Algorithms in Bioinformatics

De Novo Genome Assembly and Short Read Mapping

Carlo M. Lapid

Core Facility for Bioinformatics
Philippine Genome Center

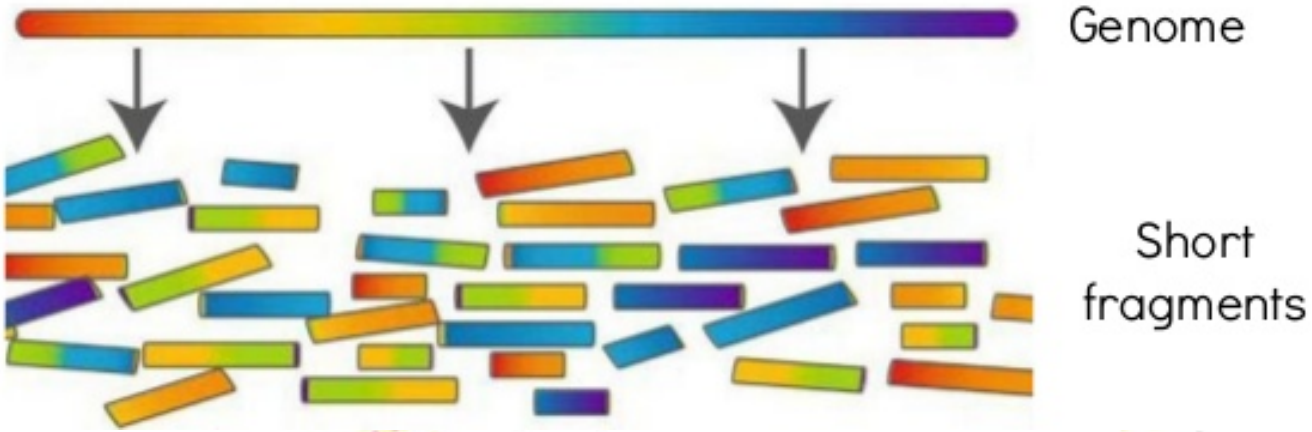
PGC
PHILIPPINE GENOME CENTER

Algorithms

1. De novo genome assembly
2. Mapping short reads to a reference genome

What is assembly?

Next-generation sequencing creates millions of short, unorganized reads with no clear biological meaning or context.



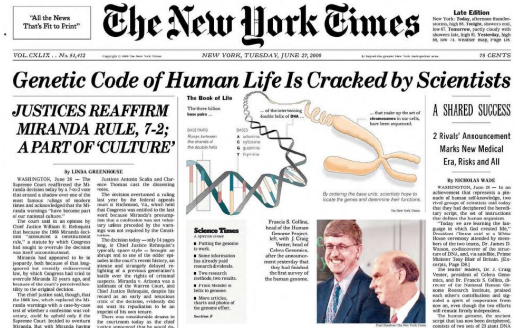
Source: https://www.slideshare.net/torstenseemann/de-novo-genome-assembly-tseemann-imb-winter-school-2016-brisbane-au-4-july-2016

PGC
PHILIPPINE GENOME CENTER

What is assembly?

Copies of genome

Sequencing reads



Assembled genome

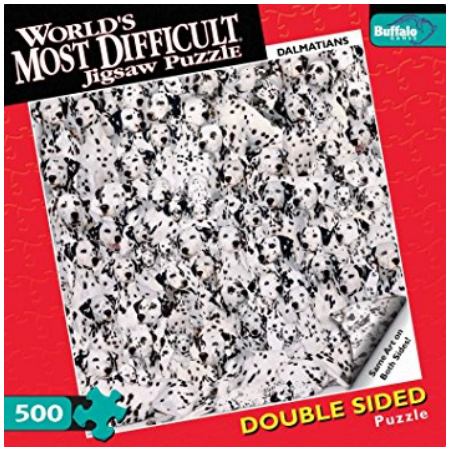
PGC
PHILIPPINE GENOME CENTER

What is assembly?

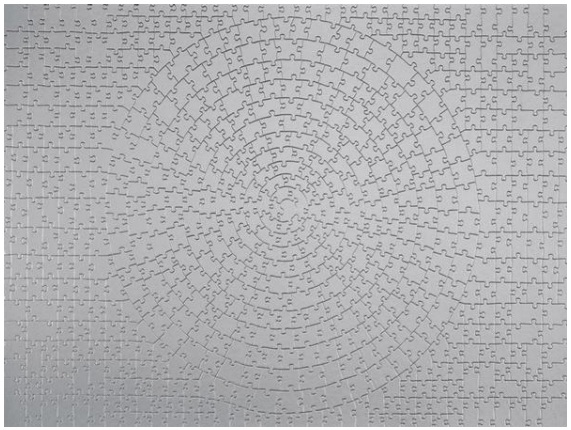
- De novo genome assembly is like solving a jigsaw puzzle...
  - with millions of pieces,
  - multiple copies of each piece,
  - pieces that are missing,
  - pieces that fit together but shouldn't,
  - pieces that don't fit together but should
  - pieces that don't belong to the puzzle,
  - and no picture on the box.

PGC
PHILIPPINE GENOME CENTER

What is assembly?



Reference-based

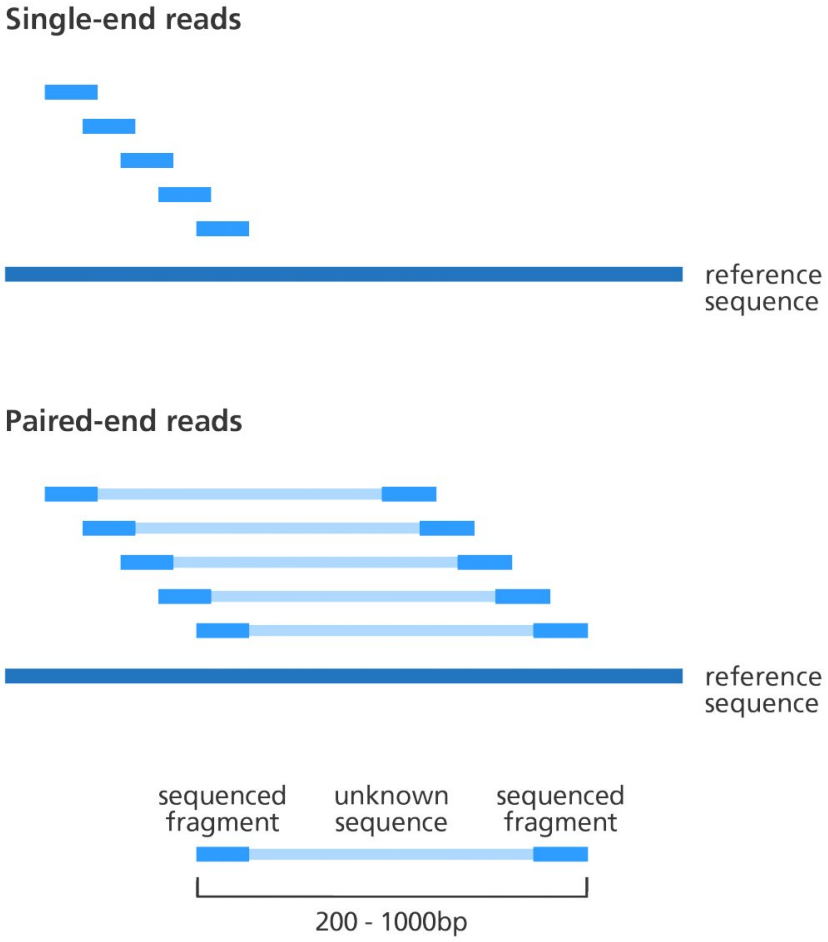


De novo

- Assembly can be performed by mapping reads to a known reference genome, or accomplished *de novo*, i.e. “from the beginning”.
- Why *de novo*?
  - Sometimes an appropriate reference genome isn't available
  - Inappropriate reference genome can mislead assembly
  - *De novo* techniques can uncover variation that might be absent in a reference genome

PGC
PHILIPPINE GENOME CENTER

De novo assembly



Depending on the sequencing protocol and technology, sequencing reads can be *single-end* or *paired-end*.

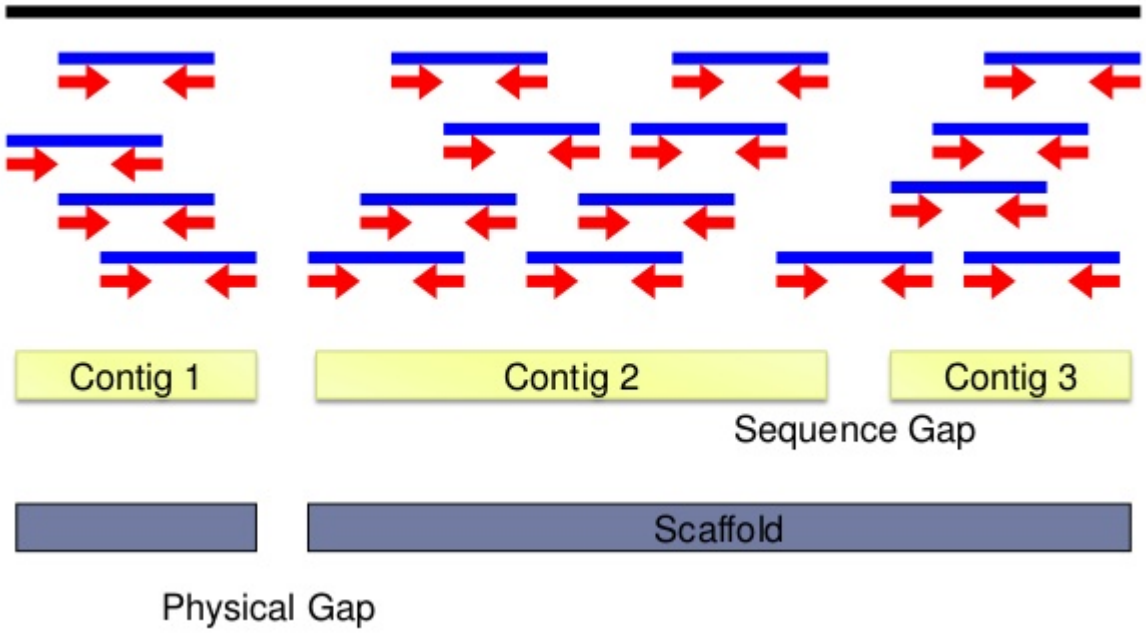
Paired-end reads are sequenced from opposite ends of the same DNA fragment, and the approximate genomic distance between them is known.

Source: https://www.yourgenome.org/facts/how-do-you-put-a-genome-back-together-after-sequencing

De novo assembly

Contigs are contiguous sequences assembled from overlapping reads.

Scaffolds are assemblies of contigs, separated by unsequenced gaps of approximately known size.



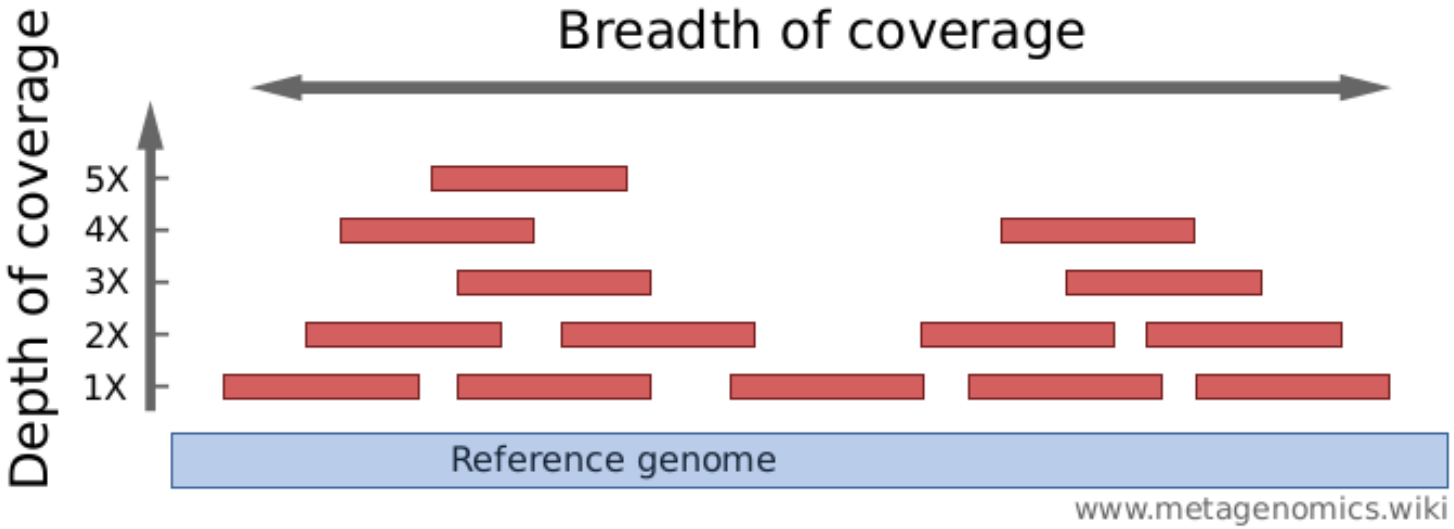
Source: https://github.com/Ecological-and-Evolutionary-Genomics/eeg2016/wiki/Mar-21-Exercise-7----SPAdes-assembler

PHILIPPINE GENOME CENTER

De novo assembly

Depth of coverage refers to how often a base of a genome is sequenced, i.e. “covered” by reads.

Breadth of coverage is the percentage of bases of the target genome covered with a certain depth.



Source: http://www.metagenomics.wiki/pdf/definition/coverage-read-depth

PGC
PHILIPPINE GENOME CENTER

Assembly: an example

Exercise: Try to assemble these five reads into a contig.

Read 1: ACGGATAC

Read 2: GTCCTTGC

Read 3: ATACAGGA

Read 4: CATAGTCC

Read 5: AGGACATA

PGC
PHILIPPINE GENOME CENTER

Assembly: an example

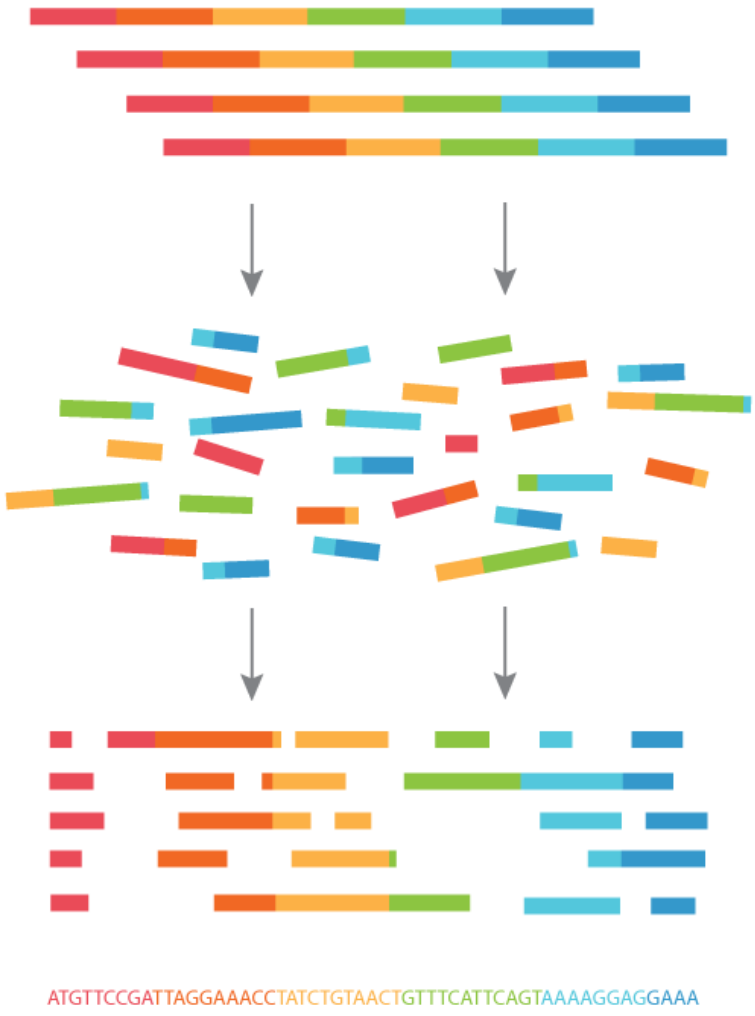
```
Read 1:       ACGGATAC
Read 3:            ATACAGGA
Read 5:                 AGGACATA
Read 4:                      CATAGTCC
Read 2:                           GTCCTTGC

Assembly:     ACGGATACAGGACATAGTCCTTGC
```

What steps did you instinctively take to arrive at this solution?

PGC
PHILIPPINE GENOME CENTER

De novo assembly



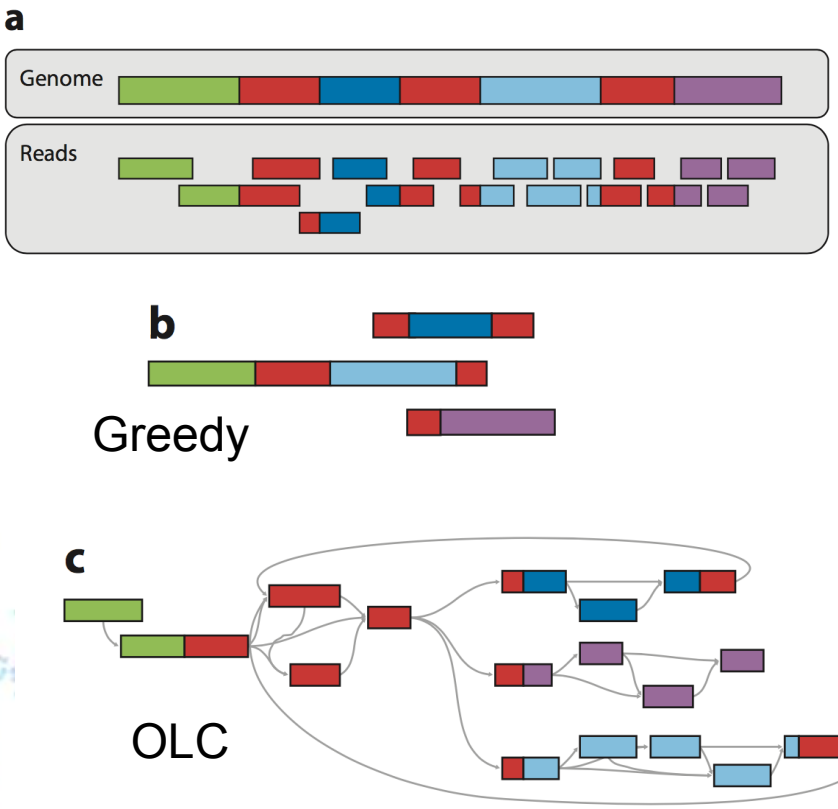
At its core, *de novo* assembly is about looking at how reads (or parts of reads) overlap with each other to determine their relative order and orientation with respect to the original genome.

Source: https://www.abmgood.com/marketing/knowledge_base/next_generation_sequencing_data_analysis.php

PGC
PHILIPPINE GENOME CENTER

De novo assembly approaches

- Greedy approaches
- Graph-based approaches
  - Overlap-Layout-Consensus (OLC)
    - String Graphs (variant)
  - De Bruijn Graphs



Source: Simpson, Jared T., and Mihai Pop. Annual review of genomics and human genetics 16 (2015): 153-172.

PHILIPPINE GENOME CENTER

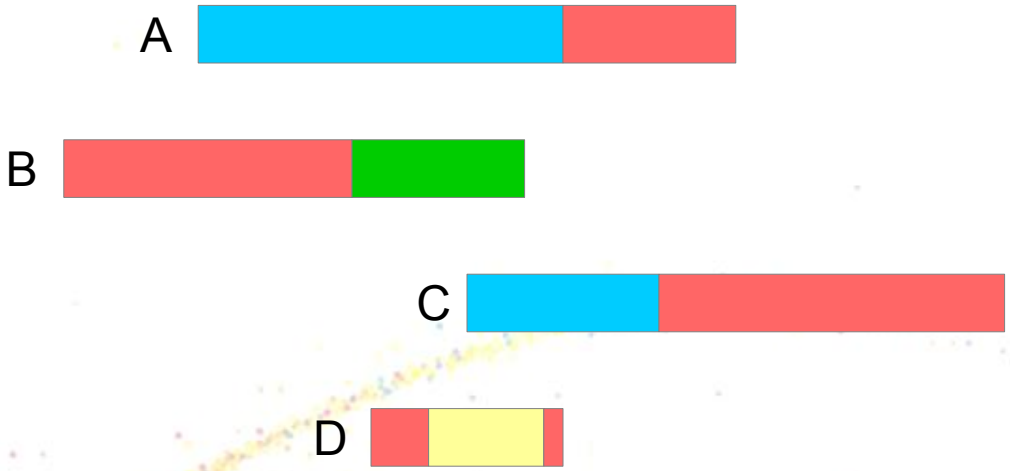
Greedy approaches

- Procedure:
  1. Sequences (reads/contigs)
  2. All-vs-all pairwise comparison
  3. Find pair with best match/overlap
  4. Collapse/assemble pair into a single contig
  5. Repeat

- Very simple
- Makes the most locally optimal (“greedy”) choice at each step
- Used to great success in early sequencing projects like the Human Genome Project

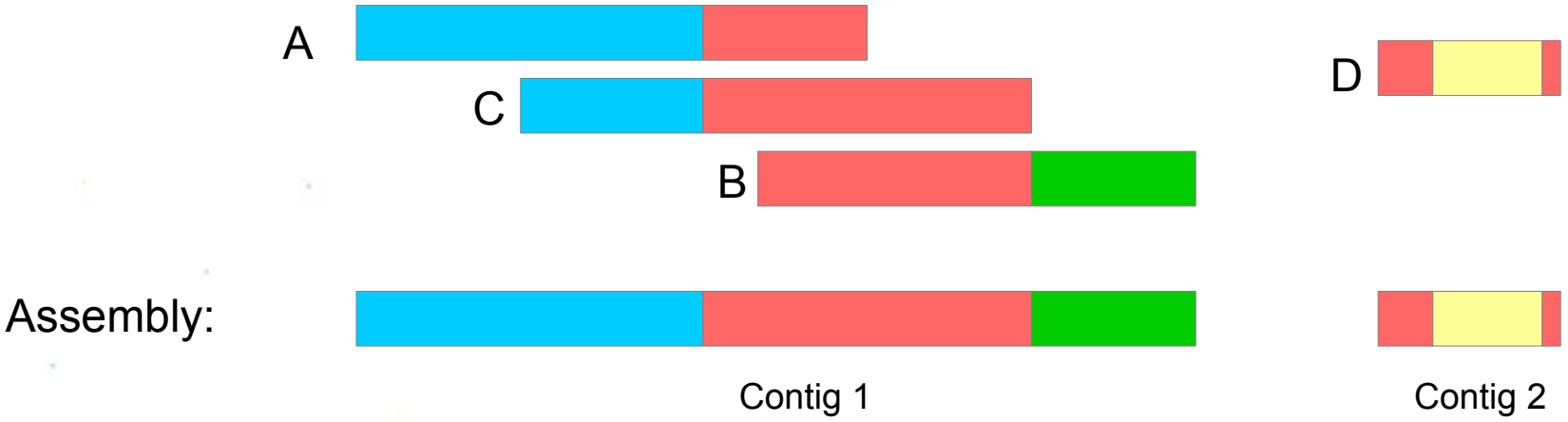
Greedy approaches

Exercise: Assemble these four contigs using a greedy approach.



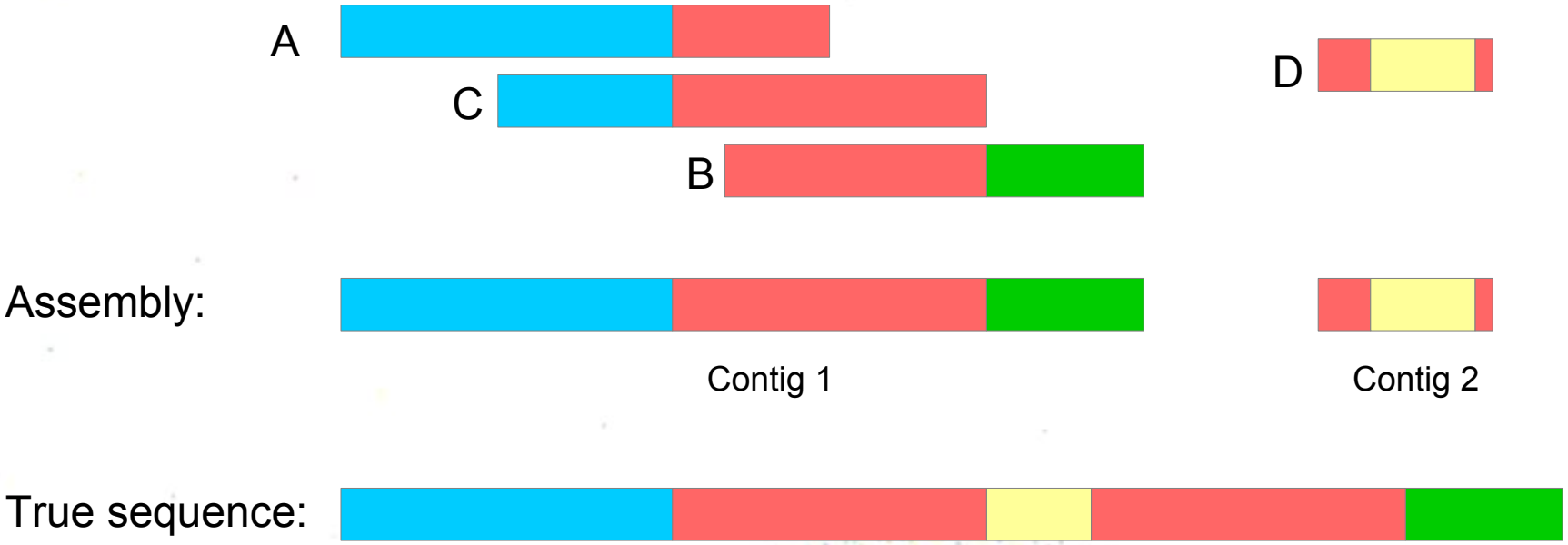
PGC
PHILIPPINE GENOME CENTER

Greedy approaches



PGC
PHILIPPINE GENOME CENTER

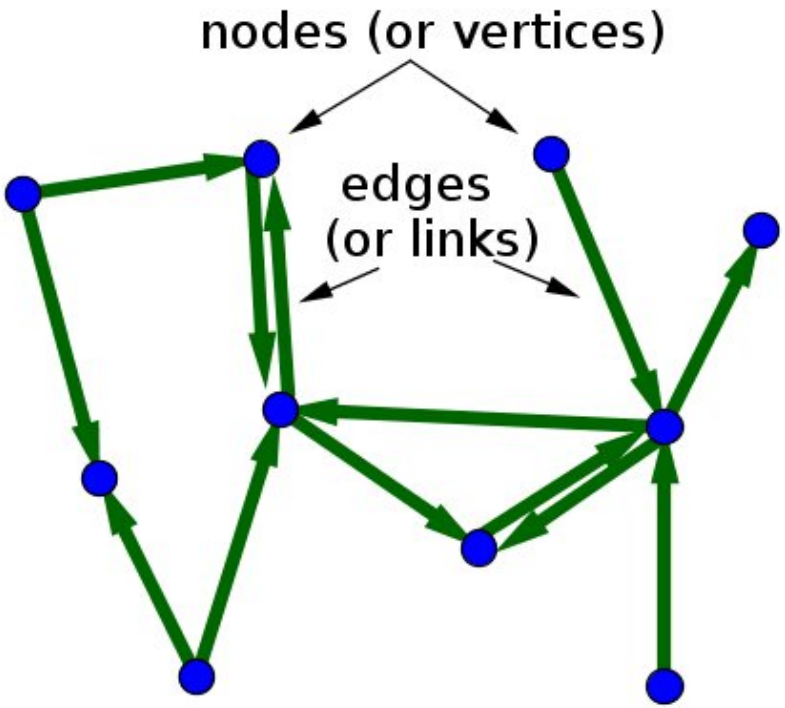
Greedy approaches

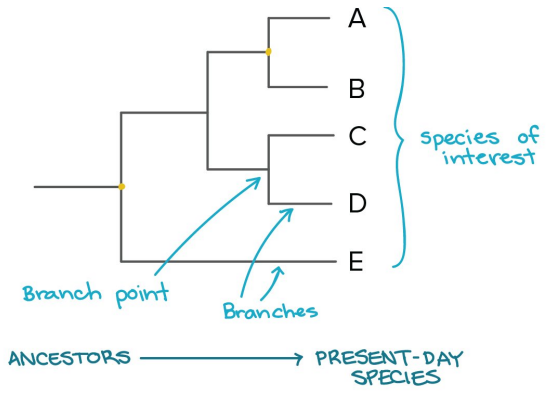


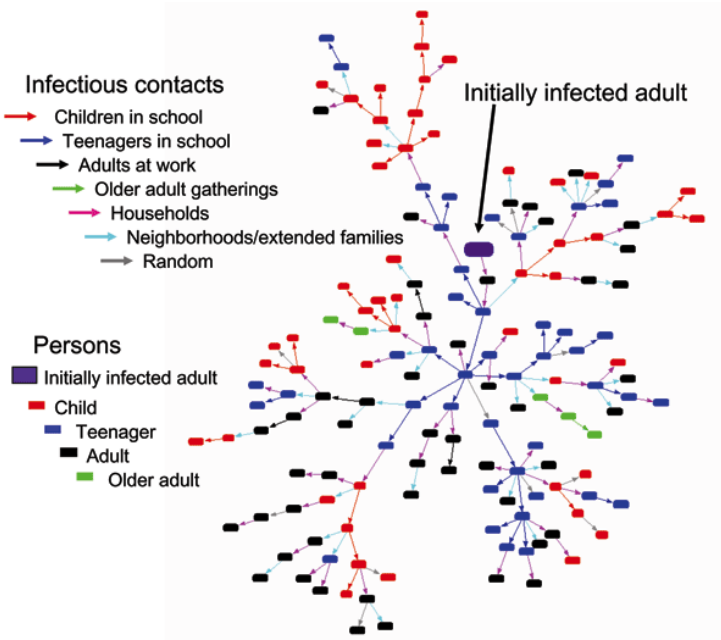
- Repeated genomic regions aren't handled well
- Paired-end information can't be easily used
- NGS produces too many reads to compare all vs all
- Now generally replaced by more complex graph-based algorithms

PGC
PHILIPPINE GENOME CENTER

Graphs







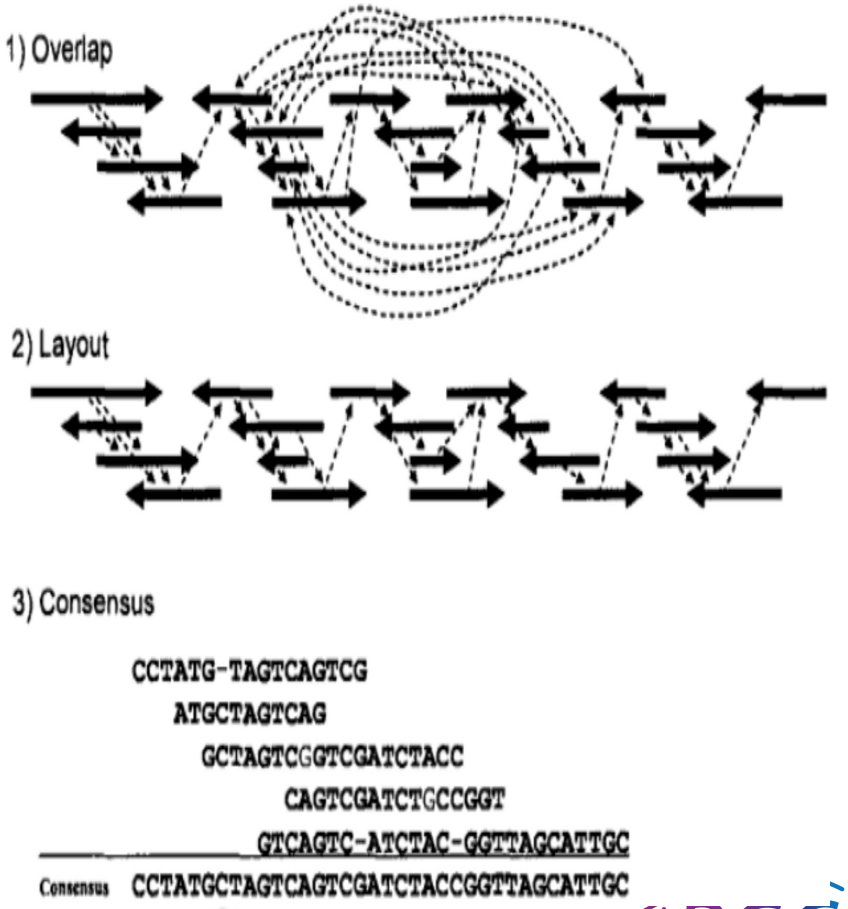
http://mathinsight.org/definition/directed_graph
https://www.khanacademy.org/science/biology/her/tree-of-life/a/phylogenetic-trees
https://wwwnc.cdc.gov/eid/article/12/11/06-0255-f4

PHILIPPINE GENOME CENTER

Overlap-Layout-Consensus

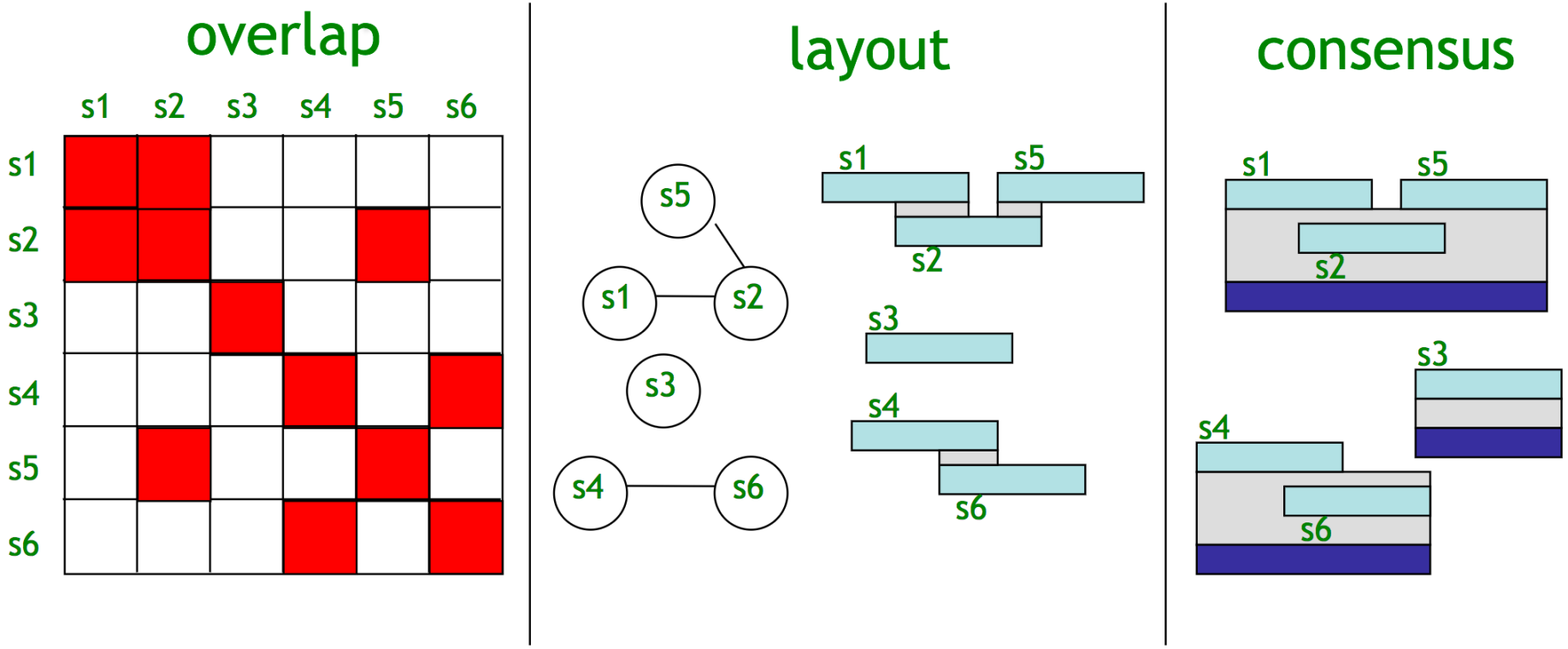
Each sequence is a vertex in the graph. Vertices are linked with an edge if they overlap.

1. Find the overlaps by aligning all the reads with each other.
2. Layout the reads based on which aligns to which.
3. Join all overlapping read sequences to get the consensus.



Source: Masahiro Kasahara, Large-Scale Genome Sequence Processing, Imperial College Press
https://www.slideshare.net/agbiotec/overview-of-genome-assembly-algorithms

PGC
PHILIPPINE GENOME CENTER

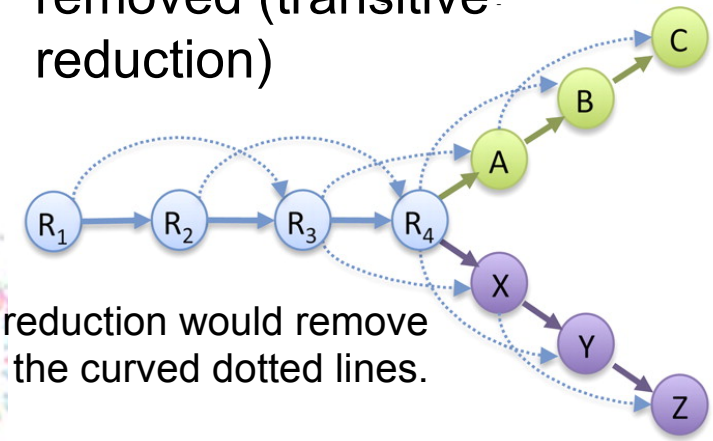


- Like greedy algorithms, OLC requires all-vs-all pairwise comparisons, which is computationally expensive.
- Finding the globally optimum solution during layout is generally impossible. Progressive pairwise alignment needed to estimate.
- Resolves repeats fairly well, using mate-pair constraints.
- Used to great success with earlier genome sequencing projects.
- Generally unsuited to modern short read NGS technologies.

Source: http://slideplayer.com/slide/3253918/

String graphs

- String graphs are made from overlap graphs, with two modifications:
  1. Contained reads – reads that are substrings of some other read – are removed
  2. Transitive edges are removed (transitive reduction)

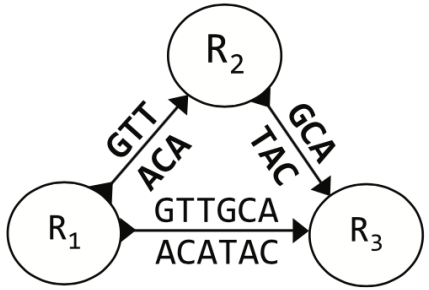


Transitive reduction would remove the curved dotted lines.

A

R_1 ACATACGATACA
R_2 TACGATACAGTT
R_3 GATACAGTTGCA

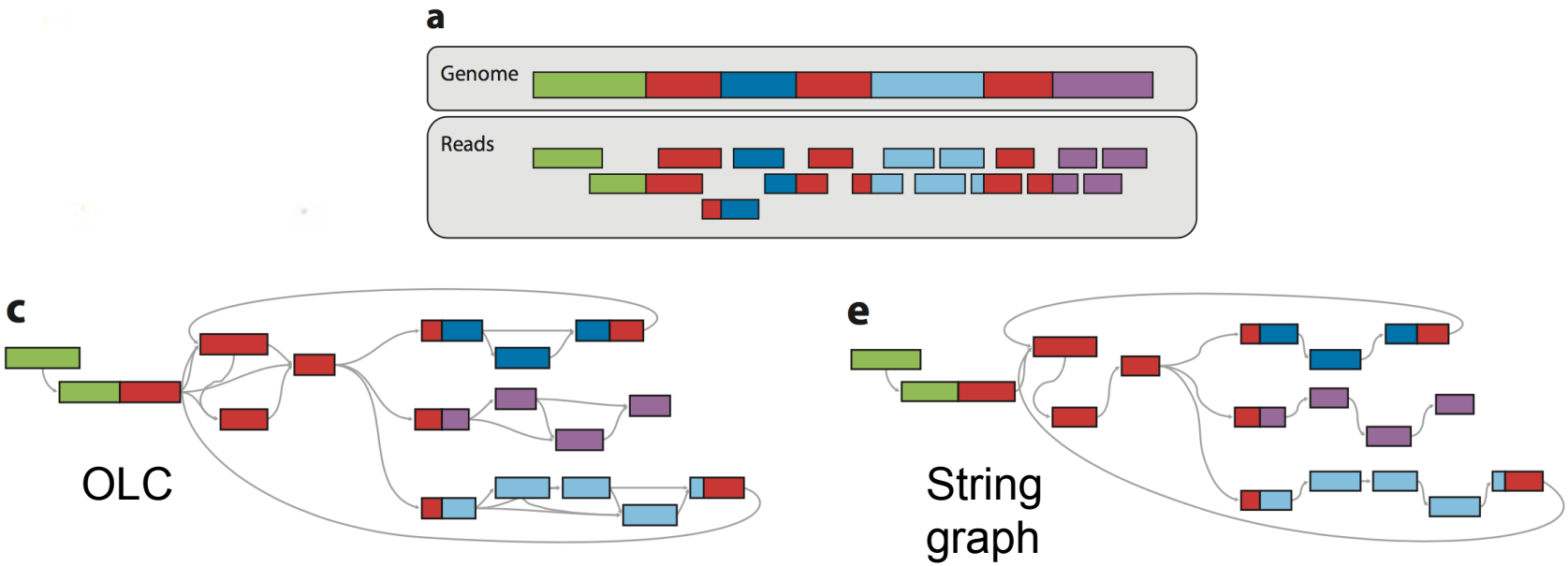
B



The edge $R_1 \rightarrow R_3$ is redundant with the path $R_1 \rightarrow R_2 \rightarrow R_3$. The edge $R_1 \rightarrow R_3$ is therefore *transitive*, and can be removed.

Source: Simpson, Jared T., and Richard Durbin. Bioinformatics 26.12 (2010): i367-i373.
Schatz, Michael C., Arthur L. Delcher, and Steven L. Salzberg. Genome research 20.9 (2010): 1165-1173.

OLC vs String graphs



String graphs represent the structure of the genome in a more concise way, addressing the computational complexity of the OLC approach. This has lead to new, more memory-efficient assemblers compatible with short read NGS technology.

Source: Simpson, Jared T., and Mihai Pop. Annual review of genomics and human genetics 16 (2015): 153-172.

De Bruijn Graphs

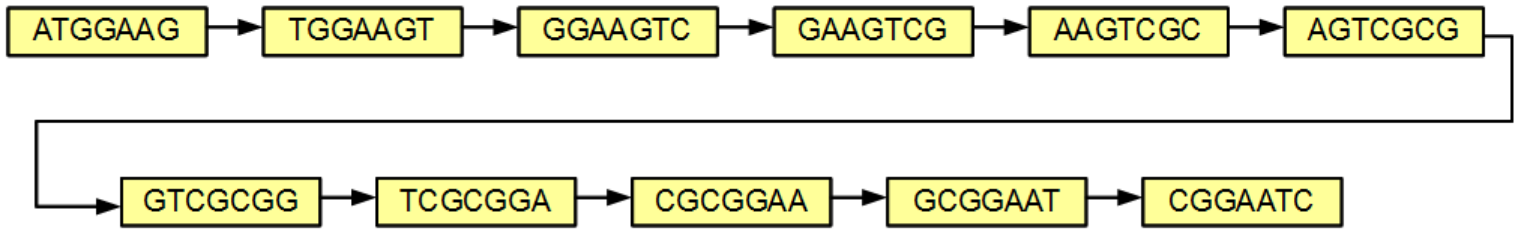
A de Bruijn graph models the relationship between exact substrings of length *k*, called *k*-mers, extracted from input reads. Vertices in the graph represent *k*-mers, and edges indicate that two *k*-mers overlap by exactly *k*-1 bases.

sequence **ATGGAAGTCGCGGAATC**

7mers

ATGGAAG
TGGAAGT
GGAAGTC
GAAGTCG
AAGTCGC
AGTCGCG
GTCGCGG
TCGCGGA
CGCGGAA
GCGGAAT
CGGAATC

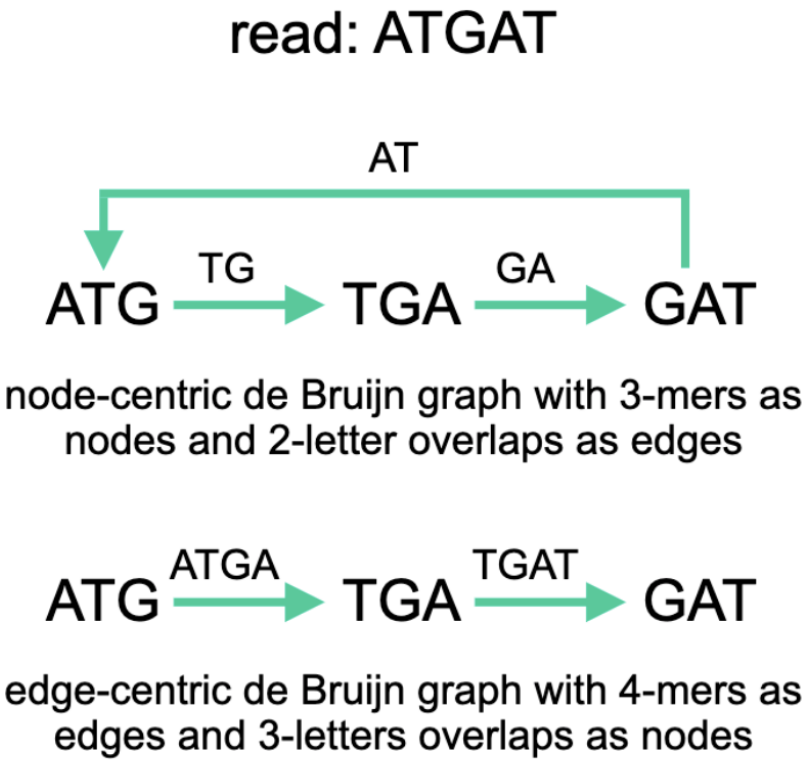
de Bruijn graph



Source: http://homolog.us/Tutorials/index.php?p=2.1&s=1

De Bruijn Graphs

- Node-centric
  - The nodes are all the unique *k*-length substrings in the read set, and an edge is created if there is a (*k*-1)-length overlap.
- Edge-centric
  - The nodes are the (*k*-1)-length substrings, and an edge connects two nodes if their (*k*-1)-mers are consecutive in some read.

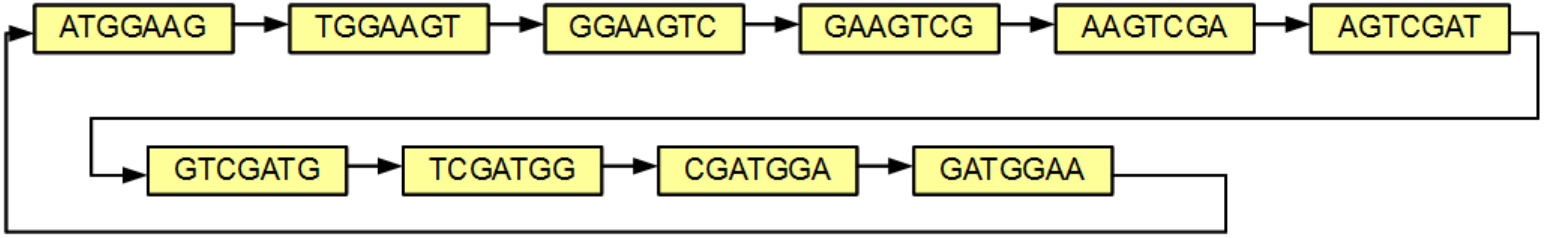


Source: Seminar by Jean-François Flot, Université libre de Bruxelles.

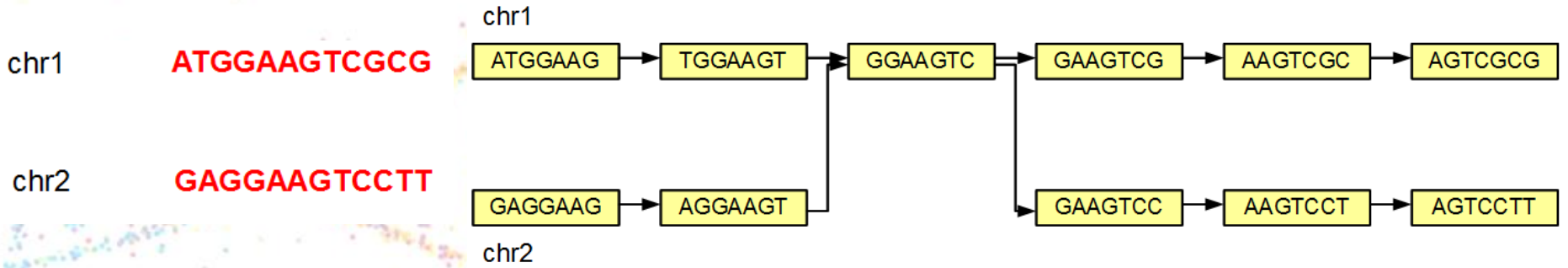
PGC
PHILIPPINE GENOME CENTER

ATGGAAGTCGATGGAAG

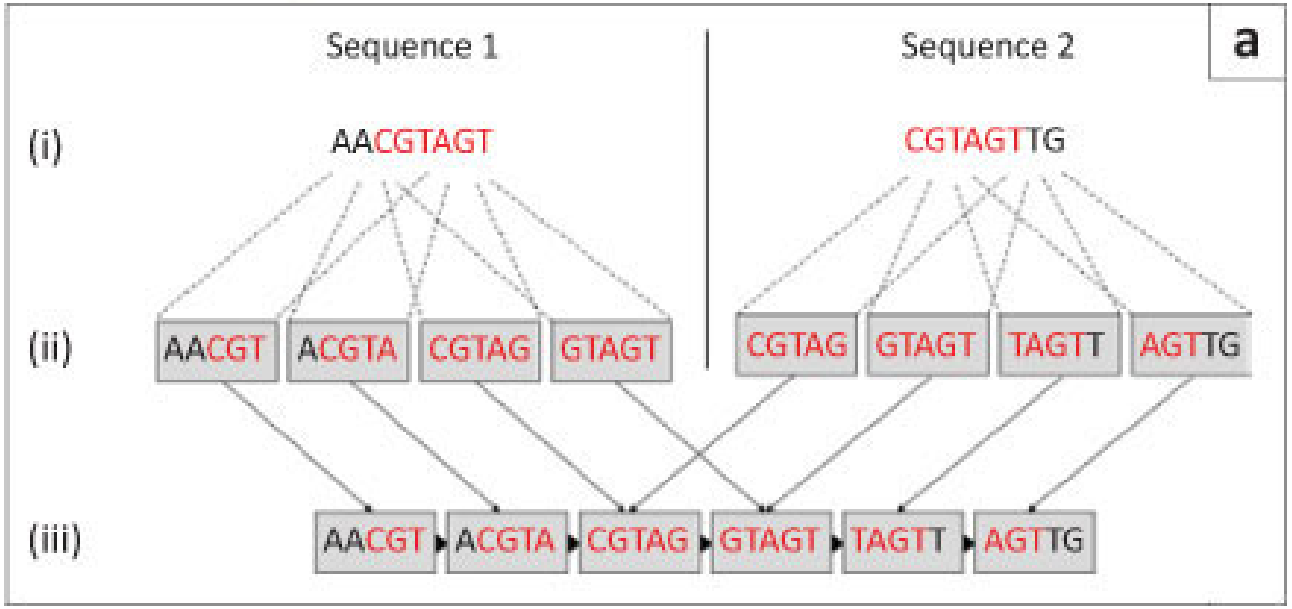
ATGGAAG
TGGAAGT
GGAAGTC
GAAGTCG
AAGTCGA
AGTCGAT
GTCGATG
TCGATGG
CGATGGA
GATGGAA
ATGGAAG



Repeating sequences can create loops or forks in the graph.

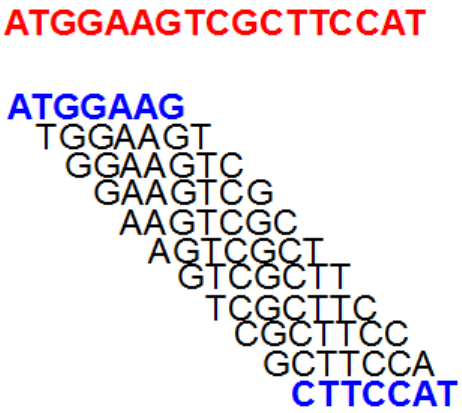


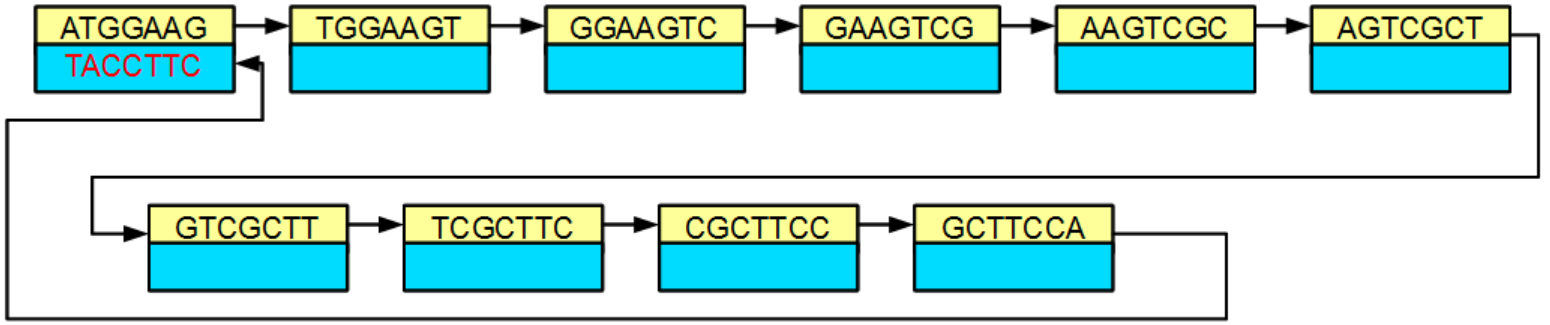
Source: http://homolog.us/Tutorials/index.php?p=2.1&s=1



By breaking down reads into *k*-mers, de Bruijn graphs allow overlapping reads to be correctly linked and ordered, without having to compute an overlap score.

Source: http://www.scielo.org.za/scielo.php?script=sci_arttext&pid=S0038-23532012000600016





- *k*-mers can be represented in the de Bruijn graph in double-stranded fashion. An edge can indicate that a *k*-mer overlaps with the reverse-complement of another.
- *k*-mers of even length can be reverse-complements of themselves (e.g. ATAT), creating ambiguity in the graph. For this reason, only *k*-mers of odd length are typically used.

Source: http://homolog.us/Tutorials/index.php?p=2.3&s=1

De Bruijn Graphs

Exercise: Make an edge-centric de Bruijn graph for the word “BANANA”. Use *k*=4.

1. Make a list of all the *k*-mers.

2. Split each *k*-mer into the left and right (*k*-1)-mer. Each (*k*-1)-mer forms a node in your graph.

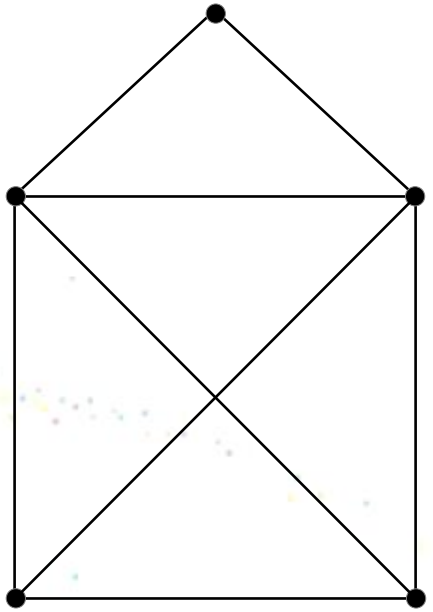
3. For each *k*-mer, draw an edge between the left and right (*k*-1)-mer.

PGC
PHILIPPINE GENOME CENTER

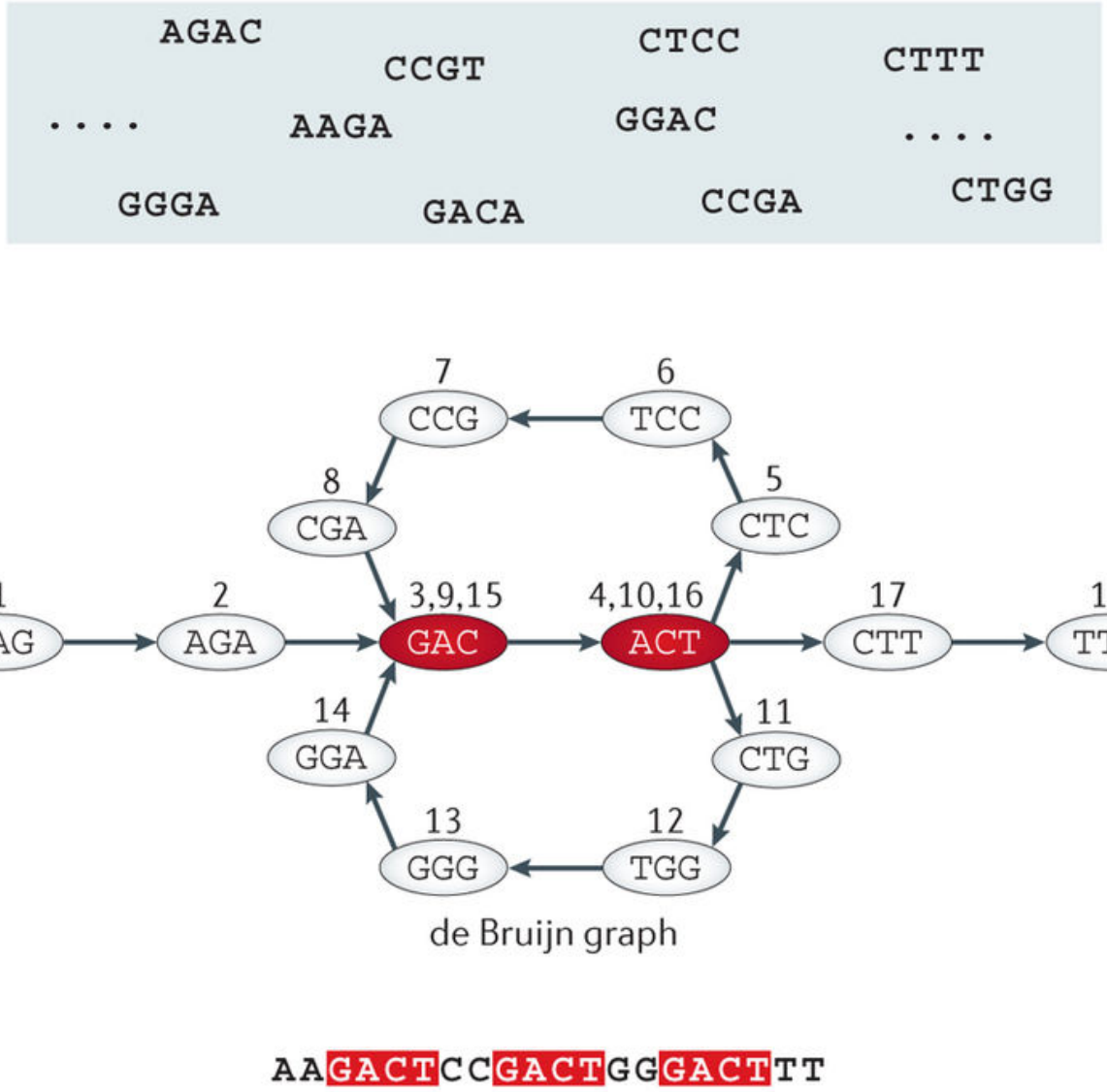
Ok, now you have a de Bruijn graph. How do you use it to assemble the original sequence?

Find the Eulerian path!

Exercise: Draw the following diagram without lifting your pen or tracing over a line more than once.



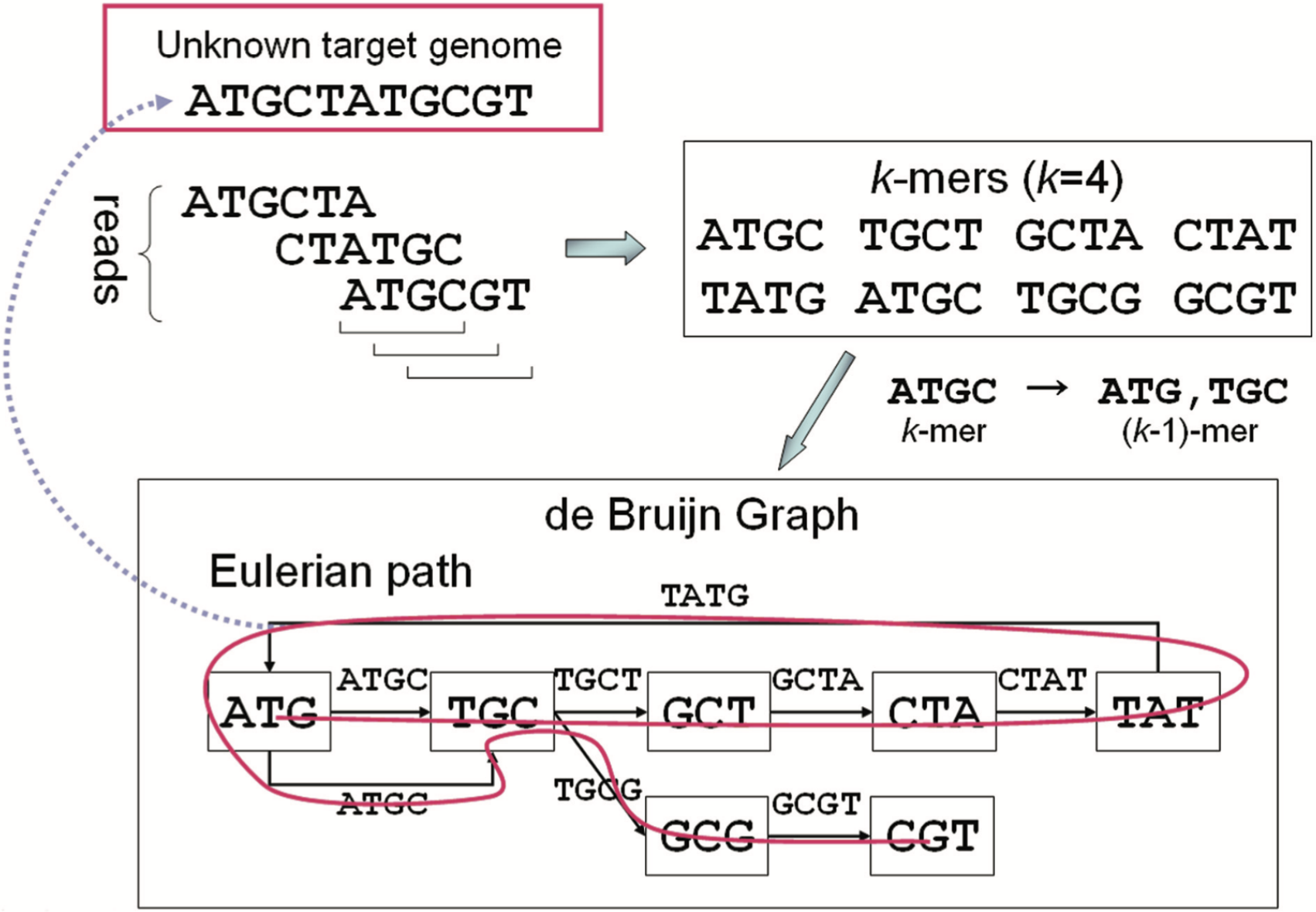
PGC
PHILIPPINE GENOME CENTER



The original sequence is reconstructed by following the Eulerian path through the graph, wherein every edge is traversed exactly once.

Source: Berger, Bonnie, Jian Peng, and Mona Singh. Nature reviews. Genetics 14.5 (2013): 333.

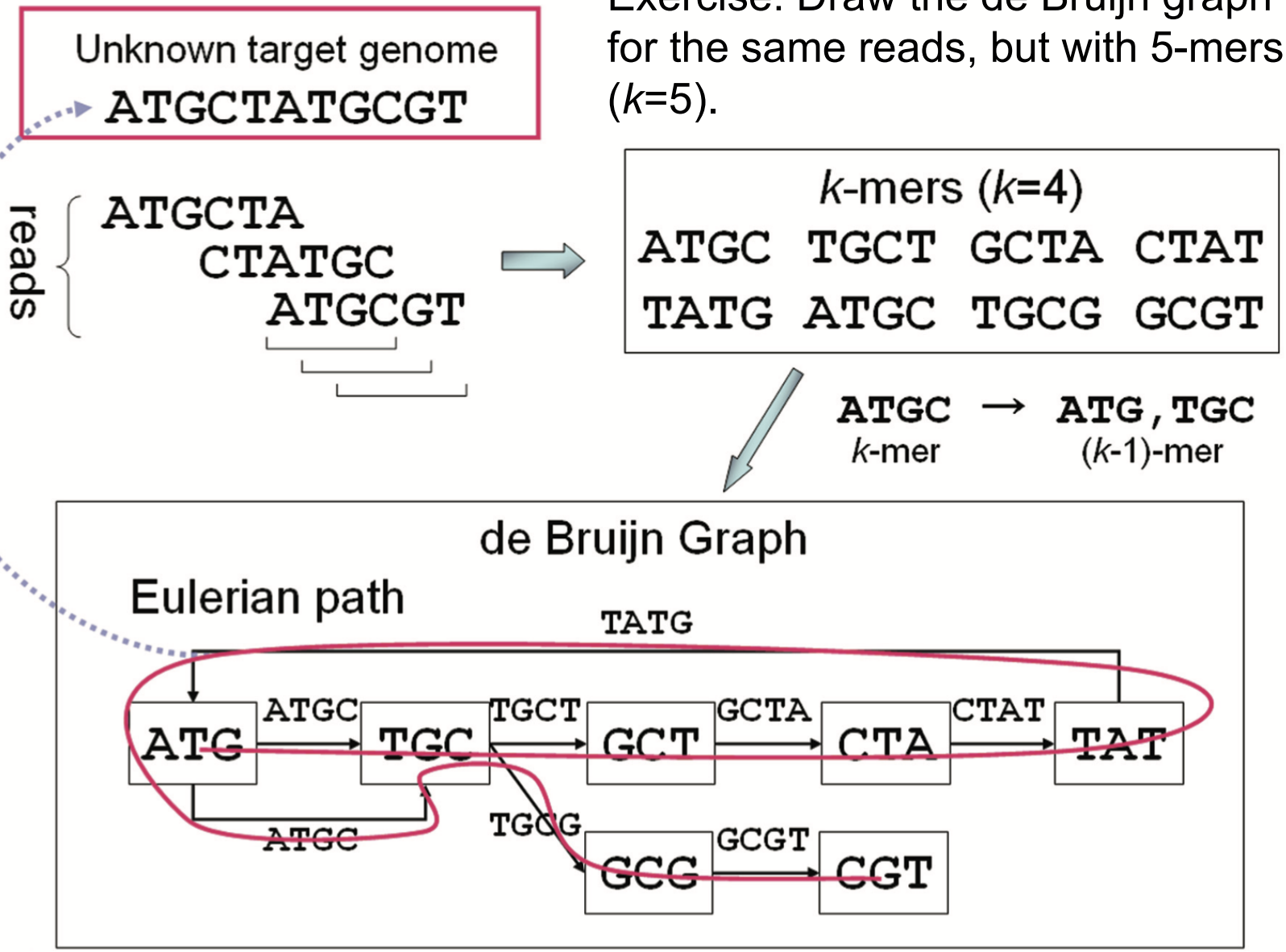
PHILIPPINE GENOME CENTER



Source: Namiki, Toshiaki, et al. Nucleic acids research 40.20 (2012): e155-e155.

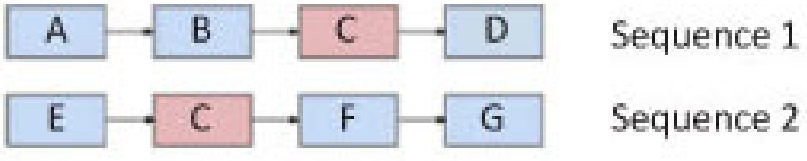
PHILIPPINE GENOME CENTER

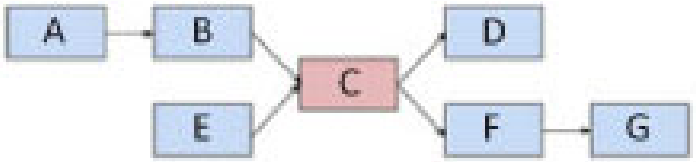
Exercise: Draw the de Bruijn graph for the same reads, but with 5-mers (*k*=5).

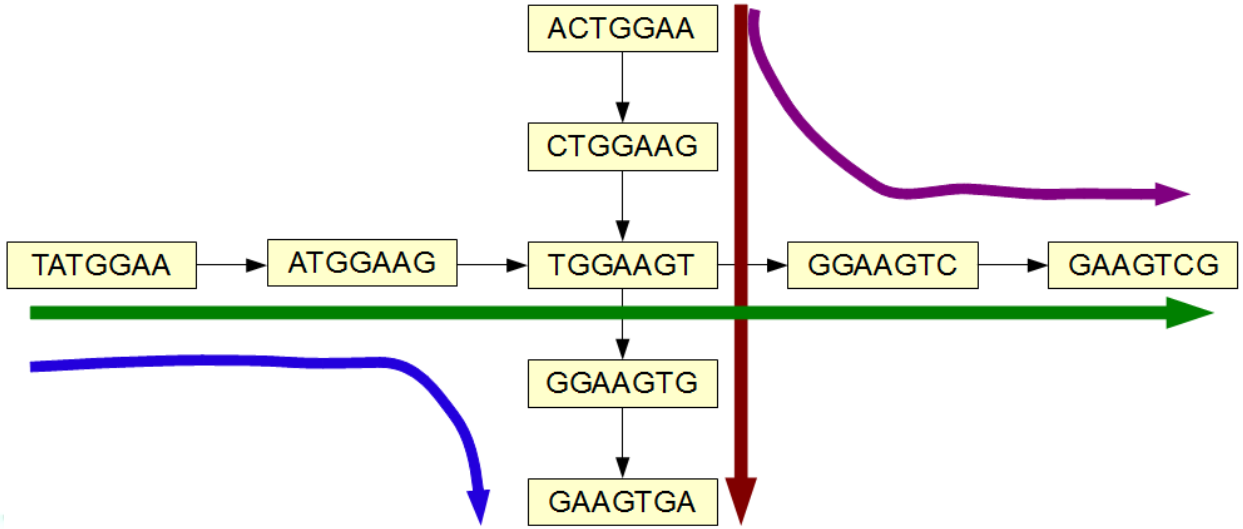


Source: Namiki, Toshiaki, et al. Nucleic acids research 40.20 (2012): e155-e155.

PHILIPPINE GENOME CENTER







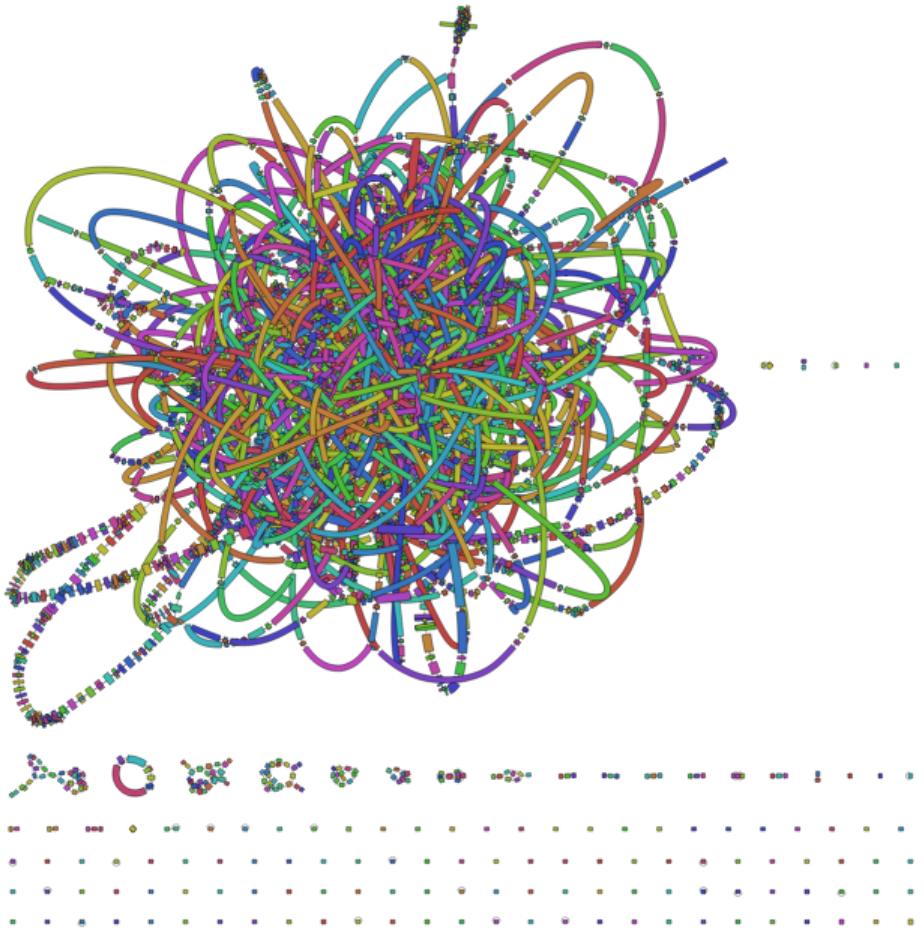
Original sequences – TATGGAAGTCG, ACTGGAAGTGA

By breaking down reads into smaller k-mers, de Bruijn graphs lose useful positional information. This can create links between unrelated sequences that cannot be entangled without consulting the original reads.

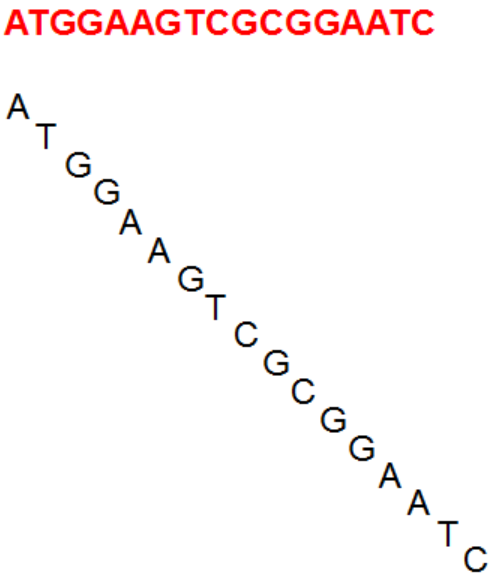
The choice of *k*-mer length is important. Here, a longer *k*-mer would have resolved the ambiguity.

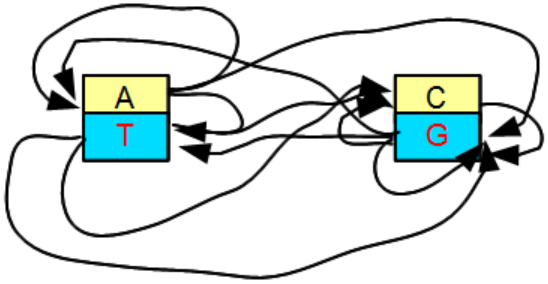
Source: http://www.scielo.org.za/scielo.php?script=sci_arttext&pid=S0038-23532012000600016
http://homolog.us/Tutorials/index.php?p=3.4&s=1

At low values for *k*, the resulting graph becomes complex and tangled from spurious associations, making final assembly difficult. At sufficiently low values of *k*, the de Bruijn graph becomes meaningless.



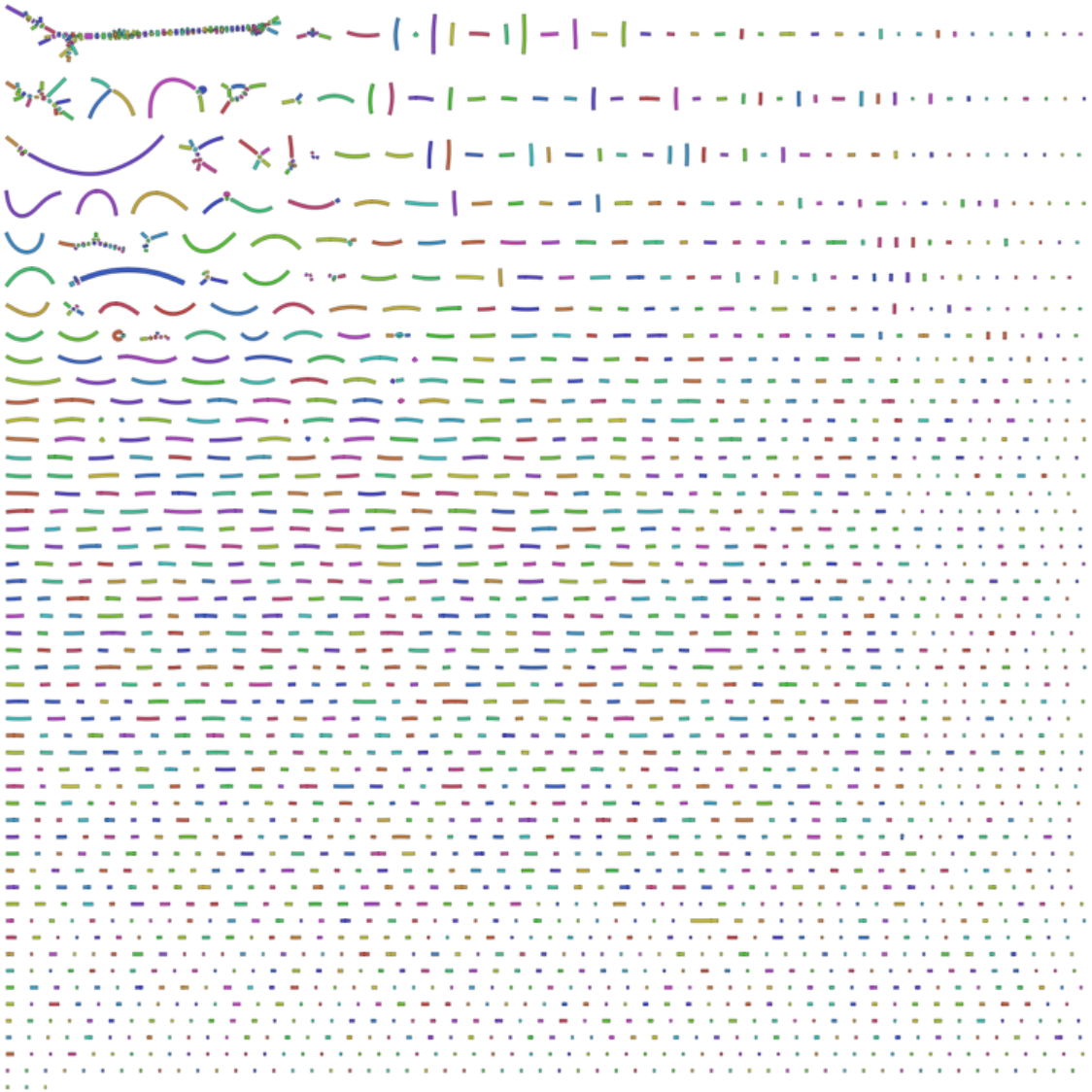
Salmonella genome assembly using 100 bp Illumina reads.
51-mer assembly with 4618 nodes and 6070 edges.





Source: https://github.com/rrwick/Bandage/wiki/Effect-of-kmer-size
http://homolog.us/Tutorials/index.php?p=2.4&s=1

PHILIPPINE GENOME CENTER

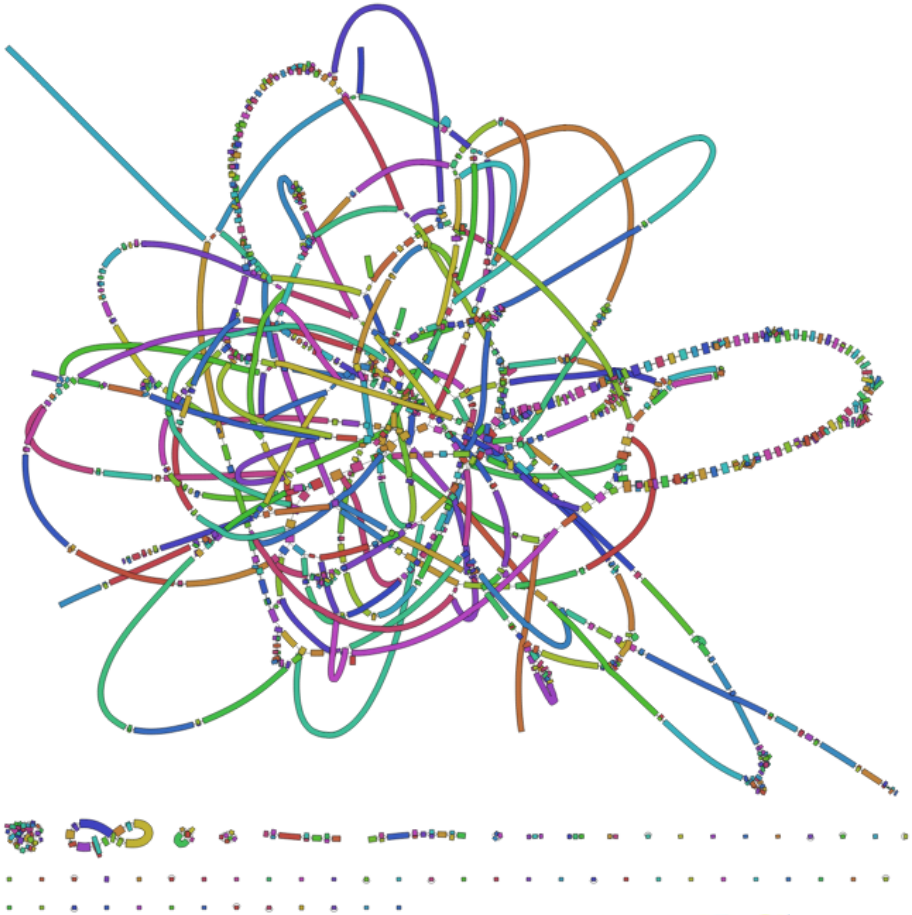


Salmonella genome assembly using 100 bp Illumina reads.
91-mer assembly with 2386 nodes and 304 edges.

Source: https://github.com/rrwick/Bandage/wiki/Effect-of-kmer-size

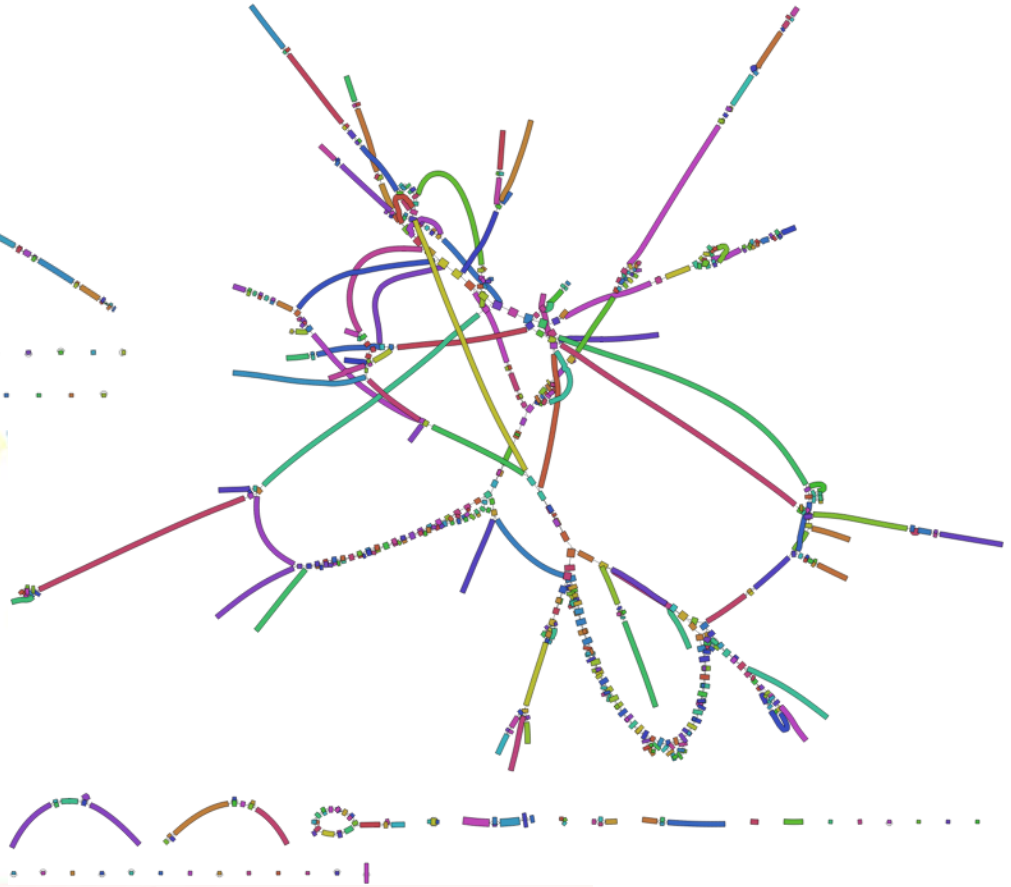
At high values of *k*, you fail to make associations between *k*-mers from different reads, since the necessary *k*-1 base overlap becomes increasingly unlikely. The resulting de Bruijn graph becomes disjointed and useless.

PGC
PHILIPPINE GENOME CENTER



61-mer assembly with 1357 nodes and 1768 edges.

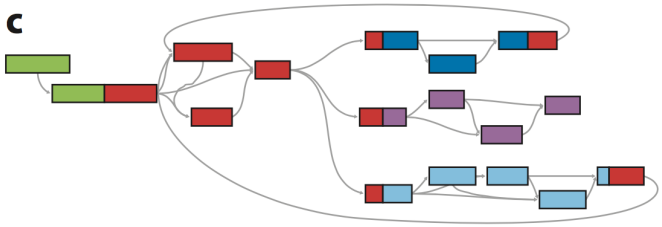
71-mer assembly with 611 nodes and 765 edges.



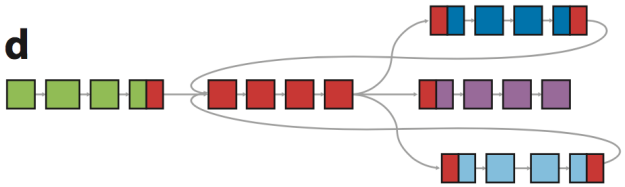
Source: https://github.com/rrwick/Bandage/wiki/Effect-of-kmer-size

OLC vs De Bruijn

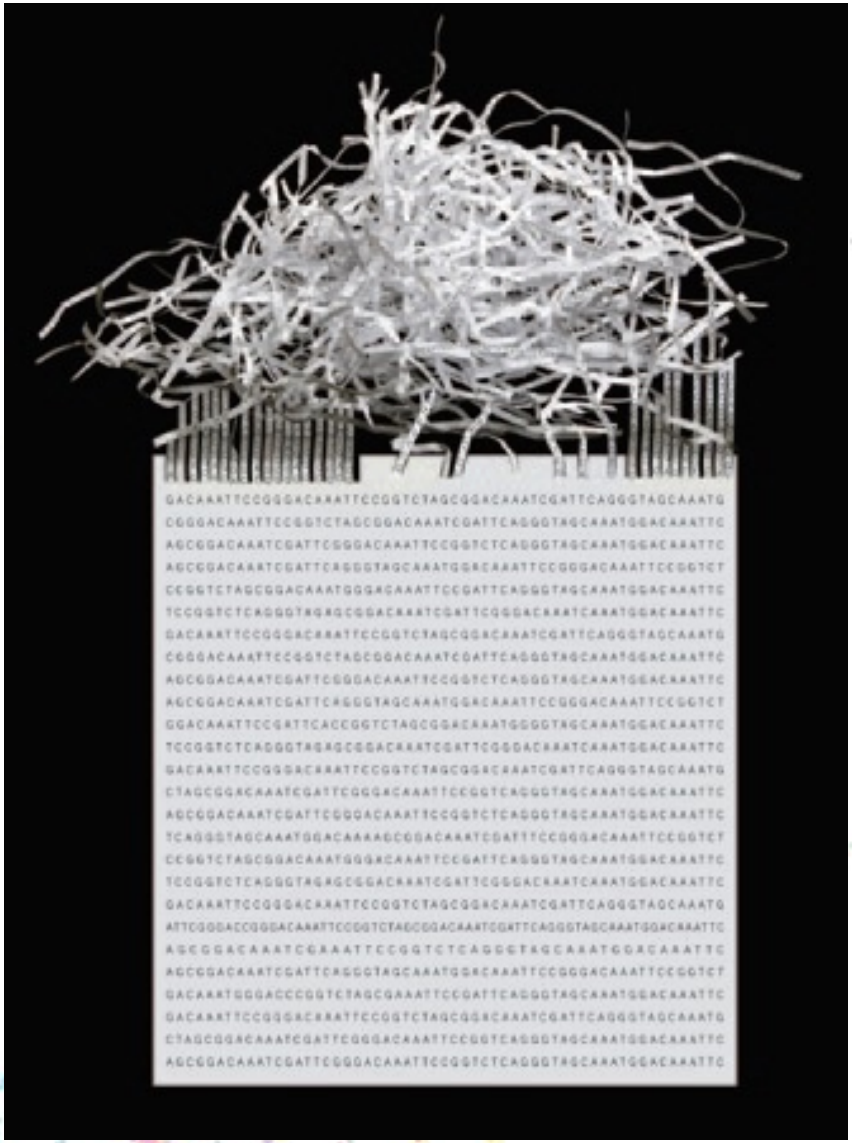
- Overlap-Layout-Consensus:
  - Appropriate for longer reads
  - Computationally expensive
  - Less sensitive to repeats and errors
  - Key parameter: minimum overlap



- De Bruijn Graphs
  - Appropriate for shorter reads
  - Avoids all-vs-all comparison
  - Repeats get collapsed, simplifying graphs but complicating assembly
  - Computationally more efficient
  - Key parameter: *k*-mer length



PGC
PHILIPPINE GENOME CENTER

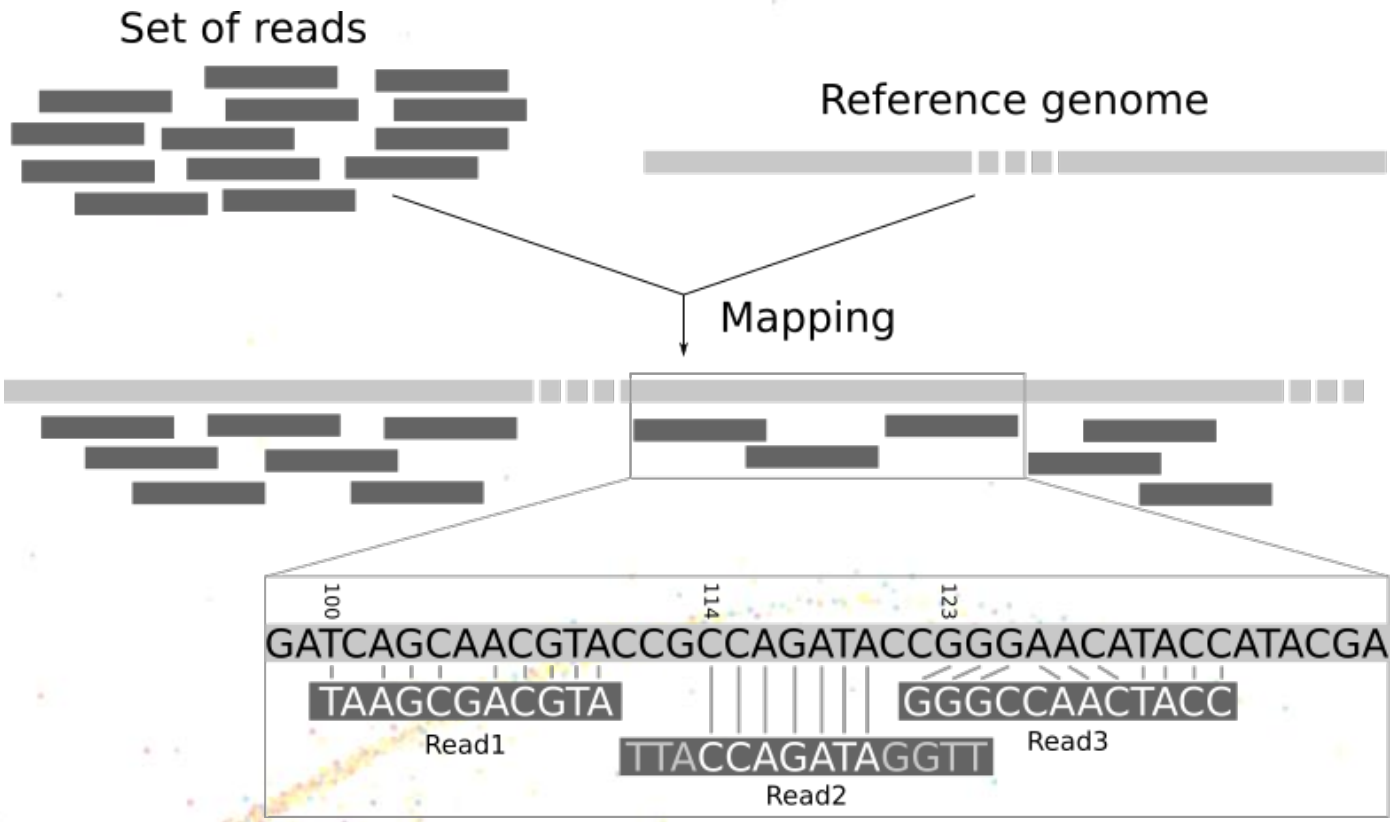


Questions?

Source: Kelly Howe, Lawrence Berkeley Laboratory
Baker, Monya. "De novo genome assembly: what every biologist should know." Nature methods 9.4 (2012): 333-337.

PGC
PHILIPPINE GENOME CENTER

Read mapping



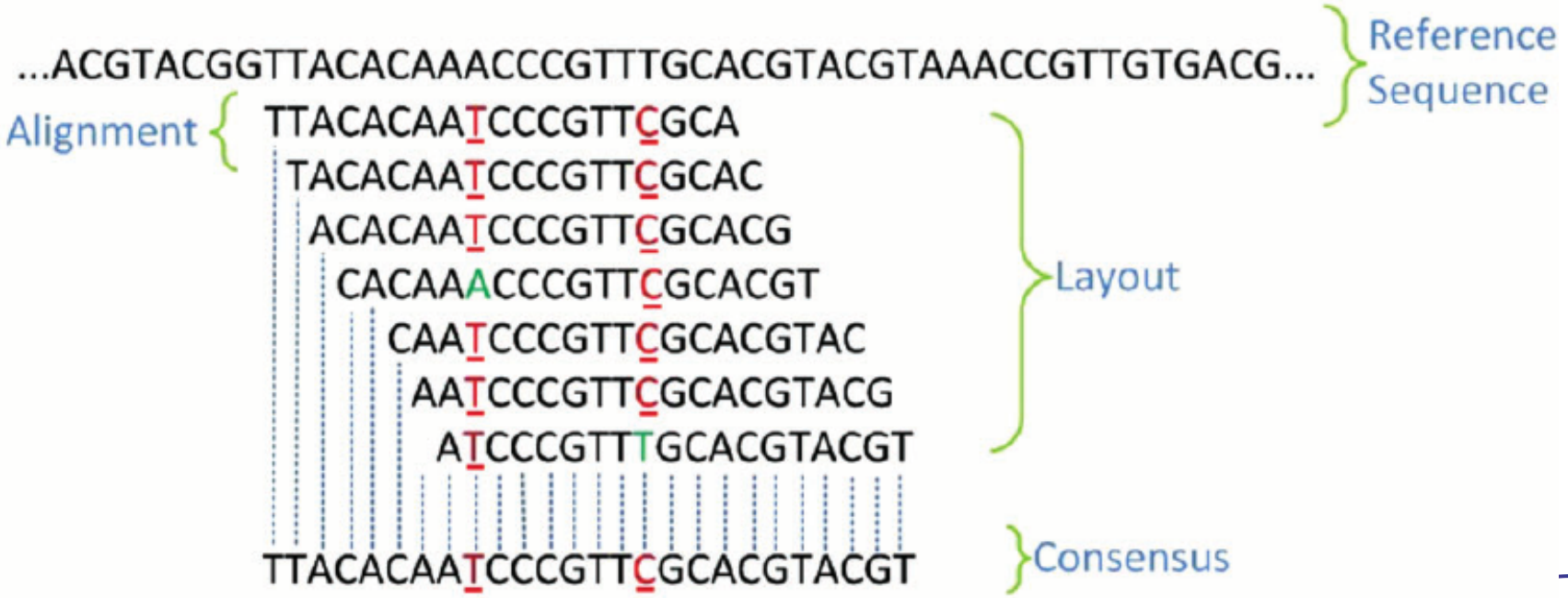
The process of aligning short reads to a longer reference sequence

Source: https://galaxyproject.github.io/training-material/topics/sequence-analysis/tutorials/mapping/tutorial.html

PGC
PHILIPPINE GENOME CENTER

Read mapping

Read mapping is the basis for reference-based genome assembly

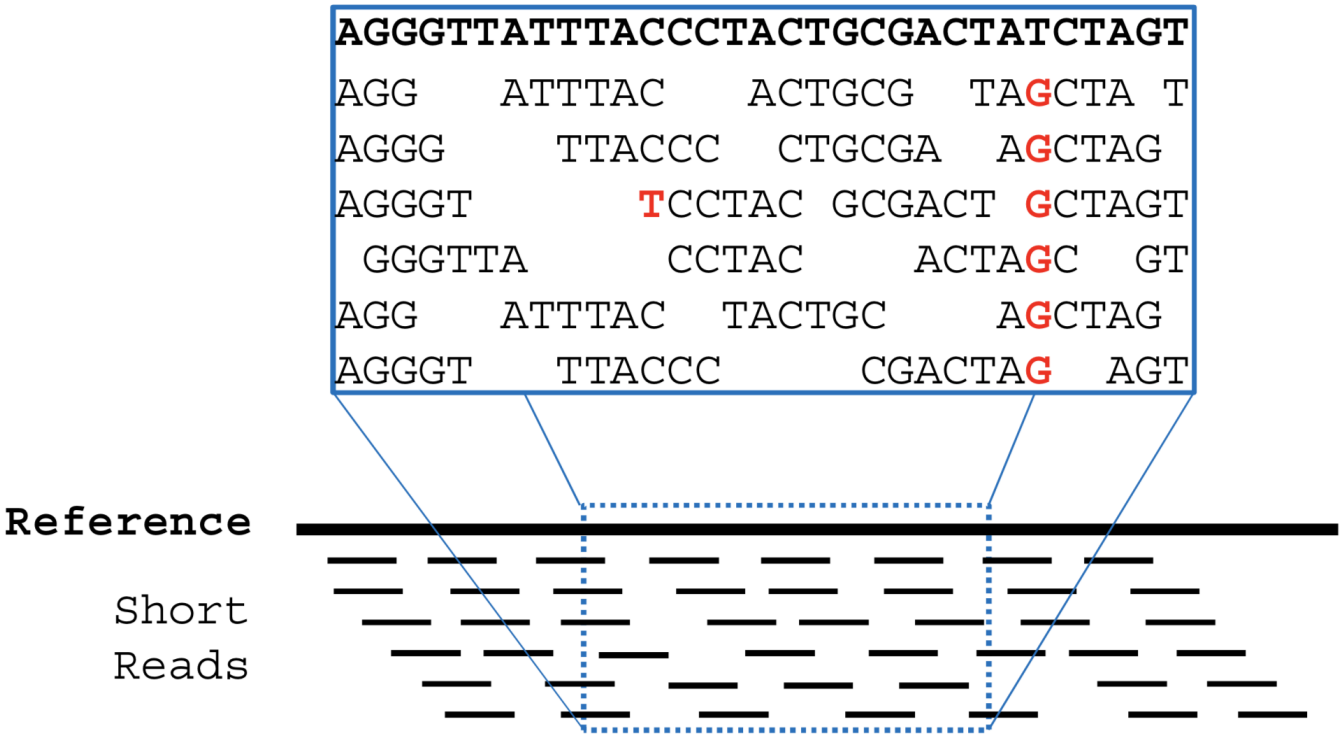


Source: Wajid, Bilal, and Erchin Serpedin. Briefings in functional genomics 15.1 (2014): 1-9.

PGC
PHILIPPINE GENOME CENTER

Read mapping

Mapping accuracy is important particularly in identifying structural and single nucleotide variations

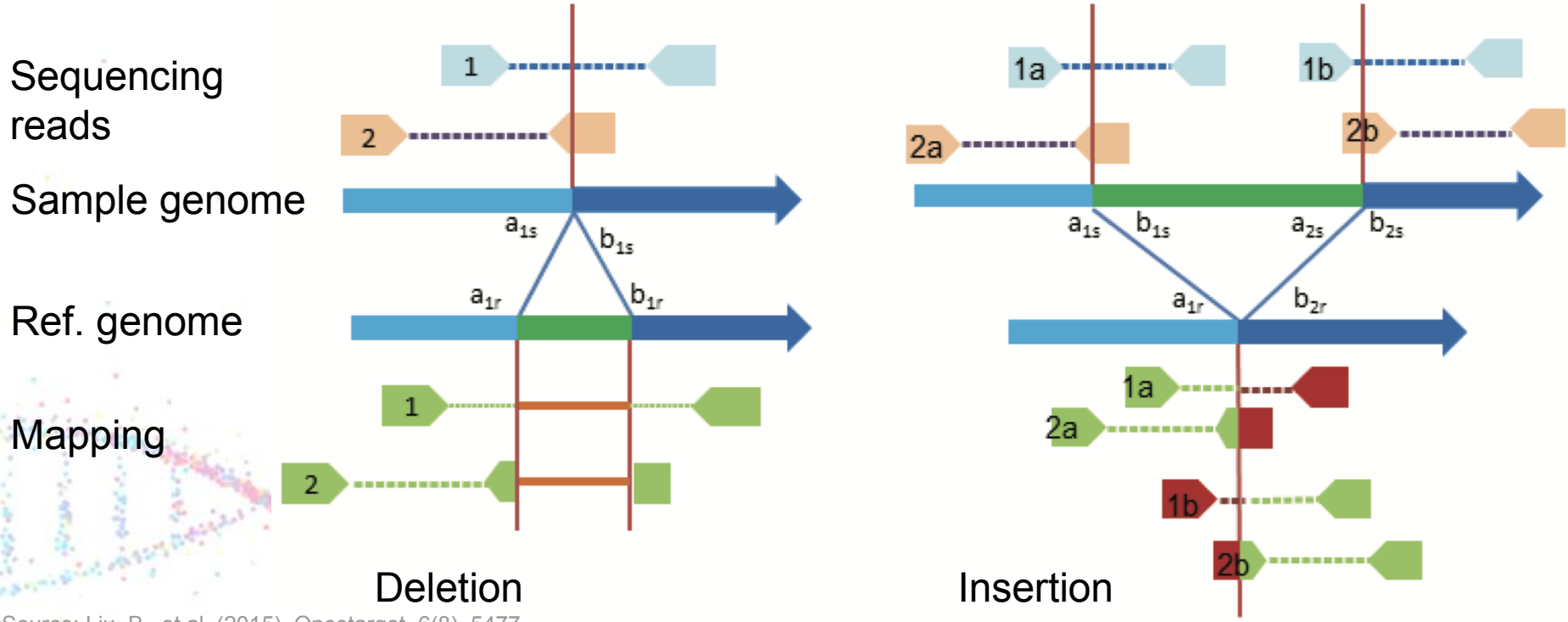


Source: Olson, C. B., et al. (2012, April). In 2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines (pp. 161-168). IEEE.

PGC
PHILIPPINE GENOME CENTER

Read mapping

Mapping accuracy is important particularly in identifying structural and single nucleotide variations



Source: Liu, B., et al. (2015). Oncotarget, 6(8), 5477.

Read mapping

```
  1 agcttttcat tctgactgca acgggcaata tgtctctgtg tggattaaaa aaagagtgtc
 61 tgatagcagc ttctgaactg gttacctgcc gtgagtaaat taaaatttta ttgacttagg
121 tcactaaata ctttaaccaa tataggcata gcgcacagac agataaaaat tacagagtac
181 acaacatcca tgaaacgcat tagcaccacc attaccacca ccatcaccat taccacaggt
241 aacggtgcgg gctgacgcgt acaggaaaca cagaaaaaag cccgcacctg acagtgcggg
```

Shown above are the first 300 bp of the *E. coli* genome.
Which of the following short reads map to reference?

A) `catgaaactca`

B) `ctgatatctgc`

C) `tgccgtgaaaa`

D) `caccattacca`

GenBank: U00096.3

PGC
PHILIPPINE GENOME CENTER

Read mapping

```
  1 agcttttcat tctgactgca acgggcaata tgtctctgtg tggattaaaa aaagagtgtc
 61 tgatagcagc ttctgaactg gttacctgcc gtgagtaaat taaaatttta ttgacttagg
121 tcactaaata ctttaaccaa tataggcata gcgcacagac agataaaaat tacagagtac
181 acaacatcca tgaaacgcat tagcaccacc attaccacca ccatcaccat taccacaggt
241 aacggtgcgg gctgacgcgt acaggaaaca cagaaaaaag cccgcacctg acagtgcggg
```

Shown above are the first 300 bp of the *E. coli* genome.
Which of the following short reads map to reference?

A) catgaaactca

B) ctgatatctgc

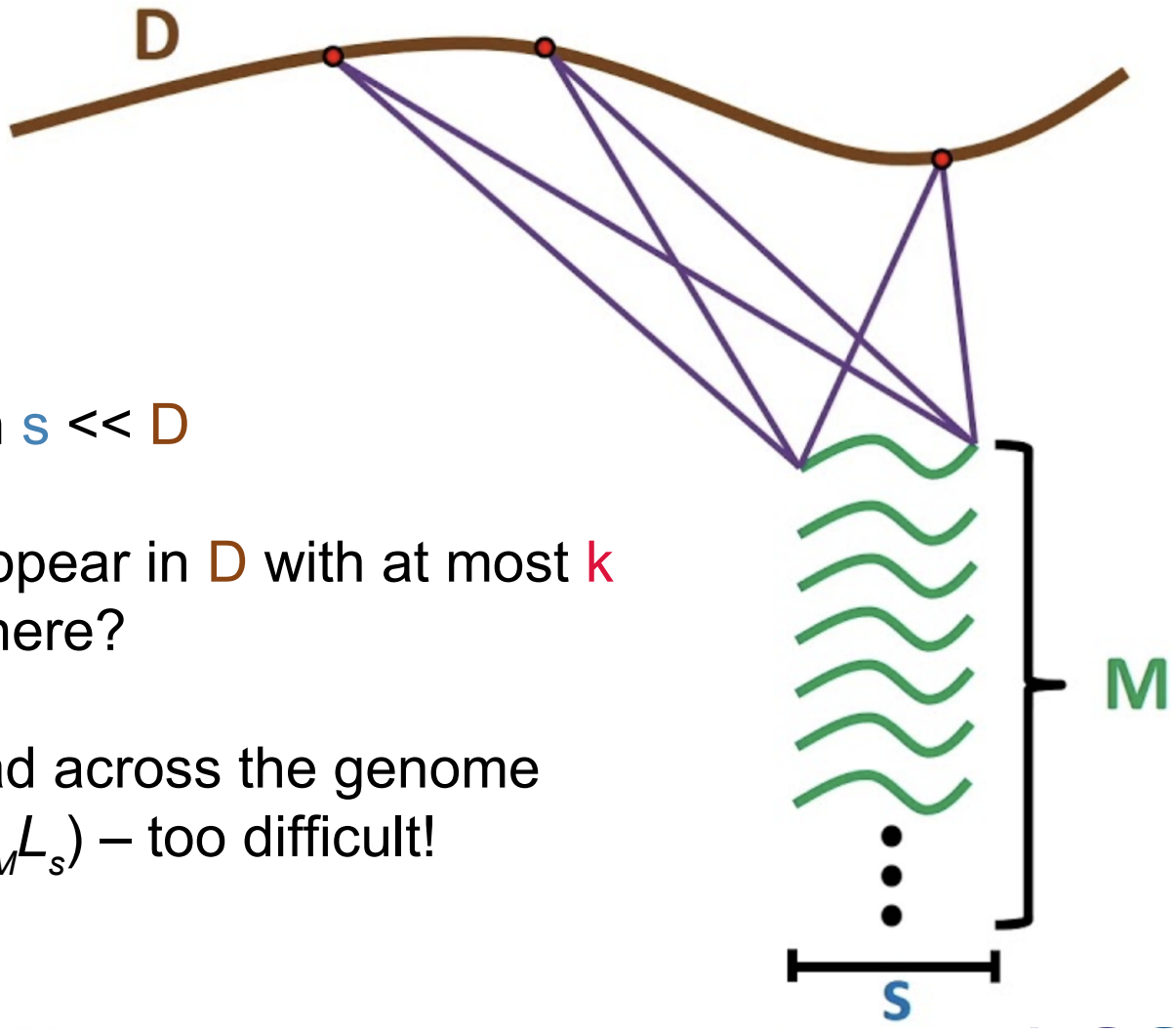
C) tgccgtgaaaa

D) caccattacca

GenBank: U00096.3

PGC
PHILIPPINE GENOME CENTER

Read mapping



Given:

- a genome database D
- M reads each of length s << D

For each read, does it appear in D with at most k differences, and if so, where?

Simply “sliding” each read across the genome has complexity of $O(L_D L_M L_s)$ – too difficult!

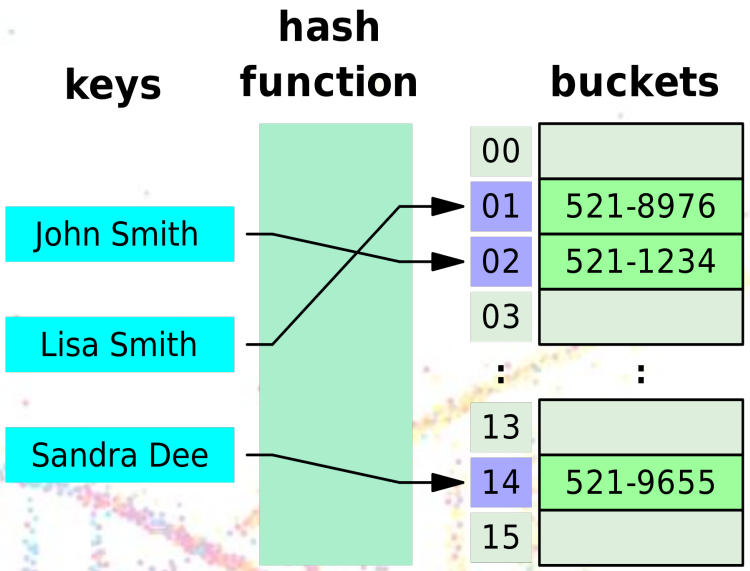
Source: https://www.youtube.com/watch?v=gqYgNXoshsQ

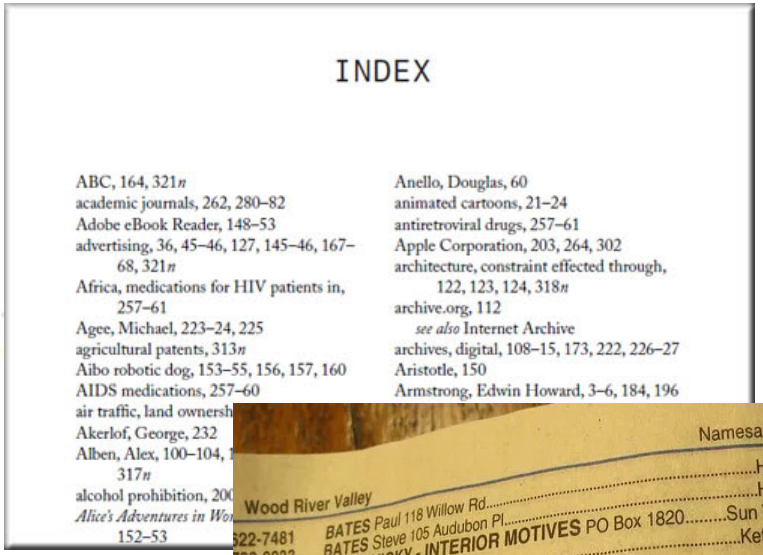
Read mapping approaches

- Hash tables
  - MAQ
  - RazerS, RazerS 3
  - RMAP
  - SHRiMP, SHRiMP2
  - SSAHA, SSAHA2
  - ZOOM
- Burrows-Wheeler Transform
  - BLASR
  - Bowtie, Bowtie 2
  - BWA, BWA-SW, BWA-MEM
  - SOAP2, SOAP3, SOAP3-dp

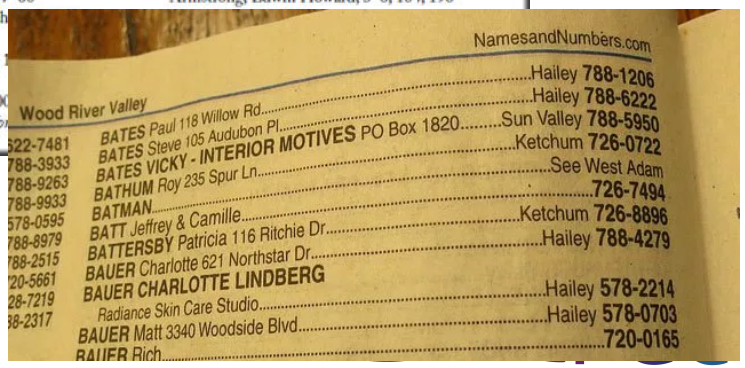
Hash tables

A hash table is a data structure that can match keys to values. When you input a key into the hash function, it quickly returns a unique location or index in computer memory where the value corresponding to that key can be found.




INDEX

ABC, 164, 321*n*
academic journals, 262, 280–82
Adobe eBook Reader, 148–53
advertising, 36, 45–46, 127, 145–46, 167–68, 321*n*
Africa, medications for HIV patients in, 257–61
Agee, Michael, 223–24, 225
agricultural patents, 313*n*
Aibo robotic dog, 153–55, 156, 157, 160
AIDS medications, 257–60
air traffic, land ownersh
Akerlof, George, 232
Alben, Alex, 100–104, 317*n*
alcohol prohibition, 20
Alice's Adventures in Wo
152–53
Anello, Douglas, 60
animated cartoons, 21–24
antiretroviral drugs, 257–61
Apple Corporation, 203, 264, 302
architecture, constraint effected through, 122, 123, 124, 318*n*
archive.org, 112
see also Internet Archive
archives, digital, 108–15, 173, 222, 226–27
Aristotle, 150
Armstrong, Edwin Howard, 3–6, 184, 196



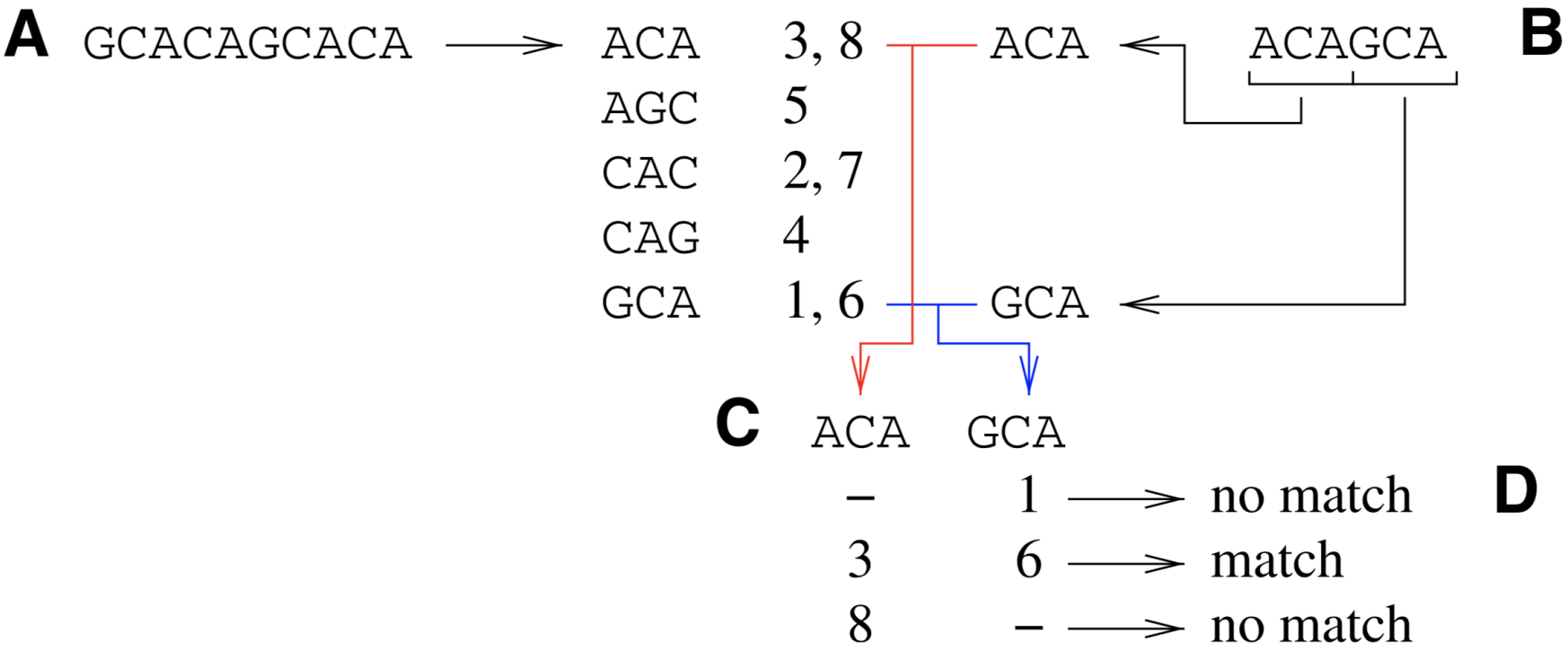
Source: https://en.wikipedia.org/wiki/Hash_table
https://www.pdfindexgenerator.com/what-is-a-book-index/
https://boingboing.net/2017/06/10/how-adam-west-played-a-prank-u.html

Hash tables

- The “obvious” approach would be to build a hash table mapping all read-length (s) subsequences of the genome to their locations.
  - Requires too much memory and disk space
  - Even harder with variable read lengths
  - Hard to account for mismatches
- Better approach is similar to assembly: break sequences down into *k*-mers of manageable length.
  - *k* significantly less than read size, i.e. $k < s$.
  - For example, for $k = 9$, there are at most $4^9 = 262{,}144$ 9-mers in the genome, regardless of the genome length.

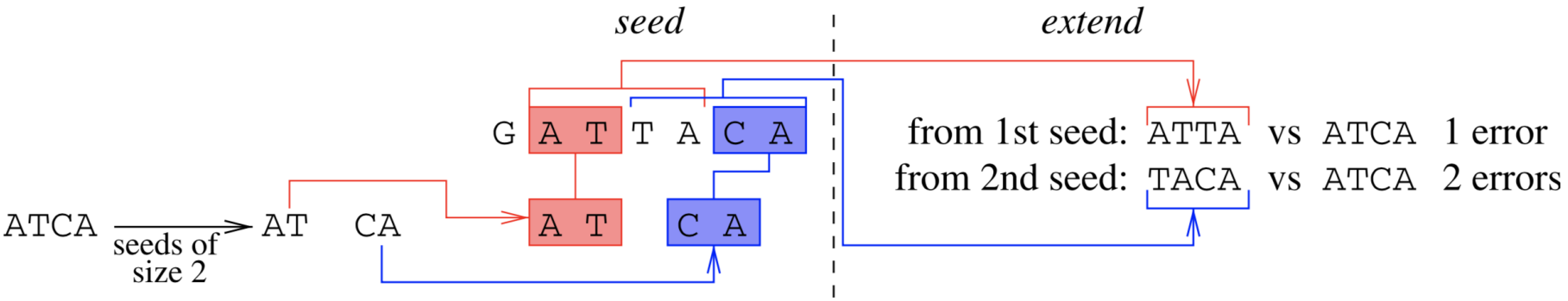
PGC
PHILIPPINE GENOME CENTER

A. Genome is cut into overlapping 3-mers. Store their respective positions in a hash table.
B. The read is cut into 3-mers. Look these 3-mers up in the hash table.
C. Positions for each hit are sorted and compared to those for other hits.
D. Compatible positions indicate a match between the read and genome.



Source: Schbath, S., et al. (2012). Journal of Computational Biology, 19(6), 796-813.

Seed and extend

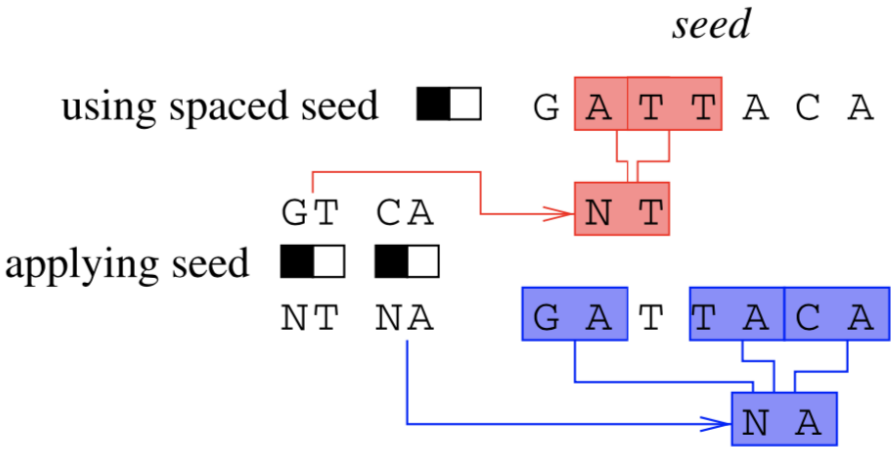


A read ATCA is sought for in GATTACA, using seeds of size 2, with one error. Each seed maps once (left part). After extension of each seed (right part), it turns out that only one mapping contains just one error.

Source: Schbath, S., et al. (2012). Journal of Computational Biology, 19(6), 796-813.

PHILIPPINE GENOME CENTER

Seed and extend



extend

from 1st seed:	ATTA	vs	GTCA	2 errors
	TTAC	vs	GTCA	3 errors
from 2nd seed:	--GA	vs	GTCA	3 errors
	ATTA	vs	GTCA	2 errors
	TACA	vs	GTCA	2 errors

As the number of allowed errors increases, the seed must shorten, which can result in too many hits. This is resolved by using “spaced seeds”.

A read GTCA is sought for in GATTACA, using spaced seeds of size 2, with two errors. The first seed maps twice, the second thrice, with one hit in common between them.

Source: Schbath, S., et al. (2012). Journal of Computational Biology, 19(6), 796-813.

PGC
PHILIPPINE GENOME CENTER

Burrows-Wheeler transform

- Invented by Michael Burrows and David Wheeler in 1994 for use in data compression.
- Because of the sorting step, BWT produces runs of repeated characters, making text easy to compress.
- The transformation is easily reversible, with no loss of information.

Source: Schbath, S., et al. (2012). Journal of Computational Biology, 19(6), 796-813.

PHILIPPINE GENOME CENTER

Burrows-Wheeler transform

genome	suffixes	sorted suffixes	positions	cylinder suffix array	B–W
GATTACA$	GATTACA$	ACA$	4	ACA$GATT	T
	ATTACA$	ATTACA$	2	ATTACA$G	G
	TTACA$	A$	6	A$GATTAC	C
	TACA$	CA$	5	CA$GATTA	A
	ACA$	GATTACA$	1	GATTACA$	$
	CA$	TACA$	4	TACA$GAT	T
	A$	TTACA$	3	TTACA$GA	A
	$	$	7	$GATTACA	A

Step 1: Mark the end of your text (i.e. your genome) with a sigil ($).

Source: Schbath, S., et al. (2012). Journal of Computational Biology, 19(6), 796-813.

PGC
PHILIPPINE GENOME CENTER

Burrows-Wheeler transform

genome	suffixes	sorted suffixes	positions	cylinder suffix array	B–W
GATTACA$	GATTACA$	ACA$	4	ACA$GATT	T
	ATTACA$	ATTACA$	2	ATTACA$G	G
	TTACA$	A$	6	A$GATTAC	C
	TACA$	CA$	5	CA$GATTA	A
	ACA$	GATTACA$	1	GATTACA$	$
	CA$	TACA$	4	TACA$GAT	T
	A$	TTACA$	3	TTACA$GA	A
	$	$	7	$GATTACA	A

Step 2: List down all the suffixes (right-most substrings) of the text.

Source: Schbath, S., et al. (2012). Journal of Computational Biology, 19(6), 796-813.

PGC
PHILIPPINE GENOME CENTER

Burrows-Wheeler transform

genome	suffixes	sorted suffixes	positions	cylinder suffix array	B–W
GATTACA$	GATTACA$	ACA$	4	ACA$GATT	T
	ATTACA$	ATTACA$	2	ATTACA$G	G
	TTACA$	A$	6	A$GATTAC	C
	TACA$	CA$	5	CA$GATTA	A
	ACA$	GATTACA$	1	GATTACA$	$
	CA$	TACA$	4	TACA$GAT	T
	A$	TTACA$	3	TTACA$GA	A
	$	$	7	$GATTACA	A

Step 3: Sort all the suffixes lexicographically (i.e. alphabetically). By convention, the sigil comes last in the lexicographic order.

Source: Schbath, S., et al. (2012). Journal of Computational Biology, 19(6), 796-813.

PGC
PHILIPPINE GENOME CENTER

Burrows-Wheeler transform

genome	suffixes	sorted suffixes	positions	cylinder suffix array	B–W
GATTACA$	GATTACA$	ACA$	4	ACA$GATT	T
	ATTACA$	ATTACA$	2	ATTACA$G	G
	TTACA$	A$	6	A$GATTAC	C
	TACA$	CA$	5	CA$GATTA	A
	ACA$	GATTACA$	1	GATTACA$	$
	CA$	TACA$	4	TACA$GAT	T
	A$	TTACA$	3	TTACA$GA	A
	$	$	7	$GATTACA	A

Step 4: Keep track of the positions for each suffix in the original text.

Source: Schbath, S., et al. (2012). Journal of Computational Biology, 19(6), 796-813.

PGC
PHILIPPINE GENOME CENTER

Burrows-Wheeler transform

genome	suffixes	sorted suffixes	positions	cylinder suffix array	B–W
GATTACA$	GATTACA$	ACA$	4	ACA$GATT	T
	ATTACA$	ATTACA$	2	ATTACA$G	G
	TTACA$	A$	6	A$GATTAC	C
	TACA$	CA$	5	CA$GATTA	A
	ACA$	GATTACA$	1	GATTACA$	$
	CA$	TACA$	4	TACA$GAT	T
	A$	TTACA$	3	TTACA$GA	A
	$	$	7	$GATTACA	A

Step 5: ”Complete” each suffix, wrapping around from the beginning of the text as if written on a cylinder.

Source: Schbath, S., et al. (2012). Journal of Computational Biology, 19(6), 796-813.

PGC
PHILIPPINE GENOME CENTER

Burrows-Wheeler transform

genome	suffixes	sorted suffixes	positions	cylinder suffix array	B–W
GATTACA$	GATTACA$	ACA$	4	ACA$GATT	T
	ATTACA$	ATTACA$	2	ATTACA$G	G
	TTACA$	A$	6	A$GATTAC	C
	TACA$	CA$	5	CA$GATTA	A
	ACA$	GATTACA$	1	GATTACA$	$
	CA$	TACA$	4	TACA$GAT	T
	A$	TTACA$	3	TTACA$GA	A
	$	$	7	$GATTACA	A

Step 6: The last column of the resulting matrix is the Burrows-Wheeler transform (BWT) of the text.

Source: Schbath, S., et al. (2012). Journal of Computational Biology, 19(6), 796-813.

PGC
PHILIPPINE GENOME CENTER

Burrows-Wheeler transform

- The transformation is easily reversible, with no loss of information.
  - Iterative sorting and concatenating of CWT returns the cylinder suffix array

B–W	sorted	concat. 1	sort	concat. 2	sort
T	A	TA	AC	TAC	ACA
G	A	GA	AT	GAT	ATT
C	A	CA	A$	CA$	A$A
A	C	AC	CA	ACA	CA$
$	G	$G	GA	$GA	GAT
T	T	TT	TA	TTA	TAC
A	T	AT	TT	ATT	TTA
A	$	A$	$A	A$A	$GA

  - Exercise: which row corresponds to the original string?

Source: Schbath, S., et al. (2012). Journal of Computational Biology, 19(6), 796-813.

PGC
PHILIPPINE GENOME CENTER

Burrows-Wheeler Transform

Exercise: Find the BWT for the word “BANANA”.

1. Mark the end of your text with a sigil ($).
2. List down all the suffixes of the text.
3. Sort all the suffixes lexicographically.
4. Keep track of the positions for each suffix in the original text (optional).
5. ”Complete” each suffix.
6. Extract the last column of the resulting matrix.

PGC
PHILIPPINE GENOME CENTER

Finding a word with BWT

B–W	sorted
T	A
G	A
C	A
A	C
$	G
T	T
A	T
A	$

B–W	sorted
T	A
G	A
C	A
A	C
$	G
T	T
A	T
A	$

B–W	sorted
T	A
G	A
C	A
A	C
$	G
T	T
A	T
A	$

B–W	sorted
T	A
G	A
C	A
A	C
$	G
T	T
A	T
A	$

B–W	sorted
T	A
G	A
C	A
A	C
$	G
T	T
A	T
A	$

Goal: Find the read “GAT” in the genome “GATTACA”.

Step 1: Starting with the BWT, sort lexicographically, reconstructing the first column of the BWT matrix.

Source: Schbath, S., et al. (2012). Journal of Computational Biology, 19(6), 796-813.

Finding a word with BWT

B–W	sorted	B–W	sorted	B–W	sorted	B–W	sorted	B–W	sorted
T	A	T	A	T	A	T	A	T	A
G	A	G	A	G	A	G	A	G	A
C	A	C	A	C	A	C	A	C	A
A	C	A	C	A	C	A	C	A	C
$	G	$	G	$	G	$	G	$	G
T	T	T	T	T	T	T	T	T	T
A	T	A	T	A	T	A	T	A	T
A	$	A	$	A	$	A	$	A	$

Goal: Find the read “GAT” in the genome “GATTACA”.

Step 2: We proceed right-to-left in the query string, starting with “T” in “GAT”. Look for all instances of this letter in the sorted BWT.

Source: Schbath, S., et al. (2012). Journal of Computational Biology, 19(6), 796-813.

PGC
PHILIPPINE GENOME CENTER

Finding a word with BWT

B–W	sorted	B–W	sorted	B–W	sorted	B–W	sorted	B–W	sorted
T	A	T	A	T	A	T	A	T	A
G	A	G	A	G	A	G	A	G	A
C	A	C	A	C	A	C	A	C	A
A	C	A	C	A	C	A	C	A	C
$	G	$	G	$	G	$	G	$	G
T	T	T	T	T	T	T	T	T	T
A	T	A	T	A	T	A	T	A	T
A	$	A	$	A	$	A	$	A	$

Goal: Find the read “GAT” in the genome “GATTACA”.

Step 3: Remember that the BWT column represents the letters immediately preceding the sorted BWT in the text, because the text “wraps” around the cylinder. So let’s look at the corresponding letters in the BWT.

Source: Schbath, S., et al. (2012). Journal of Computational Biology, 19(6), 796-813.

PHILIPPINE GENOME CENTER

Finding a word with BWT

B–W	sorted	B–W	sorted	B–W	sorted	B–W	sorted	B–W	sorted
T	A	T	A	T	A	T	A	T	A
G	A	G	A	G	A	G	A	G	A
C	A	C	A	C	A	C	A	C	A
A	C	A	C	A	C	A	C	A	C
$	G	$	G	$	G	$	G	$	G
T	T	T	T	T	T	T	T	T	T
A	T	A	T	A	T	A	T	A	T
A	$	A	$	A	$	A	$	A	$

Goal: Find the read "GAT" in the genome "GATTACA".

Step 3: The two letters that come before "T" in the text are "T" and "A". Since we're looking for "GAT", we're interested in the "A".

Source: Schbath, S., et al. (2012). Journal of Computational Biology, 19(6), 796-813.

Finding a word with BWT

B–W	sorted
T	A
G	A
C	A
A	C
$	G
T	T
A	T
A	$

B–W	sorted
T	A
G	A
C	A
A	C
$	G
T	T
A	T
A	$

B–W	sorted
T	A
G	A
C	A
A	C
$	G
T	T
A	T
A	$

?

B–W	sorted
T	A
G	A
C	A
A	C
$	G
T	T
A	T
A	$

B–W	sorted
T	A
G	A
C	A
A	C
$	G
T	T
A	T
A	$

Goal: Find the read “GAT” in the genome “GATTACA”.

Step 4: We can repeat the process to find what letter(s) come before “AT” in the text. But first, we need to find which “A” in the sorted BWT corresponds to the “A” in the BWT we found earlier.

Source: Schbath, S., et al. (2012). Journal of Computational Biology, 19(6), 796-813.

PGC
PHILIPPINE GENOME CENTER

Finding a word with BWT

B–W	sorted
T	A
G	A
C	A
A	C
$	G
T	T
A	T
A	$

B–W	sorted
T	A
G	A
C	A
A	C
$	G
T	T
A	T
A	$

B–W		sorted
T	1	A
G	2	A
C	3	A
1 A		C
$		G
T		T
2 A		T
3 A		$

B–W	sorted
T	A
G	A
C	A
A	C
$	G
T	T
A	T
A	$

B–W	sorted
T	A
G	A
C	A
A	C
$	G
T	T
A	T
A	$

Goal: Find the read “GAT” in the genome “GATTACA”.

Step 4: The “A” that we found in the BWT is the second in that column, counting downwards. So it should match the second “A” in the sorted BWT.

Source: Schbath, S., et al. (2012). Journal of Computational Biology, 19(6), 796-813.

PGC
PHILIPPINE GENOME CENTER

Finding a word with BWT

B–W	sorted	B–W	sorted	B–W	sorted	B–W	sorted	B–W	sorted
T	A	T	A	T	A	T	A	T	A
G	A	G	A	G	A	G	A	G	A
C	A	C	A	C	A	C	A	C	A
A	C	A	C	A	C	A	C	A	C
$	G	$	G	$	G	$	G	$	G
T	T	T	T	T	T	T	T	T	T
A	T	A	T	A	T	A	T	A	T
A	$	A	$	A	$	A	$	A	$

Goal: Find the read "GAT" in the genome "GATTACA".

Step 5: Repeat Step 3, and find the corresponding letter in the BWT. Since this is a "G", we know that "GAT" does exist in our genome.

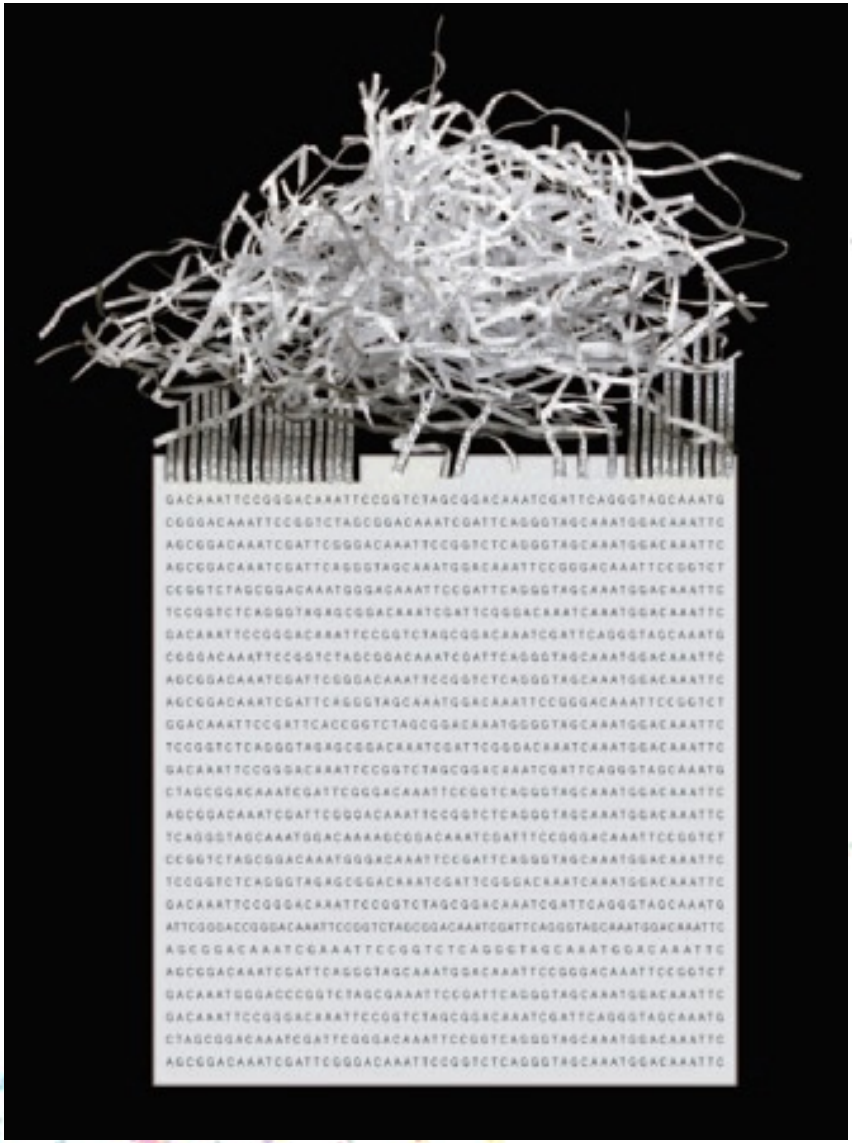
Source: Schbath, S., et al. (2012). Journal of Computational Biology, 19(6), 796-813.

PHILIPPINE GENOME CENTER

FM-index

- “Full-text index in Minute space”, invented by Paolo Ferragina and Giovanni Manzini in 2000.
- A BWT-based strategy for quickly searching for patterns within a compressed text.
- Includes additional tricks for finding locations, accounting for mismatches, increasing efficiency.
- FM-indexing lies at the core of popular read mapping software packages such as Bowtie and BWA.

	Bowtie 2	BWA
Index reference genome	`bowtie2-build`	`bwa index`
Map reads	`bowtie2`	`bwa mem`



Questions?

Source: Kelly Howe, Lawrence Berkeley Laboratory
Baker, Monya. "De novo genome assembly: what every biologist should know." Nature methods 9.4 (2012): 333-337.

PGC
PHILIPPINE GENOME CENTER