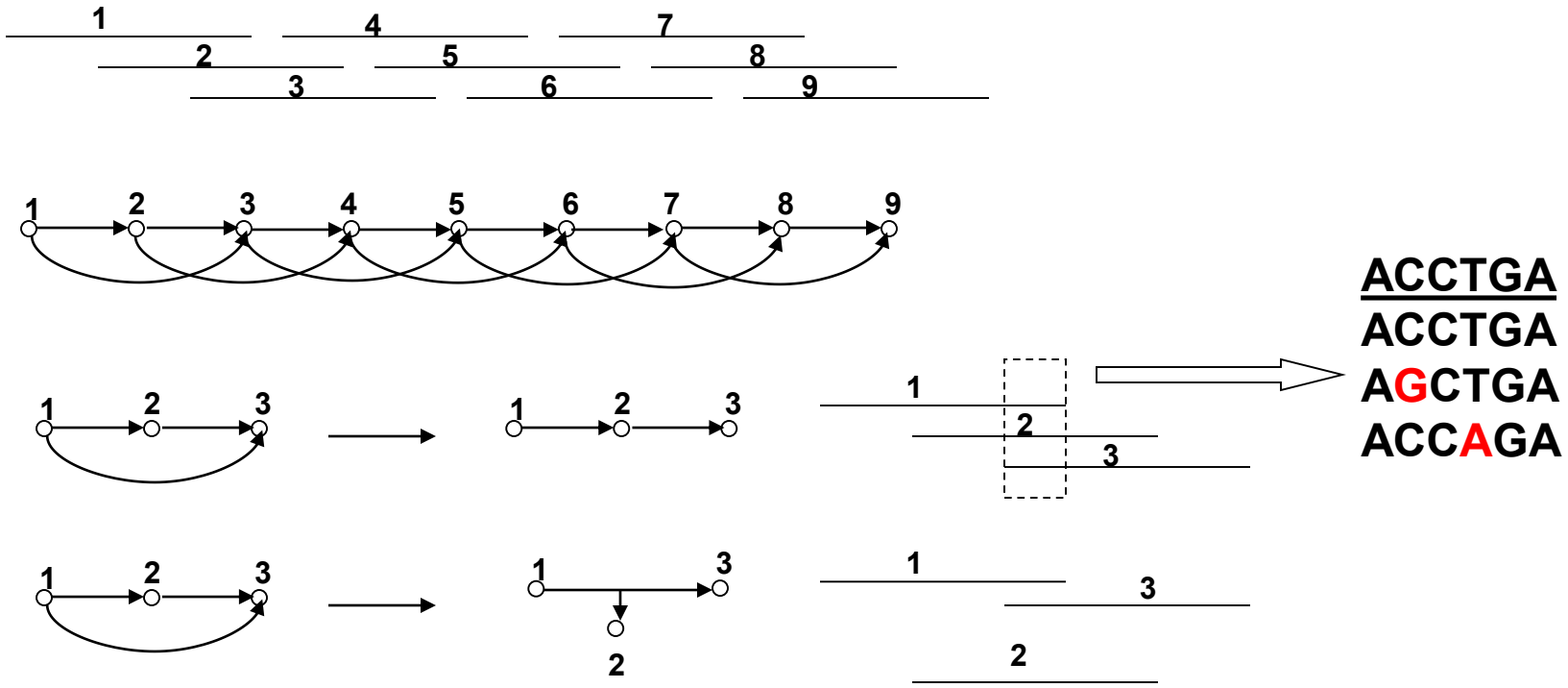
- Build a rough map of fragment overlaps
- Pick the largest scoring overlap
- Merge the two fragments
- Repeat until no more merges can be done



Slide from Mihai Pop

Overlap-layout-consensus

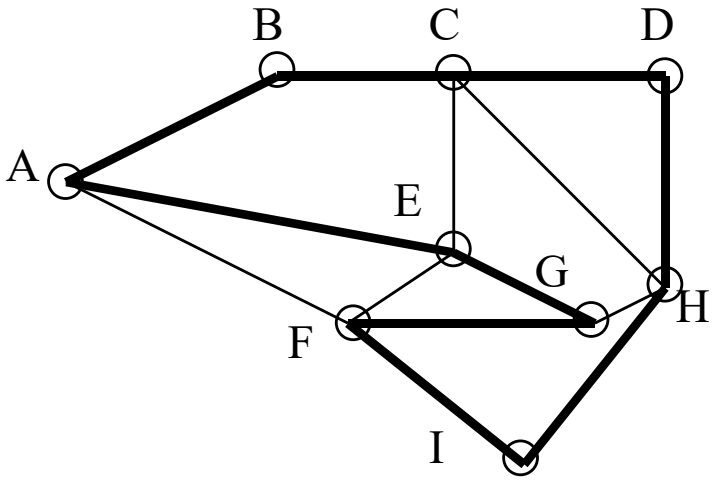
Main entity: read
Relationship between reads: overlap

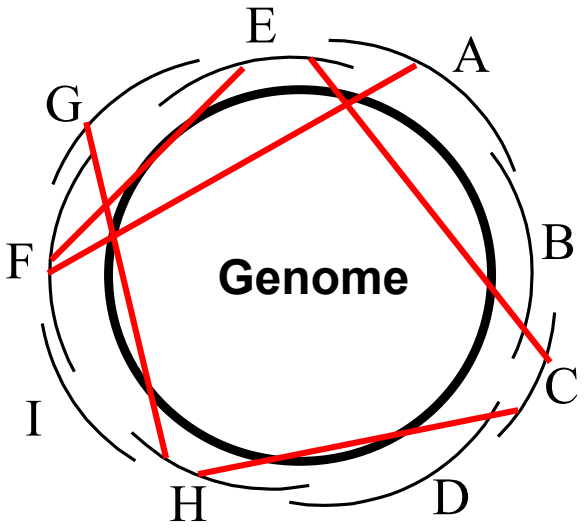


Slide from Mihai Pop

Paths through graphs and assembly

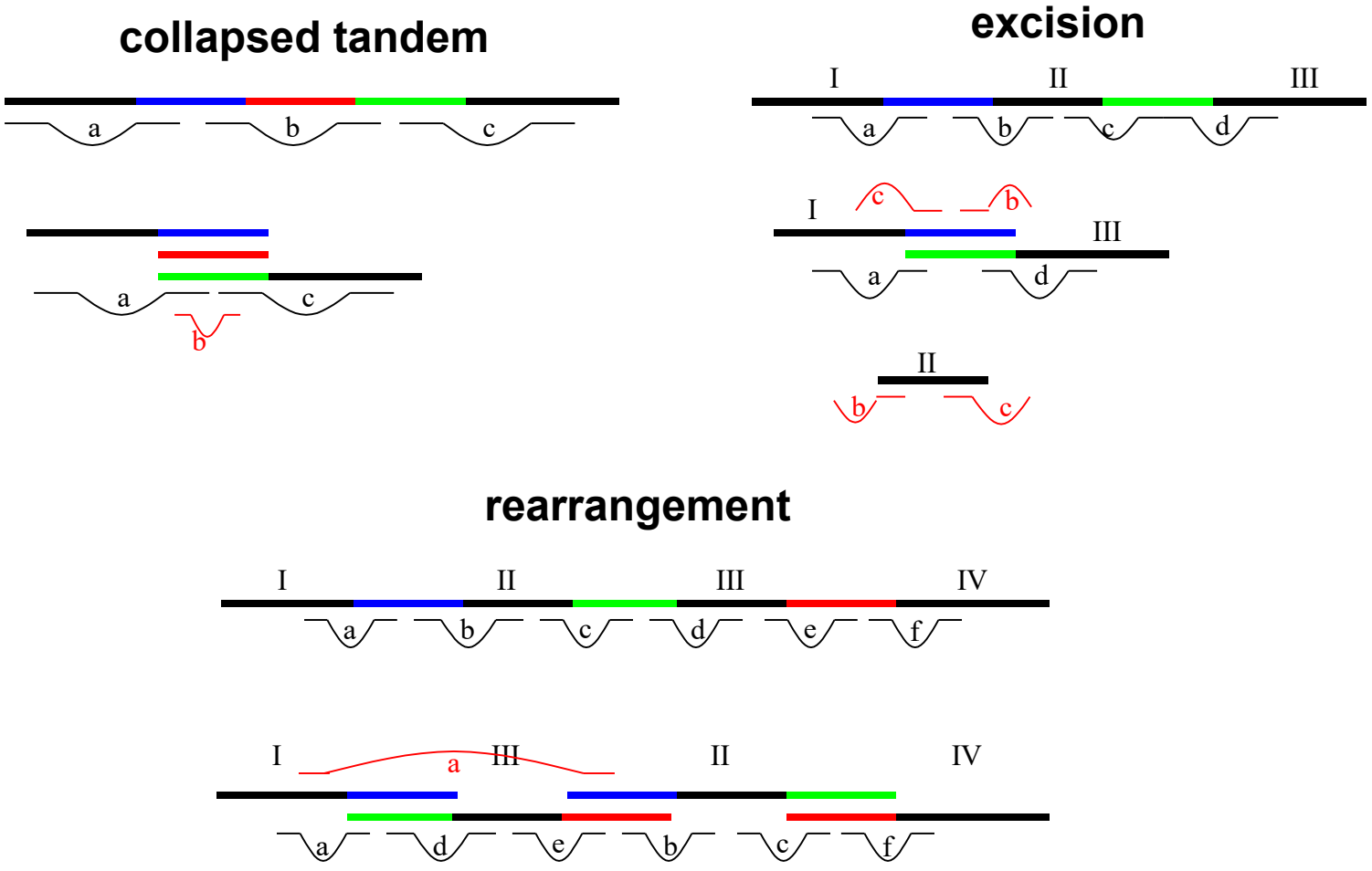
- Hamiltonian cycle: visit each node exactly once, returning to the start





REPEATS

Mis-assembled repeats



Slide from Mihai Pop