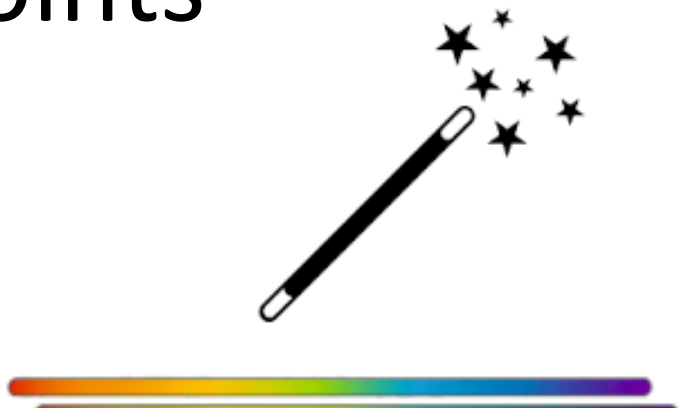
De novo Genome Assembly

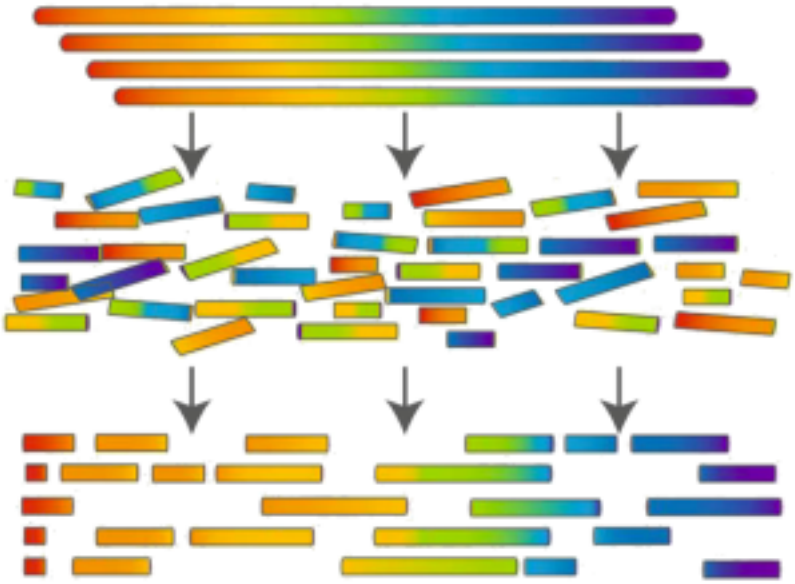
- *De novo* = starting from the beginning
- *De novo* assembly = starting with raw DNA reads, assembling them into a reference genome
- IE – not mapping reads back to an existing reference or comparing to a reference

Previous points

- Ideal: read an entire chromosome from beginning to end in one long, perfect run



- Reality: Genome assemblies are messy, leading to varying levels of completeness
  - Complete, Chromosome, Scaffold, Contigs
  - Why?



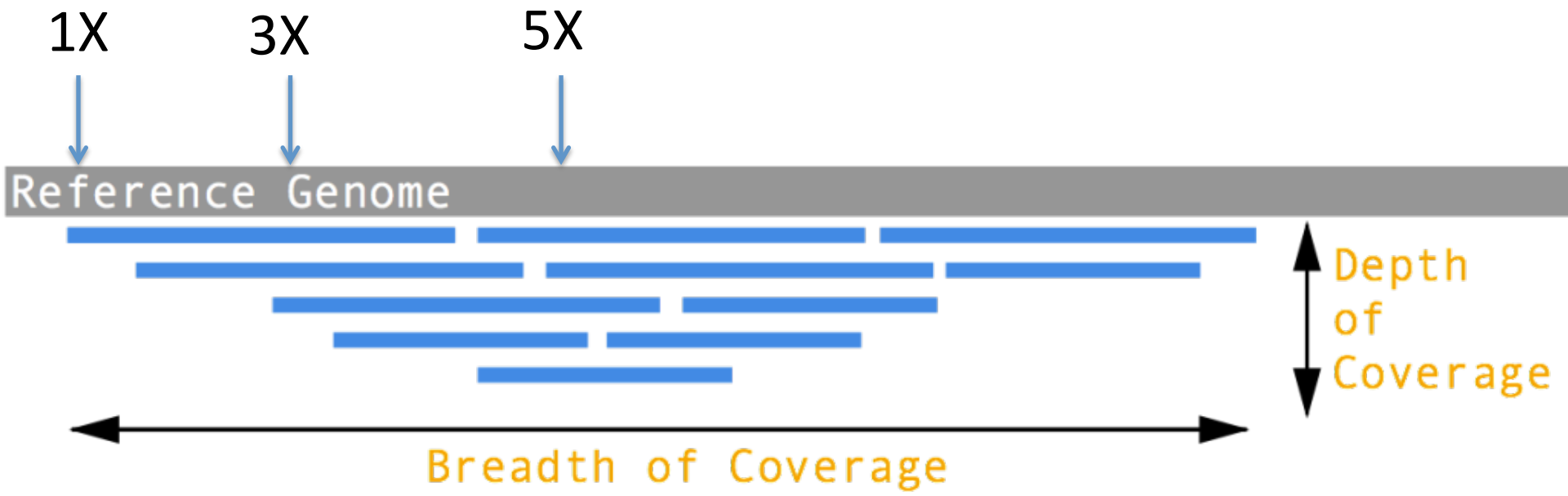
Jigsaw puzzle:

- How do the pieces fit together? (overlap)
- Missing pieces (sequencing bias)
- Dirty pieces (sequencing error, real biological variation)

Depth vs. Coverage

Often synonyms
Better to separate as two ideas:

1. Depth of coverage
How many reads cover a single base?
Average depth – how many reads cover each base on average?



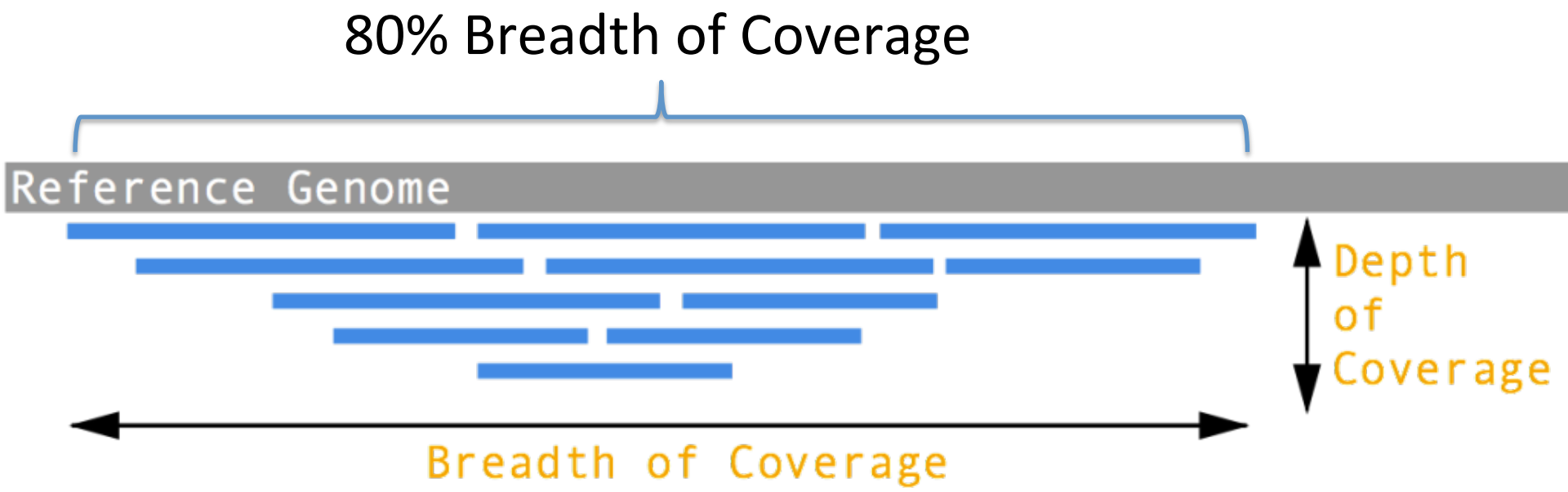
Depth vs. Coverage

Often synonyms

Better to separate as two ideas:

2. Breadth of coverage:

If less than the whole genome has read coverage, the percent with read coverage.



Difficulty 1: Volume of Information

Small genome of 25Mb

- Would like to sample each base 100 times
- 2.5 billion bases in 25 million reads (100 bases per read

Bigger genome

- Norway spruce is 20Gb
- Produced 1.9 trillion bases

100X coverage is reasonable starting point for assembly.

Difficulty 1: Volume of Information (Continued)

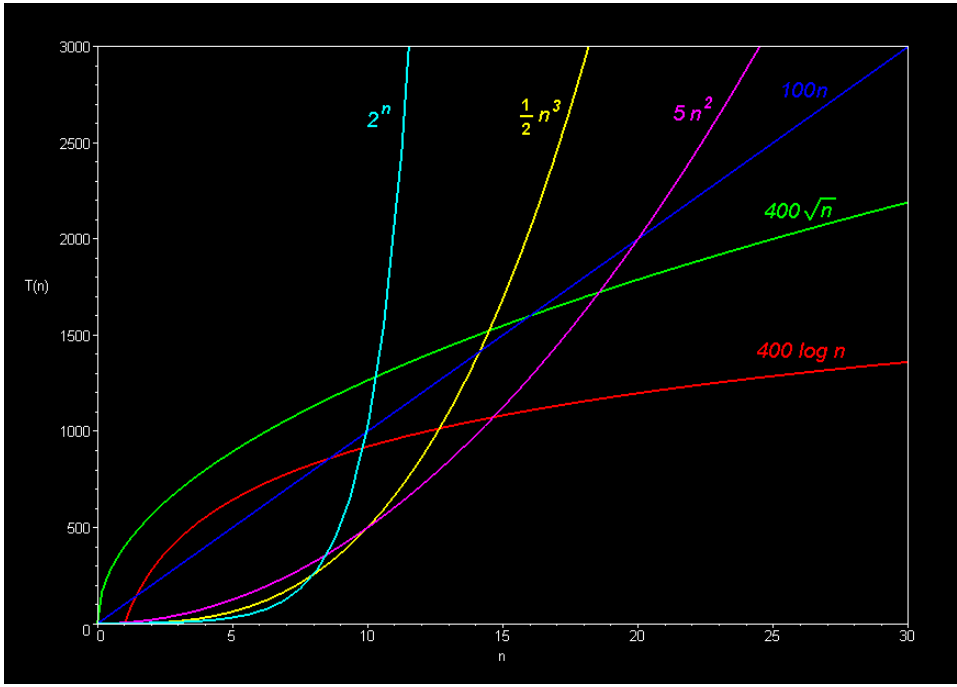
We need to examine if each read overlaps with each and every other read

Comparisons = Nx(N-1)/2
~ N^2

We need special computer algorithms and data structures to reduce the problem to a tractable level.

(Approximate methods will be involved)

N^2. Yikes!



Running time graph.

Difficulty 2: Repeats

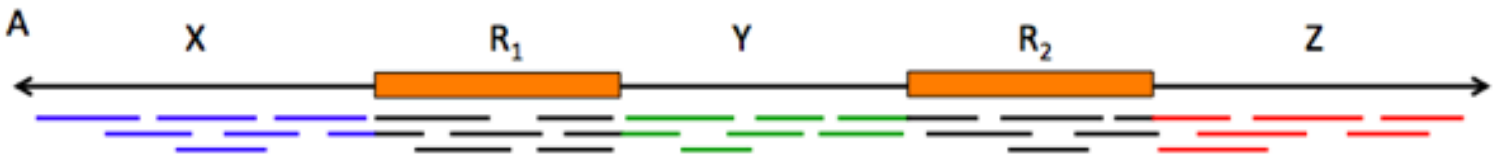
- Short repeats are problematic
- Lots of them, often longer than our (short) read length

```
?? ATATATATATATATATATATATATATATATATATATATATATATATA ??
   ATATATATATAT     ATATATATATA     TATATATATA
     ATATATATA         ATATATATATATAT    TATATAT
    TATATATATATATATATATA        TATATATATA
```

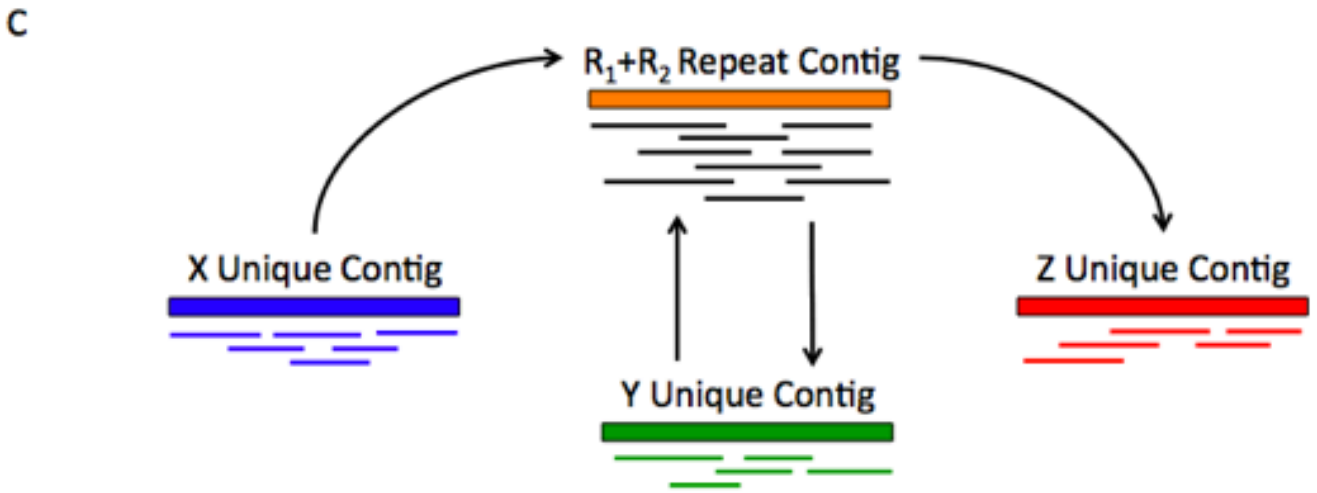
Difficulty 2: Repeats (continued)

Long repeats are problematic too.

Reality:



Assembly results:



How to fix?

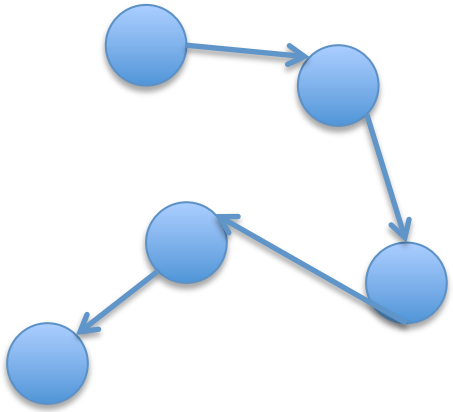
- Long reads
- Mate Pairs

Approaches

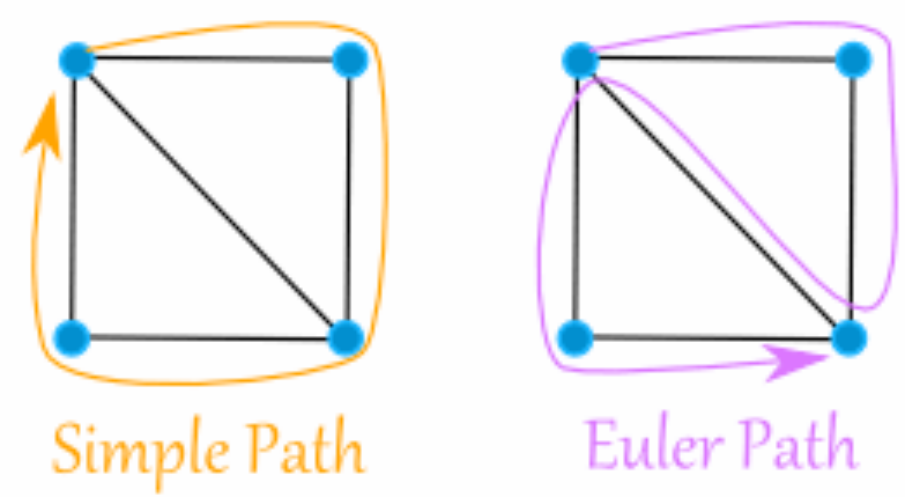
- Overlap, layout, consensus (OLC)
  - Relies on an overlap graph
  - Long reads (Sanger, 454)
- de Bruijn graph
  - Uses a k-mer graph
  - Short reads (Illumina)

Different ways to try to get to the same answer – reconstruct biological reality

Graph

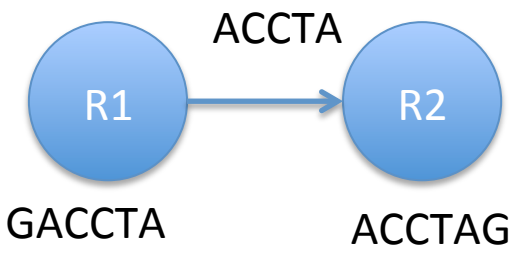


- Abstraction of data to nodes and edges
- Directed graph
  - Edges may only be traversed in one direction
  - Collections of edges form paths
- Simple path - Each node is visited once and only once
- Euler path – Each edge is visited once and only once



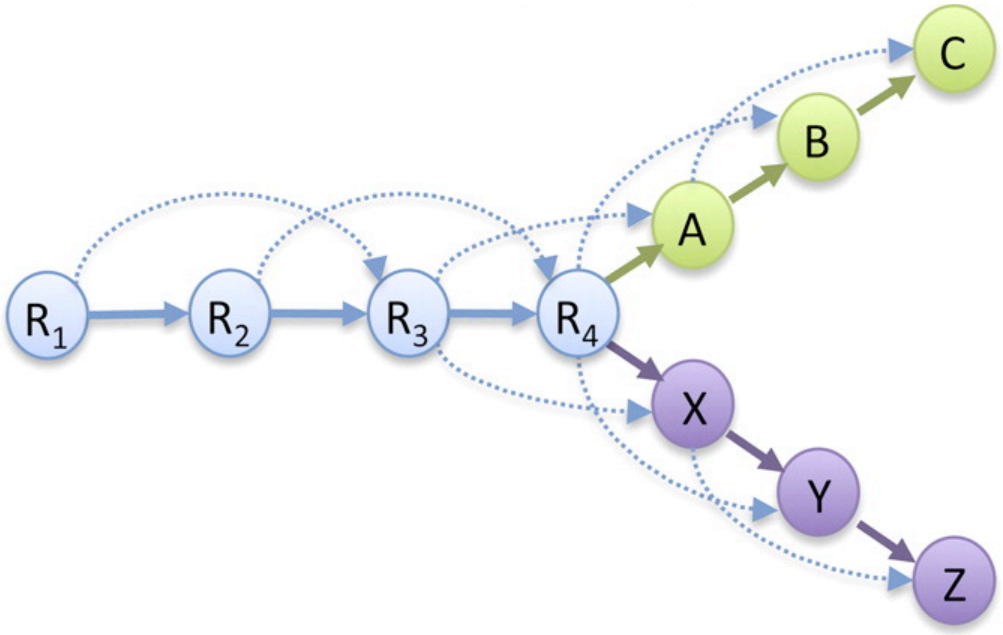
Overlap graph

- Represent sequencing reads and their overlaps
- Overlaps have to be computed by aligning all reads against all other reads
- Computationally expensive!
- Node = read
- Edge = overlap
- Paths through the graph are potential contigs



Overlap Graph

```
R1: GACCTACA
R2:    ACCTACAA
R3:     CCTACAAG
R4:      CTACAAGT
A:        TACAAGTT
B:         ACAAGTTA
C:          CAAGTTAG
X:        TACAAGTC
Y:         ACAAGTCC
Z:          CAAGTCCG
```



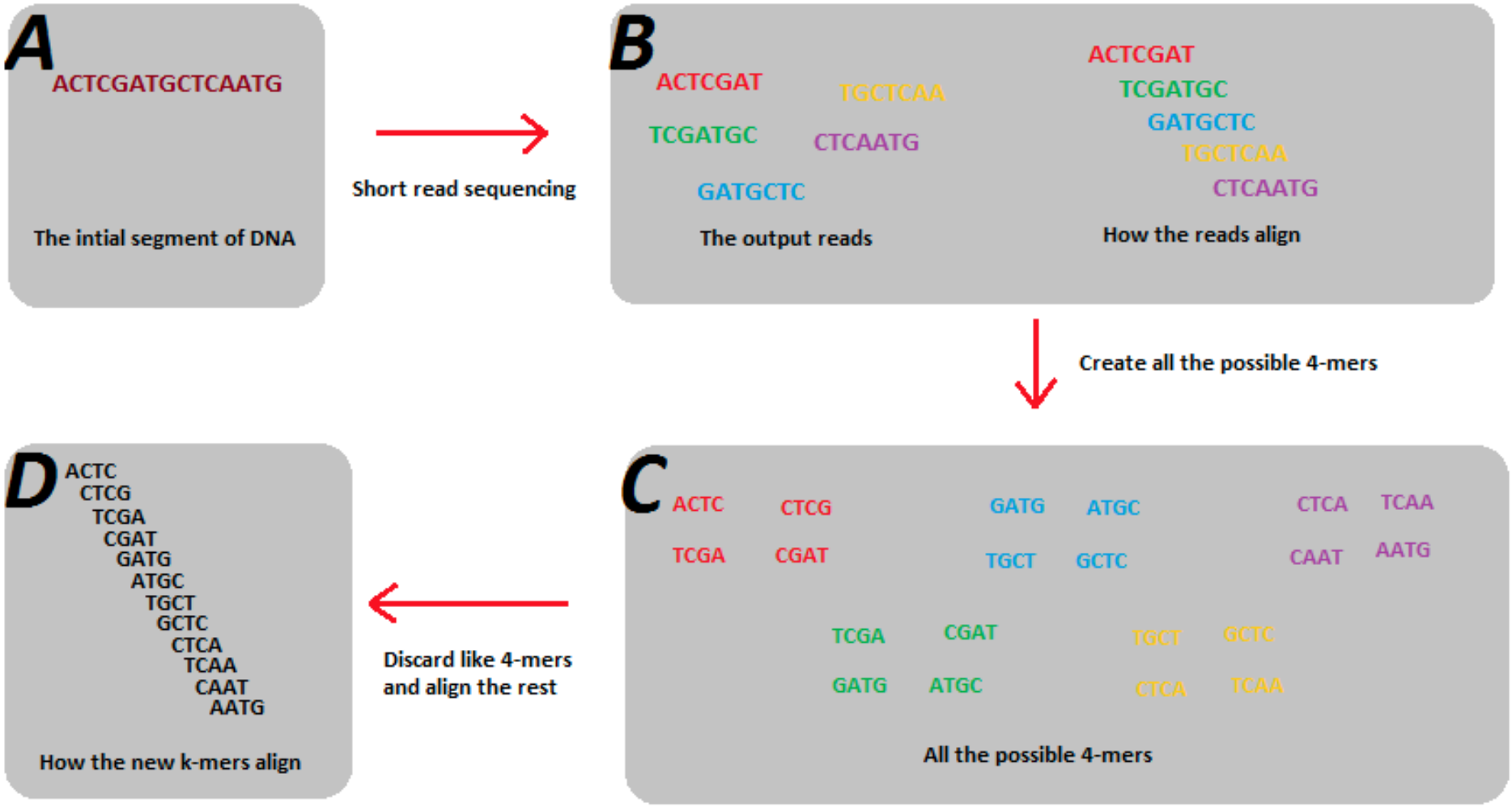
Reality could be: GACCTACAAGTTAGGACCTACAAGTCCG

Schatz et al "Assembly of large genomes using second-generation sequencing" 2010

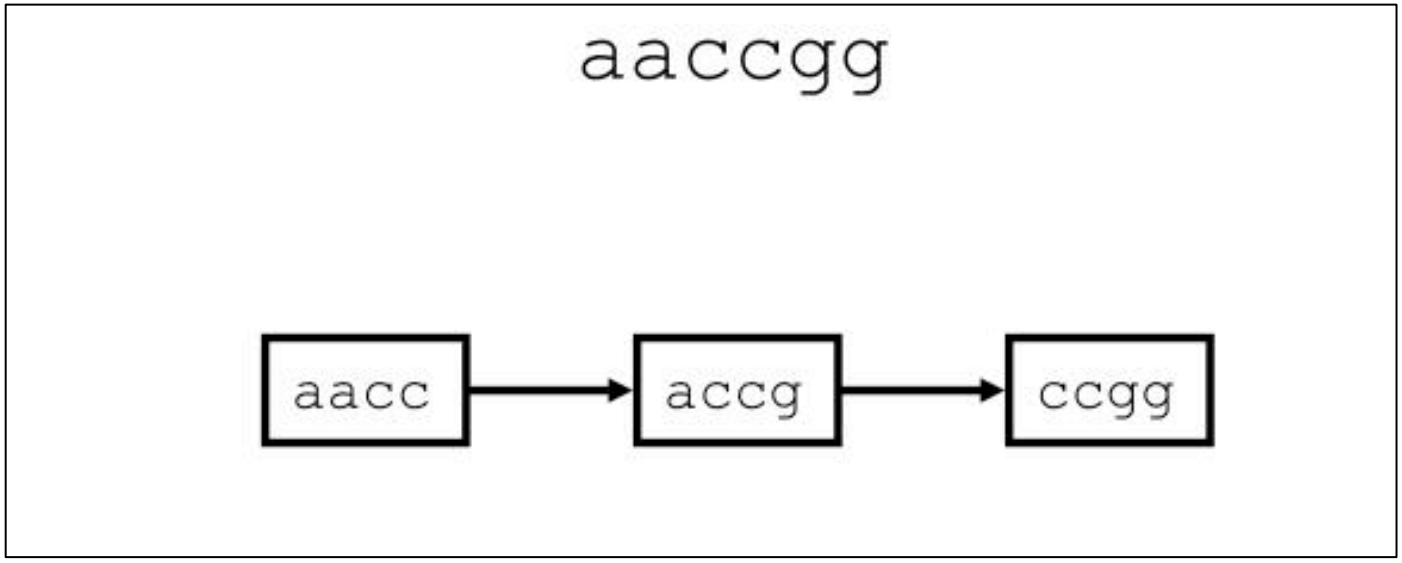
de Bruijn Graph

- Developed as a mathematic construct
- K-mer graph developed as a type of de Bruijn graph useful for assembly
- Nodes are subsequences of a longer sequence
- Edges are fixed length overlaps

K-mers


A
ACTCGATGCTCAATG
The intial segment of DNA
Short read sequencing
B
ACTCGAT
TGCTCAA
TCGATGC
CTCAATG
GATGCTC
The output reads
ACTCGAT
TCGATGC
GATGCTC
TGCTCAA
CTCAATG
How the reads align
Create all the possible 4-mers
C
ACTC
CTCG
GATG
ATGC
CTCA
TCAA
TCGA
CGAT
TGCT
GCTC
CAAT
AATG
TCGA
CGAT
TGCT
GCTC
GATG
ATGC
CTCA
TCAA
All the possible 4-mers
Discard like 4-mers
and align the rest
D
ACTC
CTCG
TCGA
CGAT
GATG
ATGC
TGCT
GCTC
CTCA
TCAA
CAAT
AATG
How the new k-mers align

A single read represented as a k-mer graph
K = 4



Miller et al. Assembly Algorithms for Next-Generation Sequencing Data. 2011

- More than one read can be represented by a de Bruijn graph
- Reads with perfect overlaps have the same path
- These overlaps are detected without calculating the alignments between every read pair
- Major computational savings!!!

Start with two reads

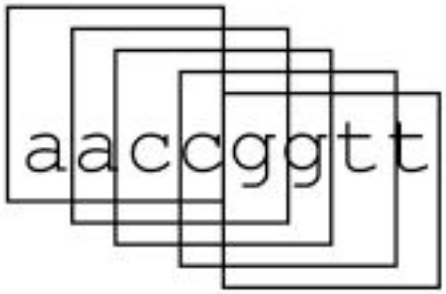
(a)

aaccgg
ccggtt

Build the graph – the graph holds the information about their overlap

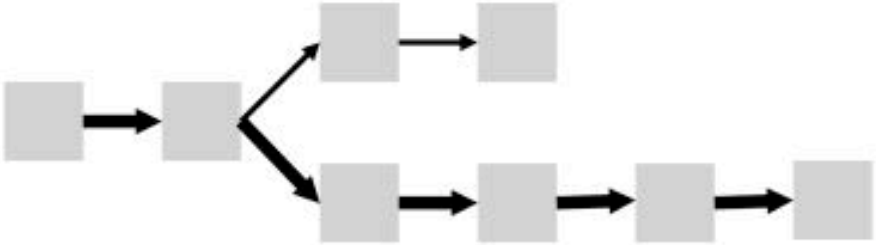
(b)

(c)

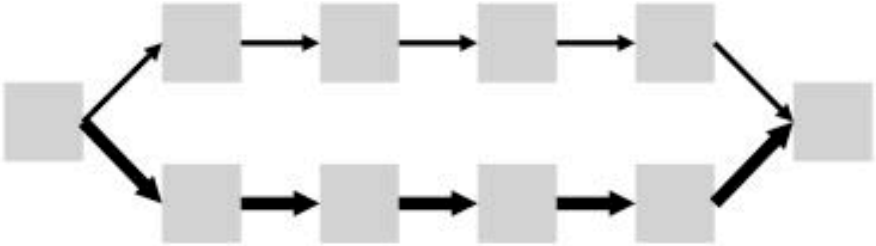


Merge the nodes to yield a sequence contig

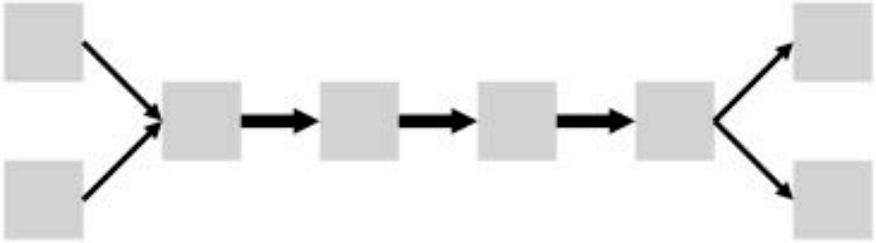
Complexity in Graphs



Error at read end causes a "spur"

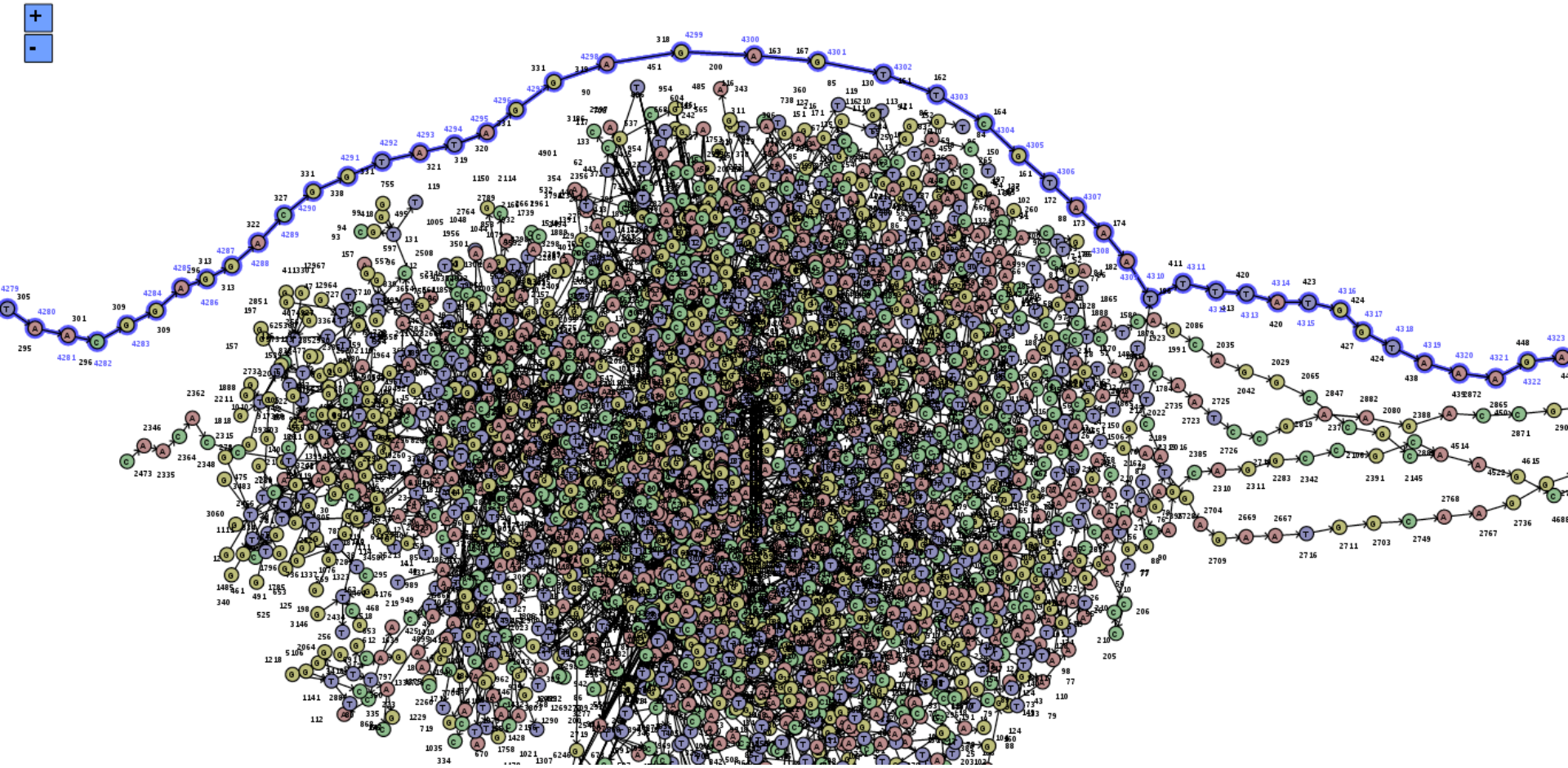


Real polymorphism or error in the middle of a read causes a "bubble"



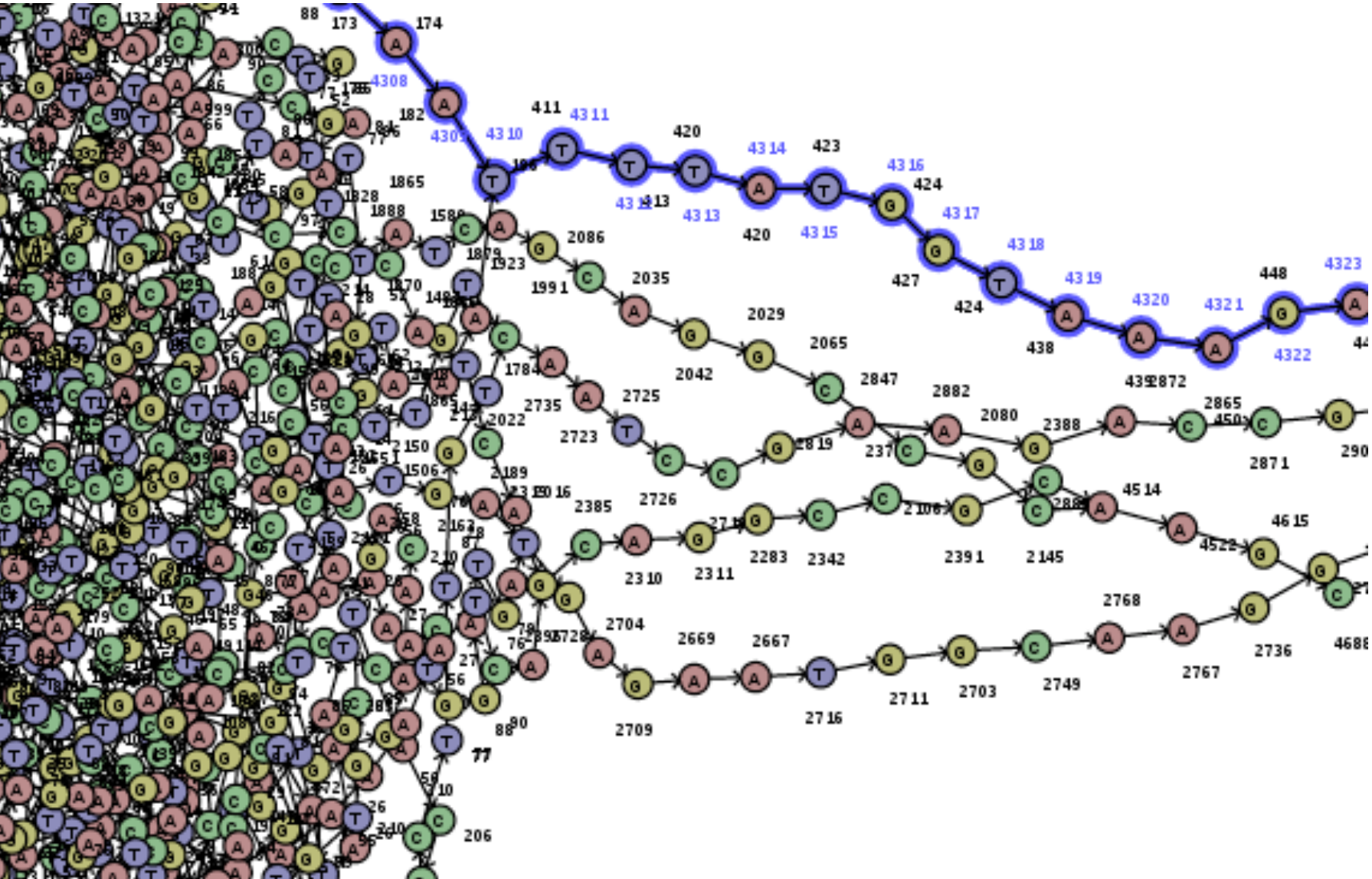
Repeats yield a "frayed rope"

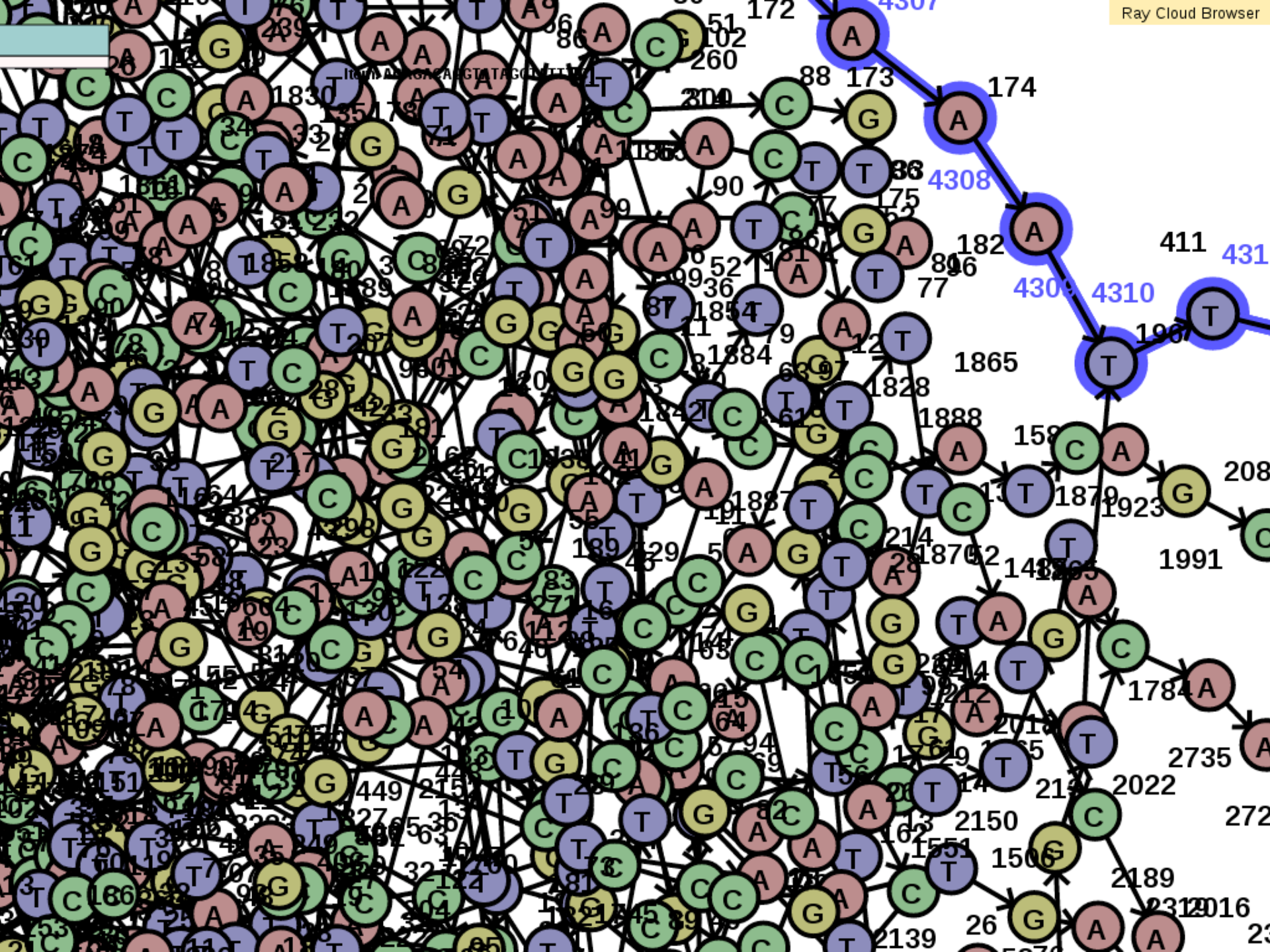
Actual Complexity Much Worse



Ray Cloud Browser
http://dskernel.blogspot.com/2013/01/metagenome-lumps-artifactual-mutations.html

Actual Complexity Much Worse




Ray Cloud Browser
4307
4308
4309
4310
411
431
1865
1991
2735
2022
2150
2189
2139

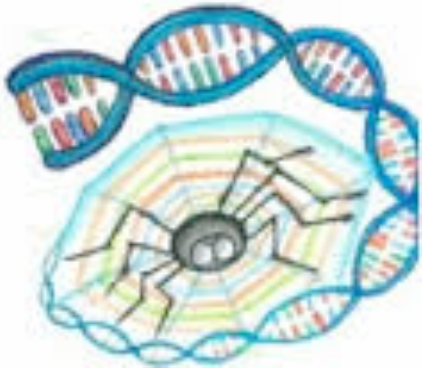
Assemblers

- To deal with complicated graphs, assemblers rely on heuristic algorithms and approximation algorithms to:
  - remove redundancy
  - repair errors
  - reduce complexity
  - enlarge simple paths
  - otherwise simplify the graph as much as possible

Software

OLC

- CABOG
- Celera
- Newbler (454)
- CAP
- Arachne



De Bruijn

- ABySS
- ALLPATHs-LG
  - DISCOVAR
- SOAPdenovo
- SPAdes
- Velvet

Probably over 100 implementations out there!

SPAdes
ABySS

- Primarily for bacteria assemblies
- Single cell sequencing as well as (normal) multi-cell sequencing
- Proper utilization of paired in data in a de Bruijn graph

ALLPATHs-LG

- Common for large eukaryotic genomes
- Requires both a paired end and a mate pair library
- Many clever improvements on memory needs and traversing the de Bruijn graph

Selecting k-mer value

- Smaller k-mers require less memory
  - But graphs are more complex and yield smaller average contig lengths
  - Repeats longer than the k-mer length cannot be resolved.
- Larger k-mers require much more memory
  - Can yield many longer contigs but also many smaller contigs
- Usually you will want to try a set of k-mer values and pick the best
  - Start with ½ to 2/3 the read length if you have sufficient RAM resources

A final step: Gap Filling

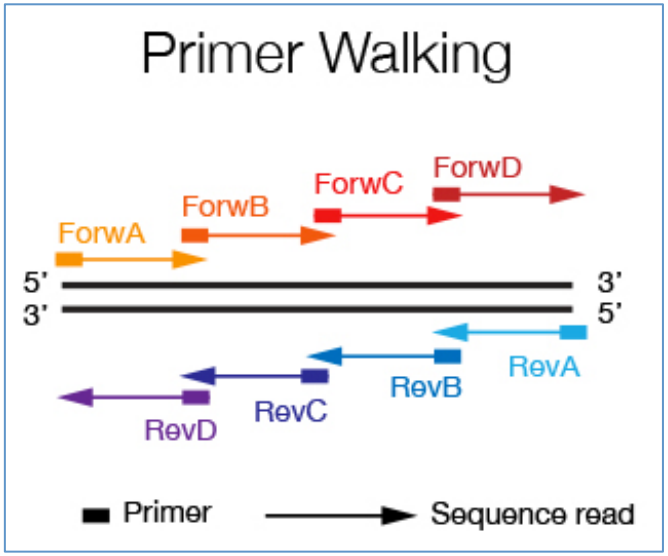
In the lab

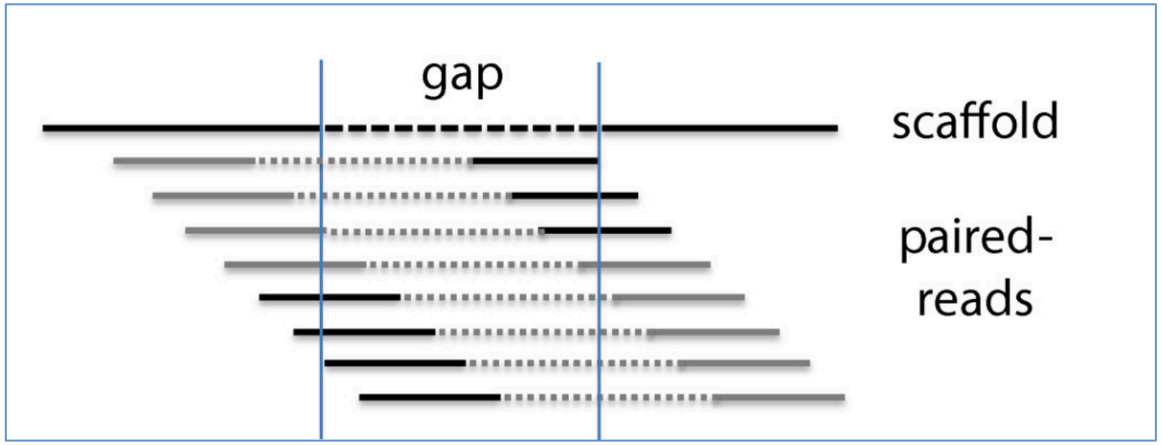
Develop primers and sequence

"Primer walking"

With software

Abyss comes with a gap-filler named Sealer





http://www.cogentech.it/sequenziamento-DNA-ita-technical-details.php

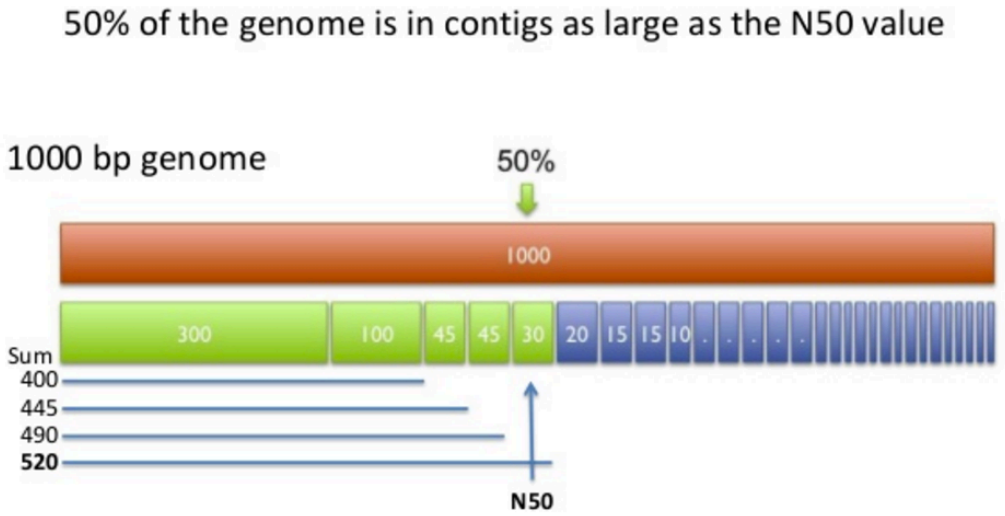
https://genomebiology.biomedcentral.com/articles/10.1186/gb-2012-13-6-r56

Assessing Genome Quality

1. Contiguity
2. Completeness
3. Correctness

Assessing Genome Quality

1. Contiguity
   - Would like fewer contigs in longer pieces
   - Assess # of contigs/scaffolds, average length, N50



Courtesy of Michael Schatz, CSHL

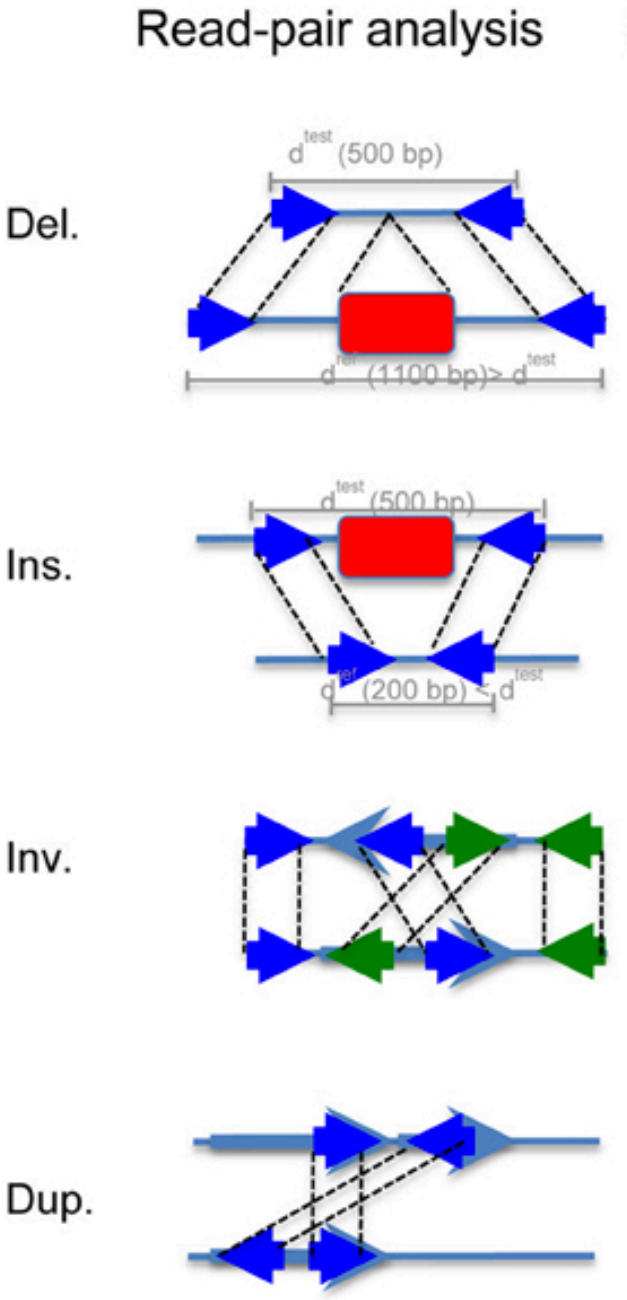
Assessing Genome Quality

2. Completeness
   - How much of the total genome was assembled?
     - Between 1 and 0
     - 80% complete vs 99% complete
     - This is based on an understanding of actual genome size
   - Many times repeats are difficult, but we're mostly concerned with genes
     - How many of the genes were captured in the assesmbly?
     - Can assess in two ways:
       - % of RNASeq reads that map to genome
       - Core conserved single copy orthologs – BUSCO
       - Make the assumption that the proportion of conserved single copy orthologs can be extrapolated to the total proportion of assembled genes

Assessing Genome Quality

3. Correctness
   - Lack of errors, such as mis-joins, collapsed repeats, miscalled bases, insertion/ deletions
   - Assess by aligning all reads back to the assembly and look for inconsistencies

(Analogous to sequencing a different individual and looking for structural variants)



Assessing Genome Quality

- QUAST
- Can be used with or without a reference genome
- Compare assembly attempts to each other
- Reports
  - Basic statistics about contigs and scaffolds
  - Comparison to a reference genome, including misassemblies and structural variation
  - Percent of reference genome covered by the assembly

Gurevich et al., 2013 QUAST: quality assessment tool for genome assemblies

Comparison of assembly software packages and approaches

Assemblathon 2: evaluating *de novo* methods of genome assembly in three vertebrate species

Keith R Bradnam[†] ✉, Joseph N Fass[†], Anton Alexandrov, Paul Baranay, Michael Bechner, Inanç Birol, Sébastien Boisvert, Jarrod A Chapman, Guillaume Chapuis, Rayan Chikhi, Hamidreza Chitsaz, Wen-Chi Chou, Jacques Corbeil, Cristian Del Fabbro, T Roderick Docking, Richard Durbin, Dent Earl, Scott Emrich, Pavel Fedotov, Nuno A Fonseca, Ganeshkumar Ganapathy, Richard A Gibbs, Sante Gnerre, Élénie Godzaridis, Steve Goldstein, Matthias Haimel, Giles Hall, David Haussler, Joseph B Hiatt, Isaac Y Ho, Jason Howard, Martin Hunt, Shaun D Jackman, David B Jaffe, Erich D Jarvis, Huaiyang Jiang, Sergey Kazakov, Paul J Kersey, Jacob O Kitzman, James R Knight, Sergey Koren, Tak-Wah Lam, Dominique Lavenier, François Laviolette, Yingrui Li, Zhenyu Li, Binghang Liu, Yue Liu, Ruibang Luo, Iain MacCallum, Matthew D MacManes, Nicolas Maillet, Sergey Melnikov, Delphine Naquin, Zemin Ning, Thomas D Otto, Benedict Paten, Octávio S Paulo, Adam M Phillippy, Francisco Pina-Martins, Michael Place, Dariusz Przybylski, Xiang Qin, Carson Qu, Filipe J Ribeiro, Stephen Richards, Daniel S Rokhsar, J Graham Ruby, Simone Scalabrin, Michael C Schatz, David C Schwartz, Alexey Sergushichev, Ted Sharpe, Timothy I Shaw, Jay Shendure, Yujian Shi, Jared T Simpson, Henry Song, Fedor Tsarev, Francesco Vezzi, Riccardo Vicedomini, Bruno M Vieira, Jun Wang, Kim C Worley, Shuangye Yin, Siu-Ming Yiu, Jianying Yuan, Guojie Zhang, Hao Zhang, Shiguo Zhou and Ian F Korf ✉

[†] Contributed equally