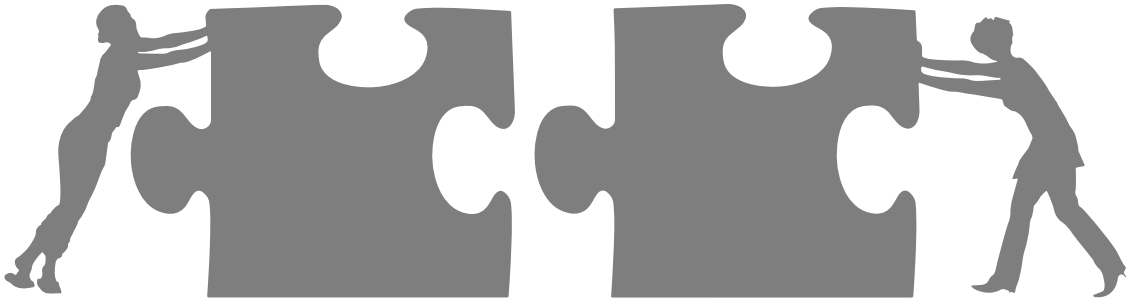
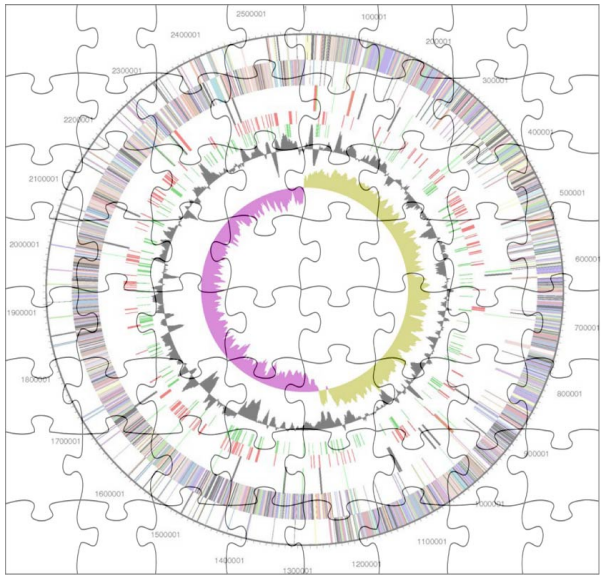
Principles of genome assembly
A journey through paradigms





http://contig.wordpress.com/2010/04/13/

UNIVERSITÉ LIBRE DE BRUXELLES

ULB

Jean-François Flot
Professor

Faculté des Sciences

Evolutionary Biology & Ecology
Av F.D. Roosevelt 50 - CP 160/12
B-1050 Brussels - Belgium
office UC.4.163
T +32 (0)2 650 40 14
F +32 (0)2 650 24 45
M jflot@ulb.be

ULB

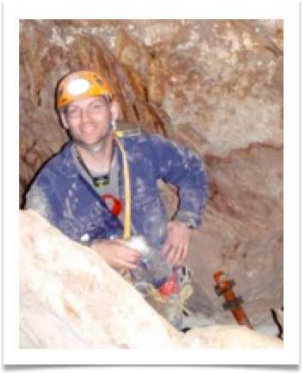
21.03.2025

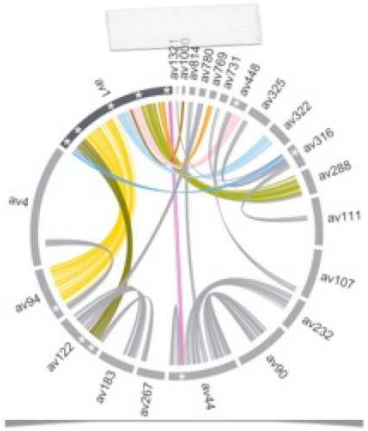
Research group “Evolutionary & Ecological Genomics” (EEG)

Group **Jean-François Flot** Ecological & evolutionary genomics

phone: +32 (0)2 650 40 14
e-mail: jflot@ulb.ac.be

http://ebe.ulb.be/ebe/Flot.html





Research interests

Molecular approaches to species delimitation - Molecular systematics - Genome assembly - Experimental evolution - Genome evolution - Bioinformatics

Research interests

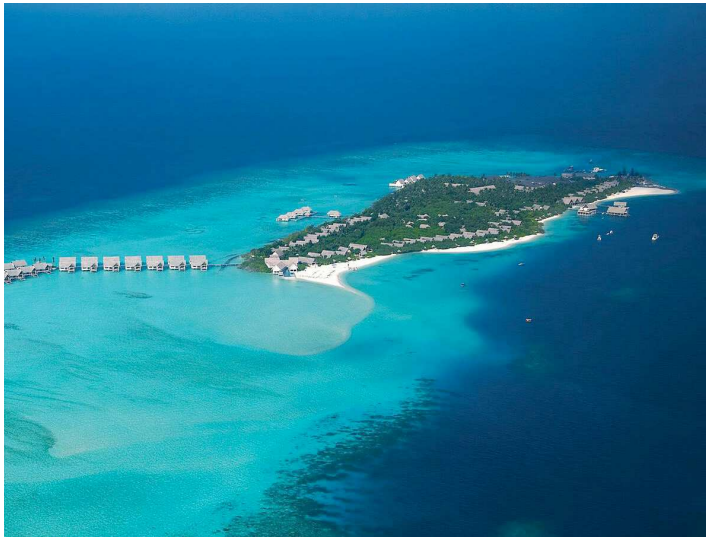
Horizontal axis: biodiversity, species delimitation, molecular taxonomy, genome size variations, environmental DNA

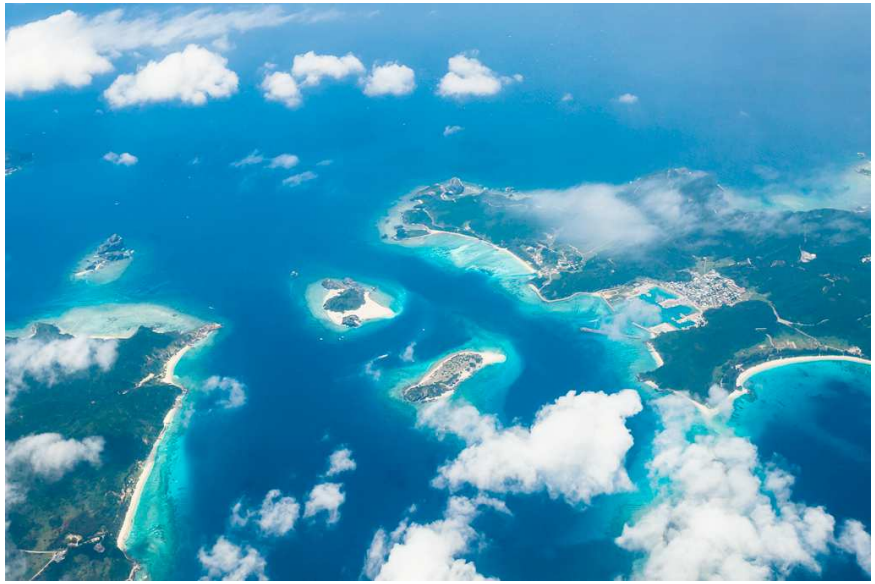
Vertical axis: genome evolution, metagenomics, computational genomics

Taxa: amphipods, ants, bacteria, corals, rotifers, marine worms, yeasts

Environments: caves, coral reefs, “extreme” environments, experimental evolution

Coral studies



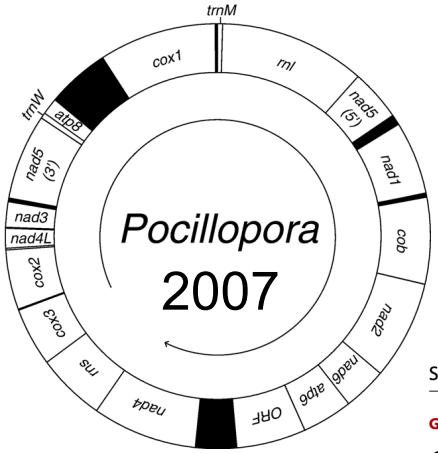


Cave studies

Niphargus gammariformis

Jean-François Flot (jflot@ulb.be)

Genome studies: animals



LETTER 2013

OPEN
doi:10.1038/nature12326

Genomic evidence for ameiotic evolution in the bdelloid rotifer *Adineta vaga*

Jean-François Flot[1,2,3,4,5,6], Boris Hespeels[1,2], Xiang Li[1,2], Benjamin Noel[3], Irina Arkhipova[7], Etienne G. J. Danchin[8,9,10], Andreas Hejnol[11], Bernard Henrissat[12], Romain Koszul[13], Jean-Marc Aury[3], Valérie Barbe[3], Roxane-Marie Barthélémy[14], Jens Bast[15], Georgii A. Bazykin[16,17], Olivier Chabrol[14], Arnaud Couloux[3], Martine Da Rocha[8,9,10], Corinne Da Silva[3], Eugene Gladyshev[7], Philippe Gouret[14], Oskar Hallatschek[6,18], Bette Hecox-Lea[7,19], Karine Labadie[3], Benjamin Lejeune[1,2], Oliver Piskurek[20], Julie Poulain[3], Fernando Rodriguez[7], Joseph F. Ryan[11], Olga A. Vakhrusheva[16,17], Eric Wajnberg[8,9,10], Bénédicte Wirth[14], Irina Yushenova[7], Manolis Kellis[21], Alexey S. Kondrashov[16,22], David B. Mark Welch[7], Pierre Pontarotti[14], Jean Weissenbach[3,4,5], Patrick Wincker[3,4,5], Olivier Jaillon[3,4,5,21]* & Karine Van Doninck[1,2]*

SCIENCE ADVANCES | RESEARCH ARTICLE

GENETICS 2021

Chromosome-level genome assembly reveals homologous chromosomes and recombination in asexual rotifer *Adineta vaga*

Paul Simion[1]*†, Jitendra Narayan[1†], Antoine Houtain[1], Alessandro Derzelle[1], Lyam Baudry[2,3], Emilien Nicolas[1,4], Rohan Arora[1,4], Marie Cariou[1,5], Corinne Cruaud[6], Florence Rodriguez Gaudray[7], Clément Gilbert[8], Nadège Guiglielmoni[7], Boris Hespeels[1], Djampa K. L. Kozlowski[9], Karine Labadie[6], Antoine Limasset[10], Marc Llirós[1,11], Martial Marbouty[2], Matthieu Terwagne[1], Julie Virgo[1], Richard Cordaux[12], Etienne G. J. Danchin[9], Bernard Hallet[13], Romain Koszul[2], Thomas Lenormand[14], Jean-Francois Flot[7,15]*, Karine Van Doninck[1,4]*

Schwager *et al. BMC Biology* (2017) 15:62
DOI 10.1186/s12915-017-0399-x

2017 BMC Biology

RESEARCH ARTICLE Open Access

The house spider genome reveals an ancient whole-genome duplication during arachnid evolution

Evelyn E. Schwager[1,2†], Prashant P. Sharma[3†], Thomas Clarke[4,5,6†], Daniel J. Leite[11], Torsten Wierschin[7†], Matthias Pechmann[8,9], Yasuko Akiyama-Oda[10,11], Lauren Esposito[12], Jesper Bechsgaard[13], Trine Bilde[13], Alexandra D. Buffry[1], Hsu Chao[14], Huyen Dinh[14], HarshaVardhan Doddapaneni[14], Shannon Dugan[14], Cornelius Eibner[15], Cassandra G. Extavour[16], Peter Funch[13], Jessica Garb[2], Luis B. Gonzalez[1], Vanessa L. Gonzalez[17], Sam Griffiths-Jones[18], Yi Han[14], Cheryl Hayashi[5,19], Maarten Hilbrant[1,9], Daniel S. T. Hughes[14], Ralf Janssen[20], Sandra L. Lee[14], Ignacio Maeso[21], Shwetha C. Murali[14], Donna M. Muzny[14], Rodrigo Nunes da Fonseca[22], Christian L. B. Paese[1], Jiaxin Qu[14], Matthew Ronshaugen[18], Christoph Schomburg[8], Anna Schönauer[1], Angelika Stollewerk[23], Montserrat Torres-Oliva[8], Natascha Turetzek[8], Bram Vanthournout[13,24], John H. Werren[25], Carsten Wolff[26], Kim C. Worley[14], Gregor Bucher[27*], Richard A. Gibbs[14*], Jonathan Coddington[17*], Hiroki Oda[10,28*], Mario Stanke[7*], Nadia A. Ayoub[4*], Nikola-Michael Prpic[8*], Jean-François Flot[29*], Nico Posnien[8*], Stephen Richards[14*] and Alistair P. McGregor[1*]

ORIGINAL ARTICLE 2022

Evolutionary Applications WILEY

Chromosomal assembly of the flat oyster (*Ostrea edulis* L.) genome as a new genetic resource for aquaculture

Isabelle Boutet[1] | Homère J. Alves Monteiro[2] | Lyam Baudry[3] | Takeshi Takeuchi[4] | Eric Bonnivard[1] | Bernard Billoud[5] | Sarah Farhat[6] | Ricardo Gonzales-Araya[7] | Benoit Salaun[7] | Ann C. Andersen[1] | Jean-Yves Toullec[1] | François H. Lallier[1] | Jean-François Flot[8] | Nadège Guiglielmoni[8] | Ximing Guo[9] | Cui Li[10] | Bassem Allam[6] | Emmanuelle Pales-Espinosa[6] | Jakob Hemmer-Hansen[2] | Pierrick Moreau[3] | Martial Marbouty[3] | Romain Koszul[3] | Arnaud Tanguy[1]

Peer Community Journal 2022

Section: Genomics

RESEARCH ARTICLE

Published
2022-07-18

Cite as
Hugo Darras, Natalia De Souza Araujo, Lyam Baudry, Nadège Guiglielmoni, Pedro Lorite, Martial Marbouty, Fernando Rodriguez, Irina Arkhipova, Romain Koszul, Jean-François Flot and Serge Aron (2022) *Chromosome-level genome assembly and annotation of two lineages of the ant* Cataglyphis hispanica*: stepping stones towards genomic studies of hybridogenesis and thermal adaptation in desert ants*, Peer Community Journal, 2: e40.

Correspondence
hgdarras@gmail.com

Chromosome-level genome assembly and annotation of two lineages of the ant *Cataglyphis hispanica*: stepping stones towards genomic studies of hybridogenesis and thermal adaptation in desert ants

Hugo Darras[#,1,2], Natalia De Souza Araujo[#,3,2], Lyam Baudry[4,5], Nadège Guiglielmoni[2,6], Pedro Lorite[7], Martial Marbouty[5], Fernando Rodriguez[8], Irina Arkhipova[8], Romain Koszul[5], Jean-François Flot[3,2], and Serge Aron[2]

Farhat *et al. BMC Genomics* (2022) 23:192
https://doi.org/10.1186/s12864-021-08262-1

2022 BMC Genomics

RESEARCH ARTICLE Open Access

Comparative analysis of the *Mercenaria mercenaria* genome provides insights into the diversity of transposable elements and immune molecules in bivalve mollusks

Sarah Farhat[1], Eric Bonnivard[2], Emmanuelle Pales Espinosa[1], Arnaud Tanguy[2], Isabelle Boutet[2], Nadège Guiglielmoni[3], Jean-François Flot[3,4] and Bassem Allam[1*]

Peer Community Journal 2024

Section: Genomics

Research article

Published
2024-03-12

Cite as
Klara Eleftheriadi, Nadège Guiglielmoni, Judit Salces-Ortiz, Carlos Vargas-Chavez, Gemma I. Martínez-Redondo, Marta Gut, Jean-François Flot, Andreas Schmidt-Rhaesa and Rosa Fernández (2024) *The genome sequence of the Montseny horsehair worm, Gordionus montsenyensis sp. nov., a key resource to investigate Ecdysozoa evolution*, Peer Community Journal, 4: e32.

Correspondence
rosa.fernandez@ibe.upf-csic.es

The genome sequence of the Montseny horsehair worm, *Gordionus montsenyensis* sp. nov., a key resource to investigate Ecdysozoa evolution

Klara Eleftheriadi[#,1], Nadège Guiglielmoni[#,2], Judit Salces-Ortiz[1], Carlos Vargas-Chavez[1], Gemma I. Martínez-Redondo[1], Marta Gut[3], Jean-François Flot[4,5], Andreas Schmidt-Rhaesa[6], and Rosa Fernández[1]

Volume 4 (2024), article e32

Genome studies: animals

scientific **data** 2024

OPEN

DATA DESCRIPTOR

Chromosome-level genome assembly of the bay scallop *Argopecten irradians*

Denis Grouzdev[1], Emmanuelle Pales Espinosa[1], Stephen Tettelbach[2], Sarah Farhat[1,3], Arnaud Tanguy[4], Isabelle Boutet[4], Nadège Guiglielmoni[5], Jean-François Flot[5,6], Harrison Tobi[2] & Bassem Allam[1]

JOURNAL ARTICLE CORRECTED PROOF

Genomic comparison of the temperate coral *Astrangia poculata* with tropical corals yields insights into winter quiescence, innate immunity, and sexual reproduction 2025

Kathryn H Stankiewicz, Nadège Guiglielmoni, Sheila A Kitchen, Jean-François Flot, Katie L Barott, Sarah W Davies, John R Finnerty, Sean P Grace, Leslie S Kaufman, Hollie M Putnam ... Show more
Author Notes

G3 Genes|Genomes|Genetics, jkaf033, https://doi.org/10.1093/g3journal/jkaf033
Published: 18 February 2025 **Article history**

Cold Spring Harbor Laboratory bioRχiv THE PREPRINT SERVER FOR BIOLOGY (submitted)

New Results Follow this preprint

Chromosome-level genome assembly and annotation of the chaetognath *Flaccisagitta enflata*

Nadège Guiglielmoni, Michael Eitel, Pierrick Moreau, Stefan Krebs, Mark Vermeij, Romain Koszul, Jean-François Flot

doi: https://doi.org/10.1101/2025.03.14.643333

Cold Spring Harbor Laboratory bioRχiv THE PREPRINT SERVER FOR BIOLOGY (submitted)

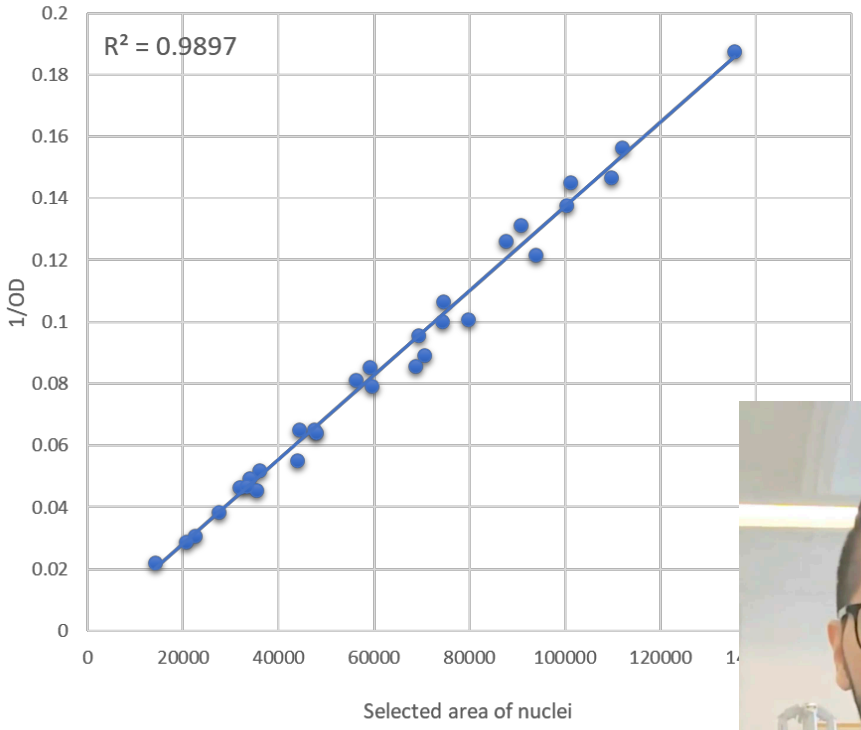
New Results Follow this preprint

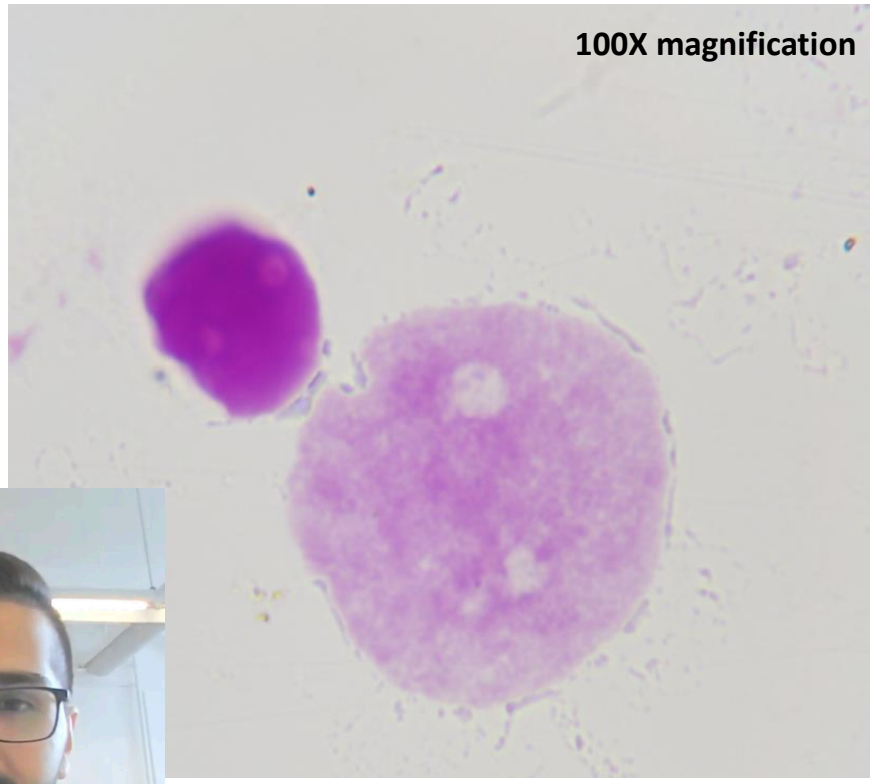
First chromosome-level genome assembly of the colonial tunicate *Botryllus schlosseri*

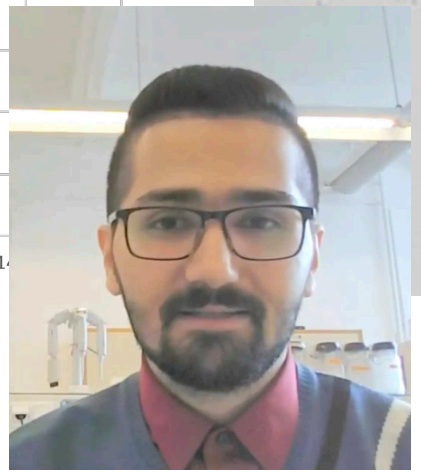
Olivier De Thier, Mohammed M.Tawfeeq, Roland Faure, Marie Lebel, Philippe Dru, Simon Blanchoud, Alexandre Alié, Federico D. Brown, Jean-François Flot, Stefano Tiozzo

doi: https://doi.org/10.1101/2024.05.29.594498

A refined Feulgen protocol to measure genome sizes



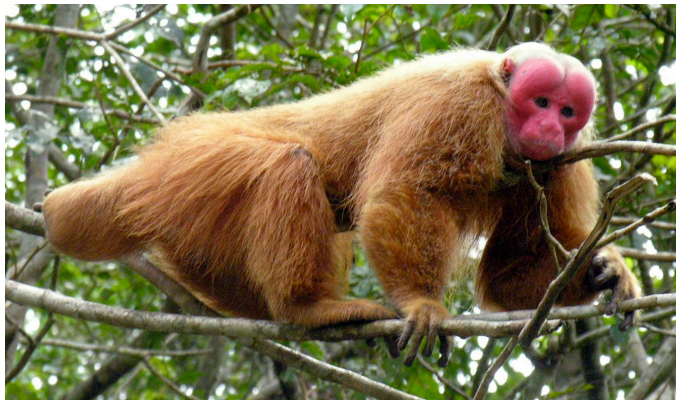




PhD student Mohammed M.Tawfeeq

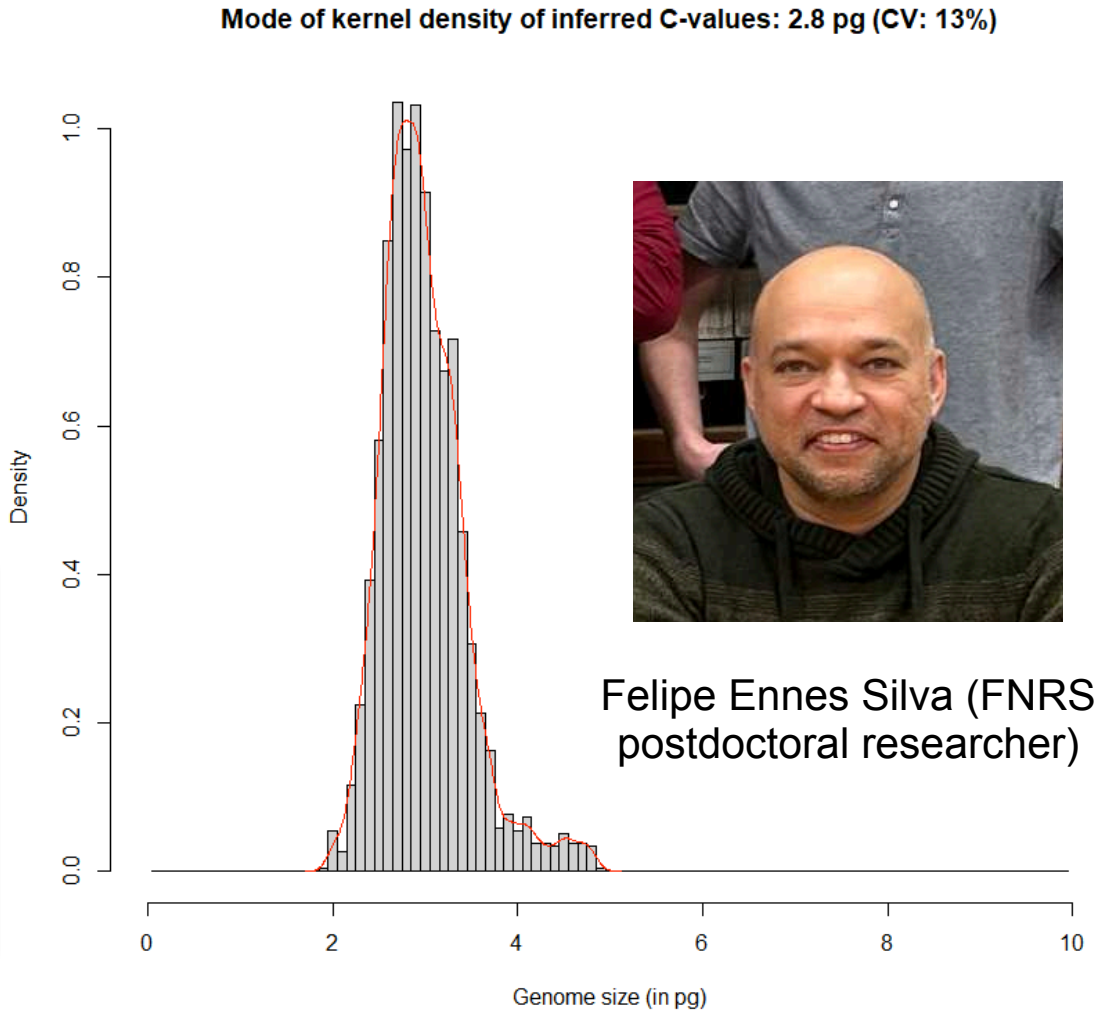
Jean-François Flot (jflot@ulb.be)

Are the resulting measurements accurate?



Cacajao rubicundus
Photo by Evgenia Kononova (public domain)

Total length:	2693375772
Fragments:	3165
Fragments N50:	11957981
Largest frg:	53347725
Scaffolds:	0
Mean coverage:	16



Felipe Ennes Silva (FNRS postdoctoral researcher)

Genome studies: bacteria

Rodriguez Jimenez *et al. BMC Genomics* (2022) 23:618
https://doi.org/10.1186/s12864-022-08842-9

2022 BMC Genomics

RESEARCH Open Access

Comparative genome analysis of *Vagococcus fluvialis* reveals abundance of mobile genetic elements in sponge-isolated strains

Ana Rodriguez Jimenez[1,2*], Nadège Guiglielmoni[2], Lise Goetghebuer[1], Etienne Dechamps[1], Isabelle F. George[1,3] and Jean-François Flot[2,4]

nature microbiology

2022

Article https://doi.org/10.1038/s41564-022-01252-3

Methanotrophy by a *Mycobacterium* species that dominates a cave microbial ecosystem

Received: 23 June 2021
Accepted: 14 September 2022
Published online: 3 November 2022
Check for updates

Rob J. M. van Spanning[1], Qingtian Guan[2], Chrats Melkonian[1,13,14], James Gallant[3], Lubos Polerecky[4], Jean-François Flot[5], Bernd W. Brandt[6], Martin Braster[1], Paul Iturbe Espinoza[1], Joost W. Aerts[1], Marion M. Meima-Franke[7], Sander R. Piersma[8], Catalin M. Bunduc[1], Roy Ummels[3], Arnab Pain[2], Emily J. Fleming[8], Nicole N. van der Wel[10], Vasile D. Gherman[11], Serban M. Sarbu[9,12], Paul L. E. Bodelier[7] and Wilbert Bitter[1,3]

www.nature.com/ismej

2023

ISME

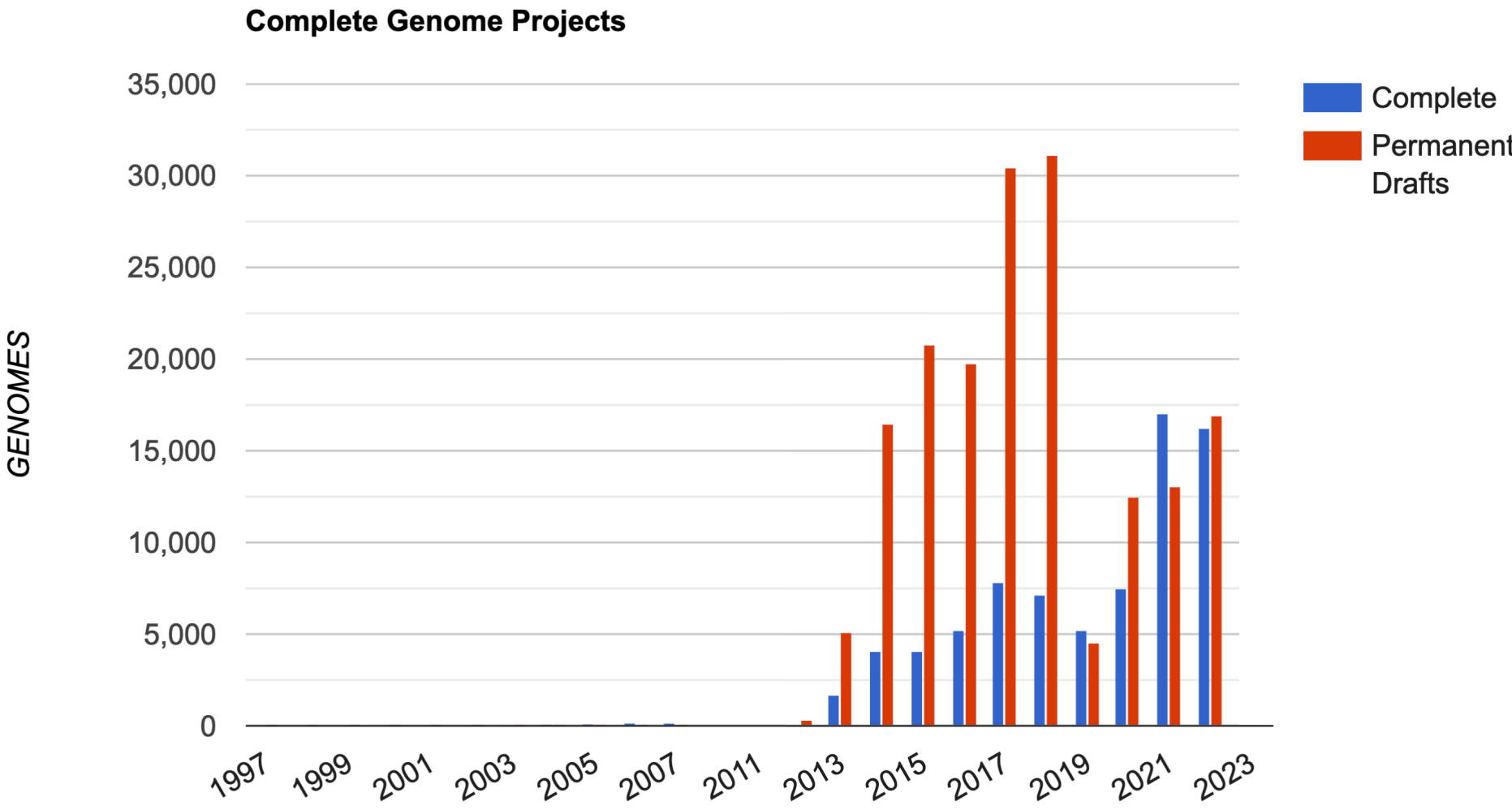
ARTICLE OPEN

Check for updates

Cave *Thiovulum* (*Candidatus* Thiovulum stygium) differs metabolically and genomically from marine species

Mina Bizic[1,2,16], Traian Brad[3,16], Danny Ionescu[1,2,16], Lucian Barbu-Tudoran[4], Luca Zoccarato[1,5], Joost W. Aerts[6], Paul-Emile Contarini[7,8], Olivier Gros[7], Jean-Marie Volland[8,9], Radu Popa[10], Jessica Ody[11], Daniel Vellone[12], Jean-François Flot[11,13], Scott Tighe[12] and Serban M. Sarbu[14,15]

Complete genome assemblies are finally becoming usual

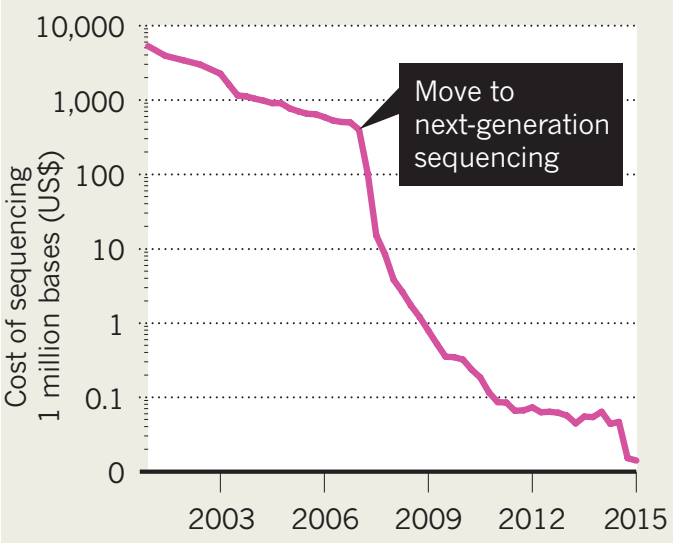


https://gold.jgi.doe.gov/statistics

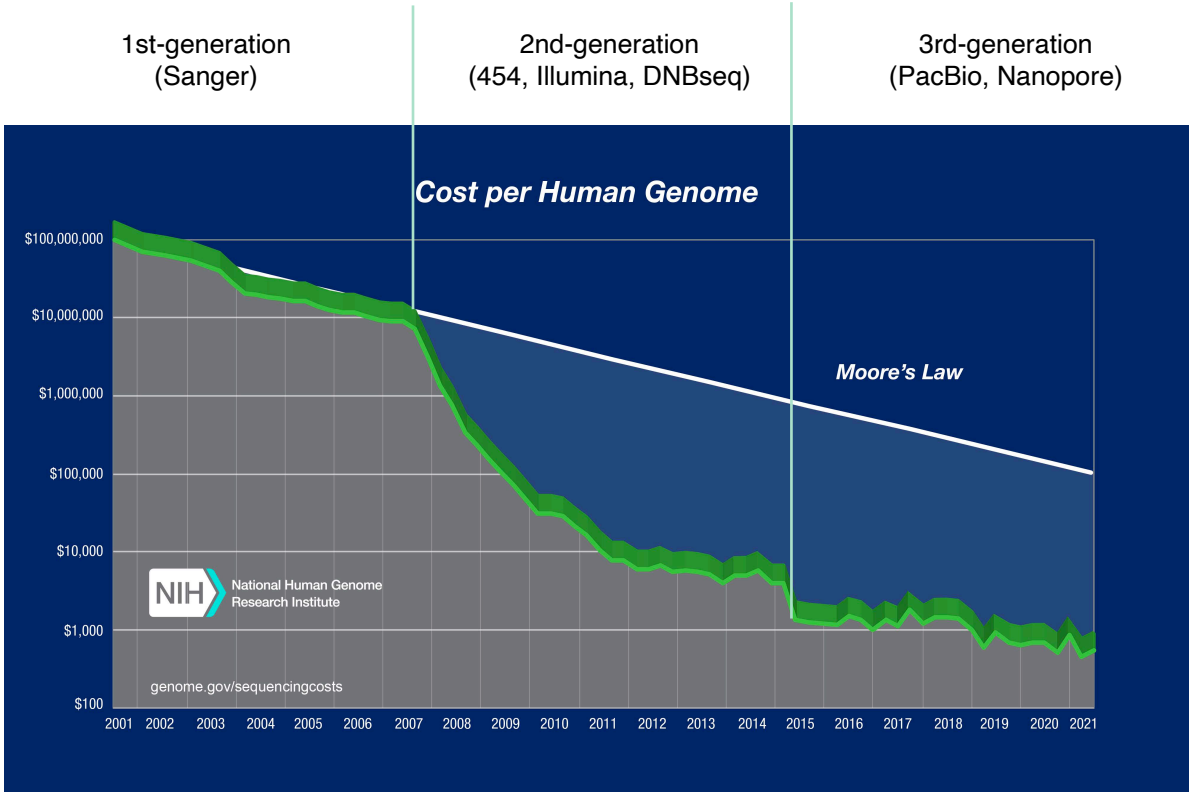
What developments have made this possible?

Reads: the raw material of genomics

"reads": any sequence that comes out of a sequencing machine



Scott (2016) Technology: Read the instructions. *Nature* 537:S54–S56



https://www.genome.gov/about-genomics/fact-sheets/Sequencing-Human-Genome-cost

What does it mean to “sequence a genome”?

1) generate a set of reads covering it nearly completely

Key term: *coverage*. Usually it’s short for *average coverage*: the average number of reads covering a position in the genome.

```
                CTAGGCCCTCAATTTTT
              CTCTAGGCCCTCAATTTTT
            GGCTCTAGGCCCTCATTTTTT
          CTCGGCTCTAGCCCCTCATTTT
        TATCTCGACTCTAGGCCCTCA          177 nucleotides
        TATCTCGACTCTAGGCC
     TCTATATCTCGGCTCTAGG
GGCGTCTATATCTCG
GGCGTCGATATCT
GGCGTCTATATCT
GGCGTCTATATCTCGGCTCTAGGCCCTCATTTTTT    35 nucleotides
```

Average coverage = 177 / 35 ≈ 7x

http://www.langmead-lab.org/teaching-materials/

The probability that a given base of the genome was not sequenced is given by

$$P_0 = e^{-c}$$

where c is the fold coverage and is given by

$$c = \frac{LN}{G}$$

and where LN is the number of bases sequenced, L being the read length and N the number of reads, G is the target sequence length, and e is the constant ~2.718. These results show that to achieve an error rate of 1 in 10,000 (0.01%), it is theoretically necessary to obtain ninefold coverage of the genome. With fivefold coverage, an error rate of 0.6% is expected.

Fold Coverage	P_0	Percent Not Sequenced	Percent Sequenced
0.25	$e^{-0.25} = 0.78$	78	22
0.5	$e^{-0.5} = 0.61$	61	39
0.75	$e^{-0.75} = 0.47$	47	53
1	$e^{-1} = 0.37$	37	63
2	$e^{-2} = 0.135$	13.5	87.5
3	$e^{-3} = 0.05$	5	95
4	$e^{-4} = 0.018$	1.8	98.2
5	$e^{-5} = 0.0067$	0.6	99.4
6	$e^{-6} = 0.0025$	0.25	99.75
7	$e^{-7} = 0.0009$	0.09	99.91
8	$e^{-8} = 0.0003$	0.03	99.97
9	$e^{-9} = 0.0001$	0.01	99.99
10	$e^{-10} = 0.000045$	0.005	99.995

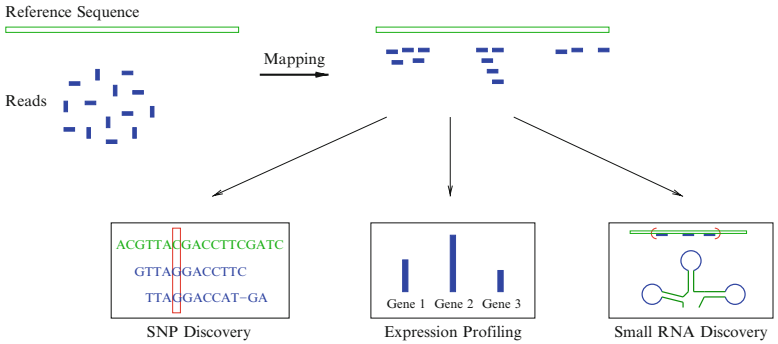
Source: Adapted from ► http://www.genome.ou.edu/poisson_calc.html and Lander and Waterman (1988).

Pevzner (2009) *Bioinformatics and Functional Genomics*

What does it mean to “sequence a genome”?

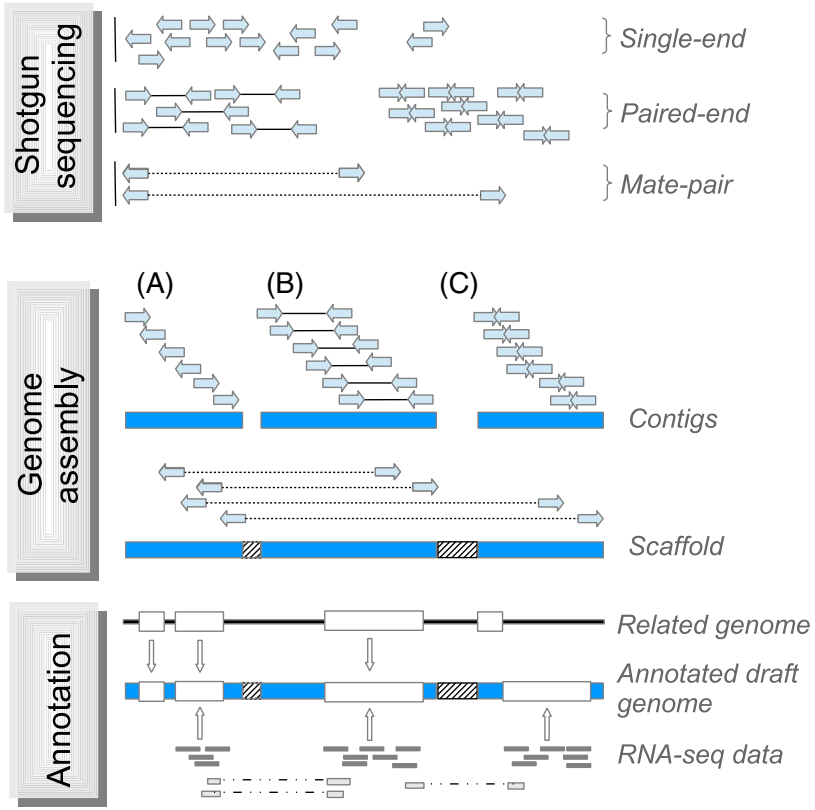
2) assemble the reads = reconstitute, from the set of reads, the suite of bases (A,T,G and C) characteristic of this genome

a) mapping assembly (by aligning the reads to a reference genome)



Nagarajan & Pop (2010) In: Fenyö (ed.) *Computational Biology*

b) *de novo* assembly (without the help of a reference sequence)



Ekblom & Wolf (2014) *Evolutionary Applications*

A contig = a set of overlapping reads, from which a contiguous consensus sequence can be obtained (http://staden.sourceforge.net/contig.html)
Staden (1980) *Nucleic Acids Research* 8: 3673-3694

A scaffold = a list of ordered and oriented contigs
Roach et al. (1995) *Genomics* 26: 345-353

Roadmap for today

1. the basic principle: looking for overlaps
2. greedy approaches
3. overlap-layout-consensus (OLC) approaches
4. de Bruijn graph-based approaches

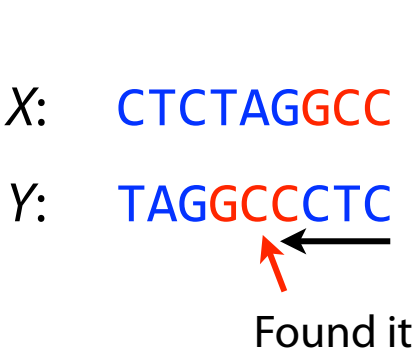
The basic idea behind assembly: looking for overlaps

Overlap: length-l suffix of X matches length-l prefix of Y, where l is given

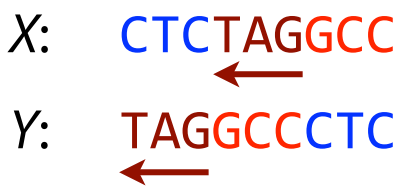
Simple idea: look in Y for occurrences of length-l suffix of X. Extend matches to the left to confirm whether entire prefix of Y matches.

Say $l = 3$

Look for this in Y, going right-to-left

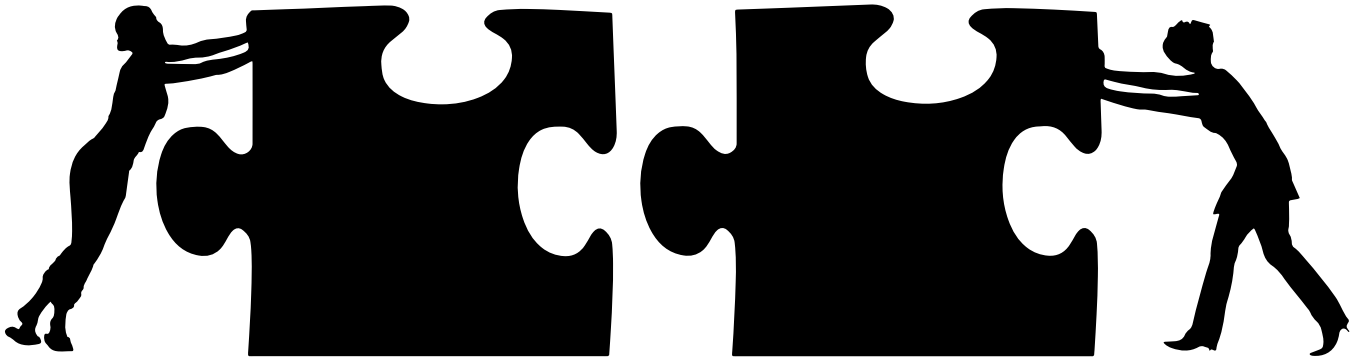


Extend to left; in this case, we confirm that a length-6 prefix of Y matches a suffix of X



http://www.langmead-lab.org/teaching-materials/

The basic idea behind assembly: looking for overlaps



In French: “chevauchements”

Because the DNA molecule is oriented, “A overlaps B” is not equivalent to “B overlaps A”!

Morris & de Groot (2009) *Lucky Luke 48: Le Bandit Manchot*

Overlap graphs

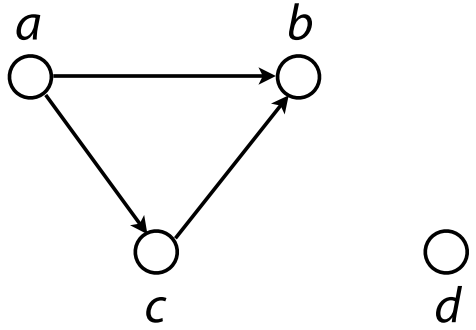
Some terminology:

Directed graph $G(V, E)$ consists of set of *vertices, V* and set of *directed edges, E*

Directed edge is an *ordered pair* of vertices.
First is the *source*, second is the *sink*.

Vertex is drawn as a circle

Edge is drawn as a line with an arrow connecting two circles



Vertex also called *node* or *point*

Edge also called *arc* or *line*

Directed graph also called *digraph*

$V = \{ a, b, c, d \}$

$E = \{ (a, b), (a, c), (c, b) \}$

Source Sink

http://www.langmead-lab.org/teaching-materials/

Overlap graphs

An overlap graph is a directed graph. The reads are represented by the vertices of the graph, and an edge s1 → s2 is present if there is an overlap between the reads s1 and s2.

Below: overlap graph, where an overlap is a suffix/prefix match of at least 3 characters

A vertex is a read, a directed edge is an overlap between suffix of source and prefix of sink

Vertices (reads): { *a:* CTCTAGGCC, *b:* GCCCTCAAT, *c:* CAATTTTT }

Edges (overlaps): { (*a*, *b*), (*b*, *c*) }


```
CTCTAGGCC          GCCCTCAAT
   |||                ||||
GCCCTCAAT          CAATTTTT
```

http://www.langmead-lab.org/teaching-materials/

Exercise 1: finding overlaps

Build a graph representing all the (perfect) overlaps of a minimum of 3 bases between the following reads:

ATTATAT CGCGTAC ATTGCGC GCATTAT ACGGCGC TATATTG GTACGGC GCGTACG ATATTGC

Hint: your final graph should comprise a total of 14 such overlaps.

Exercise 1: finding overlaps

CGCGTAC

ATTATAT

ATTGCGC

GCATTAT

TATATTG

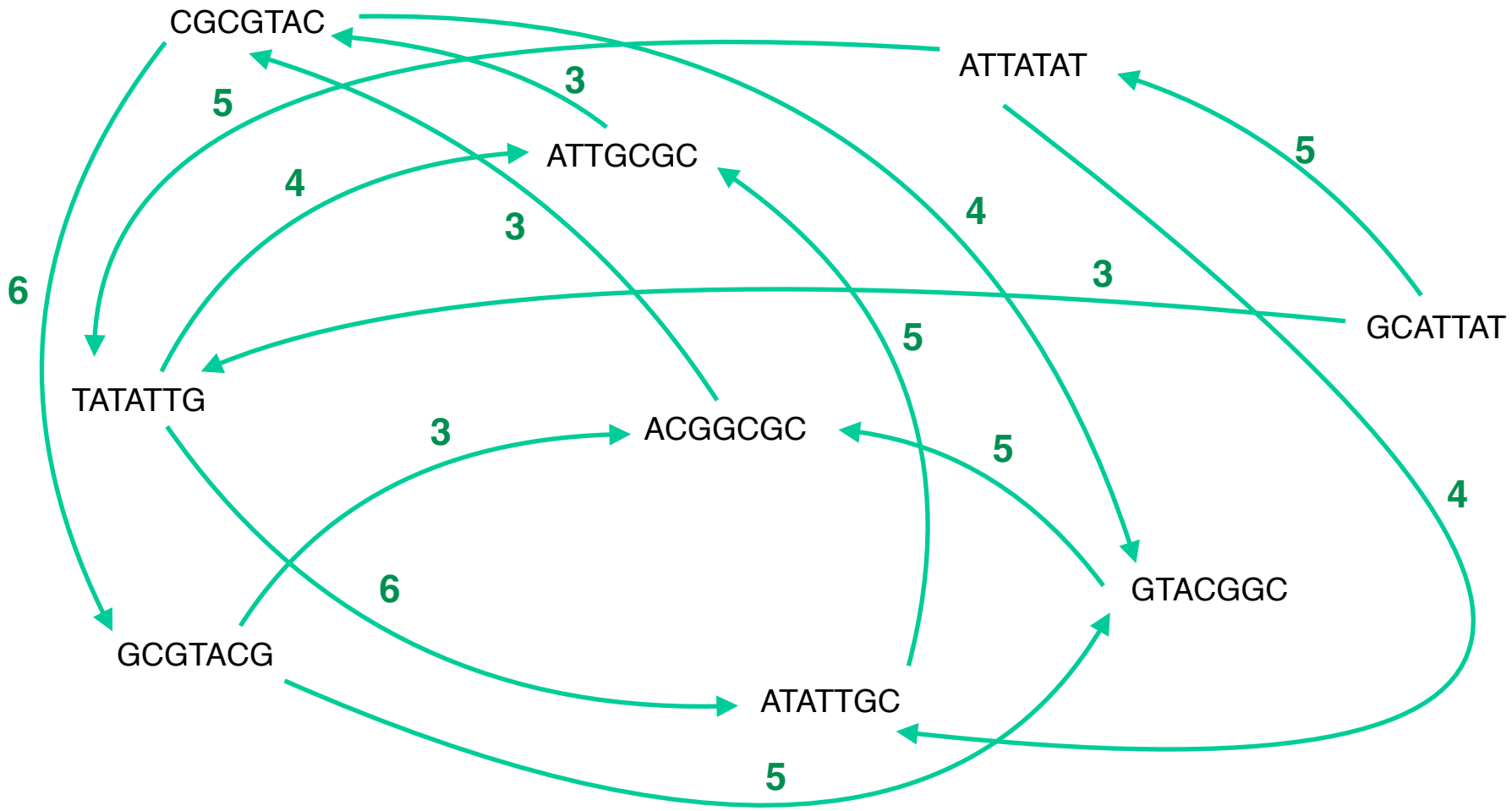
ACGGCGC

GTACGGC

GCGTACG

ATATTGC

Exercise 1: solution



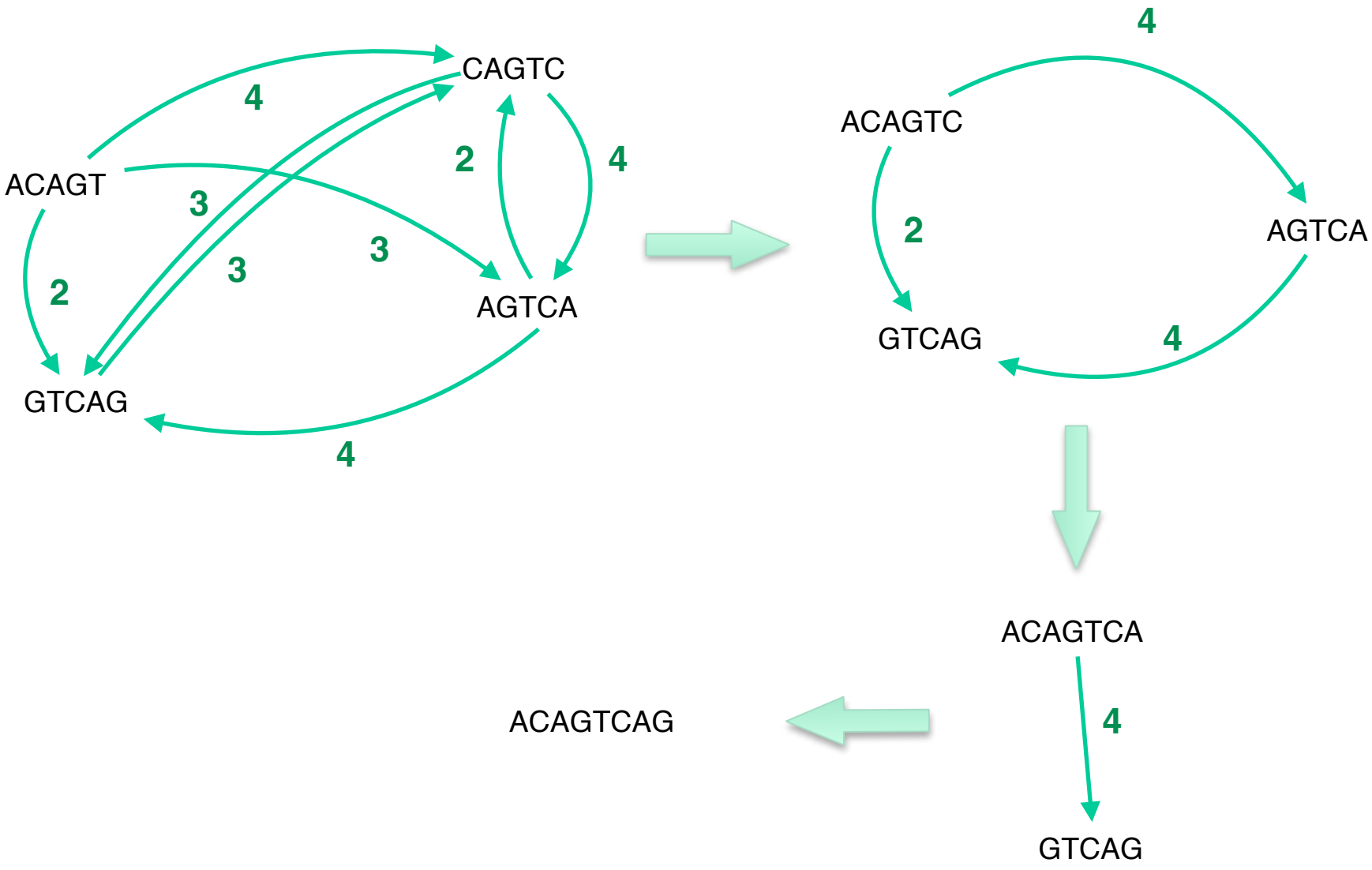
The greedy approach to assembly

Greedy assemblers try to find the **shortest common superstring** of a set of sequences (i.e., given a set of reads, the shortest string T such that every read is a substring of T).

This problem is **NP-complete**, i.e., it belongs to **both NP-hard** (there is no known algorithm to solve it in polynomial time) and **P** (one can check a proposed solution in polynomial time). For this reason, greedy assemblers use heuristics to try to approach the solution of the problem: at each step, the greedy algorithm (Tarhio & Ukkonen 1988) picks up the two sequences that overlap best and merges them.

Tarhio & Ukkonen (1988) A greedy approximation algorithm for constructing shortest common superstrings. *Theoretical Computer Science* 57:131-145

The greedy approach to assembly



The greedy approach to assembly

The greedy algorithm does not find always the shortest common superstring: by merging the two best-overlapping reads, greedy approaches can get caught into **local optima** instead of returning the overall best assembly.

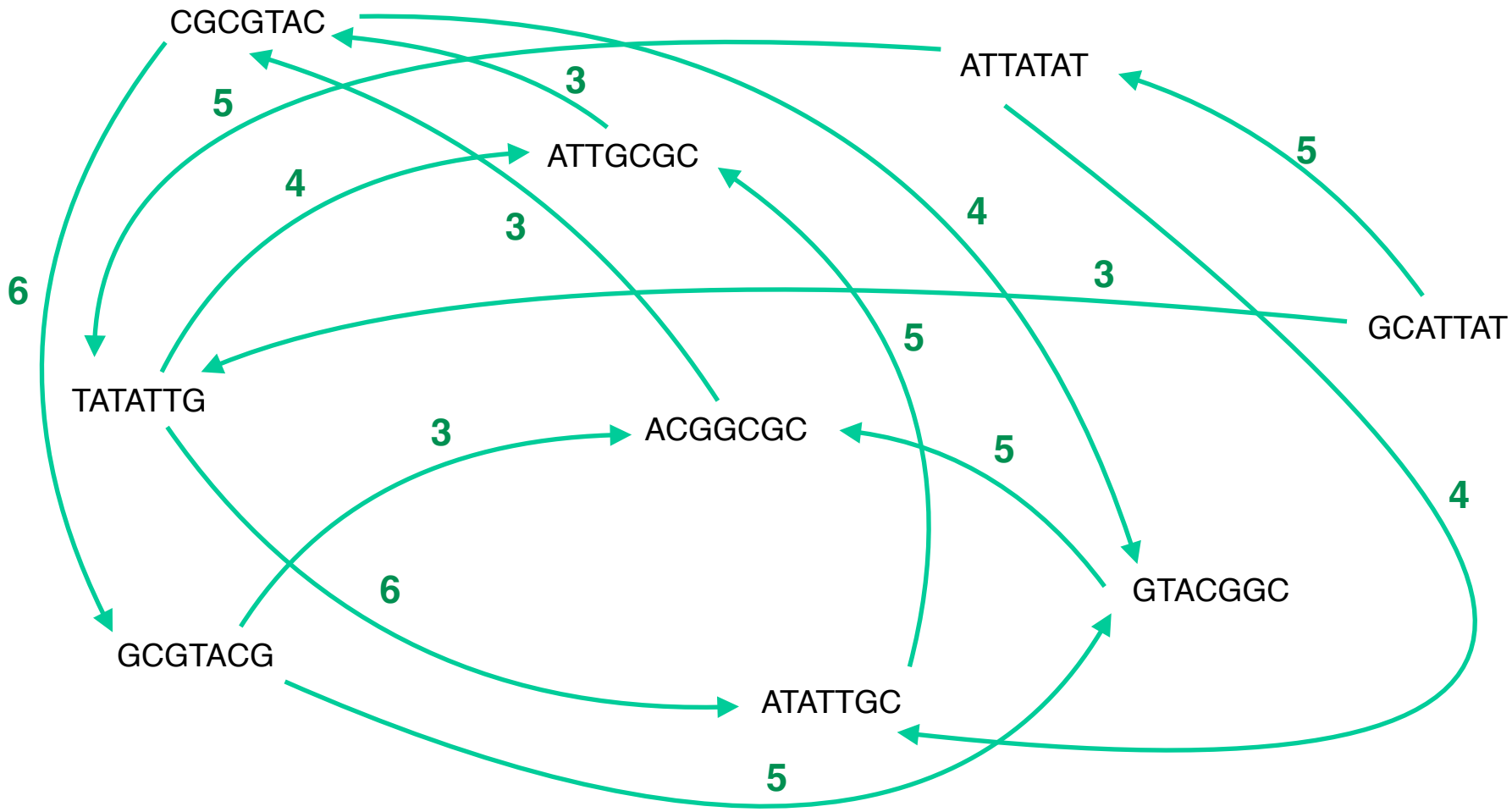
However, the greedy algorithm has been showed to produce a common superstring that is **at most four times longer** than the shortest common superstring (Blum et al. 1994), whereas in practice it seems to never produce a common superstring **more than two times longer** than this optimal target (Tarhio & Ukkonen 1988).

Blum, Jiang, Li, Tromp & Yannakakis (1994) Linear approximation of shortest superstrings. *Journal of the ACM* 41:630-647

Tarhio & Ukkonen (1988) A greedy approximation algorithm for constructing shortest common superstrings. *Theoretical Computer Science* 57:131-145

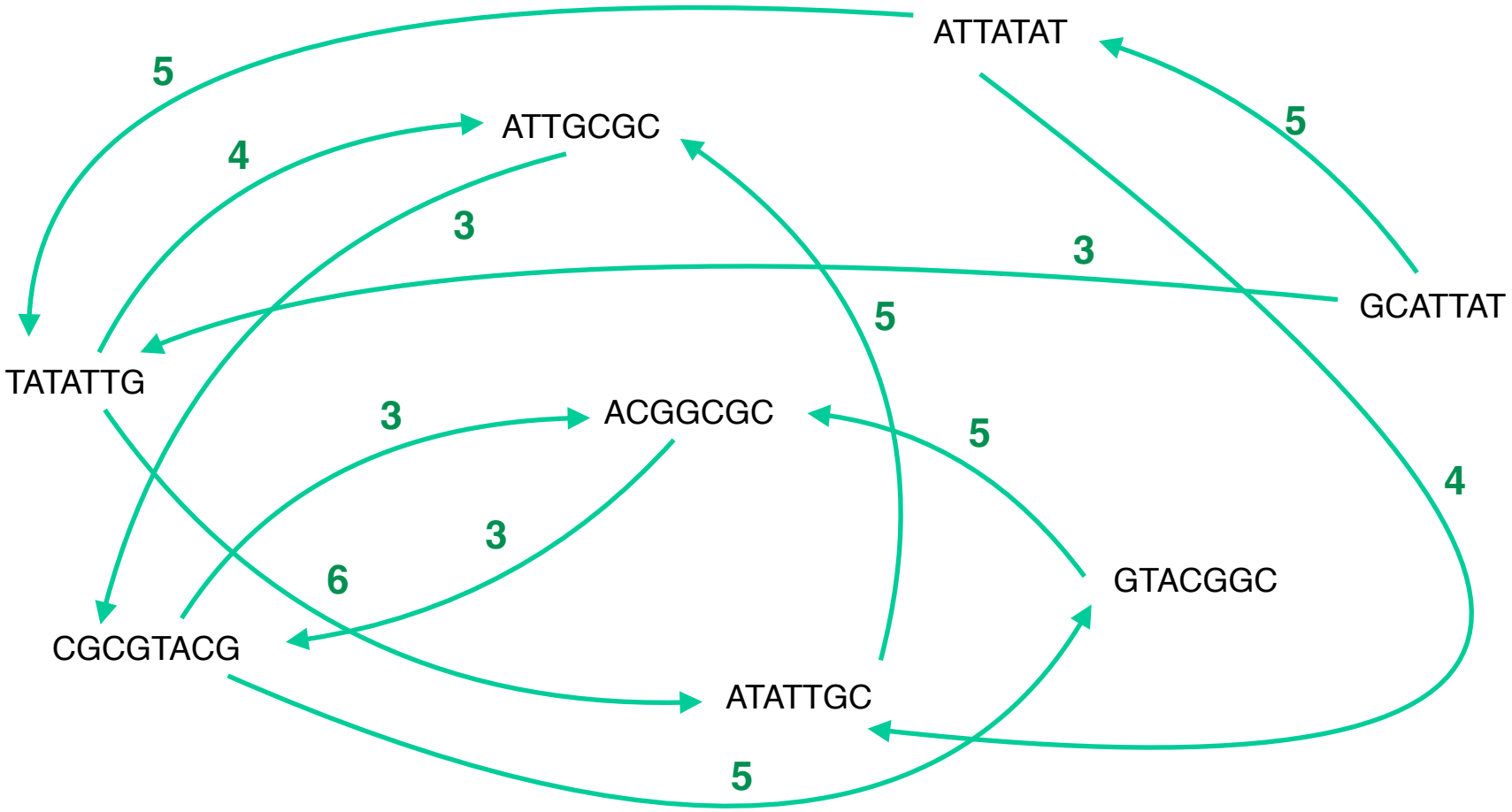
Exercise 2: greedy assembly

Assemble in a greedy fashion the same set of reads.

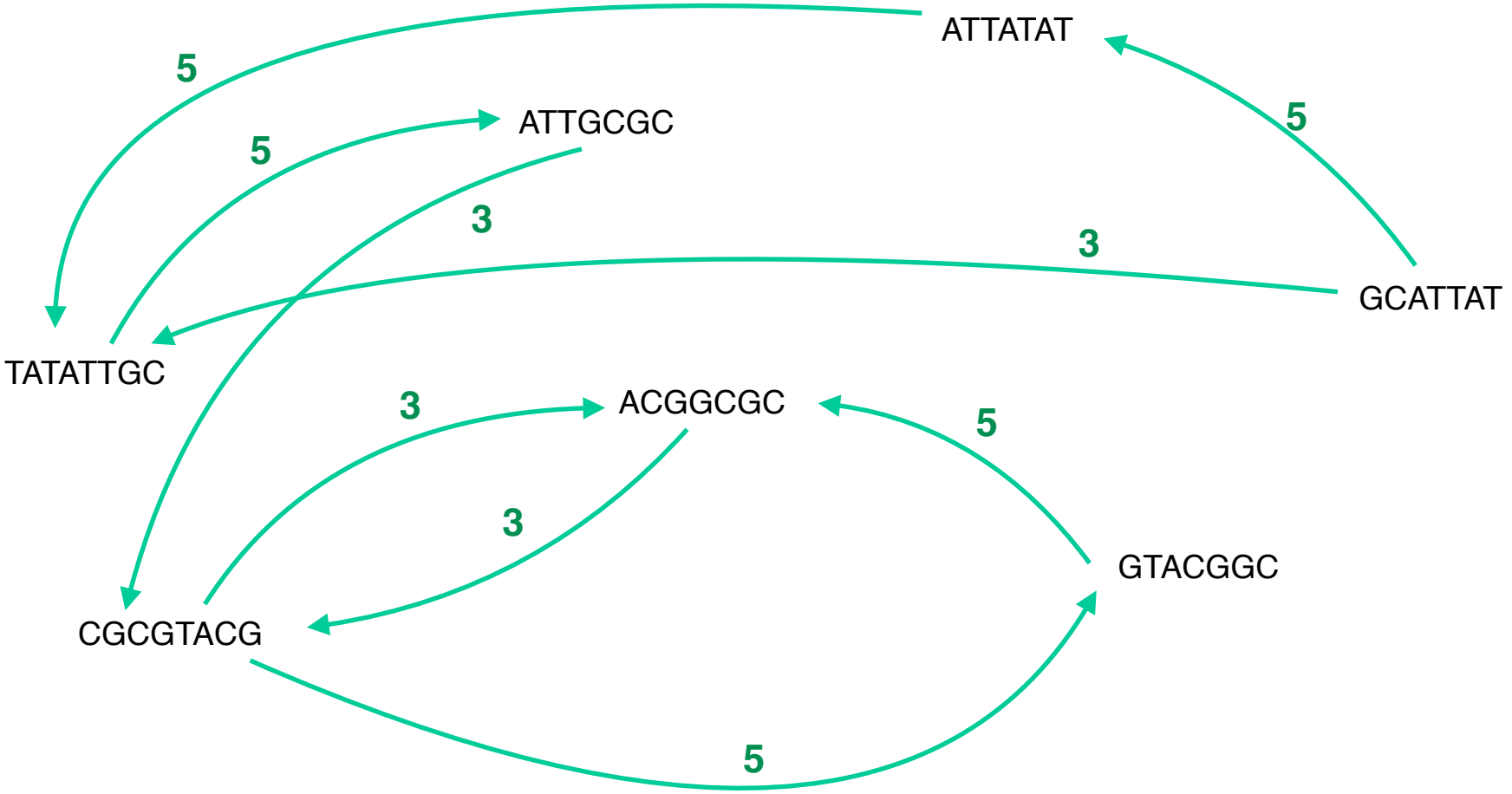


Hint: the solution is 23 bp long.

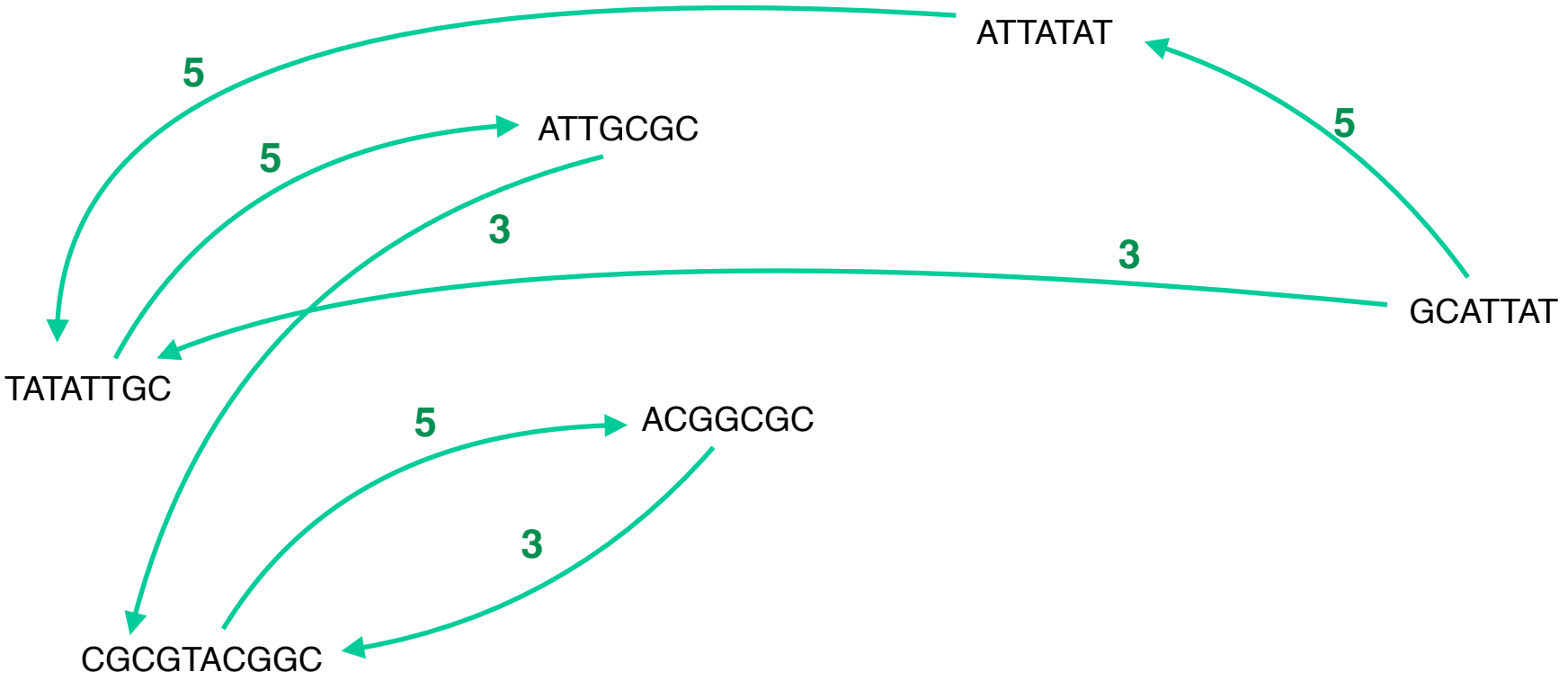
Exercise 2: solution



Exercise 2: solution

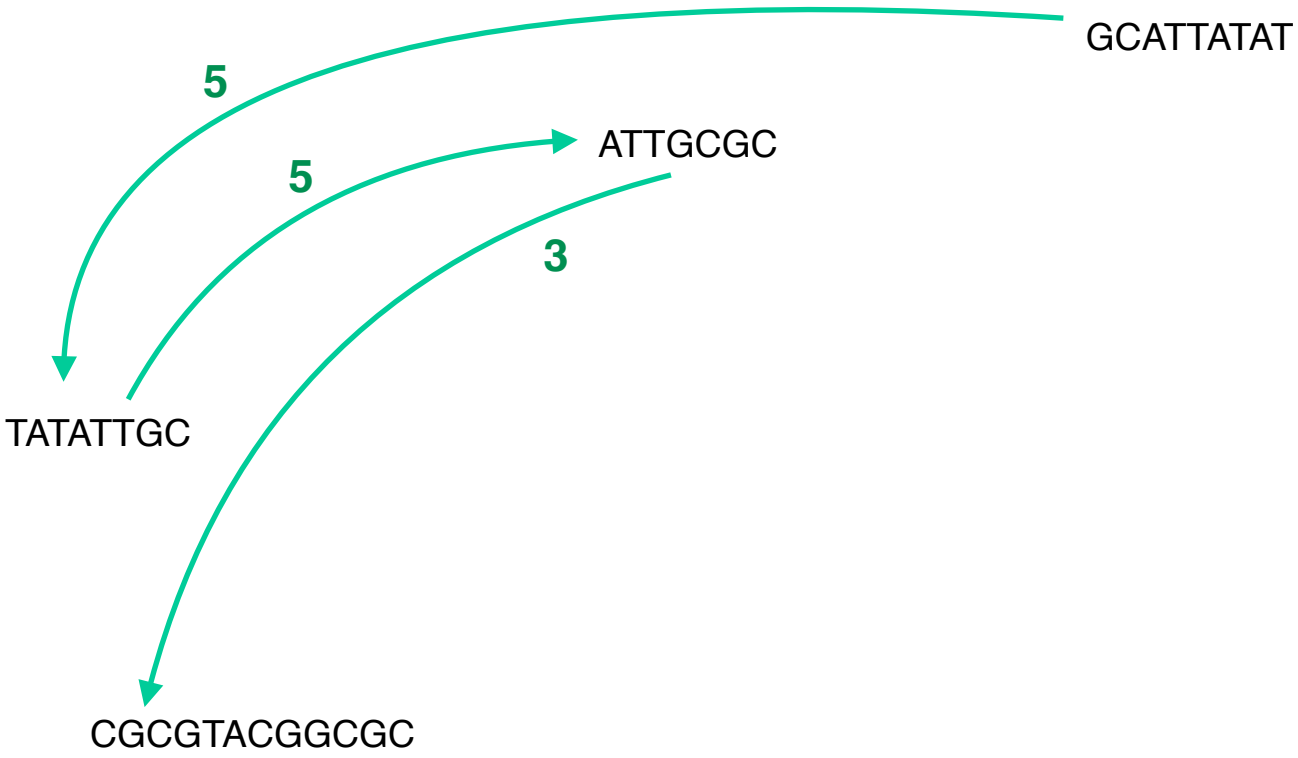


Exercise 2: solution

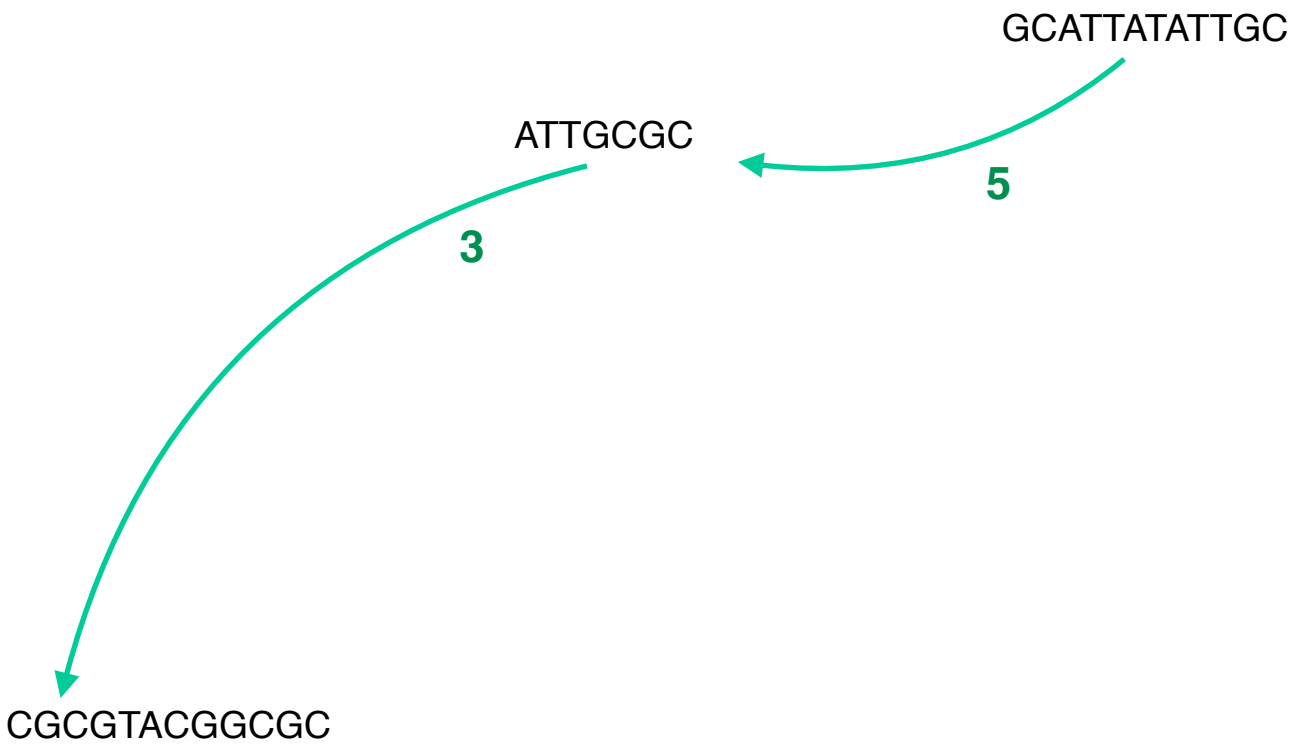


Exercise 2: solution

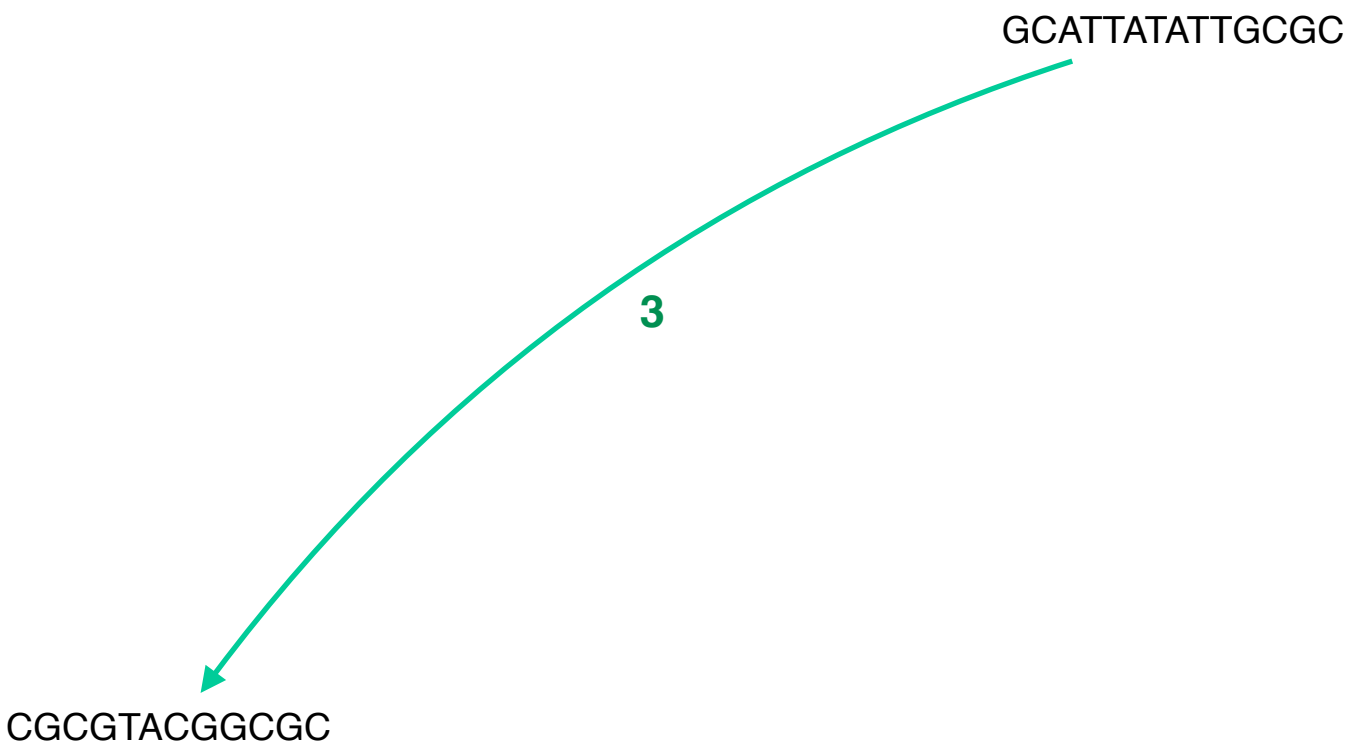
Exercise 2: solution



Exercise 2: solution



Exercise 2: solution



Exercise 2: solution

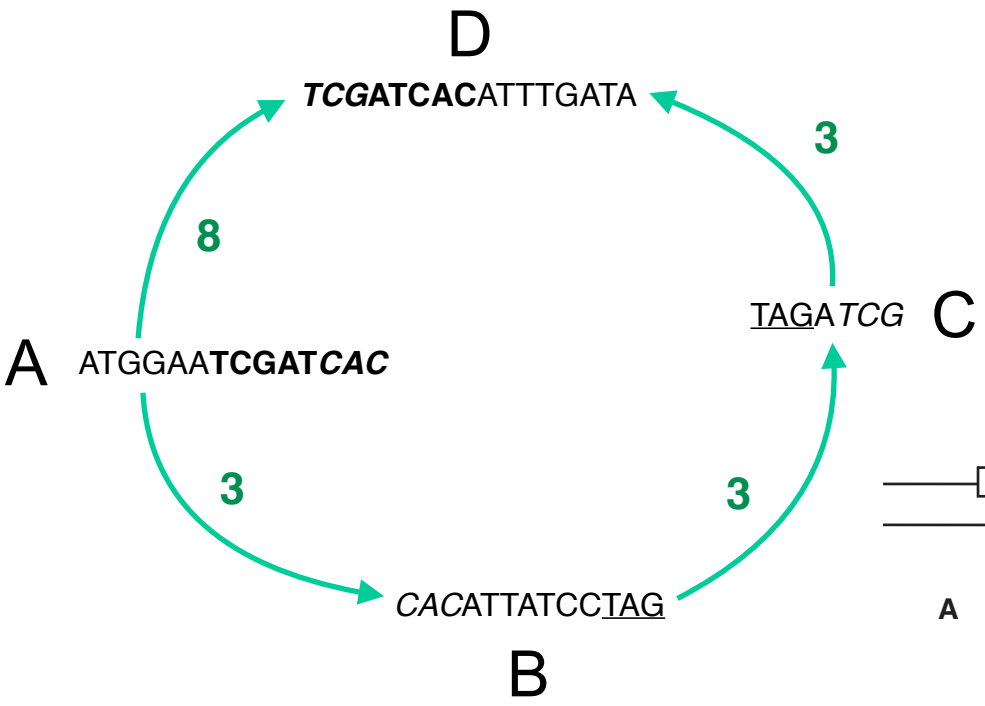
GCATTATATTGCGCGTACGGCGC

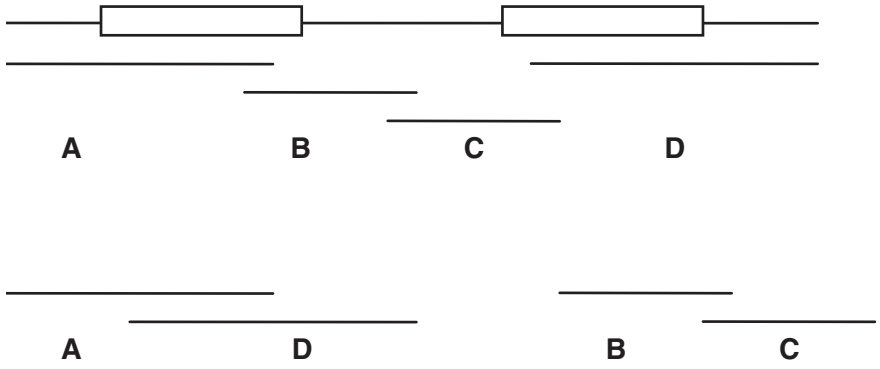
Problem: greedy assemblers fail on genome repeats

Greedy assemblers are fast and efficient, hence the popular programs to assemble Sanger data at the end of the 20th century (such as phrap, CAP3, and the TIGR assembler) were greedy. A recently published assembler using this paradigm is NOVOplasty, which specializes in assembling the circular genomes of mitochondria and chloroplasts.

However, as greedy assemblers aim to find the shortest common superstring they fail in case genomes are **not parcimonious**, e.g. when they contain **large, identical repeats**.

Problem: greedy assemblers fail on genome repeats





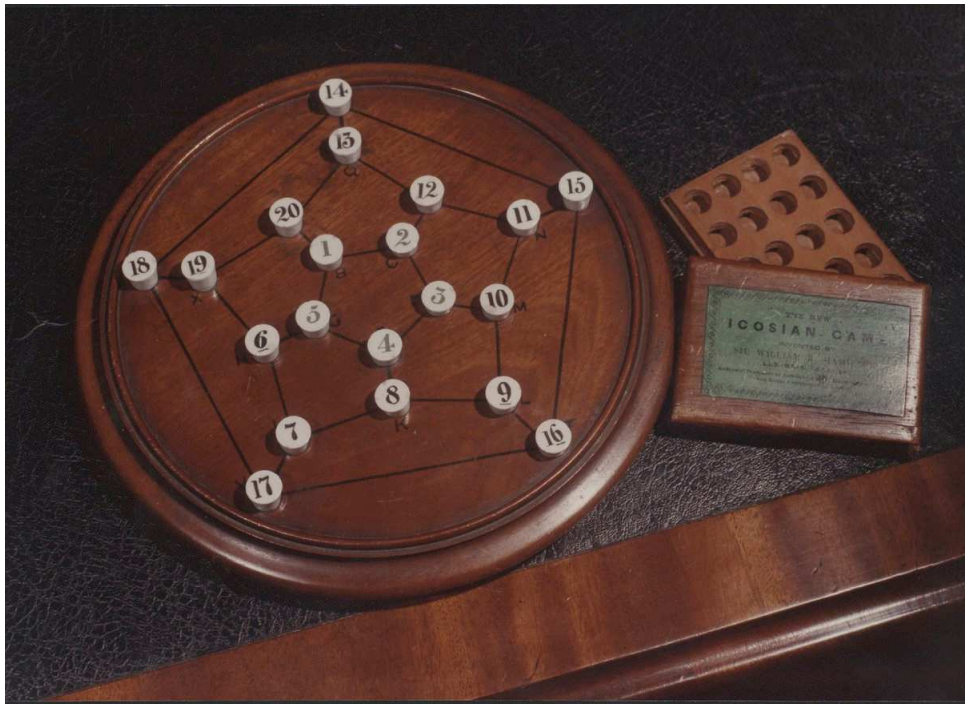
Pop (2009) *Briefings in Bioinformatics*

The OLC (overlap-layout-consensus) approach

OLC assemblers disantangle repeats by looking for the **shortest generalised Hamiltonian path** in the overlap graph (i.e., the shortest path that visits each node at least once, similar to the traveling salesman problem that looks for a generalised Hamiltonian cycle). Unfortunately, this is a **NP-hard** problem, i.e., there is no polynomial-time algorithm available to solve it (or even to check whether a proposed solution is really the shortest path).

Sir William Rowan Hamilton (1805-1865)

Source: Wikipedia



Icosian game invented by Hamilton in 1857

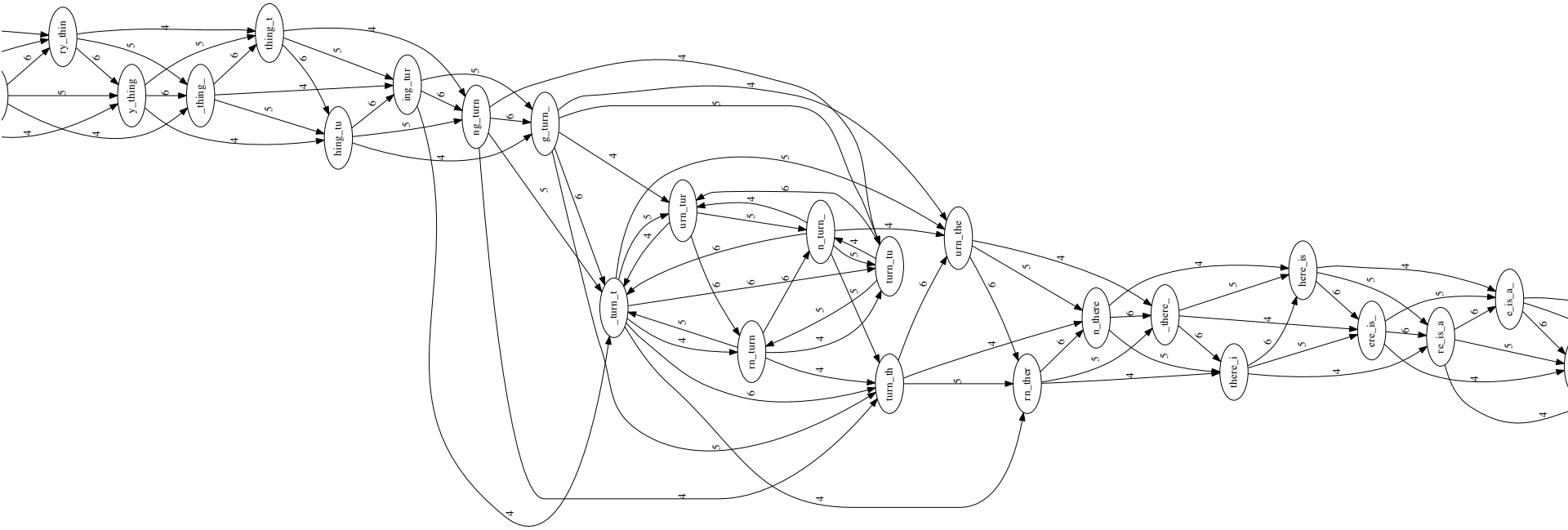
The OLC (overlap-layout-consensus) approach

Overlap graphs are often very complex.

Below: part of the overlap graph for

`to_every_thing_turn_turn_turn_there_is_a_season`

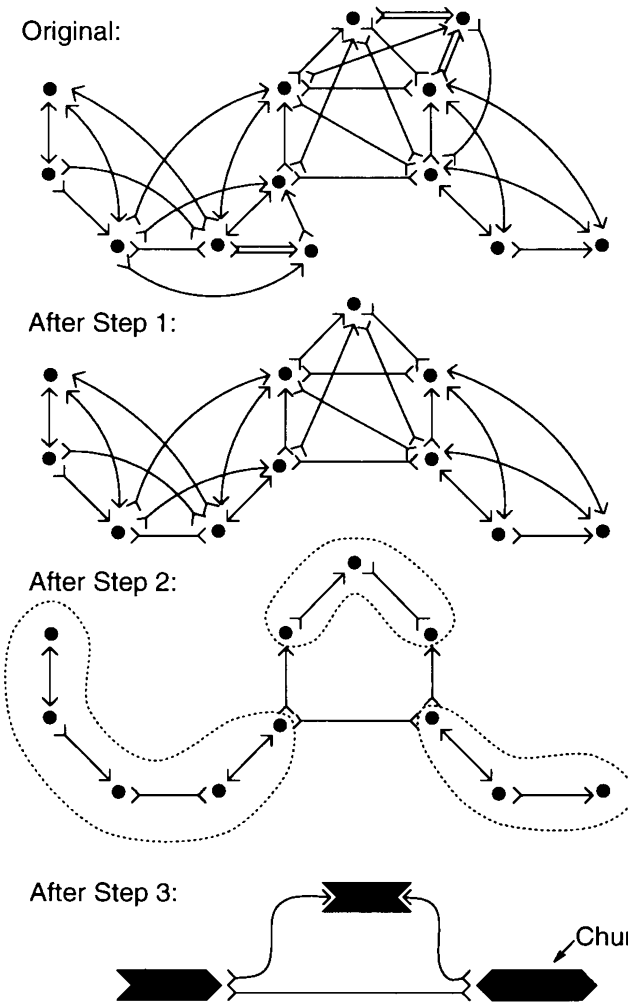
$l = 4, k = 7$

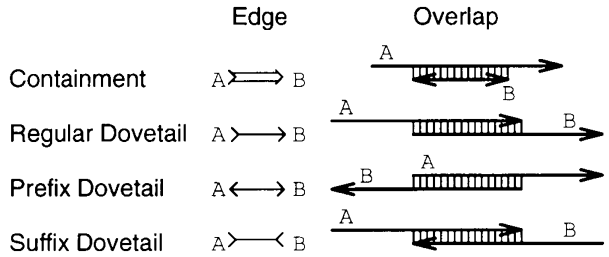


http://www.langmead-lab.org/teaching-materials/

The OLC (overlap-layout-consensus) approach

Layout step: simplifying the overlap graph by turning it into a string graph





JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 2, Number 2, 1995
Mary Ann Liebert, Inc.
Pp. 275–290

Toward Simplifying and Accurately Formulating Fragment Assembly

EUGENE W. MYERS

The OLC (overlap-layout-consensus) approach

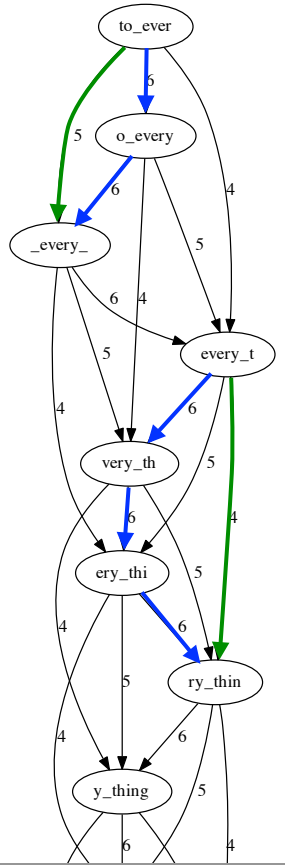
Layout step: simplifying the overlap graph by turning it into a string graph

The string graph is a directed graph whose set of vertices and edges are included in those of the overlap graph, but contained reads (reads which are included in other reads) and transitive edges (edges of the form $s_1 \rightarrow s_3$ when there exists s_2 such that $s_1 \rightarrow s_2$ and $s_2 \rightarrow s_3$) are removed.

Anything redundant about this part of the overlap graph?

Some edges can be *inferred* (*transitively*) from other edges

E.g. green edge can be inferred from blue

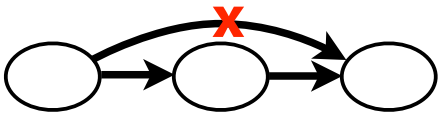


http://www.langmead-lab.org/teaching-materials/

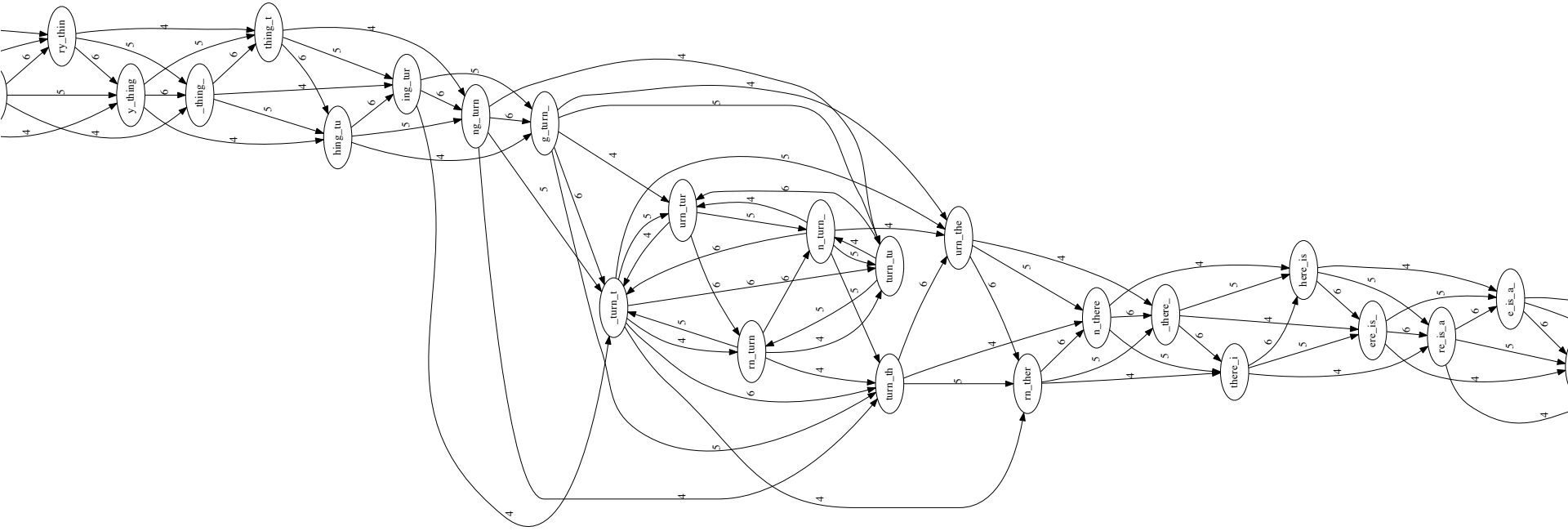
The OLC (overlap-layout-consensus) approach

Layout step: simplifying the overlap graph by turning it into a string graph

Remove transitively-inferrible edges, starting with edges that skip one node:



Before:

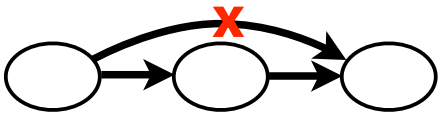


http://www.langmead-lab.org/teaching-materials/

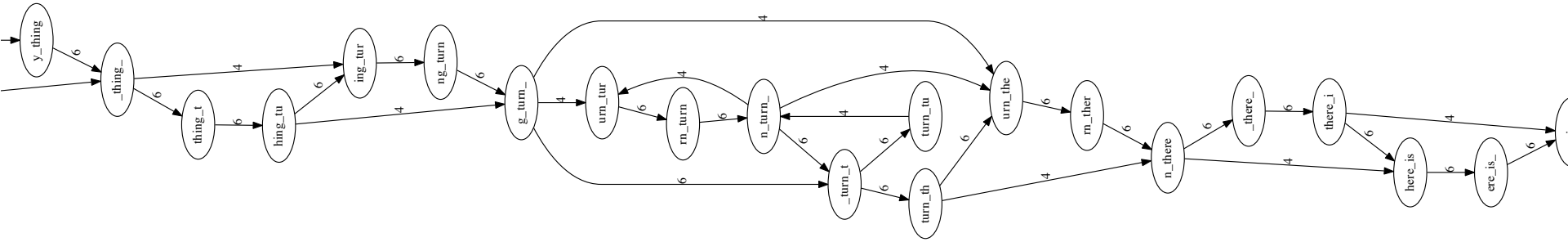
The OLC (overlap-layout-consensus) approach

Layout step: simplifying the overlap graph by turning it into a string graph

Remove transitively-inferrible edges, starting with edges that skip one node:



After:

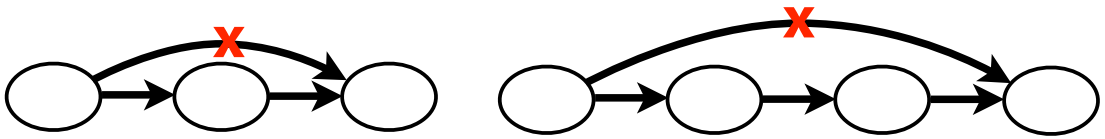


http://www.langmead-lab.org/teaching-materials/

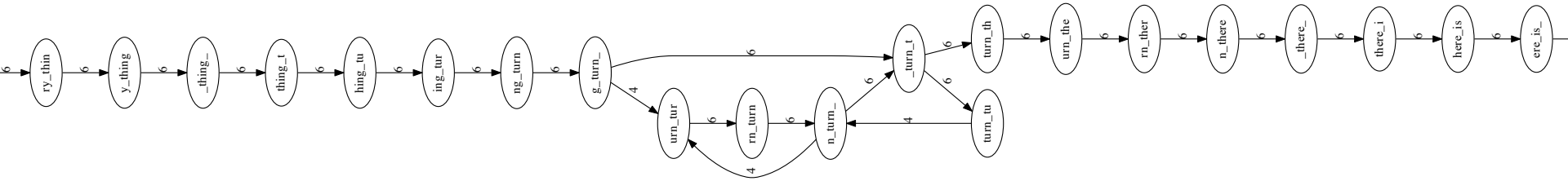
The OLC (overlap-layout-consensus) approach

Layout step: simplifying the overlap graph by turning it into a string graph

Remove transitively-inferrible edges, starting with edges that skip one *or two* nodes:



After:



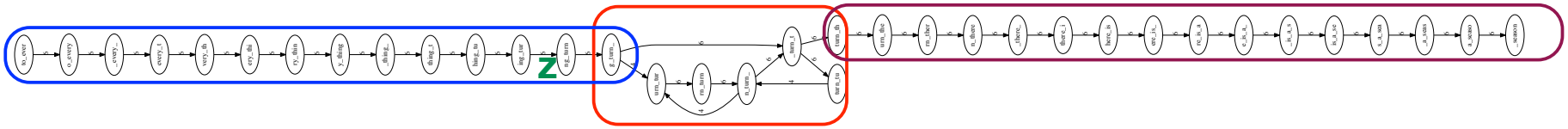
Even simpler

http://www.langmead-lab.org/teaching-materials/

The OLC (overlap-layout-consensus) approach

Layout step: simplifying the overlap graph by turning it into a string graph

Emit *contigs* corresponding to the non-branching stretches



Contig 1
`to_every_thing_turn_`

Contig 2
`turn_there_is_a_season`

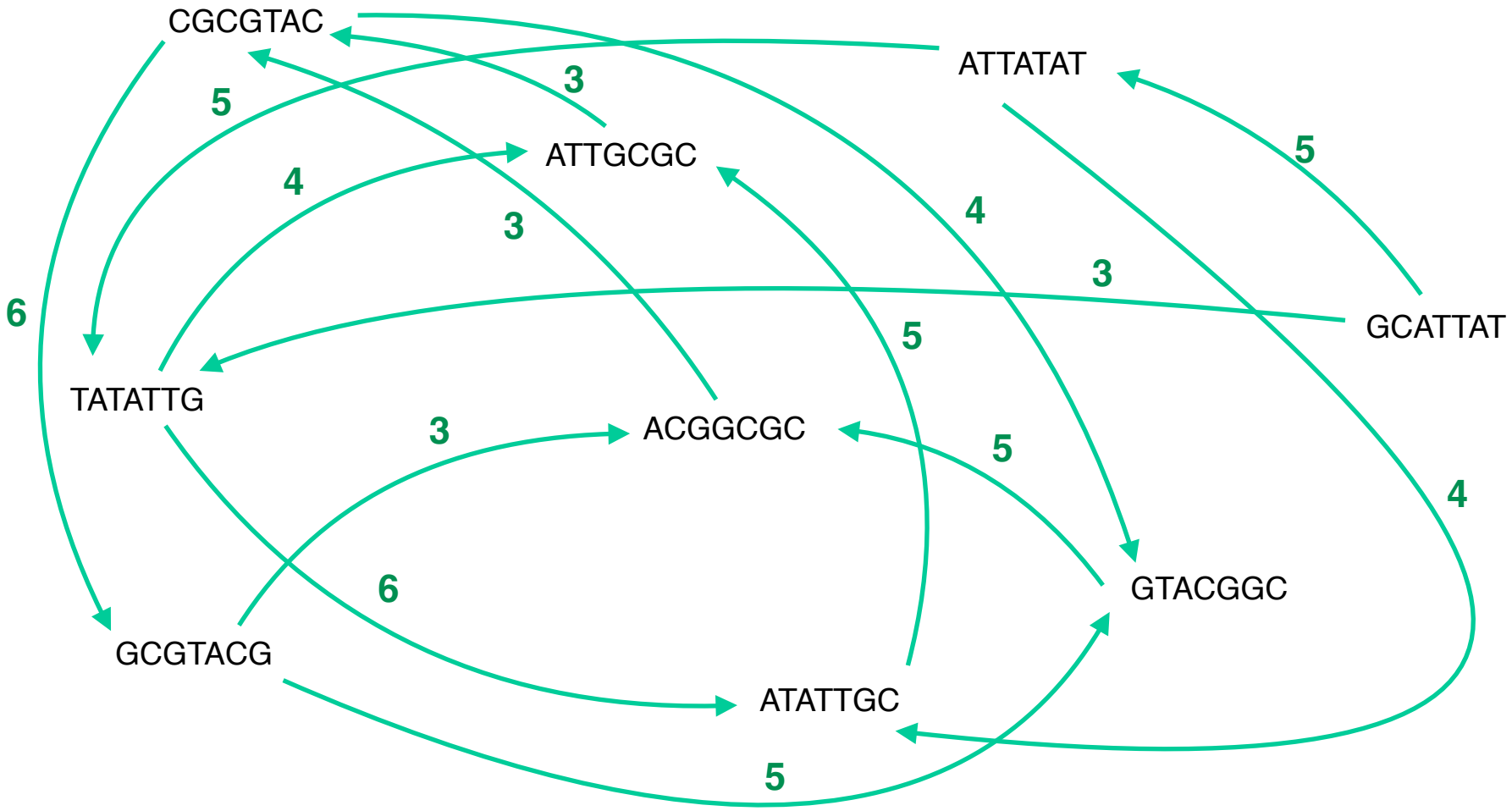
Unresolvable repeat

http://www.langmead-lab.org/teaching-materials/

Some assemblers call these non-branching stretches “unitigs” (short for “uniquely assemblable contigs”) or “chunks”.

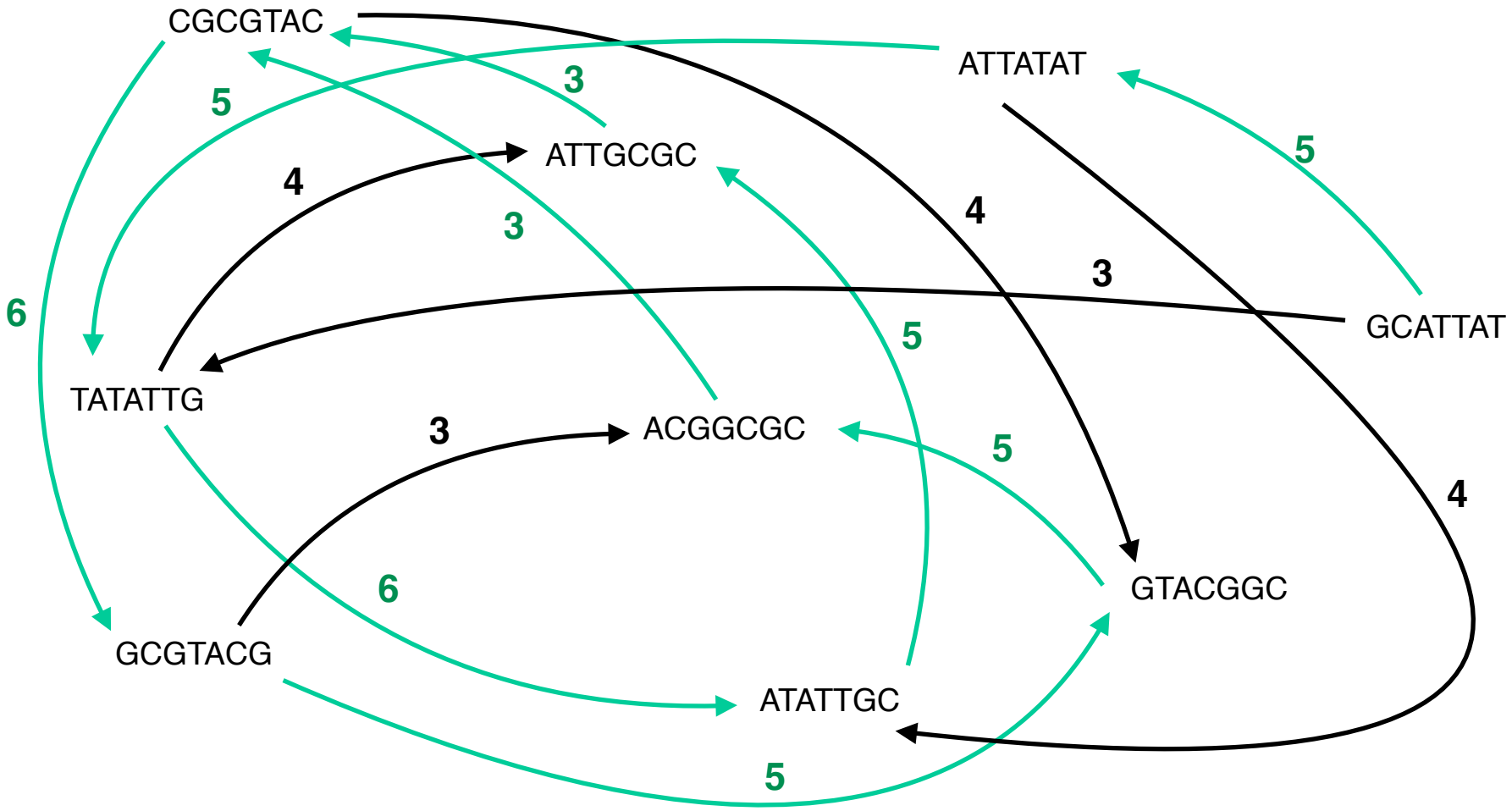
Exercise 4: turning an overlap graph into a string graph

Turn the overlap graph built previously into a string graph by removing all redundant information.



Exercise 4: turning an overlap graph into a string graph

Turn the overlap graph built previously into a string graph by removing all redundant information.



The OLC (overlap-layout-consensus) approach

Third step: consensus

At each position, ask: what nucleotide (and/or gap) is here?

Complications: (a) sequencing error, (b) ploidy

Exercise 5: practising consensus

Build a graph representing all the overlaps of a minimum of 5 bases with at most one mismatch between the following reads, then assemble them in the OLC way.

TATATTAA ATTATAT ATGTTAAC TAAATGT ATATGT TGTTAACG

Hint: there are 8 overlaps to find, two of which will be removed to yield the string graph. The final consensus should be 14 bp long.

Exercise 5: practising consensus

Build a graph representing all the overlaps of a minimum of 5 bases with at most one mismatch between the following reads, then assemble them in the OLC way.

TATATTAA

ATTATAT

TAAATGT

TGTTAACG

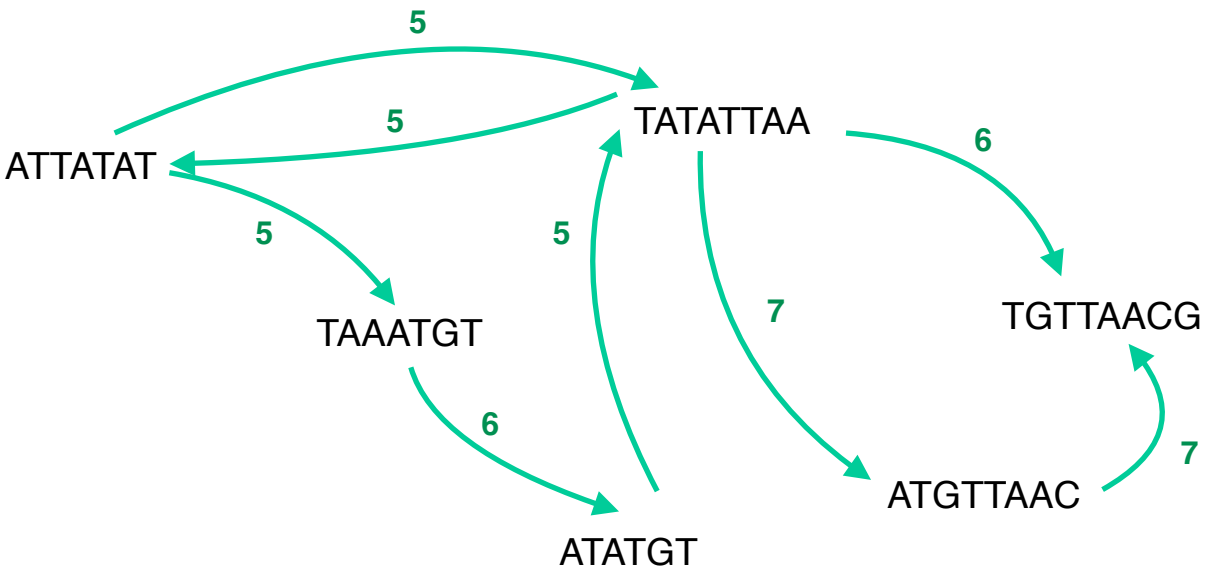
ATGTTAAC

ATATGT

Hint: there are 8 overlaps to find, two of which will be removed to yield the string graph. The final consensus should be 14 bp long.

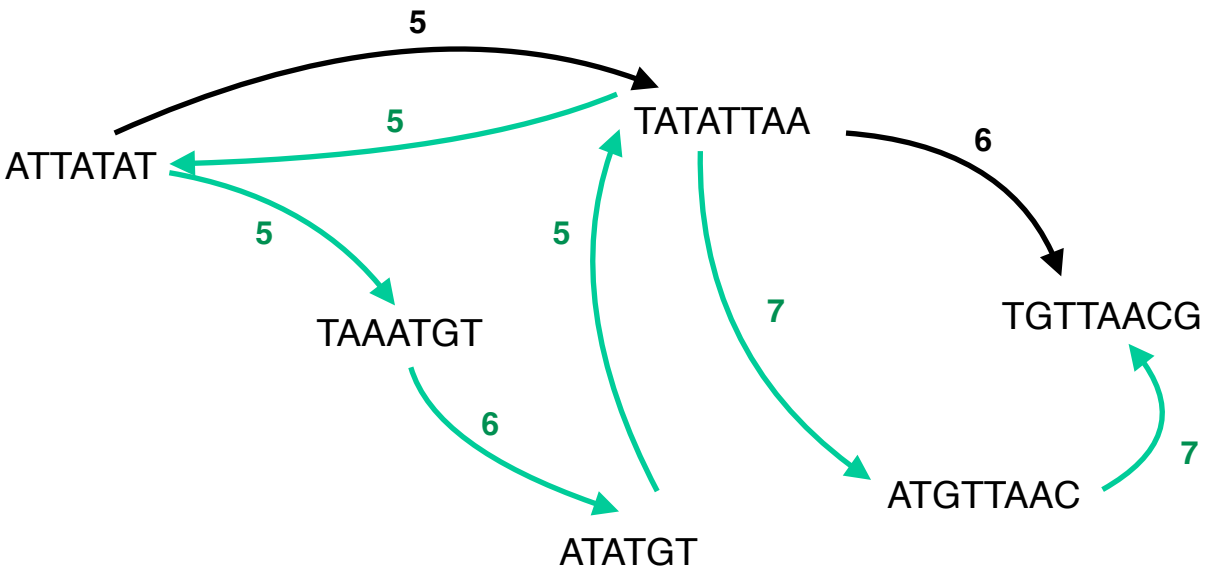
Exercise 5: solution

Build a graph representing all the overlaps of a minimum of 5 bases with at most one mismatch between the following reads, then assemble them in the OLC way.



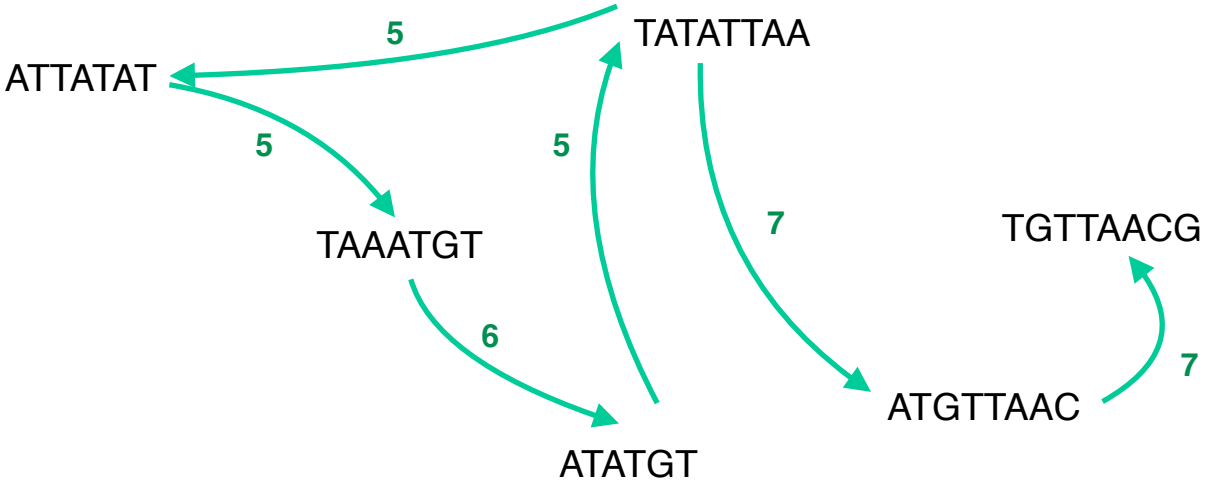
Exercise 5: solution

Build a graph representing all the overlaps of a minimum of 5 bases with at most one mismatch between the following reads, then assemble them in the OLC way.



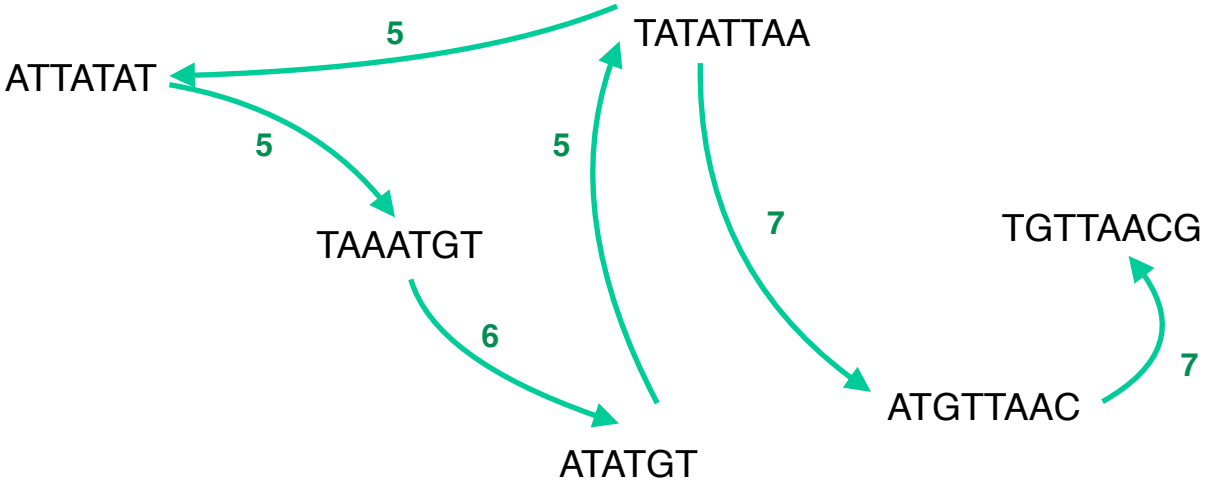
Exercise 5: solution

Build a graph representing all the overlaps of a minimum of 5 bases with at most one mismatch between the following reads, then assemble them in the OLC way.



Exercise 5: solution

Build a graph representing all the overlaps of a minimum of 5 bases with at most one mismatch between the following reads, then assemble them in the OLC way.




```
ATTATAT
  TAAATGT
   ATATGT
    TATATTAA
     ATGTTAAC
      TGTTAACG
---------------
ATTATATGTTAACG
```

The problem with OLC assemblers

OLC assemblers are well able to disentangle repeats. However, they require looking for overlaps among all reads then finding a Hamiltonian path in the graph, hence the computational cost is very high when there are lots of reads. They are therefore best suited to small genomes (< ca. 200 Mb) and/or long reads (PacBio, Nanopore).

Popular OLC genome assemblers are: Canu, SMARTdenovo, Falcon (for PacBio and Nanopore reads), Raven (formerly known as Ra); minimap2/ miniasm/minipolish implement the overlap step (minimap) then the layout step (miniasm) and finally the consensus step (minipolish). Omega is a OLC assembler specially designed for metagenomic datasets. Peregrine and Hifiasm deal only with PacBio HiFi data.

A third way: de Bruijn graphs

An Eulerian path approach to DNA fragment assembly

Pavel A. Pevzner*, Haixu Tang†, and Michael S. Waterman†‡§

www.pnas.org/cgi/doi/10.1073/pnas.171285098

*Department of Computer Science and Engineering, University of California, San Diego, La Jolla, CA; and Departments of †Mathematics and ‡Biological Sciences, University of Southern California, Los Angeles, CA

9748–9753 | PNAS | August 14, 2001 | vol. 98 | no. 17

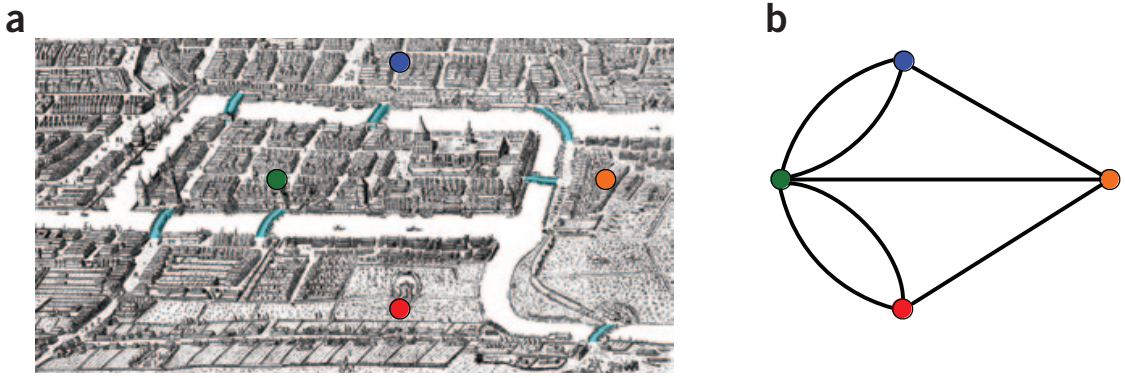


Figure 1 Bridges of Königsberg problem. (**a**) A map of old Königsberg, in which each area of the city is labeled with a different color point. (**b**) The Königsberg Bridge graph, formed by representing each of four land areas as a node and each of the city's seven bridges as an edge.

Leonhard Euler (1707-1783)
portrait byJakob Emanuel Handmann - Kunstmuseum Basel

The Königsberg Bridges Problem (*briefly* KBP):
Is it possible for a pedestrian to walk across all seven bridges in Königsberg without crossing any bridge twice?

Eulerian Path *(or Walk)* Problem (briefly EPP):
Is it possible to traverse a graph passing through every edge exactly once?

Grötschel & Yuan (2012) Euler, Mei-Ko Kwan, Königsberg, and a Chinese Postman. *Documenta Mathematica* Extra Volume:43-50

A third way: de Bruijn graphs

If it is possible to pass through some edges **more than once**, the problem become the **Generalized** Eulerian Path Problem. Finding a **generalized Eulerian cycle** (i.e. a generalized Eulerian path that starts and end at the same point) is also called the **Chinese Postman problem**, in honor of the Chinese mathematician **Meigu Guan** who formalised it.

第 10 卷 第 3 期 数 学 学 报 Vol. 10, No. 3
1960 年 12 月 ACTA MATHEMATICA SINICA Dec., 1960

奇 偶 点 图 上 作 业 法*

管 梅 谷
(山 东 师 范 学 院)

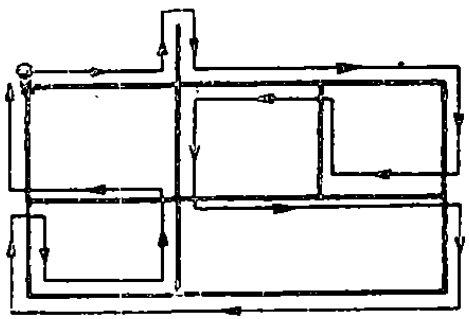
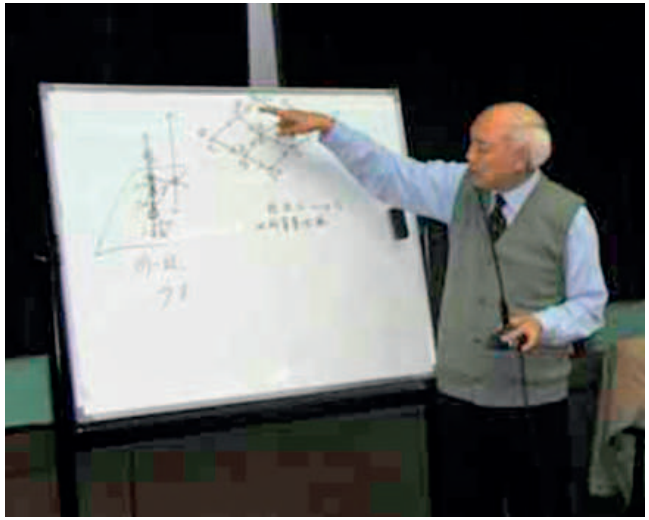


图 5

管 梅谷
= Mei-Ko Kwan
= Meigu Guan



Grötschel & Yuan (2012) Euler, Mei-Ko Kwan, Königsberg, and a Chinese Postman. *Documenta Mathematica* Extra Volume:43-50

A third way: de Bruijn graphs

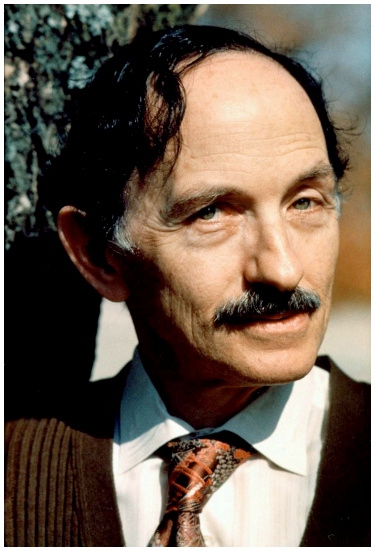
Nicolaas Govert de Bruijn (1918-2012)
http://tueinbeeld.tue.nl/search.pp?showpicture=39279&page=1&pos=5#

de Bruijn developed a method for finding the **shortest circular superstring containing all the possible k-mers of a given alphabet**. His method involved build a graph with **all possible k-mers as edges** and **their (k-1)-mer overlaps as nodes**, then looking for the shortest circular path trough the graph crossing each edge at least one, i.e., **a generalised Eulerian cycle**.

Mathematics. — *A combinatorial problem.* By N. G. DE BRUIJN. (Communicated by Prof. W. VAN DER WOUDE.)

(Communicated at the meeting of June 29, 1946.)

de Bruijn NG (1946) *Proceedings of the Section of Sciences of the Koninklijke Nederlandse Akademie van Wetenschappen te Amsterdam* 49: 758-764



Irving John Good (1916-2009)
https://vtspecialcollections.wordpress.com/2015/01/15/i-j-jack-good-virginia-techs-own-bletchley-park-connection/#jp-carousel-2099

The same year, British mathematician Irving John Good published the same idea independently. Both built on previous work by the French mathematician Camille Flye Sainte-Marie

A third way: de Bruijn graphs

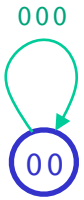
In the simplest case of a binary alphabet
with k=3 there are 8 possible 3-mers:

000
001
010
011
100
101
110
111

A third way: de Bruijn graphs

In the simplest case of a binary alphabet
with k=3 there are 8 possible 3-mers:

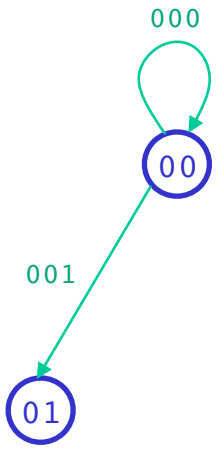
000 corresponds to an edge 00→00
001
010
011
100
101
110
111



A third way: de Bruijn graphs

In the simplest case of a binary alphabet with k=3 there are 8 possible 3-mers:

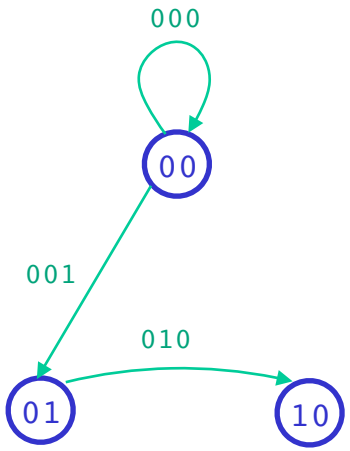
000 corresponds to an edge 00→00
001 corresponds to an edge 00→01
010
011
100
101
110
111



A third way: de Bruijn graphs

In the simplest case of a binary alphabet with k=3 there are 8 possible 3-mers:

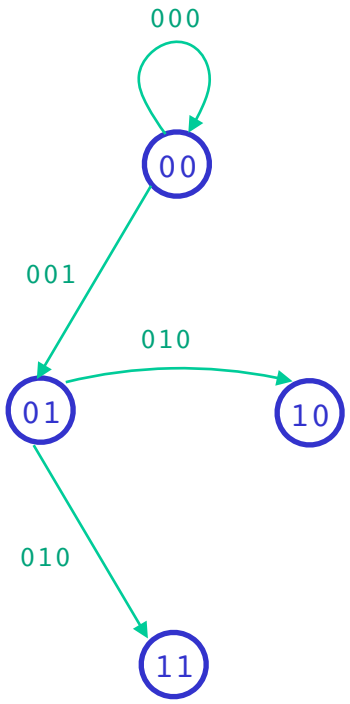
000 corresponds to an edge 00→00
001 corresponds to an edge 00→01
010 corresponds to an edge 01→10
011
100
101
110
111



A third way: de Bruijn graphs

In the simplest case of a binary alphabet with k=3 there are 8 possible 3-mers:

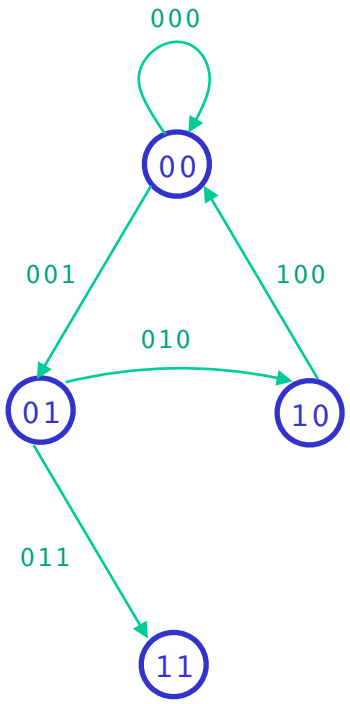
000 corresponds to an edge 00→00
001 corresponds to an edge 00→01
010 corresponds to an edge 01→10
011 corresponds to an edge 01→11
100
101
110
111



A third way: de Bruijn graphs

In the simplest case of a binary alphabet with k=3 there are 8 possible 3-mers:

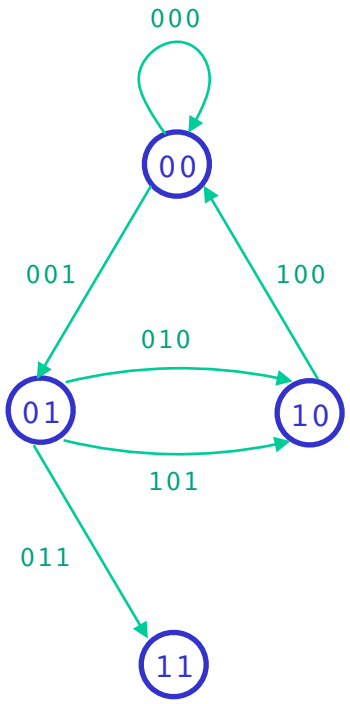
000 corresponds to an edge 00→00
001 corresponds to an edge 00→01
010 corresponds to an edge 01→10
011 corresponds to an edge 01→11
100 corresponds to an edge 10→00
101
110
111



A third way: de Bruijn graphs

In the simplest case of a binary alphabet with k=3 there are 8 possible 3-mers:

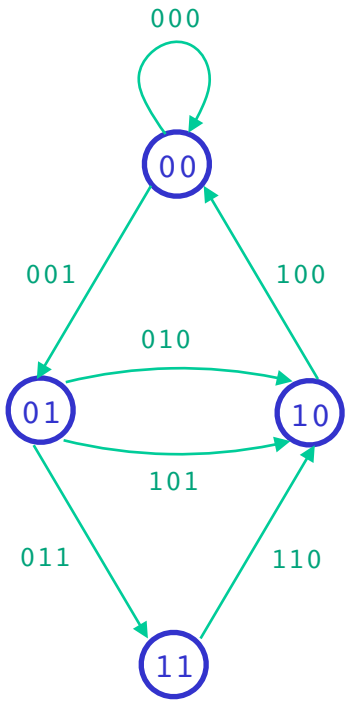
000 corresponds to an edge 00→00
001 corresponds to an edge 00→01
010 corresponds to an edge 01→10
011 corresponds to an edge 01→11
100 corresponds to an edge 10→00
101 corresponds to an edge 10→01
110
111



A third way: de Bruijn graphs

In the simplest case of a binary alphabet with k=3 there are 8 possible 3-mers:

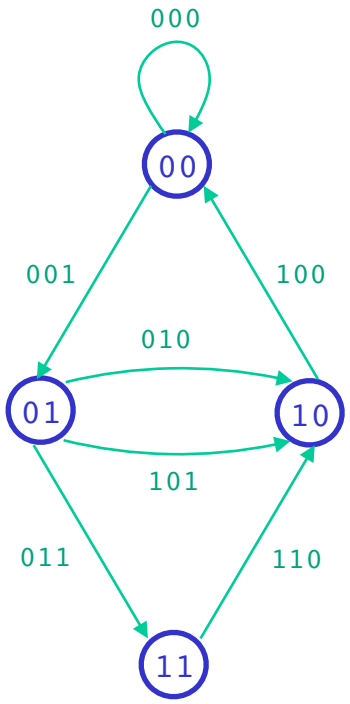
000 corresponds to an edge 00→00
001 corresponds to an edge 00→01
010 corresponds to an edge 01→10
011 corresponds to an edge 01→11
100 corresponds to an edge 10→00
101 corresponds to an edge 10→01
110 corresponds to an edge 11→10
111



A third way: de Bruijn graphs

In the simplest case of a binary alphabet with k=3 there are 8 possible 3-mers:

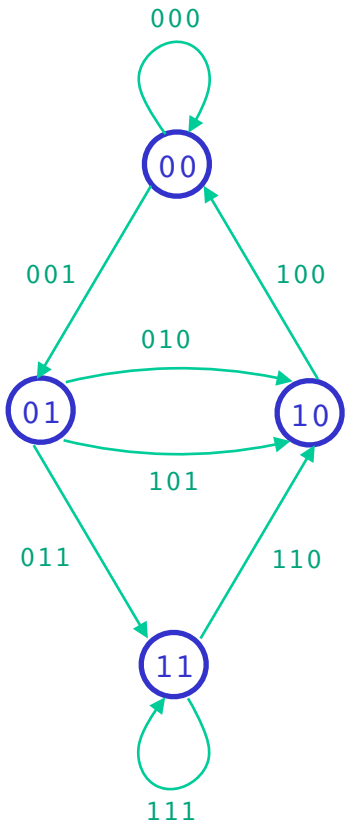
000 corresponds to an edge 00→00
001 corresponds to an edge 00→01
010 corresponds to an edge 01→10
011 corresponds to an edge 01→11
100 corresponds to an edge 10→00
101 corresponds to an edge 10→01
110 corresponds to an edge 11→10
111 corresponds to an edge 11→11



A third way: de Bruijn graphs

In the simplest case of a binary alphabet with k=3 there are 8 possible 3-mers:

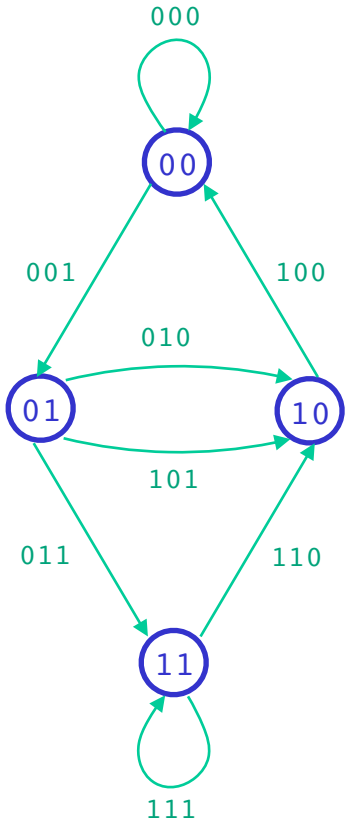
000 corresponds to an edge 00→00
001 corresponds to an edge 00→01
010 corresponds to an edge 01→10
011 corresponds to an edge 01→11
100 corresponds to an edge 10→00
101 corresponds to an edge 10→01
110 corresponds to an edge 11→10
111 corresponds to an edge 11→11



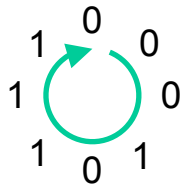
A third way: de Bruijn graphs

In the simplest case of a binary alphabet with k=3 there are 8 possible 3-mers:

000 corresponds to an edge 00→00
001 corresponds to an edge 00→01
010 corresponds to an edge 01→10
011 corresponds to an edge 01→11
100 corresponds to an edge 10→00
101 corresponds to an edge 10→01
110 corresponds to an edge 11→10
111 corresponds to an edge 11→11



Resulting de Bruijn sequence:



de Bruijn graphs for DNA sequences

Tetranucleotide alphabet
with k=3

Possible 3-mers:

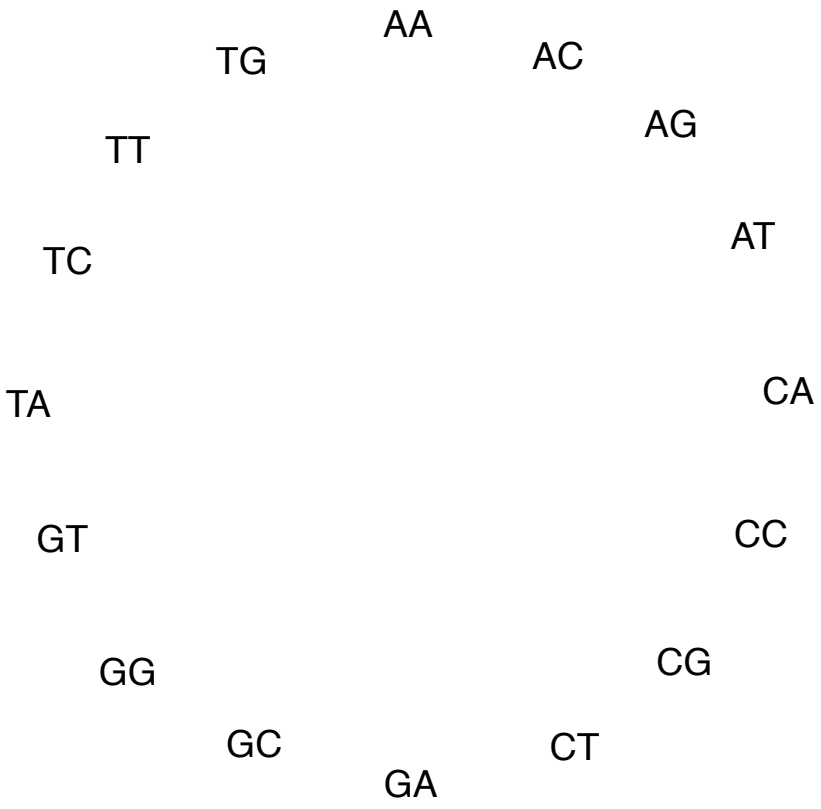
AAA	CAA	GAA	TAA
AAC	CAC	GAC	TAC
AAG	CAG	GAG	TAG
AAT	CAT	GAT	TAT
ACA	CCA	GCA	TCA
ACC	CCC	GCC	TCC
ACG	CCG	GCG	TCG
ACT	CCT	GCT	TCT
AGA	CGA	GGA	TGA
AGC	CGC	GGC	TGC
AGG	CGG	GGG	TGG
AGT	CGT	GGT	TGT
ATA	CTA	GTA	TTA
ATC	CTC	GTC	TTC
ATG	CTG	GTG	TTG
ATT	CTT	GTT	TTT

Possible 2-mer overlaps:

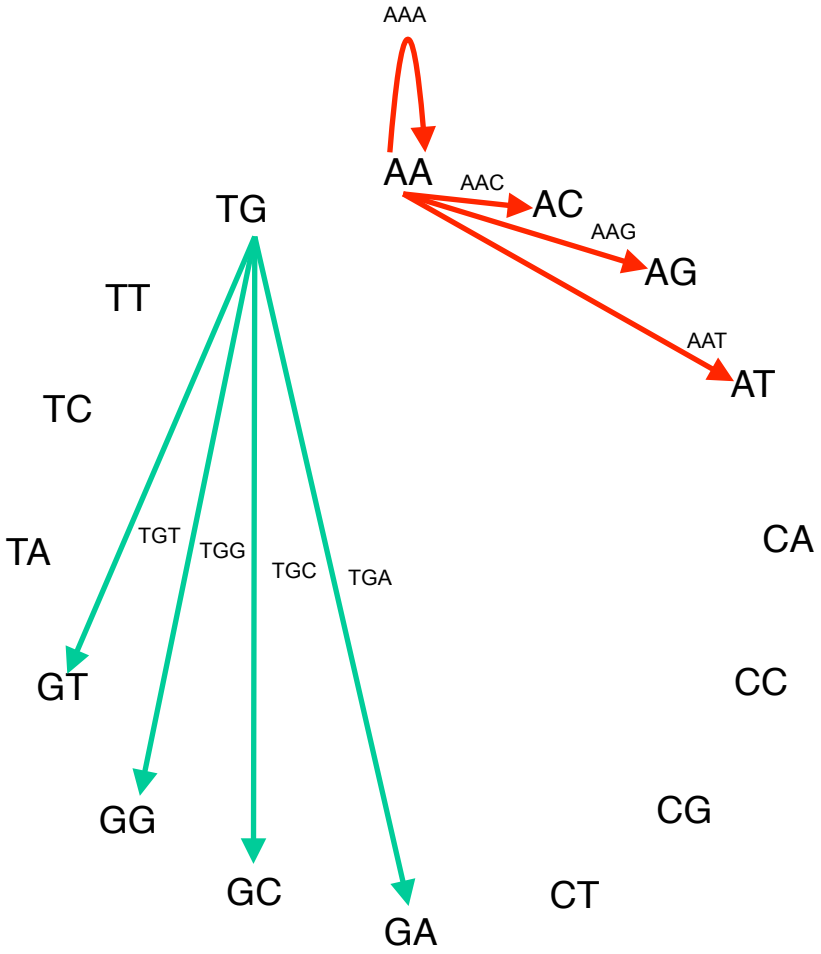
AA	GA
AC	GC
AG	GG
AT	GT
CA	TA
CC	TC
CG	TG
CT	TT

AA	AC	AG	AT
CA	CC	CG	CT
GA	GC	GG	GT
TA	TC	TG	TT

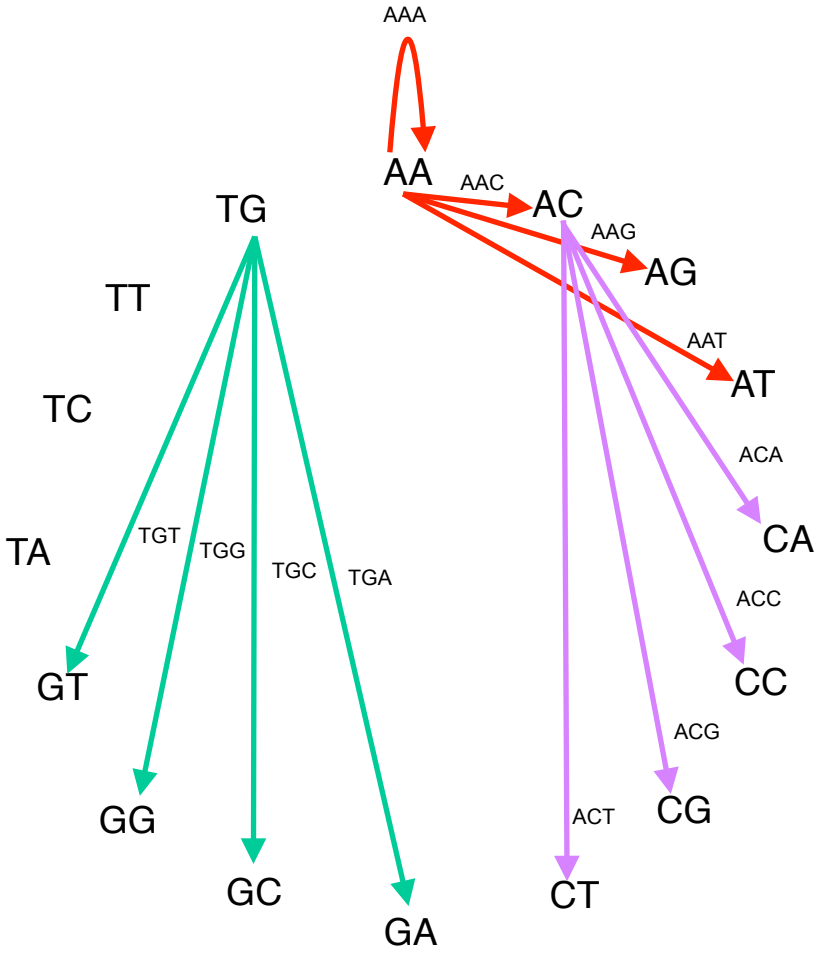
de Bruijn graphs for DNA sequences



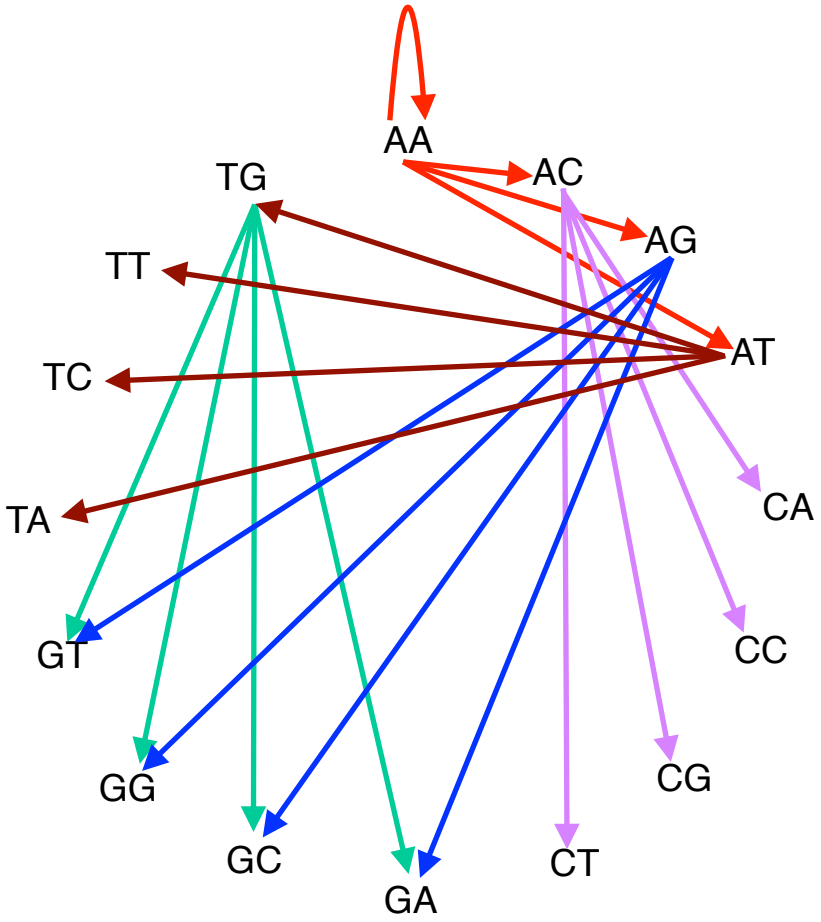
de Bruijn graphs for DNA sequences



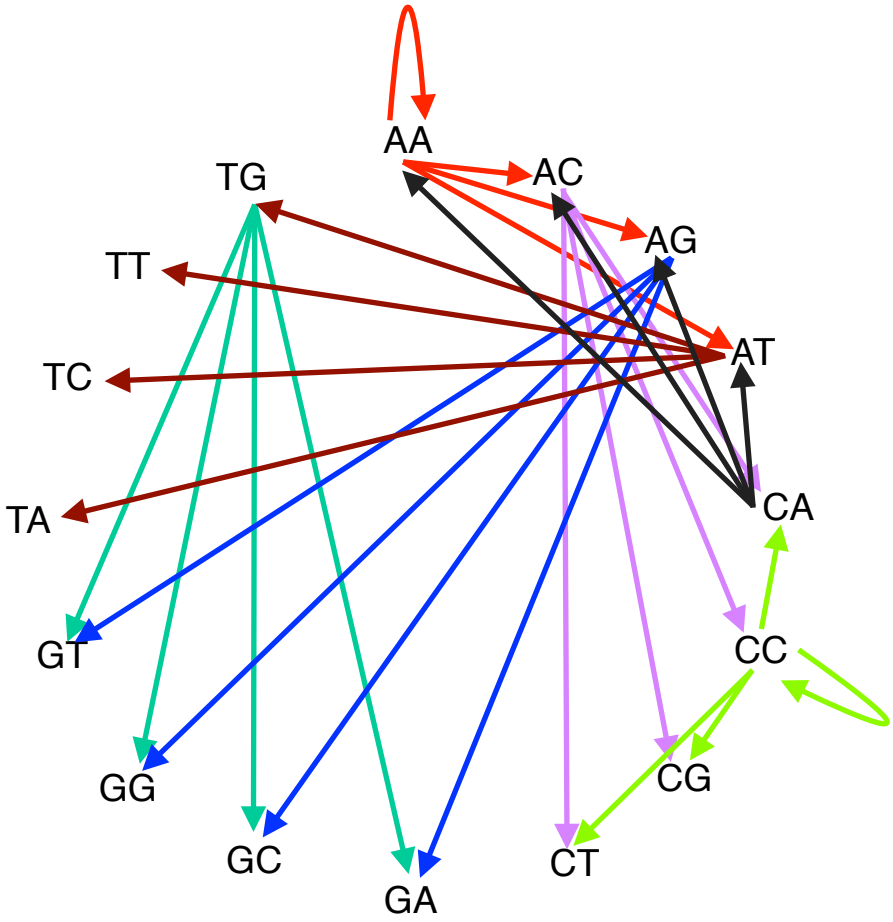
de Bruijn graphs for DNA sequences



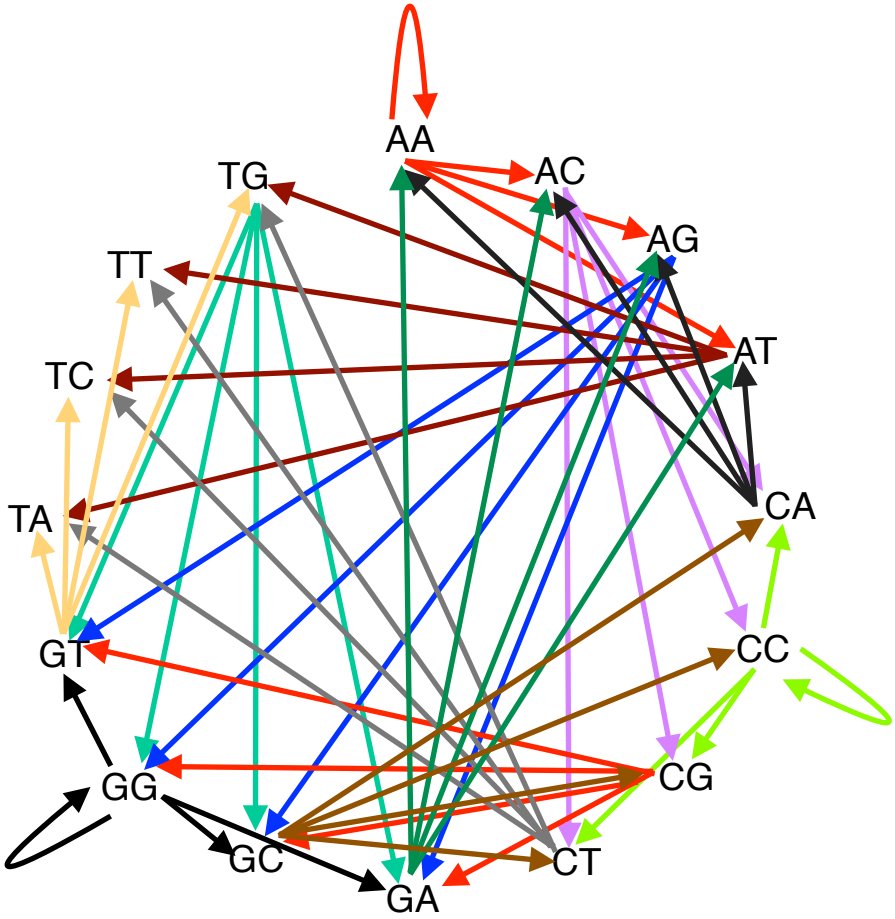
de Bruijn graphs for DNA sequences



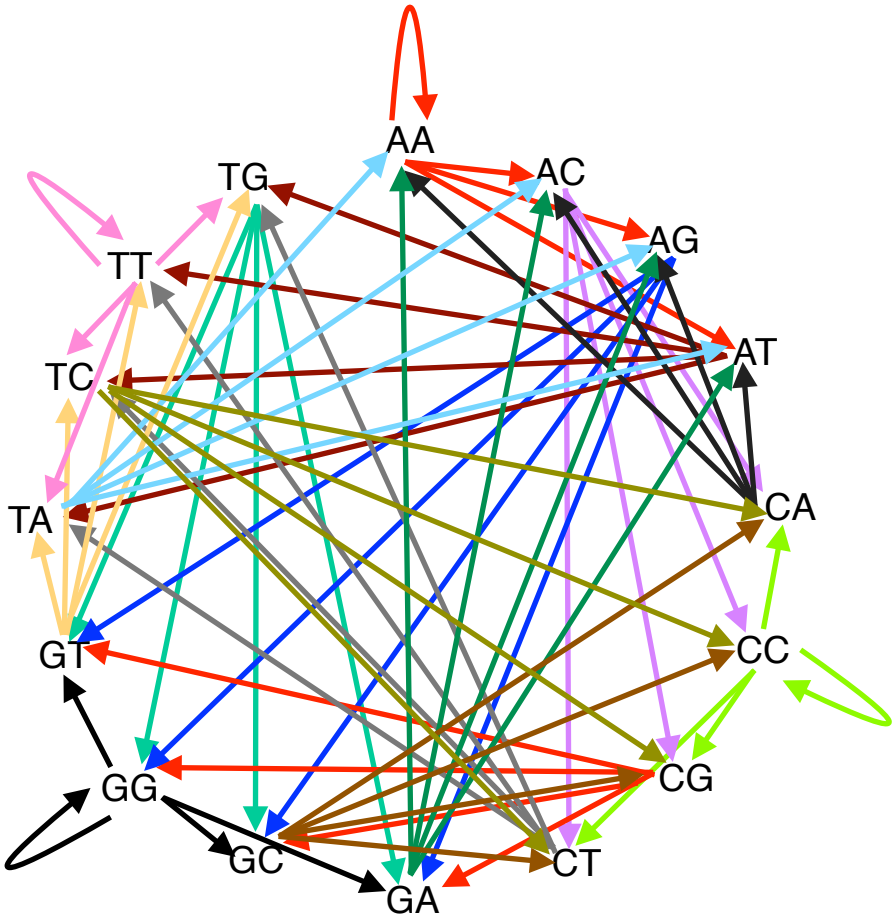
de Bruijn graphs for DNA sequences



de Bruijn graphs for DNA sequences



de Bruijn graphs for DNA sequences



The corresponding de Bruijn sequence is 64-letter long,
see e.g. https://jgeisler0303.github.io/deBruijnDecode/

Using de Bruijn graphs for genome assembly

JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 2, Number 2, 1995
Mary Ann Liebert, Inc.
Pp. 291–306

A New Algorithm for DNA Sequence Assembly

RAMANA M. IDURY and MICHAEL S. WATERMAN

ABSTRACT

Since the advent of rapid DNA sequencing methods in 1976, scientists have had the problem of inferring DNA sequences from sequenced fragments. Shotgun sequencing is a well-established biological and computational method used in practice. Many conventional algorithms for shotgun sequencing are based on the notion of pairwise fragment overlap. While shotgun sequencing infers a DNA sequence given the sequences of overlapping fragments, a recent and complementary method, called sequencing by hybridization (SBH), infers a DNA sequence given the set of oligomers that represents all subwords of some fixed length, *k*. In this paper, we propose a new computer algorithm for DNA sequence assembly that combines in a novel way the techniques of both shotgun and SBH methods. Based on our preliminary investigations, the algorithm promises to be very fast and practical for DNA sequence assembly.

Edge-centric DBGs

In genomics, we use de Bruijn graphs (DBGs) to represent the overlaps between the k-mers **present in a sequence or set of reads**. This is different from the de Bruijn graphs we looked at until this point, which comprise all possible k-mers in a given alphabet are are sometimes called for this reason “full de Bruijn graphs”.

The most frequent representation of DBGs is called **edge-centric DBGs**, the edges of which are the k-mers present in the initial sequence(s), whereas the nodes are all their possible (k-1)-length overlaps (i.e., all their left and right (k-1)-mers). In edge-centric DBGs nodes can be deduced from edges, i.e. it is sufficient to store the edges as a representation of the DBG.

read: ATGAT

3-mers: ATG, TGA, GAT
Left and right 2-mers: AT→TG; TG→GA; GA→AT

<u>edge-centric DBG for k=3:</u>

edge-centric de Bruijn graph with 3-mers as edges
and all possible 2-letters overlaps as nodes

4-mers: ATGA, TGAT
Left and right 3-mers: ATG→TGA; TGA→GAT

<u>edge-centric DBG for k=4:</u>

edge-centric de Bruijn graph with 4-mers as edges
and all possible 3-letters overlaps as nodes

Why DBGs?

To assemble a genome using an **edge-centric DBG**, one needs to find a **generalized Eulerian path** through the graph.

Unlike finding Hamiltonian paths on an overlap graph, which is NP-hard, the problem of finding an Eulerian path in a graph is part of P, i.e. can be solved in polynomial time (actually, linear time).

Medvedev & Pop (2021) What do Eulerian and Hamiltonian cycles have to do with genome assembly? *PLOS Computational Biology* 17:e1008928

A practical example of de Bruijn graph construction

Let us build a de Bruijn graph for the following set of reads for k=3

ATTATAT CGCGTAC ATTGCGC GCATTAT ACGGCGC TATATTG GTACGGC GCGTACG ATATTGC

3-mers in the reads (each arrow represents one 3-mer edge in the DBG):

AT→TT→TA→AT→TA→AT

CG→GC→CG→GT→TA→AC

AT→TT→TG→GC→CG→GC

GC→CA→AT→TT→TA→AT

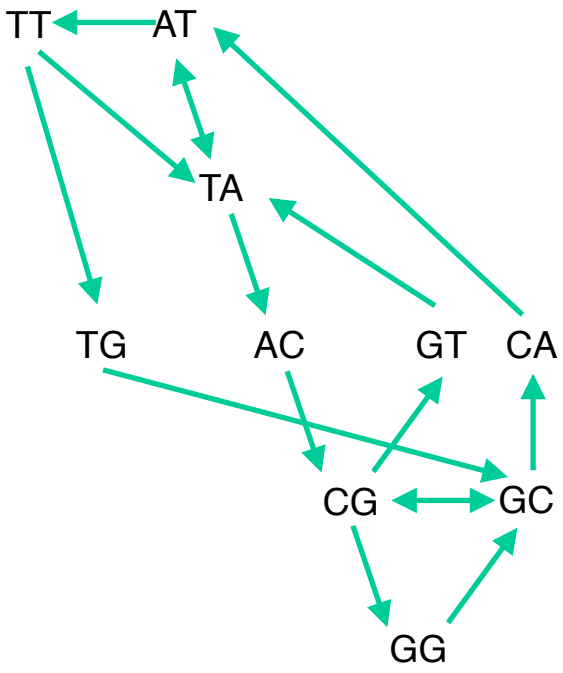
AC→CG→GG→GC→CG→GC

TA→AT→TA→AT→TT→TG

GT→TA→AC→CG→GG→GC

GC→CG→GT→TA→AC→CG

AT→TA→AT→TT→TG→GC

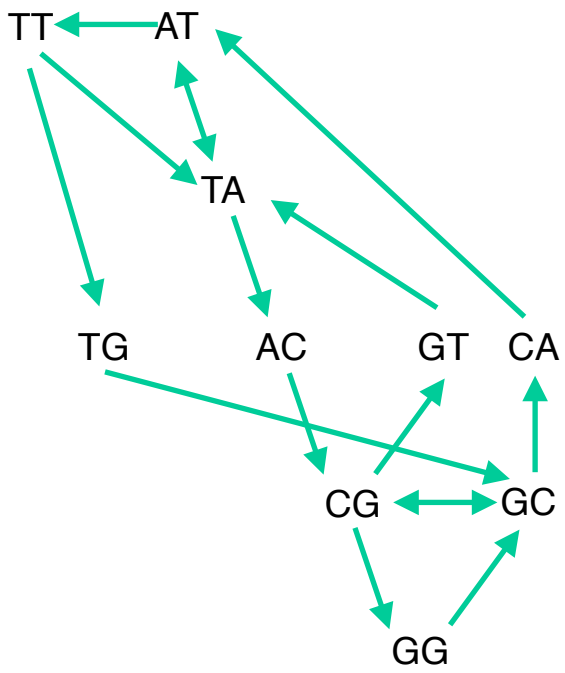


A practical example of de Bruijn graph construction

Let us build a de Bruijn graph for the following set of reads for k=3

ATTATAT CGCGTAC ATTGCGC GCATTAT ACGGCGC TATATTG GTACGGC GCGTACG ATATTGC

Problem: there are usually several generalised Eulerian paths for a given graph, some of which are inconsistent with the initial set of reads. For this reason, DBG assemblers emit **unitigs** (non-branched elementary paths through the DBG, which can also be found in linear time).



Unitigs: GCAT, TACG, TTGC, CGGC, CGTA

A practical example of de Bruijn graph construction

Same for k=4

ATTATAT CGCGTAC ATTGCGC GCATTAT ACGGCGC TATATTG GTACGGC GCGTACG ATATTGC

suites of 4-mers in the reads (each arrow represents one 4-mer edge in the DBG):

ATT→TTA→TAT→ATA→TAT

CGC→GCG→CGT→GTA→TAC

ATT→TTG→TGC→GCG→CGC

GCA→CAT→ATT→TTA→TAT

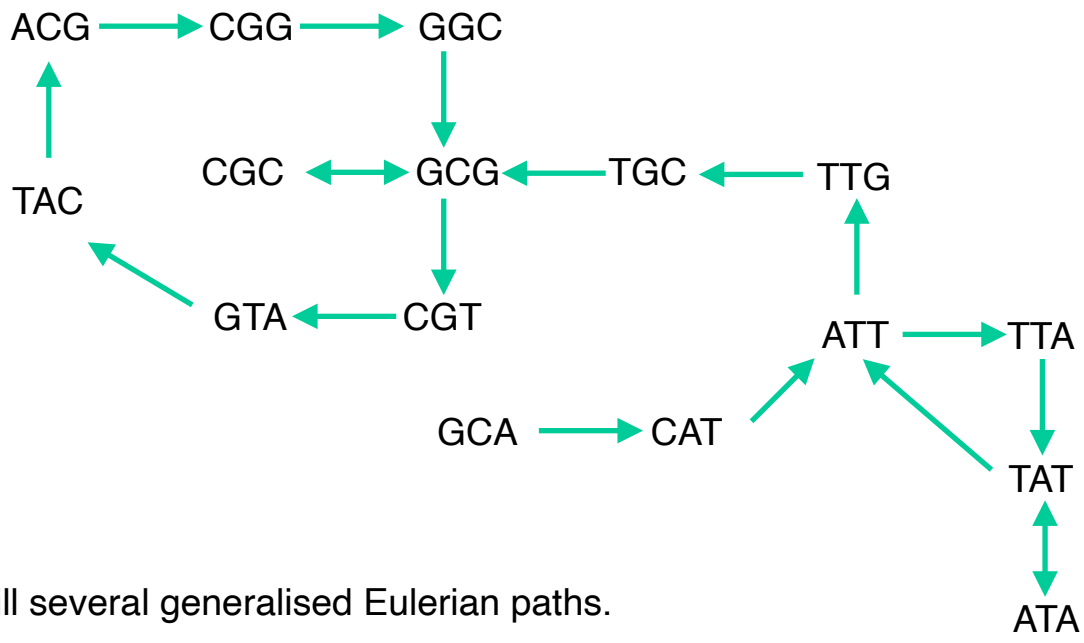
ACG→CGG→GGC→GCG→CGC

TAT→ATA→TAT→ATT→TTG

GTA→TAC→ACG→CGG→GGC

GCG→CGT→GTA→TAC→ACG

ATA→TAT→ATT→TTG→TGC



Here there are still several generalised Eulerian paths.

A practical example of de Bruijn graph construction

k=4

ATTATAT CGCGTAC ATTGCGC GCATTAT ACGGCGC TATATTG GTACGGC GCGTACG ATATTGC

suites of 4-mers in the reads (each arrow represents one 4-mer edge in the DBG):

ATT→TTA→TAT→ATA→TAT

CGC→GCG→CGT→GTA→TAC

ATT→TTG→TGC→GCG→CGC

GCA→CAT→ATT→TTA→TAT

ACG→CGG→GGC→GCG→CGC

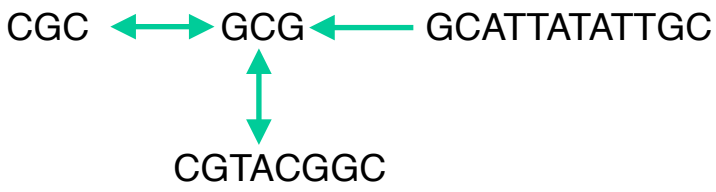
TAT→ATA→TAT→ATT→TTG

GTA→TAC→ACG→CGG→GGC

GCG→CGT→GTA→TAC→ACG

ATA→TAT→ATT→TTG→TGC

After compaction:



Because of the repeat GCG, there are two possible generalised Eulerian paths:
GCATTATATT**GCGCG**TACG**GCG**
GCATTATATT**GCG**TACG**GCGCG**

Unitigs (non-branched elementary paths through the de Bruijn graph):
GCATTATATT**GCG GCGCG GCG**TACGGCG

Exercise 6: constructing and compacting de Bruijn graphs

Assemble the same series of reads for k=5

ATTATAT CGCGTAC ATTGCGC GCATTAT ACGGCGC TATATTG GTACGGC GCGTACG ATATTGC

How many paths does this DBG yield?

Some assemblers (such as MEGAHIT) perform assemblies using several increasing k values, considering at each step the unitigs inferred from k<n as additional reads when constructing the k=n DBG…

Let's try it: how many paths do you obtain if you build a DBG for k=5 taking into account the reads + the unitigs obtained from the k=4 DBG?

ATTATAT CGCGTAC ATTGCGC GCATTAT ACGGCGC TATATTG GTACGGC GCGTACG ATATTGC

+ GCATTATATTGCG GCGCG GCGTACGGCG

Exercise 6: solution

k=5

ATTATAT CGCGTAC ATTGCGC GCATTAT ACGGCGC TATATTG GTACGGC GCGTACG ATATTGC

suites of 5-mers in the reads (each arrow represents one 5-mer edge in the DBG):

ATTA→TTAT→TATA→ATAT

CGCG→GCGT→CGTA→GTAC

ATTG→TTGC→TGCG→GCGC

GCAT→CATT→ATTA→TTAT

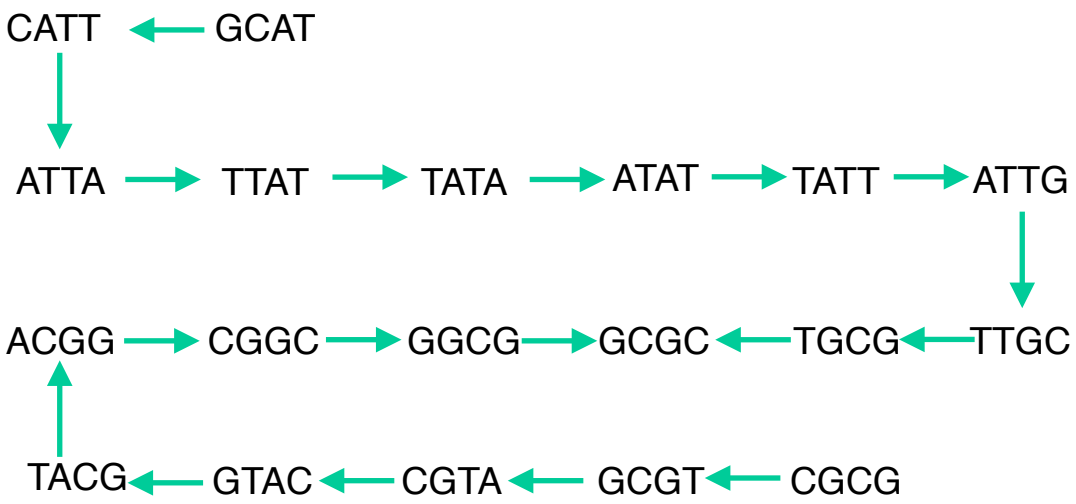
ACGG→CGGC→GGCG→GCGC

TATA→ATAT→TATT→ATTG

GTAC→TACG→ACGG→CGGC

GCGT→CGTA→GTAC→TACG

ATAT→TATT→ATTG→TTGC



After compaction: GCATTATATTGCG → GCGC ← CGCGTACGGCG

Here the DBG yields two paths. This is because the chosen k was too high given the number of reads (i.e., the coverage depth), resulting in missing k-mers.

unitigs: GCATTATATTGCGC & CGCGTACGGCGC

Exercise 6: solution

k=5 + unitigs obtained from k=4

ATTATAT CGCGTAC ATTGCGC GCATTAT ACGGCGC TATATTG GTACGGC GCGTACG ATATTGC
+ GCATTATATTGCG GCGCG GCGTACGGCG

suites of 5-mers in the reads:

ATTA→TTAT→TATA→ATAT

CGCG→GCGT→CGTA→GTAC

ATTG→TTGC→TGCG→GCGC

GCAT→CATT→ATTA→TTAT

ACGG→CGGC→GGCG→GCGC

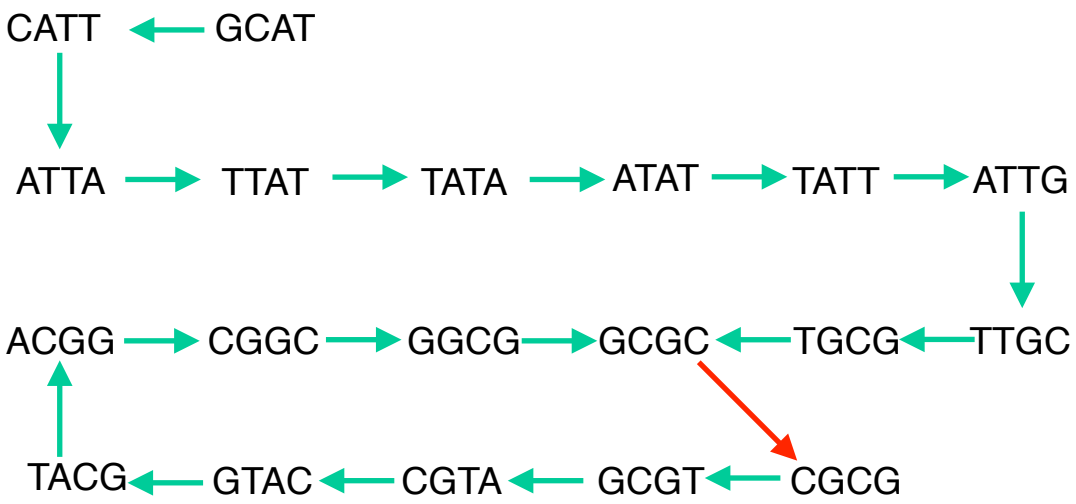
TATA→ATAT→TATT→ATTG

GTAC→TACG→ACGG→CGGC

GCGT→CGTA→GTAC→TACG

ATAT→TATT→ATTG→TTGC

GCGC→CGCG



Now the DBG yields only a single Eulerian path (thanks to the addition of the paths inferred from a smaller k value).

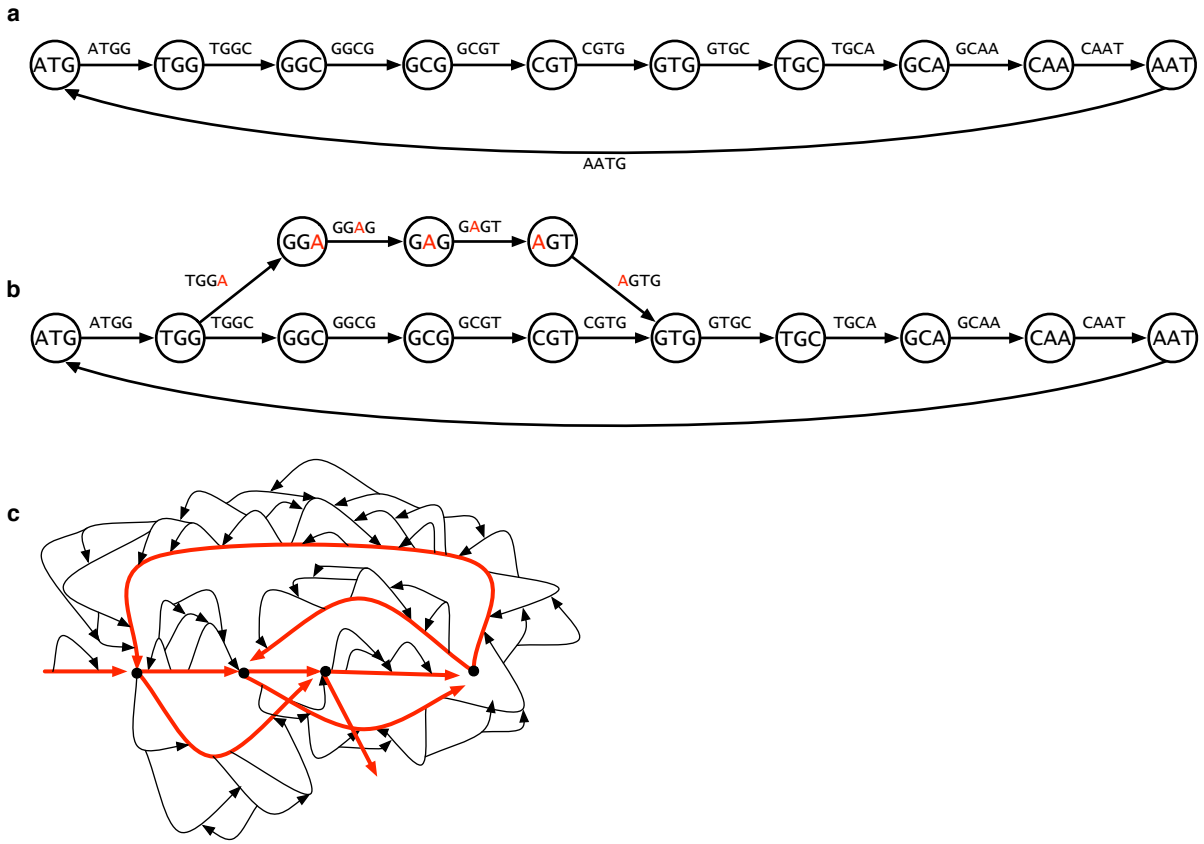
unitig: GCATTATATTGCGCGTACGGCGC

The problem with de Bruijn graphs

de Bruijn graph assemblers are much faster than OLC assemblers when dealing with large amount of reads. For this reason, most present-day assemblers for Illumina data use a de Bruijn-graph approach (ALLPATHS-LG, Minia, IDBA_UD, SPAdes, Platanus,…).

However, because reads are decomposed into k-mers, it is more difficult to solve repeats. Increasing the value of k improves the situation by resolving the repeats that are shorted than k bases, but then sequencing errors pose problem by creating **a huge number of erroneous k-mers**.

The problem with de Bruijn graphs



Supplementary Figure 1. De Bruijn graph from reads with sequencing errors. (**a**) A de Bruijn graph E on our set of reads with $k = 4$. Finding an Eulerian cycle is already a straightforward task, but for this value of k, it is trivial. (**b**) If TGG***A***GTG is incorrectly sequenced as a sixth read (in addition to the correct TGG***C***GTG read), then the result is a *bulge* in the de Bruijn graph, which complicates assembly. (**c**) An illustration of a de Bruijn graph E with many bulges. The process of bulge removal should leave only the red edges remaining, yielding an Eulerian path in the resulting graph.

Compeau (2012) *Nature Biotechnology*

Error correction

How can we try to reduce the amount of errors in the reads before downstream analyses?

As an analogy, how would you design a spell-checker for a language that you do not know?

Error correction using k-mer spectra

How can we try to reduce the amount of errors in the reads before downstream analyses?

As an analogy, how would you design a spell-checker for a language that you do not know?

Answer: build a dictionary (here, a **k-mer** dictionary). Words that are encountered only once are most probably errors!

An Eulerian path approach to DNA fragment assembly

Pavel A. Pevzner*, Haixu Tang†, and Michael S. Waterman†‡§

*Department of Computer Science and Engineering, University of California, San Diego, La Jolla, CA; and Departments of †Mathematics and ‡Biological Sciences, University of Southern California, Los Angeles, CA

Contributed by Michael S. Waterman, June 7, 2001

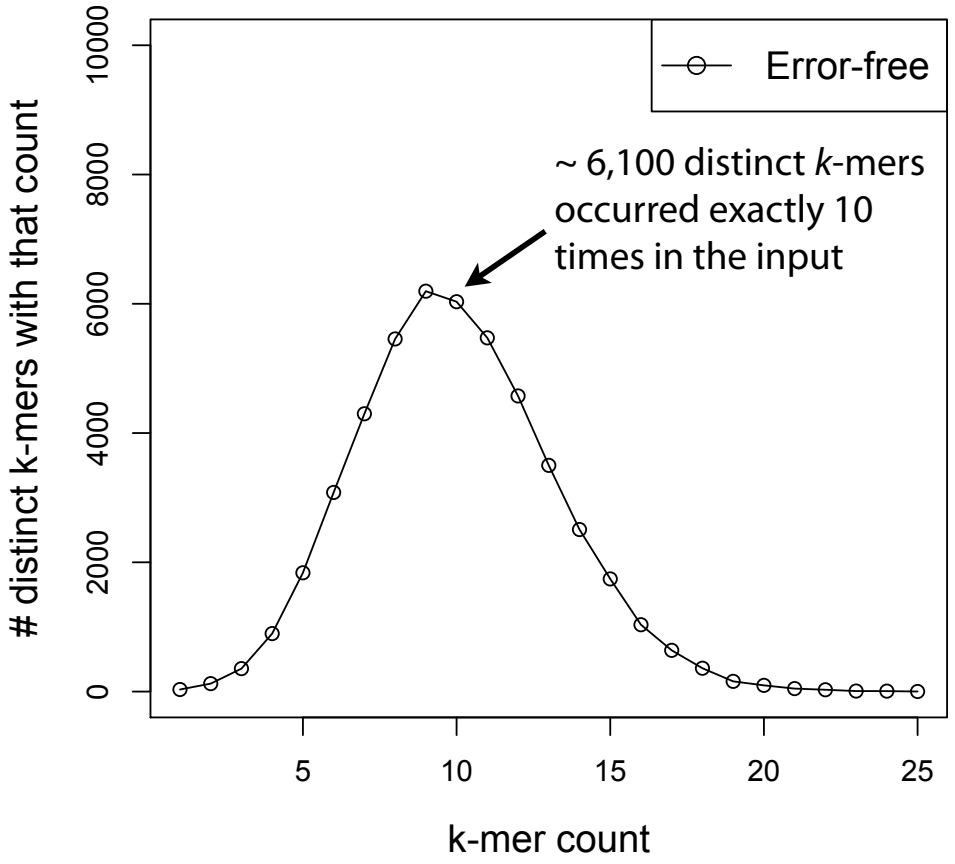
9748–9753 | PNAS | **August 14, 2001** | vol. 98 | no. 17

Error correction using k-mer spectra

Say we have error-free sequencing reads drawn from a genome.
The amount of sequencing is such that average coverage = 200.
Let $k = 20$

How would the picture change for data with 1% error rate?

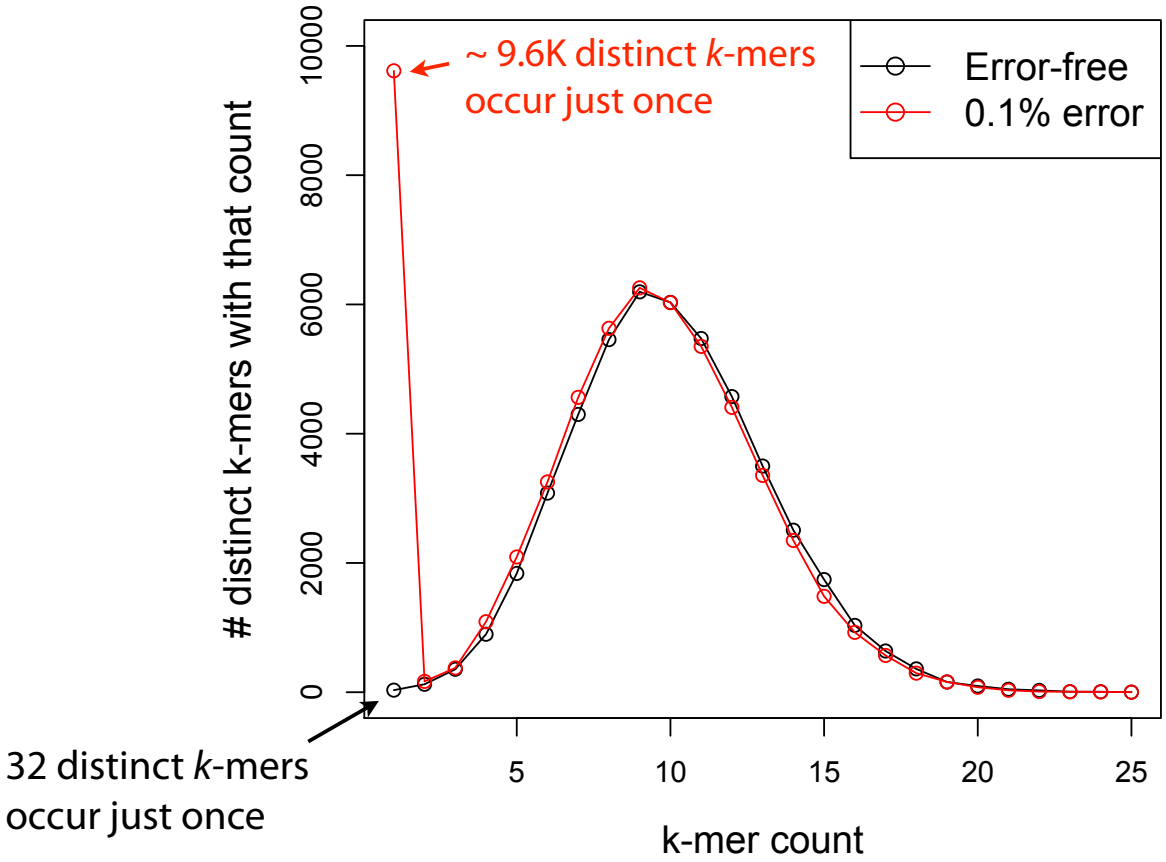
Hint: errors usually change high-count *k*-mer into low-count *k*-mer



http://www.langmead-lab.org/teaching-materials/

Error correction using k-mer spectra

k-mers with errors usually occur fewer times than error-free *k*-mers

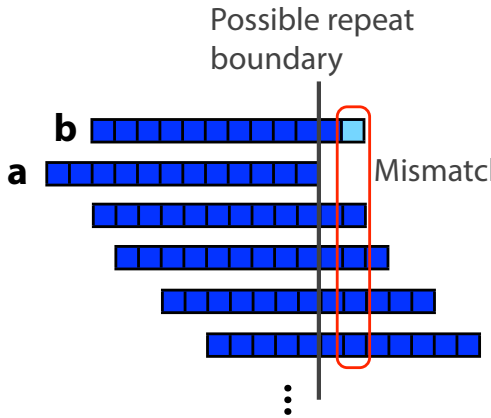


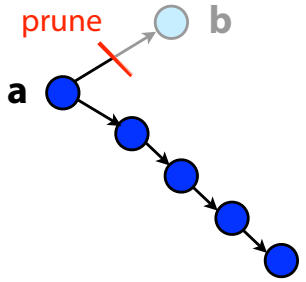
http://www.langmead-lab.org/teaching-materials/

Error correction using graphs

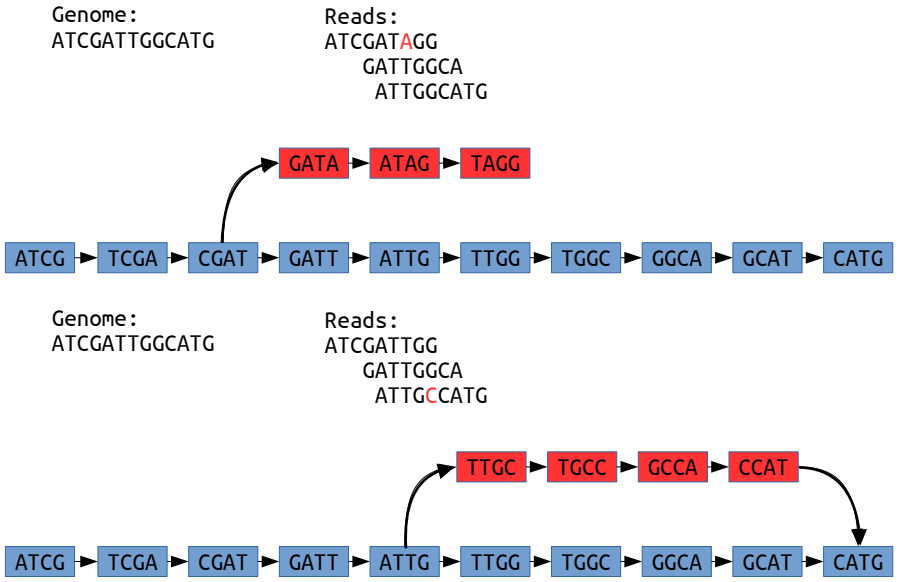
Using overlap graphs:

http://www.langmead-lab.org/teaching-materials/





Using DBGs:



A tip = a unitig of length inferior to 2(k-1) that has no successor at one of its extremities
Limasset et al. (2020) *Bioinformatics*

Bcool: a DBG-based approach to correct reads

Bioinformatics, 36(5), 2020, 1374–1381
doi: 10.1093/bioinformatics/btz102
Advance Access Publication Date: 20 February 2019
Original Paper

OXFORD

Sequence analysis

Toward perfect reads: self-correction of short reads via mapping on de Bruijn graphs

Antoine Limasset [1,*], Jean-François Flot [1,2,†] and Pierre Peterlongo [3,†]

[1]Evolutionary Biology & Ecology, Université Libre de Bruxelles (ULB), Bruxelles, Belgium, [2]Interuniversity Institute of Bioinformatics in Brussels – (IB)[2], Brussels, Belgium and [3]Inria, CNRS, University of Rennes, IRISA, Rennes, France

*To whom correspondence should be addressed.
[†]The authors wish it to be known that, in their opinion, the last two authors should be regarded as Joint Last Authors.

Associate Editor: Alfonso Valencia

Received on July 23, 2018; revised on January 7, 2019; editorial decision on February 7, 2019; accepted on February 18, 2019

Abstract

Motivation: Short-read accuracy is important for downstream analyses such as genome assembly and hybrid long-read correction. Despite much work on short-read correction, present-day correctors either do not scale well on large datasets or consider reads as mere suites of *k*-mers, without taking into account their full-length sequence information.
Results: We propose a new method to correct short reads using de Bruijn graphs and implement it as a tool called Bcool. As a first step, Bcool constructs a compacted de Bruijn graph from the reads. This graph is filtered on the basis of *k*-mer abundance then of unitig abundance, thereby removing most sequencing errors. The cleaned graph is then used as a reference on which the reads are mapped to correct them. We show that this approach yields more accurate reads than *k*-mer-spectrum correctors while being scalable to human-size genomic datasets and beyond.
Availability and implementation: The implementation is open source, available at http://github.com/Malfoy/BCOOL under the Affero GPL license and as a Bioconda package.
Contact: antoine.limasset@gmail.com
Supplementary information: Supplementary data are available at *Bioinformatics* online.

Bcool: a DBG-based approach to correct short reads

Fig. 2. Bcool workflow. The white boxes are FASTA files and the grey boxes represent the tools that process or generate them. ntCard (Mohamadi *et al.*, 2017) is used to select the best-suited *k*-mer size. A compacted DBG is then constructed using Bcalm2 (Chikhi *et al.*, 2016). The Btrim module cleans the graph, and the reads are finally mapped back on the DBG using Bgreat2

Bcool: de Bruijn graph correction from graph alignment

available from Github (https://github.com/Malfoy/BCOOL) and bioconda

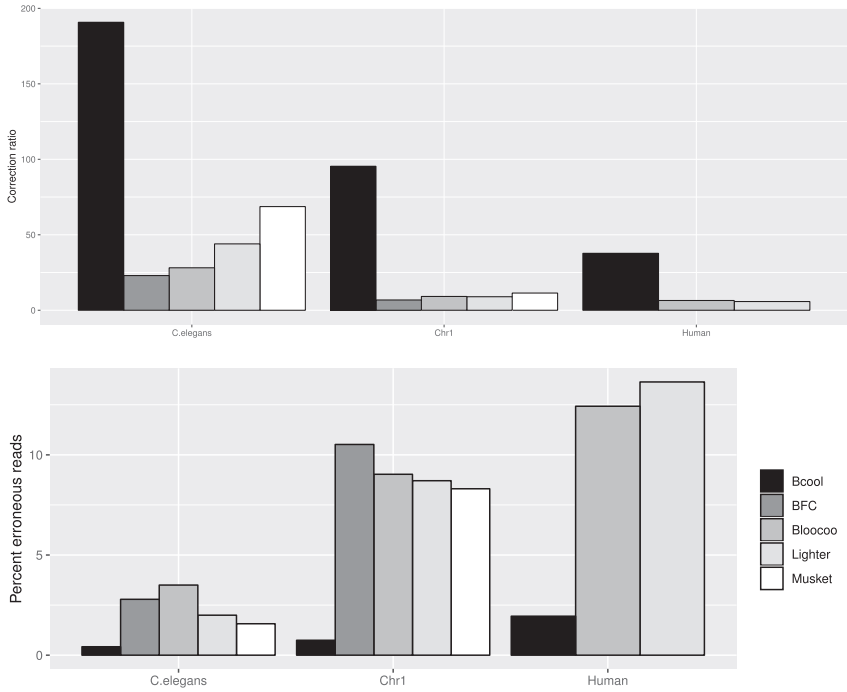


Fig. 5. Correction ratios (top) and percentage of erroneous reads after correction (bottom) for different correctors on our three simulated haploid datasets. We simulated 100× of 150 bp reads with a 1% error rate. BFC and Musket ran out of memory on all full human datasets

Limasset et al. (2020) *Bioinformatics*

What about noisy long reads?

Various approaches have been proposed to adapt the DBG paradigm to noisy reads with ~10% errors.

ABruijn: approximate DBG approach that only use “solid” k-mer that are well represented in the data; implemented in the program **Flye**.

Wtdbg2 (aka **Redbean**): “fuzzy DBG” that allows mismatches and indels, and collapses a set of closely similar k-mers into a single node.

As most errors in long, noisy reads at indels at homopolymers, **Shasta** uses homopolymers compression to reduce the error rate and perform a DBG assembly.

Conclusions and take-home messages

Current-day genomes assembly tools are **powered by mathematical principles** going back decades or even centuries.

From initial **greedy** assemblers, research shifted to **OLC** assemblers at the time of 1st-generation sequencing, then to **DBG** assemblers during 2nd-generation sequencing, then back to **OLC** assemblers with the advent of 3rd-generation sequencing. However, **DBG** assemblers are making a comeback thanks to new computational approaches to deal with erroneous reads and can be expected to gain ground in the future as error rates become further reduced.

It is an exciting time to work on genome assembly!


Das Ende
הסוף
Fin
結束
จบ
Wakas
That's all Folks!
終わり
النهاية
кінець
끝
Finito