BI694

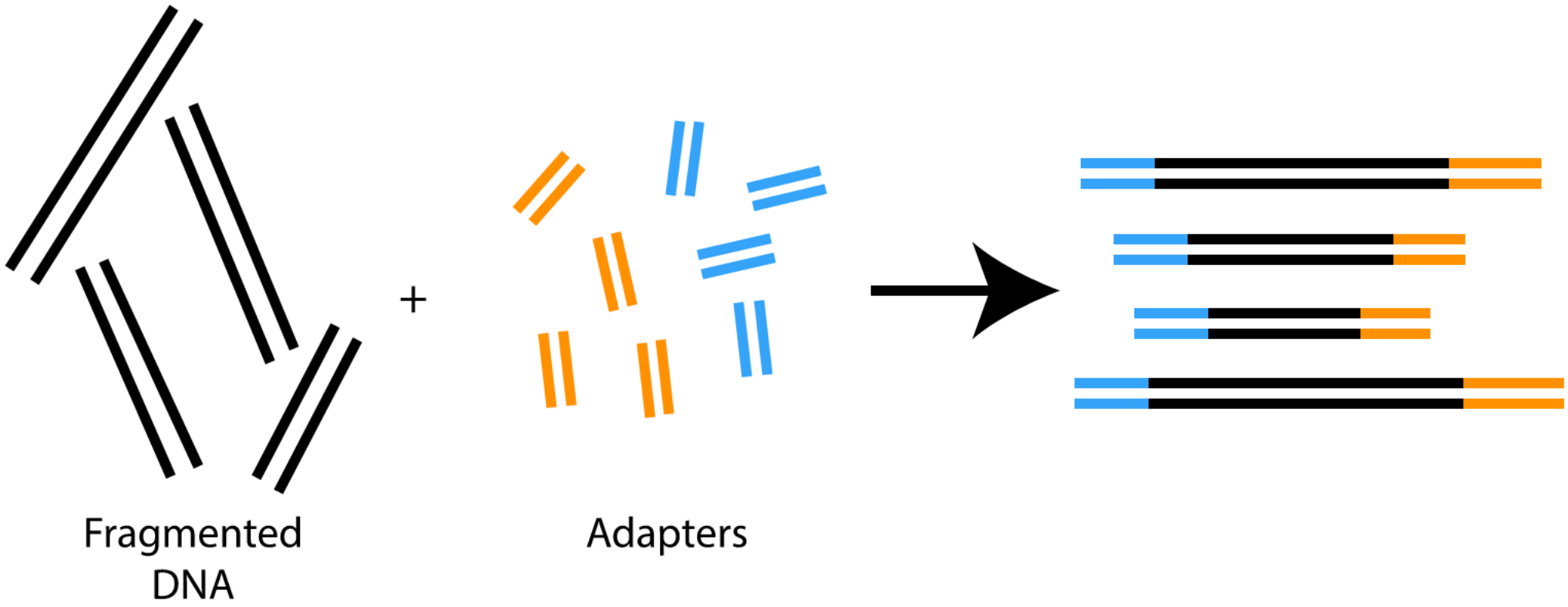
Bioinformatics & Phylogenetics

Winter Semester 2017

WEEK 8

FastQ File Format, Quality Encoding, Genome/Transcriptome Assembly

Sequencing



Adapters are attached to fragmented DNA.

FastQ File Format…

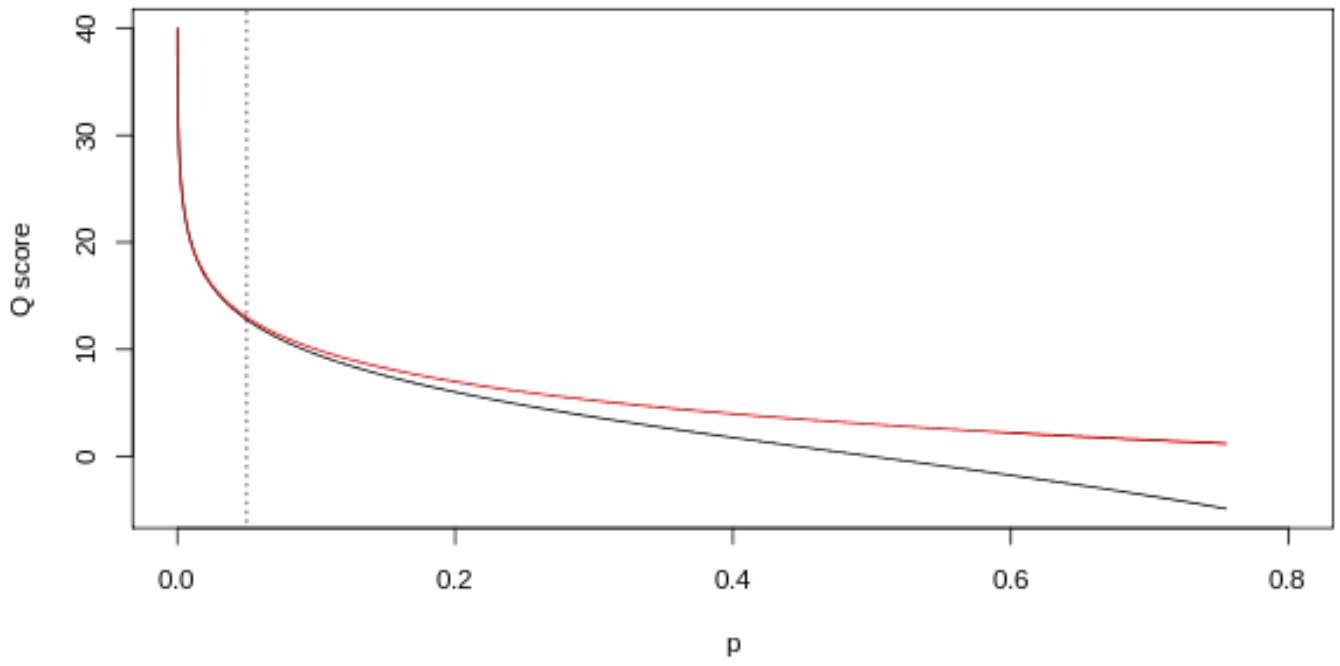
```
@SEQ_ID
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT
+
!''*((((***+))%%%++)(%%%%).1***-+*''))**55CCF>>>>>>CCCCCCC65
```

Phred Quality Scores

$$Q_{\text{sanger}} = -10 \log_{10} p$$

$$Q_{\text{solexa-prior to v.1.3}} = -10 \log_{10} \frac{p}{1-p}$$

p corresponds to the probability of the basecall being incorrect; the old Solexa (Illumina) encoding reports the odds that the basecall is incorrect.



Phred Quality Scores – ASCII offset encoding

```
@SEQ_ID
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCA
+
!''*((((***+))%%%++)(%%%%).1***-+*''))**55CCF>>>>
```

Phred score of 0 is encoded by ASCII Character 33.

What is the phred score of *?

0	<NUL>	32	<SPC>	64	@	96	`	128	Ä	160	†	192	¿	224	‡
1	<SOH>	33	!	65	A	97	a	129	Å	161	°	193	¡	225	·
2	<STX>	34	"	66	B	98	b	130	Ç	162	¢	194	¬	226	‚
3	<ETX>	35	#	67	C	99	c	131	É	163	£	195	√	227	„
4	<EOT>	36	$	68	D	100	d	132	Ñ	164	§	196	ƒ	228	‰
5	<ENQ>	37	%	69	E	101	e	133	Ö	165	•	197	≈	229	Â
6	<ACK>	38	&	70	F	102	f	134	Ü	166	¶	198	Δ	230	Ê
7	<BEL>	39	'	71	G	103	g	135	á	167	ß	199	«	231	Á
8	<BS>	40	(	72	H	104	h	136	à	168	®	200	»	232	Ë
9	<TAB>	41	)	73	I	105	i	137	â	169	©	201	…	233	È
10	<LF>	42	*	74	J	106	j	138	ä	170	™	202		234	Í
11	<VT>	43	+	75	K	107	k	139	ã	171	´	203	À	235	Î
12	<FF>	44	,	76	L	108	l	140	å	172	¨	204	Ã	236	Ï
13	<CR>	45	-	77	M	109	m	141	ç	173	≠	205	Õ	237	Ì
14	<SO>	46	.	78	N	110	n	142	é	174	Æ	206	Œ	238	Ó
15	<SI>	47	/	79	O	111	o	143	è	175	Ø	207	œ	239	Ô
16	<DLE>	48	0	80	P	112	p	144	ê	176	∞	208	–	240	
17	<DC1>	49	1	81	Q	113	q	145	ë	177	±	209	—	241	Ò
18	<DC2>	50	2	82	R	114	r	146	í	178	≤	210	“	242	Ú
19	<DC3>	51	3	83	S	115	s	147	ì	179	≥	211	”	243	Û
20	<DC4>	52	4	84	T	116	t	148	î	180	¥	212	‘	244	Ù
21	<NAK>	53	5	85	U	117	u	149	ï	181	µ	213	’	245	ı
22	<SYN	54	6	86	V	118	v	150	ñ	182	∂	214	÷	246	ˆ
23	<ETB>	55	7	87	W	119	w	151	ó	183	∑	215	◊	247	˜
24	<CAN>	56	8	88	X	120	x	152	ò	184	∏	216	ÿ	248	¯
25	<EM>	57	9	89	Y	121	y	153	ô	185	π	217	Ÿ	249	˘
26	<SUB>	58	:	90	Z	122	z	154	ö	186	∫	218	/	250	˙
27	<ESC>	59	;	91	[	123	{	155	õ	187	ª	219	€	251	°
28	<FS>	60	<	92	\	124	|	156	ú	188	º	220	‹	252	¸
29	<GS>	61	=	93	]	125	}	157	ù	189	Ω	221	›	253	˝
30	<RS>	62	>	94	^	126	~	158	û	190	æ	222	fi	254	˛
31	<US>	63	?	95	_	127	<DEL>	159	ü	191	ø	223	fl	255	ˇ

What to do with these sequences?

Assembly

```
GGCGTCTATATCTCGGCTCTAGGCCCTCATTTTTT
GGCGTCTATATCT
GGCGTCTATATCTCG
            TATCTCGGCTCTAGG
            TATCTCAGCTCTAGGCC
            TATCTCAGCTCTAGGCCCTCA
                CTCGGCTCTAGGCCCTCATTTT
                    GGCTCTAGGCCCTCATTTTTT
                        CTCTAGGCCCTCATTTTTT
                           CTAGGCCCTCATTTTTT
```

Assembly

```
GGCGTCTATATCTCGGCTCTAGGCCCTCATTTTTT   Genome
GGCGTCTATATCT
GGCGTCTATATCTCG
            TATCTCGGCTCTAGG
            TATCTCAGCTCTAGGCC                 Reads
            TATCTCAGCTCTAGGCCCTCA
                CTCGGCTCTAGGCCCTCATTTT
                    GGCTCTAGGCCCTCATTTTTT
                        CTCTAGGCCCTCATTTTTT
                           CTAGGCCCTCATTTTTT
```

Assembly

```
GGCGTCTATATCTCGGCTCTAGGCCCTCATTTTTT   Reconstruct this…
GGCGTCTATATCT
GGCGTCTATATCTCG
            TATCTCGGCTCTAGG
            TATCTCAGCTCTAGGCC                 …from this
            TATCTCAGCTCTAGGCCCTCA
                CTCGGCTCTAGGCCCTCATTTT
                    GGCTCTAGGCCCTCATTTTTT
                        CTCTAGGCCCTCATTTTTT
                           CTAGGCCCTCATTTTTT
```

Assembly

??????????????????????????????????????? Reconstruct this…

GGCGTCTATATCT
GGCGTCTATATCTCG
TATCTCGGCTCTAGG
TATCTCAGCTCTAGGCC …from this
TATCTCAGCTCTAGGCCCTCA
CTCGGCTCTAGGCCCTCATTTT
GGCTCTAGGCCCTCATTTTTT
CTCTAGGCCCTCATTTTTT
CTAGGCCCTCATTTTTT

Assembly

```
GGCGTCTATATCTCGGCTCTAGGCCCTCATTTTTT
GGCGTCTATATCT
GGCGTCTATATCTCG
            TATCTCGGCTCTAGG
            TATCTCAGCTCTAGGCC
            TATCTCAGCTCTAGGCCCTCA
                CTCGGCTCTAGGCCCTCATTTT
                    GGCTCTAGGCCCTCATTTTTT
                        CTCTAGGCCCTCATTTTTT
                            CTAGGCCCTCATTTTTT
```

Coverage = amount of redundant information

Assembly

```
GGCGTCTATATCTCGGCTCTAGGCCCTCATTTTTT
GGCGTCTATATCT
GGCGTCTATATCTCG
            TATCTCGGCTCTAGG
            TATCTCAGCTCTAGGCC
            TATCTCAGCTCTAGGCCCTCA
                CTCGGCTCTAGGCCCTCATTTT
                    GGCTCTAGGCCCTCATTTTTT
                        CTCTAGGCCCTCATTTTTT
                            CTAGGCCCTCATTTTTT
```

Coverage = 5

Assembly

```
GGCGTCTATATCTCGGCTCTAGGCCCTCATTTTTT
GGCGTCTATATCT
GGCGTCTATATCTCG
            TATCTCGGCTCTAGG
            TATCTCAGCTCTAGGCC
            TATCTCAGCTCTAGGCCCTCA
                CTCGGCTCTAGGCCCTCATTTT
                  GGCTCTAGGCCCTCATTTTTT
                     CTCTAGGCCCTCATTTTTT
                        CTAGGCCCTCATTTTTT
```

Coverage = 5

Assembly

```
GGCGTCTATATCTCGGCTCTAGGCCCTCATTTTTT
GGCGTCTATATCT
GGCGTCTATATCTCG
            TATCTCGGCTCTAGG
            TATCTCAGCTCTAGGCC
            TATCTCAGCTCTAGGCCCTCA
                CTCGGCTCTAGGCCCTCATTTT
                    GGCTCTAGGCCCTCATTTTTT
                        CTCTAGGCCCTCATTTTTT
                           CTAGGCCCTCATTTTTT
```

Average Coverage = Length of all reads / length of genome

Assembly

```
GGCGTCTATATCTCGGCTCTAGGCCCTCATTTTTT   35 bases
GGCGTCTATATCT
GGCGTCTATATCTCG
            TATCTCGGCTCTAGG
            TATCTCAGCTCTAGGCC                 177 bases
            TATCTCAGCTCTAGGCCCTCA
                CTCGGCTCTAGGCCCTCATTTT
                    GGCTCTAGGCCCTCATTTTTT
                        CTCTAGGCCCTCATTTTTT
                           CTAGGCCCTCATTTTTT
```

Average Coverage = 177 / 35 = 5-fold

Assembly

```
TATCTCAGCTCTAGGCC
   |||  |||||||||
   CTCGGCTCTAGGCCCTCATTTT
```

Suffix of one read is very similar to the prefix of another!

1st Law of Assembly

If suffix of read A is similar to prefix of read B…

```
TATCTCAGCTCTAGGCC
   |||   |||||||||
   CTCGGCTCTAGGCCCTCATTTT
```

…then A and B might *overlap* in the genome

```
          TATCTCAGCTCTAGGCC
GGCGTCTATATCTCGGCTCTAGGCCCTCATTTTTT
                CTCGGCTCTAGGCCCTCATTTT
```

1st Law of Assembly

```
TATCTCAGCTCTAGGCC
   | | |   | | | | | | | |
   CTCGGCTCTAGGCCCTCATTTT
```

Why do we observe differences/mismatches?

1st Law of Assembly

```
TATCTCAGCTCTAGGCC
   | | |   | | | | | | | |
   CTCGGCTCTAGGCCCTCATTTT
```

Why do we observe differences/mismatches?

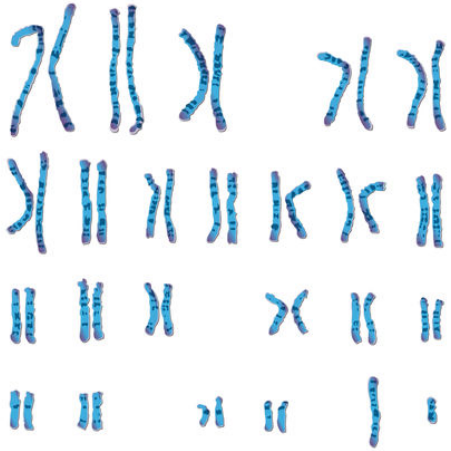
1. Sequencing errors

1st Law of Assembly

```
TATCTCAGCTCTAGGCC
   | | |   | | | | | | | |
   CTCGGCTCTAGGCCCTCATTTT
```

Why do we observe differences/mismatches?

1. Sequencing errors
2. Polyploidy



2nd Law of Assembly

More coverage leads to more and longer overlaps

```
                    CTCTAGGCCCTCATTTTTT
          TATCTCGGCTCTAGG
GGCGTCTATATCTCG
GGCGTCTATATCTCGGCTCTAGGCCCTCATTTTTT
GGCGTCTATATCT
GGCGTCTATATCTCG
          TATCTCGGCTCTAGG
          TATCTCAGCTCTAGGCCCTCA
              CTCGGCTCTAGGCCCTCATTTT
                  GGCTCTAGGCCCTCATTTTTT
                      CTCTAGGCCCTCATTTTTT
```

less coverage

more coverage

Greedy Assembly

GGCGTCTATATCT
GGCGTCTATATCTCG
TATCTCGGCTCTAGG
TATCTCAGCTCTAGGCCCTCA
CTCGGCTCTAGGCCCTCATTTT
GGCTCTAGGCCCTCATTTTTT
CTCTAGGCCCTCATTTTTT

Greedy Assembly

GGCGTCTATATCT
GGCGTCTATATCTCG
TATCTCGGCTCTAGG
TATCTCAGCTCTAGGCCCTCA
CTCGGCTCTAGGCCCTCATTTT
GGCTCTAGGCCCTCATTTTTT
CTCTAGGCCCTCATTTTTT

Given any read: find match and extend

Greedy Assembly

TATCTCAGCTCTAGGCCCTCA
    CTCGGCTCTAGGCCCTCATTTT

Given any read: find match and extend

Greedy Assembly

```
TATCTCAGCTCTAGGCCCTCA
    CTCGGCTCTAGGCCCTCATTTT
        GGCTCTAGGCCCTCATTTTTT
```

Given any read: find match and extend

Greedy Assembly

Prone to get stuck in local optima!

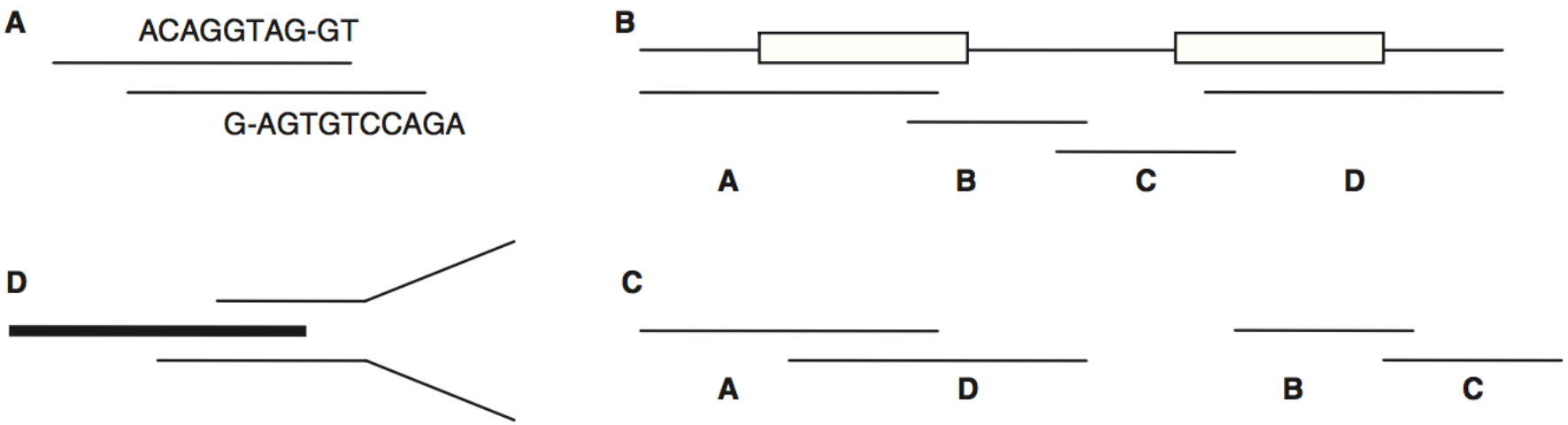


Figure I: (**A**) Overlap between two reads—note that agreement within overlapping region need not be perfect; (**B**) Correct assembly of a genome with two repeats (boxes) using four reads A–D; (**C**) Assembly produced by the greedy approach. Reads A and D are assembled first, incorrectly, because they overlap best and (**D**) Disagreement between two reads (thin lines) that could extend a contig (thick line), indicating a potential repeat boundary. Contig extension must be terminated in order to avoid misassemblies.

Pop (2009) Briefings in Bioinformatics

Overlap Layout Consensus (OLC)

3 Phases:

1) Search for read overlaps using short seeds (k-mers); extend seeds (sounds a bit like BLAST…)
2) Construct a graph of overlapping reads
3) Perform a multiple sequence alignment and calculate consensus sequence

Overlap Layout Consensus (OLC)

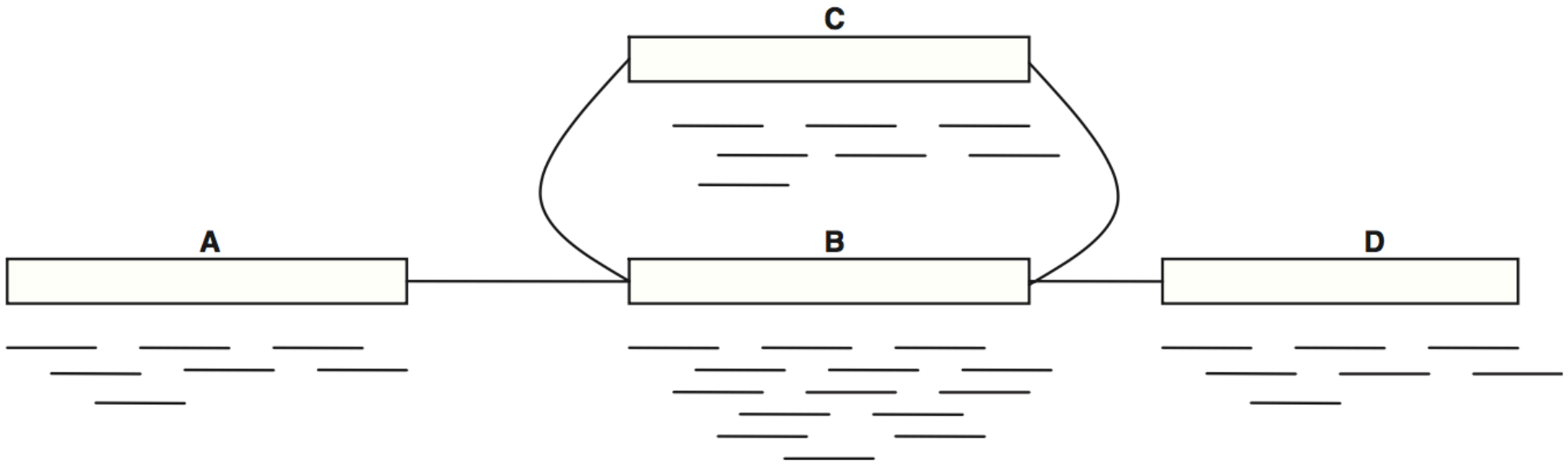


Figure 2: Overlap graph of a genome containing a two-copy repeat (B). Note the increased depth of coverage within the repeat. The correct reconstruction of this genome spells the sequence ABCBD, while conservative assembly approaches would lead to a fragmented reconstruction.

Pop (2009) Briefings in Bioinformatics

de Bruijn graph assemblers

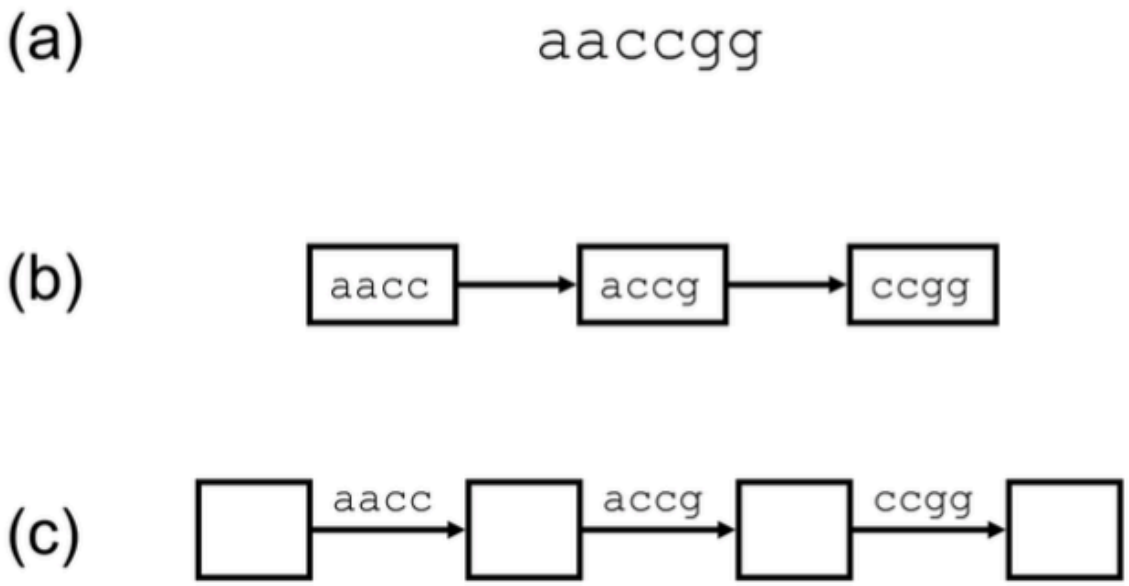


Figure 1.
A read represented by K-mer graphs. (a) The read is represented by two types of K-mer graph with K=4. Larger values of K are used for real data. (b) The graph has a node for every K-mer in the read plus a directed edge for every pair of K-mers that overlap by K-1 bases in the read. (c) An equivalent graph has an edge for every K-mer in the read and the nodes implicitly represent overlaps of K-1 bases. In these examples, the paths are simple because the value K=4 is larger than the 2bp repeats in the read. The read sequence is easily reconstructed from the path in either graph.

Miller *et al.* (2010) Genomics

de Bruijn graph assemblers

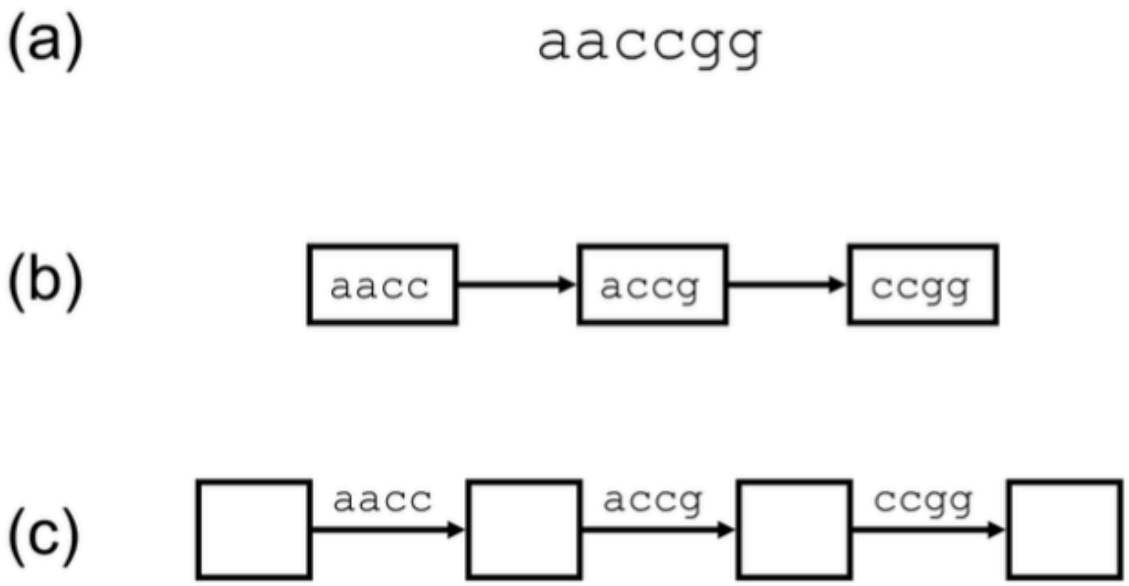


Figure 1.
A read represented by K-mer graphs. (a) The read is represented by two types of K-mer graph with K=4. Larger values of K are used for real data. (b) The graph has a node for every K-mer in the read plus a directed edge for every pair of K-mers that overlap by K-1 bases in the read. (c) An equivalent graph has an edge for every K-mer in the read and the nodes implicitly represent overlaps of K-1 bases. In these examples, the paths are simple because the value K=4 is larger than the 2bp repeats in the read. The read sequence is easily reconstructed from the path in either graph.

Every k-mer becomes a node

Two nodes are connected if they share a k-1-mer

Miller *et al.* (2010) Genomics

de Bruijn graph assemblers

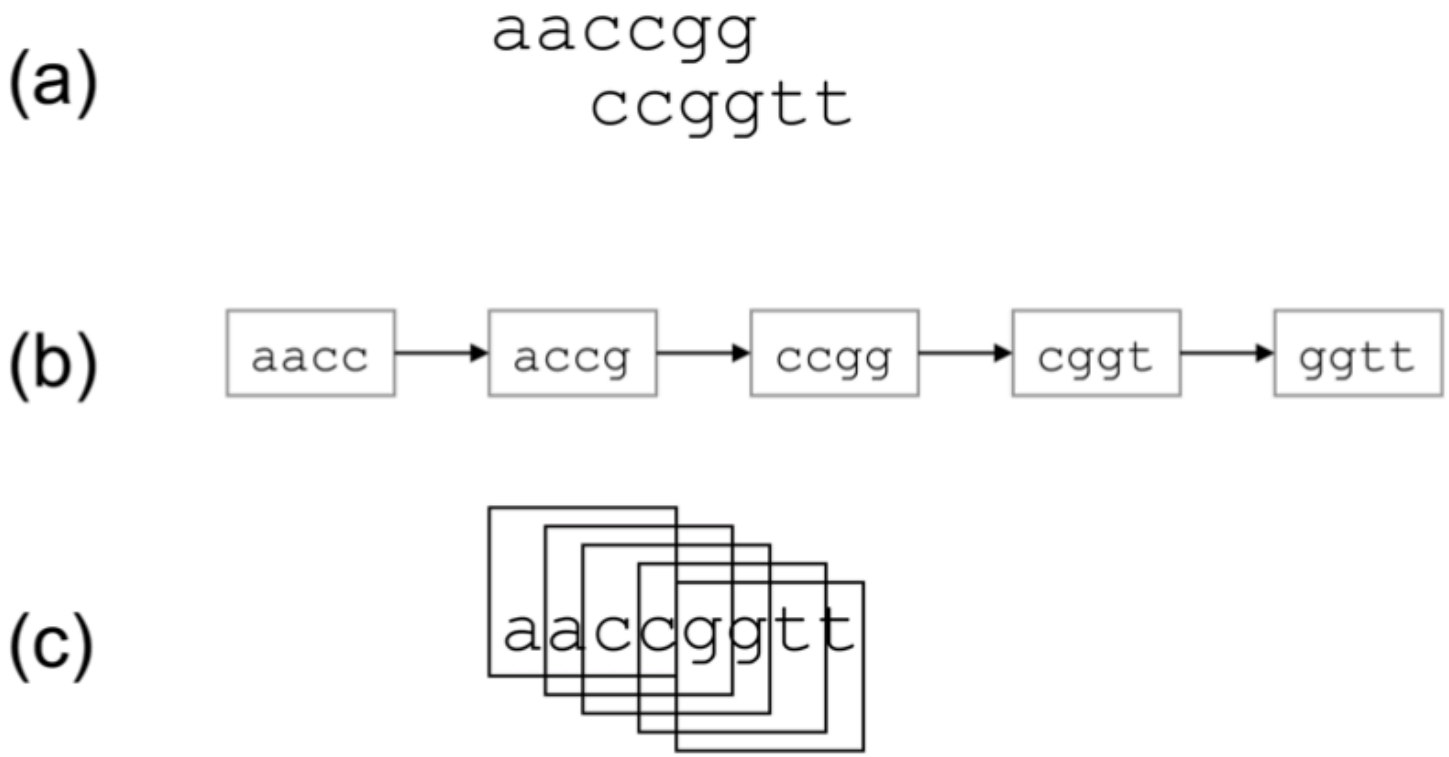


Figure 2.
A pair-wise overlap represented by a K-mer graph. (a) Two reads have an error-free overlap of 4 bases. (b) One K-mer graph, with K=4, represents both reads. The pair-wise alignment is a by-product of the graph construction. (c) The simple path through the graph implies a contig whose consensus sequence is easily reconstructed from the path.

Miller *et al.* (2010) Genomics

de Bruijn graph assemblers

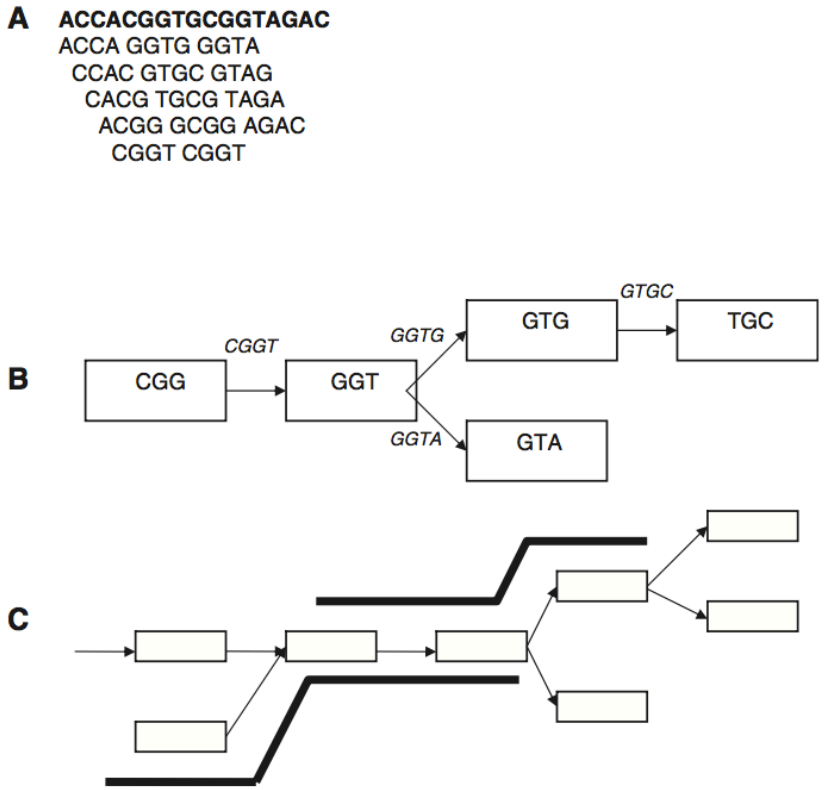


Figure 3: (**A**) *k*-mer spectrum of a DNA string (bold) for $k=4$; (**B**) Section of the corresponding deBruijn graph. The edges are labeled with the corresponding *k*-mer and (**C**) Overlap between two reads (bold) that can be inferred from the corresponding paths through the deBruijn graph.

Pop (2009) Briefings in Bioinformatics

de Bruijn graph assemblers

Why bother with a *de Bruijn* graph?

de Bruijn graph assemblers

Why bother with a *de Bruijn* graph?

This approach to assembly fits into small memory space by creating a hash table (think Python dictionary) to store k-mer/read associations.

Given error free data with reads that span across repeats, the *de Bruijn* graph would equal the k-mer graph.

de Bruijn graph assemblers

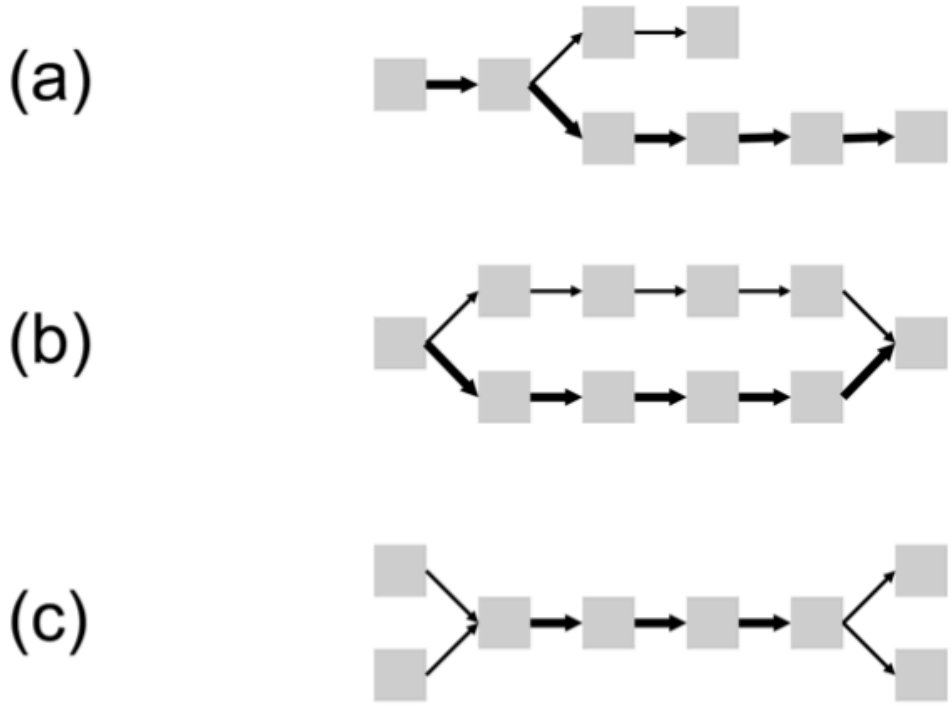


Figure 3.
Complexity in K-mer graphs can be diagnosed with read multiplicity information. In these graphs, edges represented in more reads are drawn with thicker arrows. (a) An errant base call toward the end of a read causes a "spur" or short dead-end branch. The same pattern could be induced by coincidence of zero coverage after polymorphism near a repeat. (b) An errant base call near a read middle causes a "bubble" or alternate path. Polymorphisms between donor chromosomes would be expected to induce a bubble with parity of read multiplicity on the divergent paths. (c) Repeat sequences lead to the "frayed rope" pattern of convergent and divergent paths.

Miller *et al.* (2010) Genomics

de Bruijn graph assemblers

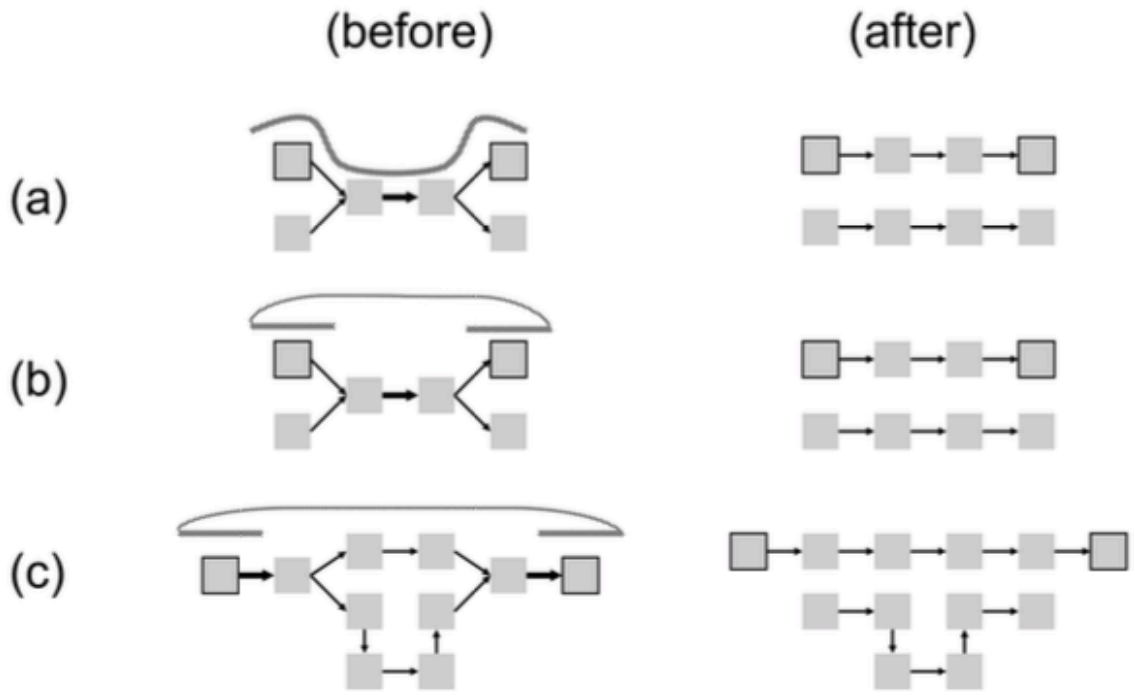


Figure 4.
Three methods to resolve graph complexity. (a) Read threading joins paths across collapsed repeats that are shorter than the read lengths. (b) Mate threading joins paths across collapsed repeats that are shorter than the paired-end distances. (c) Path following chooses one path if its length fits the paired-end constraint. Reads and mates are shown as patterned lines. Not all tangles can be resolved by reads and mates. The non-branching paths are illustrative; they could be simplified to single edges or nodes.

The k-mer graph is validated against the actual reads!

Miller *et al.* (2010) Genomics

de Bruijn graph assemblers

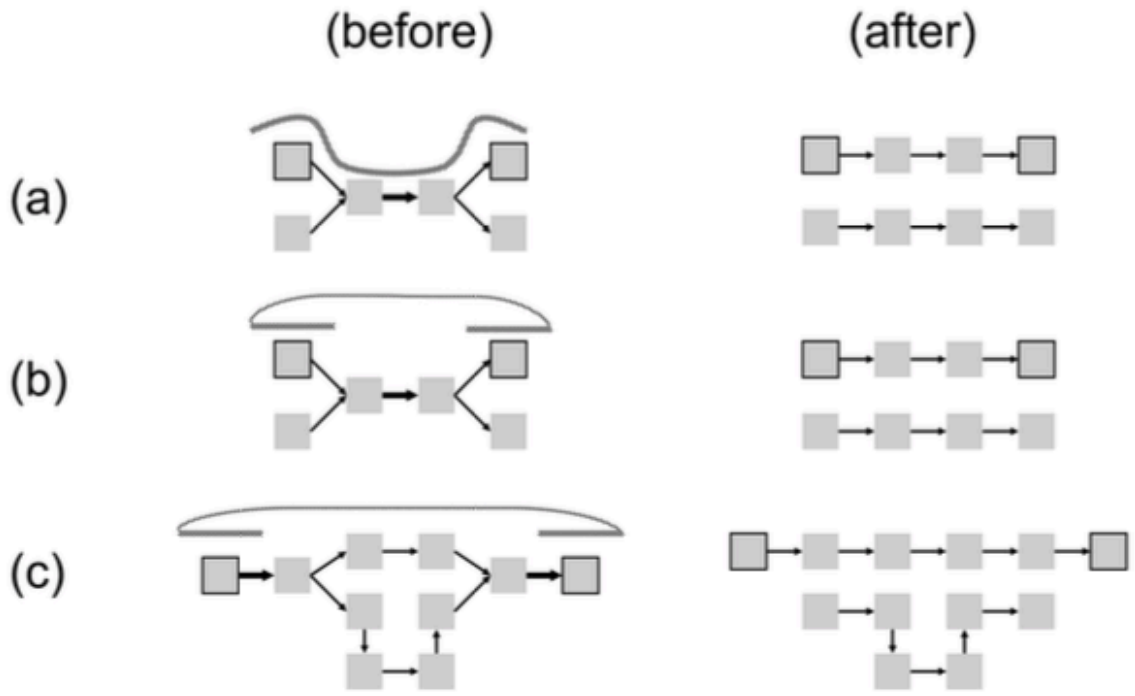
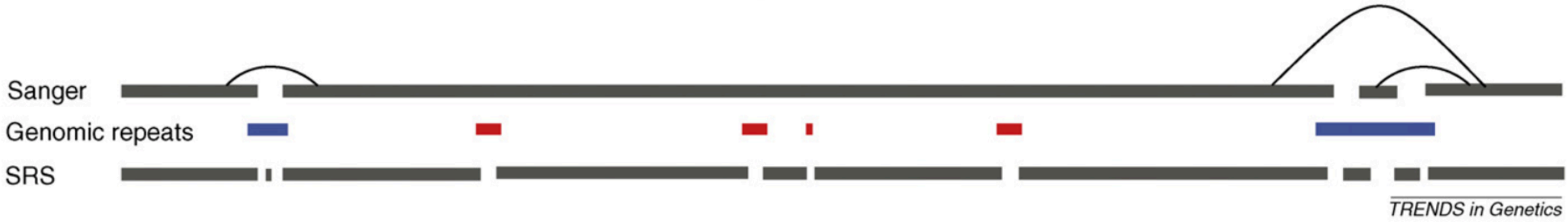


Figure 4.
Three methods to resolve graph complexity. (a) Read threading joins paths across collapsed repeats that are shorter than the read lengths. (b) Mate threading joins paths across collapsed repeats that are shorter than the paired-end distances. (c) Path following chooses one path if its length fits the paired-end constraint. Reads and mates are shown as patterned lines. Not all tangles can be resolved by reads and mates. The non-branching paths are illustrative; they could be simplified to single edges or nodes.

Contigs, Scaffolds, and Long Reads

Miller *et al.* (2010) Genomics

Short Reads and Their Drawbacks



Pop & Salzberg (2008) Trends in Genetics

Transcriptome Assembly

Genome assembly graphs should be long while transcriptome graphs should be short – chromosomes vs. transcripts.

Grabherr *et al.* (2013) Nat. Biotechnol.

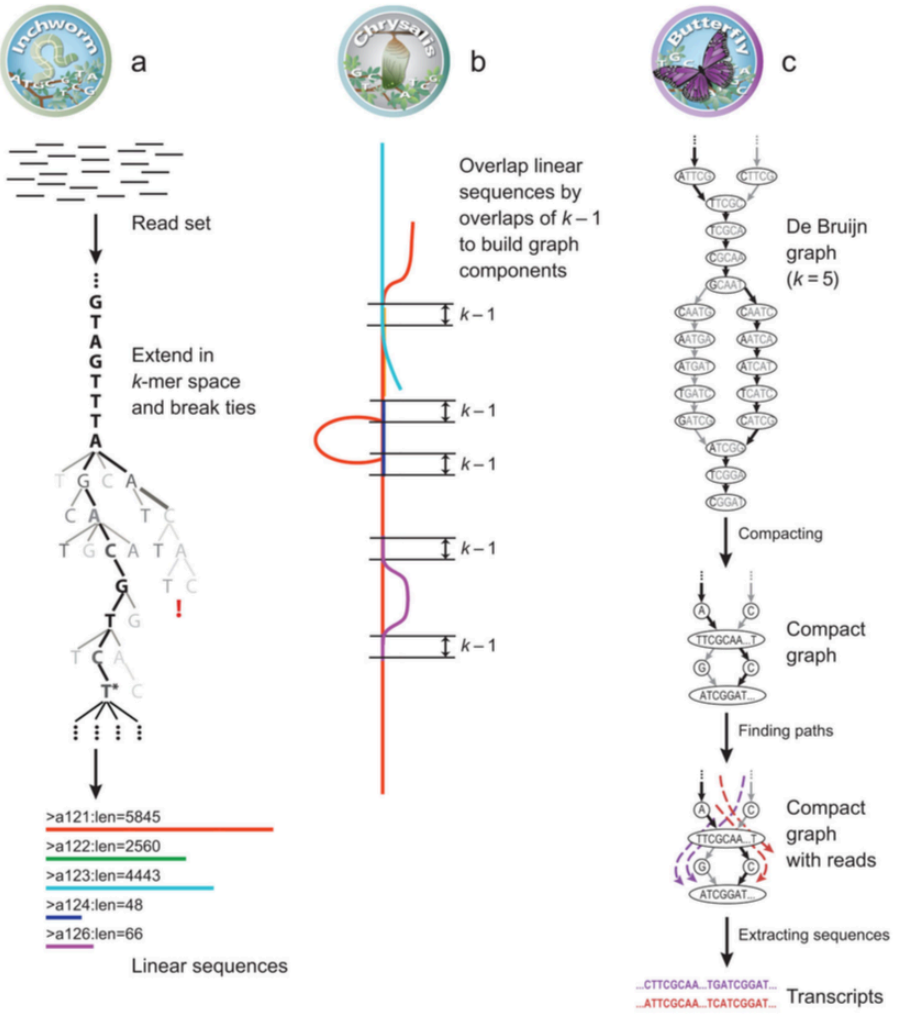


Figure 1. Overview of Trinity
(**a**) Inchworm assembles the read data set (short black line, top) by greedily searching for paths in a k-mer graph (middle), resulting in a collection of linear contigs (color lines, bottom), with each k-mer present only once in the contigs. (**b**) Chrysalis pools contigs if they share at least one k-1-mer and reads span the join, and builds individual de Bruijn graphs from each pool (colored lines). (**c**) Butterfly takes each de Bruijn graph from Chrysalis (top), and trims spurious edges and compacts linear paths (middle). It then reconciles the graph with reads (dashed colored arrows, bottom) and pairs (not shown), and outputs one linear sequence for each splice form and/or paralogous transcript reflected in the graph (bottom, colored sequences).

Transcriptome Assembly

Transcriptomes contain paralogs, orthologs, splice variants.

Grabherr *et al.* (2013) Nat. Biotechnol.

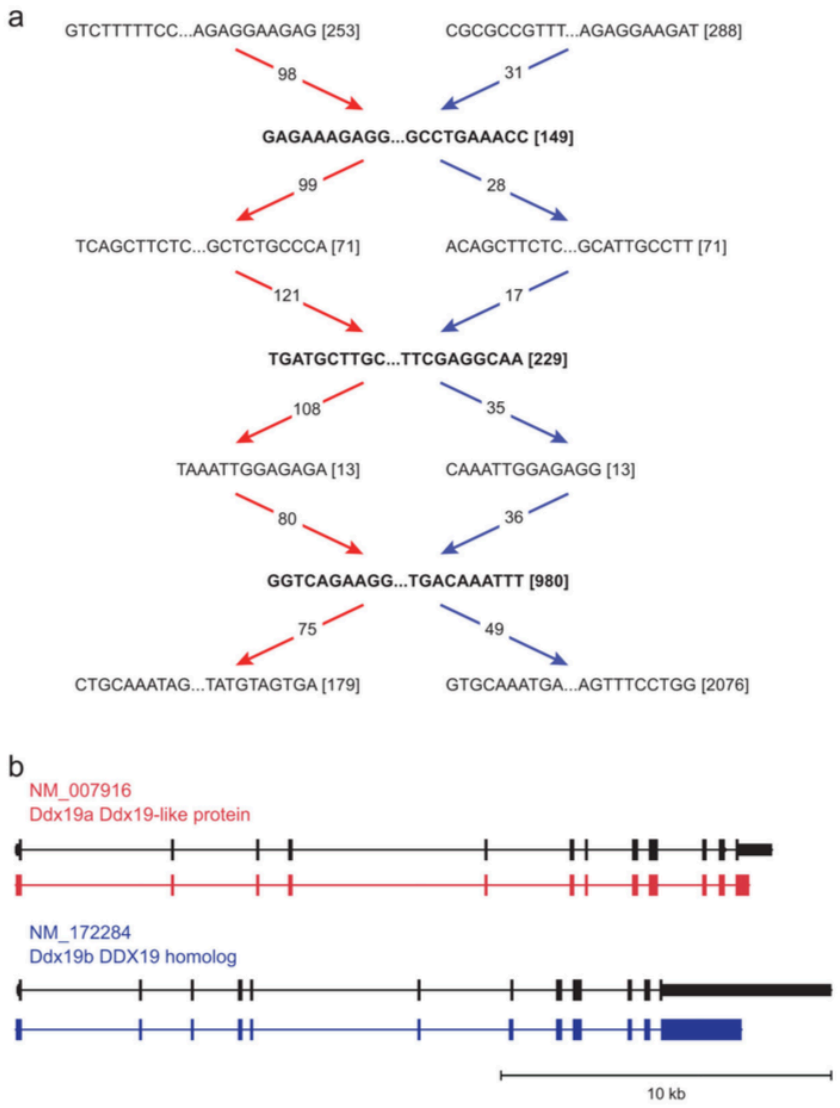


Figure 4. Trinity resolves closely paralogous genes
(**a**) Shown is the compacted component graph for two paralogous mouse genes, Ddx19a and Ddx19b (93% identity), highlighting the two paths (red and blue) chosen by Trinity out of the 64 possible paths in this portion alone. (**b**) Shown are the alignments between the transcripts represented by the red and blue paths in (**a**) and the paralogous genes Ddx19a and Ddx19b relative to the mouse reference genome (genome alignment shown for graphical clarity only; no alignments were used to generate the assemblies).

Next Time: Read Mapping

```
GGCGTCTATATCTCGGCTCTAGGCCCTCATTTTTT
GGCGTCTATATCT
GGCGTCTATATCTCG
              TATCTCGGCTCTAGG
              TATCTCAGCTCTAGGCC
              TATCTCAGCTCTAGGCCCTCA
                  CTCGGCTCTAGGCCCTCATTTT
                      GGCTCTAGGCCCTCATTTTTT
                          CTCTAGGCCCTCATTTTTT
                             CTAGGCCCTCATTTTTT
```