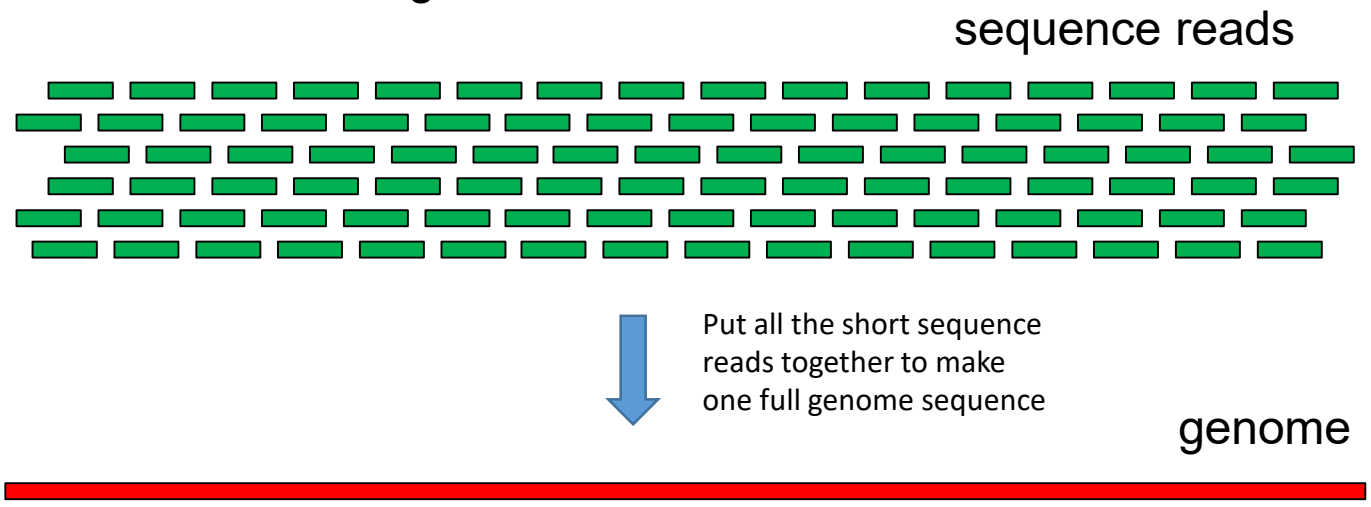
Ab initio Genome Assembly

Hugh Patterton
hpatterton@sun.ac.za

Room 2010 AI Perold Building

Ab initio Genome Assembly

- NGS sequencing produce sequence reads
- If we assemble the sequence by identifying overlaps between all sequences, we will potentially have **N^2 alignments** to test
- This problem is **tractable** with a **small number** of large **fragments**
- Human genome = 3.3×10^9 bp
- NGS produces ~100bp fragments
- With $10\times$ coverage, we have 3.3×10^8 fragments and 1×10^{17} comparisons
- If each comparison takes 1 ns, it will require ~3 years to do one assembly. **Not tractable**
- We need a better algorithm!



Eulerian path algorithm

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

For the last 20 years, fragment assembly in DNA sequencing followed the "overlap–layout–consensus" paradigm that is used in all currently available assembly tools. Although this approach proved useful in assembling clones, it faces difficulties in genomic shotgun assembly. We abandon the classical "overlap–layout–consensus" approach in favor of a new EULER algorithm that, for the first time, resolves the 20-year-old "repeat problem" in fragment assembly. Our main result is the reduction of the fragment assembly to a variation of the classical Eulerian path problem that allows one to generate accurate solutions of large-scale sequencing problems. EULER, in contrast to the CELERA assembler, does not mask such repeats but uses them instead as a powerful fragment assembly tool.

Children like puzzles, and they usually assemble them by trying all possible pairs of pieces and putting together pieces that match. Biologists assemble genomes in a surprisingly similar way, the major difference being that the number of pieces is larger. For

Because the Eulerian path approach transforms a once difficult layout problem into a simple one, a natural question is: "Could the Eulerian path approach be applied to fragment assembly?" Idury and Waterman, mimicked fragment assembly as an SBH problem (11) by representing every read of length n as a collection of $n - l + 1$ overlapping l-tuples (continuous short strings of fixed length l). At first glance, this transformation of every read into a collection of l-tuples (breaking the puzzle into smaller pieces) is a very short-sighted procedure, because information about the sequencing reads is lost. However, the loss of information is minimal for large l and is well paid for by the computational advantages of the Eulerian path approach. In addition, lost information can be restored at later stages.

Unfortunately, the Idury–Waterman approach, although very promising, did not scale up well. The problem is that sequencing errors transform a simple de Bruijn graph (corresponding to an error-free SBH) into a tangle of erroneous edges. Moreover, repeats pose serious challenges even in the case of error-free data

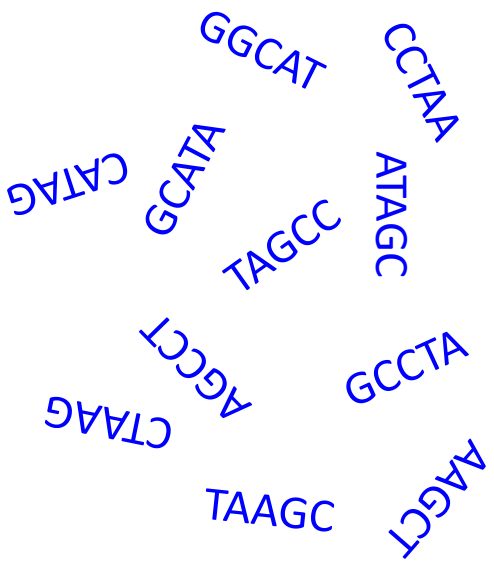
this natural approach—we never even try to match the pairs of fragments, and we do not have the overlap step at all. Instead, we do a very counterintuitive (some would say childish) thing: we cut the existing pieces of a puzzle into even smaller pieces of regular shape. Although it indeed looks childish and irresponsible, we do it on purpose rather than for the fun of it. This operation transports the puzzle assembly from the world of a difficult Layout Problem

PNAS 2001 98:9748-9753

Break each sequence read into over-lapping k-mers

GGCATAGCCTAAGCT length *L*

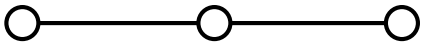
GGCAT
GCATA
CATAG
ATAGC
TAGCC
AGCCT
GCCTA
CCTAA
CTAAG
TAAGC
AAGCT



- Given **any sequence** read of length *L*, we can **represent** it as ***k*-mers** (5-mers in this case), **over-lapping by k-1 nt**
- All the ***k*-mers** are written to a **hash table**
- A hash table is an table where the *k*-mer sequence is "hashed" to make a key, which points to an address where the *k*-mer sequence is stored
- A ***k*-mer** is then **chosen**, to start with, and a **de Brijn graph**, is **constructed**

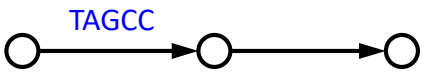
Using a de Brijn graph to represent over-lapping *k*-mers

The de Brijn graph consists of **nodes** (circles) and **edges** (connecting lines)

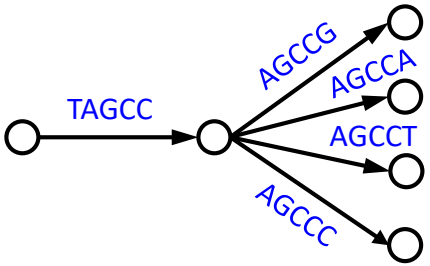


In a **directed de Brijn graph** each **edge** (line) has a **direction**

For the *k*-mer that you have chosen, say TAGCC, write the sequence on an edge



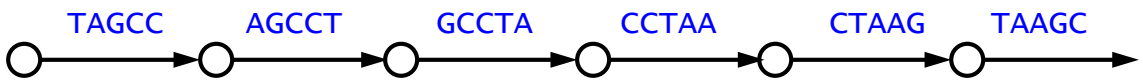
The ***k*-mer** that **follows** TAGCC will be AGCC with the last nucleotide G, A, T or C.
Look in the **hash table** to see what *k*-mers you **have**. If you have all four, indicate that



With this scheme we will expand 4^n for every step, and will get very complex graphs or run out of memory!

Use *k*-mers of a length where the *k*-mers-1 are unique

- If **each *k*-mer-1 is unique**, there can only be one **overlapping *k*-mer** in the **hash table**



- If you choose *k*-mers that are **too long**, there will be a small collection of unique *k*-mers, but **not all *k*-mers** will **have a over-lapping *k*-mer**
- If you do not have the overlapping *k*-mer, you **cannot extend** the **de Brijn graph**
- There is thus a **balance** of having a *k*-mer length giving *k*-mers that are as **unique as possible**, but simultaneously also have **over-lapping *k*-mers** for as many *k*-mers as possible
- For a **linear** sequence, you **keep on adding** overlapping ***k*-mers** until you **run out** of *k*-mers → you then have the linear sequence
- For a **circular** sequence, you continue adding *k*-mers until you **return** to your **starting *k*-mer**

Effect of *k*-mer length on uniqueness

ATAGATAAATACATAT
ATA
TAG
AGA
GAT
ATA
TAA
AAA
AAT
ATA
TAC
ACA
CAT
ATA
TAT

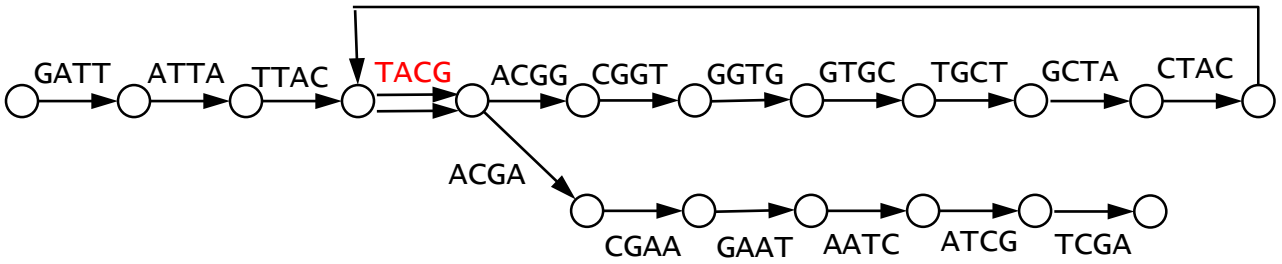
3-mers generate 4 identical *k*-mers

ATAGATAAATACATAT
ATAG
TAGA
AGAT
GATA
ATAA
TAAA
AAAT
AATA
ATAC
TACA
ACAT
CATA
ATAT

4-mers from the same sequence are all unique

The superpath through a de Brijn graph is the sequence

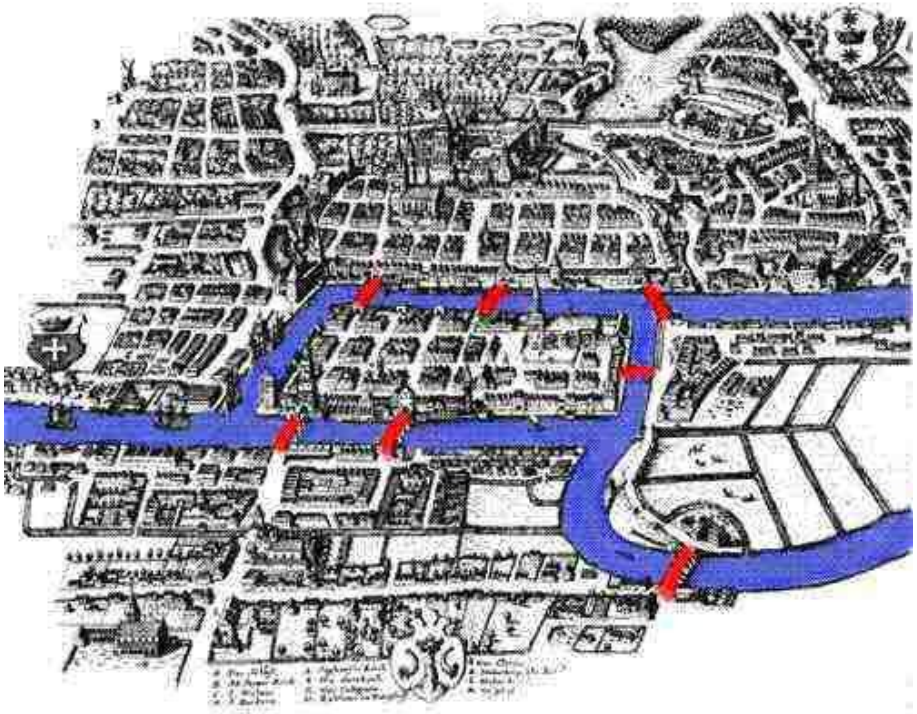
GATTACGGTGCTACGAATCGA
GATT
ATTA
TTAC
TACG
ACGG
CGGT
GGTG
GTGC
TGCT
GCTA
CTAC
TACG
ACGA
CGAA
GAAT
AATC
ATCG
TCGA



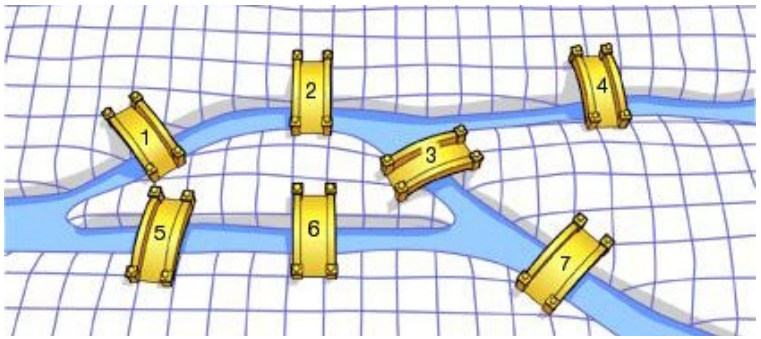
- If a *k*-mer (or group of *k*-mers) occur **more than once**, use the corresponding **number of edges**
- **Insert** branches and **returns**, as required
- If each node **has as many entering** as **exiting** arrows, the de Brijn graph is **balanced**, and a single path visiting all edges of the graph exists (**eulerian walk**)
- This **eulerian walk** or **superpath** is the **sequence**
- If **exactly two nodes** have an **odd number** of **inward** and **outward** arrows, these nodes are the **sequence termini**

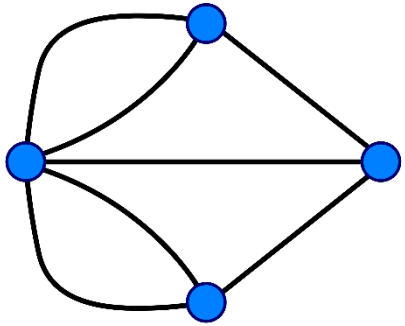
The bridges of Koningsberg problem

Can I do a tour of Koningsberg where I cross each bridge once?



Swiss mathematician





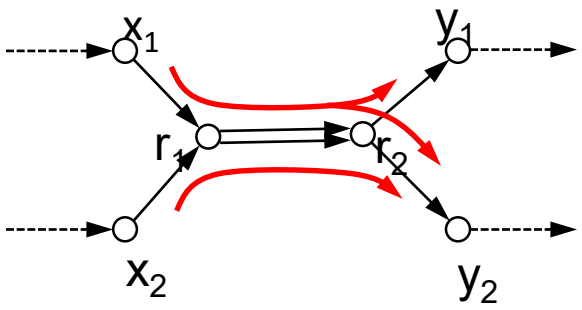
- If a **connected graph** has **any node** with an **odd number** of edges, there **is no eulerian cycle**
- The fabled Koningsberg tour is **not possible**

When will a de Bruijn graph have a eulerian cycle or walk?

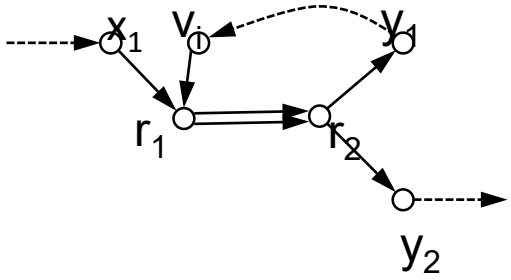
- A de Bruijn graph where **all nodes** are **connected** by edges is a **connected graph**
- **Undirected** de Bruijn graph: **no direction** is given to edges
- **Directed** de Bruijn graph: can cross an edge only in the defined direction
- A **eulerian cycle** is a **path through the entire graph**, **crossing each edge once**, and returning to the starting node
- A **undirected**, connected graph has an eulerian cycle if all nodes contain an **even number of edges**
- A **directed**, connected de Bruijn graph has a eulerian cycle if **the number of edges entering each node equals the number of edges exiting each node for all nodes** in the graph
- If the de Bruijn graph is **unconnected** (there are two nodes with an unequal number of entering and exiting edges), an eulerian walk (path traversing the entire graph once) exists if the **number of entering edges equals the number of exiting edges for all nodes, except two** (the “starting” and “ending” node)

Compeau et al. 2011 Nature Biotech

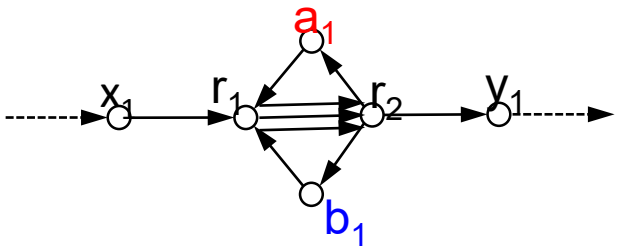
Repeats can pose a problem



$x_1r_1r_2y_1$?
$x_1r_1r_2y_2$?



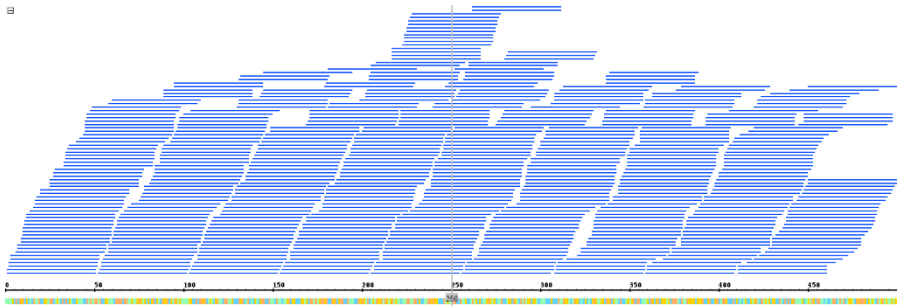
Sometimes knowledge of the global structure leads to the correct solution:
$x_1r_1r_2y_1v_ir_1r_2y_2$



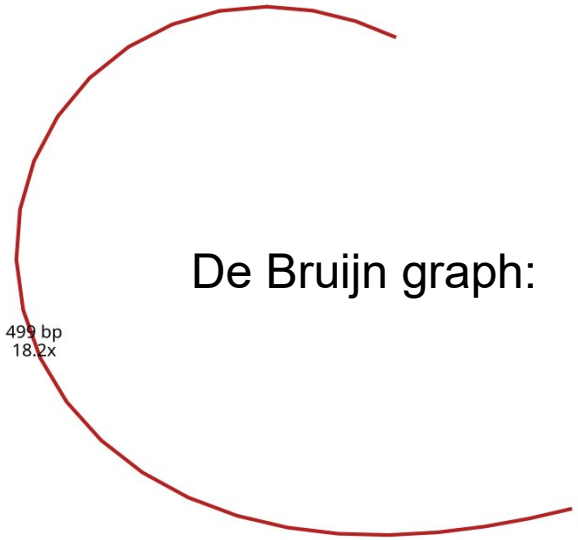
$x_1r_1r_2a_1r_1r_2b_1r_1r_2y_1$?
$X_1r_1r_2b_1r_1r_2a_1r_1r_2y_1$?

Test of a genome assembly program

Give SPAdes 50nt “reads” selected at random positions from a 500 nt sequence at 50 X coverage



De Bruijn graph:



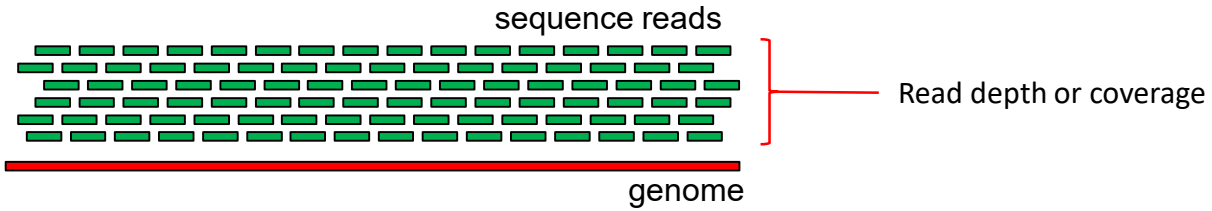

```
>>NODE_1_length_499_cov_18.2403                          (499 nt)
 Waterman-Eggert score: 2495;  570.0 bits; E(1) <  6.5e-167
100.0% identity (100.0% similar) in 499 nt overlap (2-500:1-499)

               10        20        30        40        50        60
seq     TCATCTACGTTTAATCCCCGTTTCTTCGCTGTTCACTCCACACCTCTGAGCTGCGTAGGC
        ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
NODE_1  TCATCTACGTTTAATCCCCGTTTCTTCGCTGTTCACTCCACACCTCTGAGCTGCGTAGGC
               10        20        30        40        50        60

               70        80        90       100       110       120
seq     CAAGAGTTACCTCCCCGTTTTTCCTCGCCGATAAACTACTCAATCGTCCGAAAAGACATC
        ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
NODE_1  CAAGAGTTACCTCCCCGTTTTTCCTCGCCGATAAACTACTCAATCGTCCGAAAAGACATC
               70        80        90       100       110       120

              130       140       150       160       170       180
seq     TGGAGGTCAGAGAGGGCAGTGTGGCTAGCGAAACTCGGTCCGGCTTAACGAATACCTCGA
        ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
NODE_1  TGGAGGTCAGAGAGGGCAGTGTGGCTAGCGAAACTCGGTCCGGCTTAACGAATACCTCGA
              130       140       150       160       170       180

              190       200       210       220       230       240
seq     CTTTAACGTCTGCAGGTTGGGTGCAGTCGACGTTATATCTACATACATTGATAGTATTAG
        ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
NODE_1  CTTTAACGTCTGCAGGTTGGGTGCAGTCGACGTTATATCTACATACATTGATAGTATTAG
              190       200       210       220       230       240

              250       260       270       280       290       300
seq     AACGGGTGATTTACAGATCTAGGTTGCCGACACAGAAGAAAACCCTGTGTACCTAAGGCG
        ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
NODE_1  AACGGGTGATTTACAGATCTAGGTTGCCGACACAGAAGAAAACCCTGTGTACCTAAGGCG
              250       260       270       280       290       300

              310       320       330       340       350       360
seq     CGACGCTATTAGTCCCTTCAACGTGTCCTTTGGTCTGTAATTCTCGTTTTTAAAGCAGCC
        ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
NODE_1  CGACGCTATTAGTCCCTTCAACGTGTCCTTTGGTCTGTAATTCTCGTTTTTAAAGCAGCC
              310       320       330       340       350       360

              370       380       390       400       410       420
seq     GGCATAAAAAGGTGCTAGGAGTAATGAAAGAACACGAGCGATGGGGAATCACTTGTTGCA
        ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
NODE_1  GGCATAAAAAGGTGCTAGGAGTAATGAAAGAACACGAGCGATGGGGAATCACTTGTTGCA
              370       380       390       400       410       420

              430       440       450       460       470       480
seq     ATATAATGACGGCGAACCACATGTTTTGATATCCTTAGGCCTAAGCCGGCTCCCCCGTGT
        ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
NODE_1  ATATAATGACGGCGAACCACATGTTTTGATATCCTTAGGCCTAAGCCGGCTCCCCCGTGT
              430       440       450       460       470       480

              490       500
seq     TGGAAATAGAAATTCTCCT
        ::::::::::::::::::::
NODE_1  TGGAAATAGAAATTCTCCT
              490
```

Genome assembly programs

Program	Genome	Technology	Link
ABySS	(large) genomes	Solexa, SOLiD	http://www.bcgsc.ca/platform/bioinfo/software/abyss
ALLPATHS-LG	(large) genomes	Solexa, SOLiD	http://software.broadinstitute.org/allpaths-lg/blog/
Newbler	genomes, ESTs	454, Sanger	https://swes.cals.arizona.edu/maier_lab/kartchner/documentation/index.php/home/docs/newbler
SOAPdenovo	genomes	Solexa	http://soap.genomics.org.cn/soapdenovo.html
SPAdes	(small) genomes, single-cell	Illumina, Solexa, Sanger, 454, Ion Torrent, PacBio, Oxford Nanopore	http://cab.spbu.ru/software/spades/
Velvet	(small) genomes	Sanger, 454, Solexa, SOLiD	https://www.ebi.ac.uk/~zerbino/velvet/

Fold coverage or read depth

- FASTQ file gives the quality of assigned nucleotides as a Phred score
- Alignment of reads to a reference genome gives mapping quality as a MAPQ score
- The statistical significance of the presence of a specific nucleotide or distribution of nucleotides at a given position also depends on the **number of times** that the **specific nucleotide** has been **included** in an independent **sequence read**, i.e., **the number of time a nucleotide has been sequenced**
- This number is given by the **average sequence coverage**

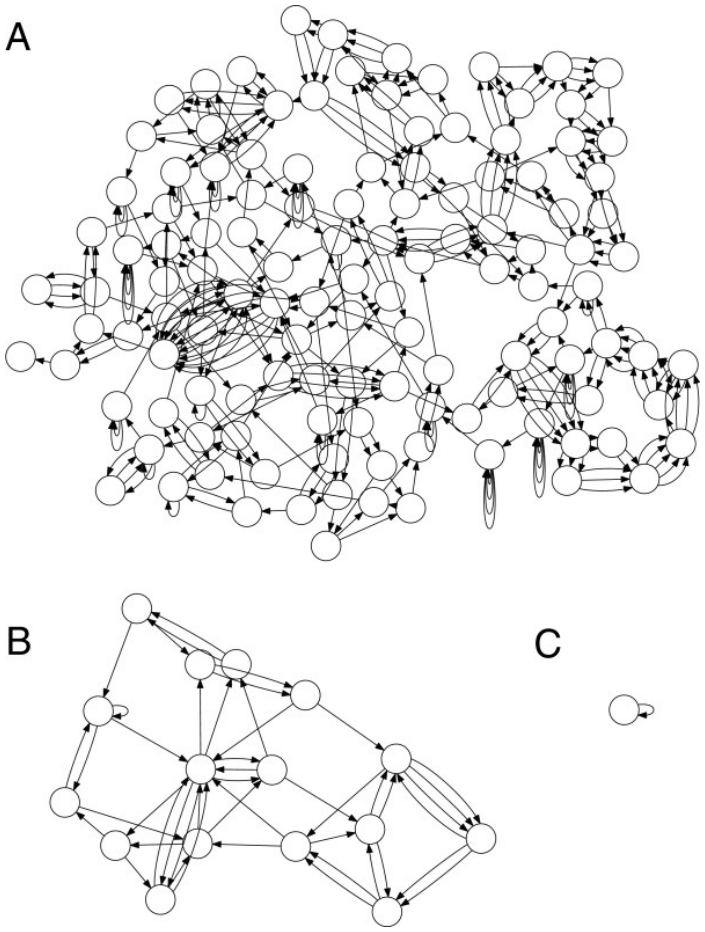


- The sequence coverage *c* is given by the **total number of nucleotides sequenced** divided by the **size of the sequenced genome**

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

- Where *L* is the read length (50 nt, 100 nt etc.), *N* is the number of reads, and *G* is the genome size
- **Different** coverage **values** are required for **different applications**, and typically vary from 10× to 100× coverage

What is the quality of a genome assembly?



- De Brijn graph **complexity decreases** with **increase** in **sequence run length**
- When an assembly produces several **separate de Brijn graphs**, each section represents a **contig**
- **Genome sequences** are typically composed of **many contigs**
- With **paired-end reads** spanning contigs, contigs can be assembled as a **scaffold**