Fragment assembly of DNA

INDRAPRASTHA INSTITUTE *of* INFORMATION TECHNOLOGY **DELHI**

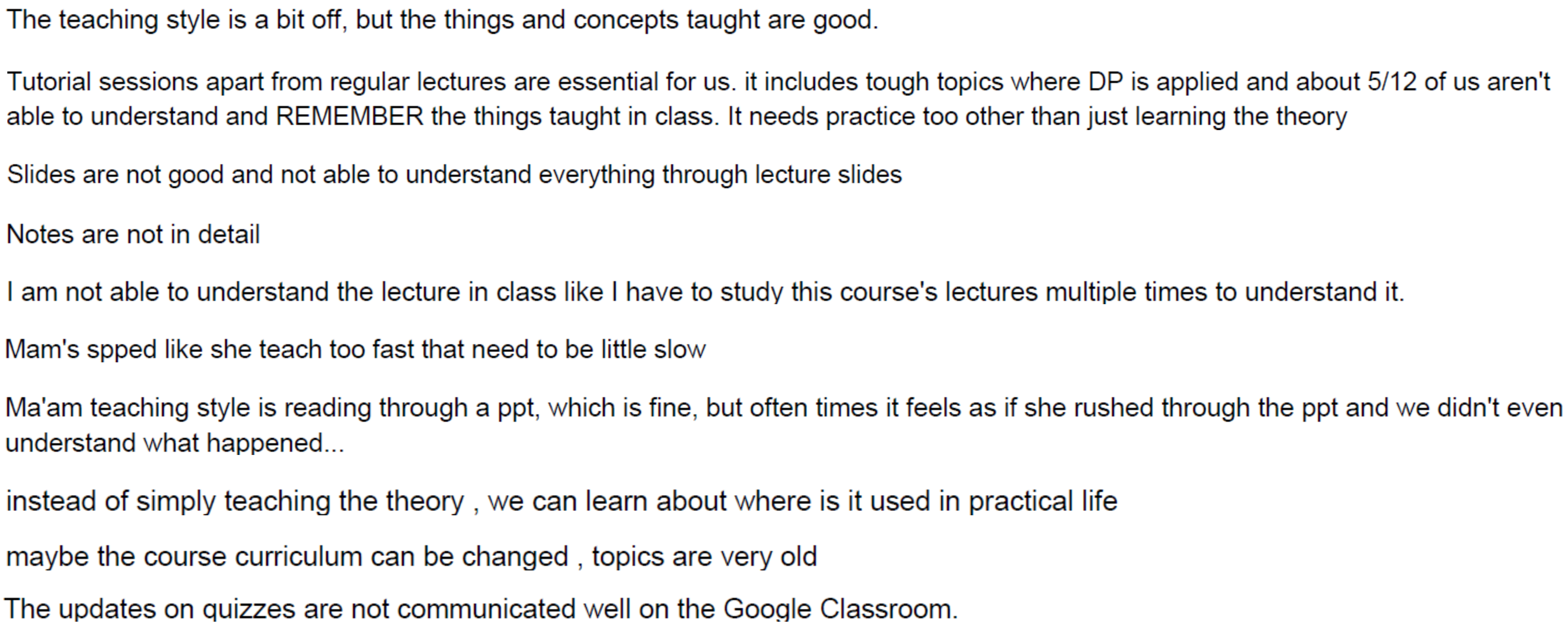
Dr. Jaspreet Kaur Dhanjal

Assistant Professor, Center for Computational Biology

Email ID: jaspreet@iiitd.ac.in

September 16, 2025

Mid-sem Feedback



The teaching style is a bit off, but the things and concepts taught are good.

Tutorial sessions apart from regular lectures are essential for us. it includes tough topics where DP is applied and about 5/12 of us aren't able to understand and REMEMBER the things taught in class. It needs practice too other than just learning the theory

Slides are not good and not able to understand everything through lecture slides

Notes are not in detail

I am not able to understand the lecture in class like I have to study this course's lectures multiple times to understand it.

Mam's spped like she teach too fast that need to be little slow

Ma'am teaching style is reading through a ppt, which is fine, but often times it feels as if she rushed through the ppt and we didn't even understand what happened...

instead of simply teaching the theory , we can learn about where is it used in practical life

maybe the course curriculum can be changed , topics are very old

The updates on quizzes are not communicated well on the Google Classroom.

De Bruijn graph assembly

k-mer

"*k*-mer" is a substring of length *k*

S: GGCGATTCATCG

A 4-mer of *S*: ATTC

All 3-mers of *S*:
GGC
GCG
CGA
GAT
ATT
TTC
TCA
CAT
ATC
TCG

mer: from Greek meaning "part"

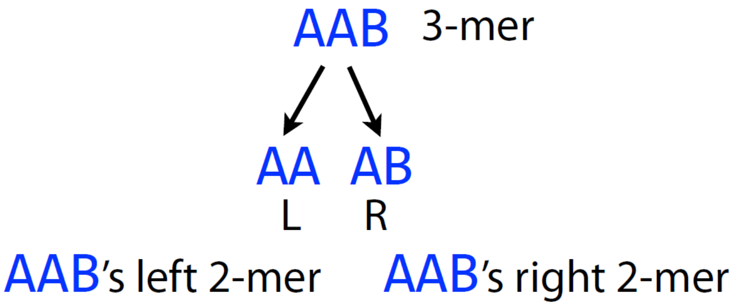
I'll use "*k*-1-mer" to refer to a substring of length *k* - 1

De Bruijn graph

As usual, we start with a collection of reads, which are substrings of the reference genome.

AAA, AAB, ABB, BBB, BBA

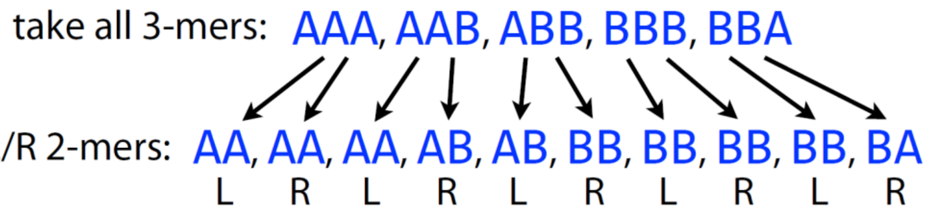
AAB is a *k*-mer (*k* = 3). AA is its *left* *k*-1-mer, and AB is its right *k*-1-mer.



De Bruijn graph

Take each length-3 input string and split it into two overlapping substrings of length 2. Call these the *left* and *right 2-mers*.

AAABBBA

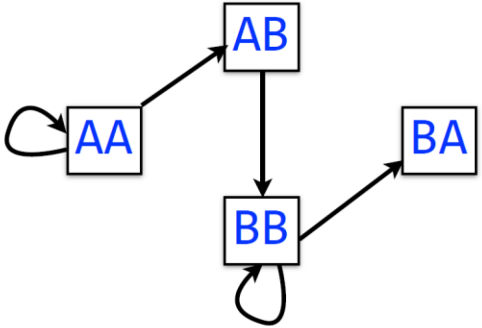


take all 3-mers: AAA, AAB, ABB, BBB, BBA

form L/R 2-mers: AA, AA, AA, AB, AB, BB, BB, BB, BB, BA

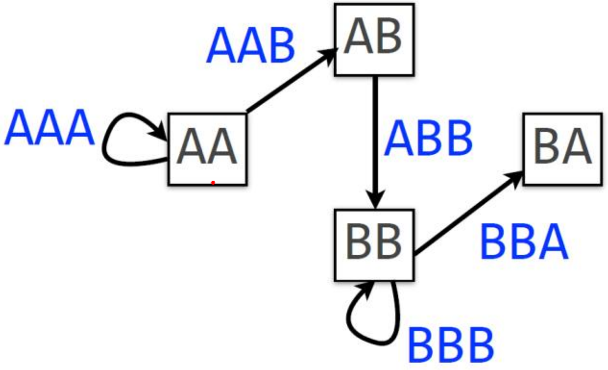
L R L R L R L R L R

Let 2-mers be nodes in a new graph. Draw a directed edge from each left 2-mer to corresponding right 2-mer:



Each *edge* in this graph corresponds to a length-3 input string

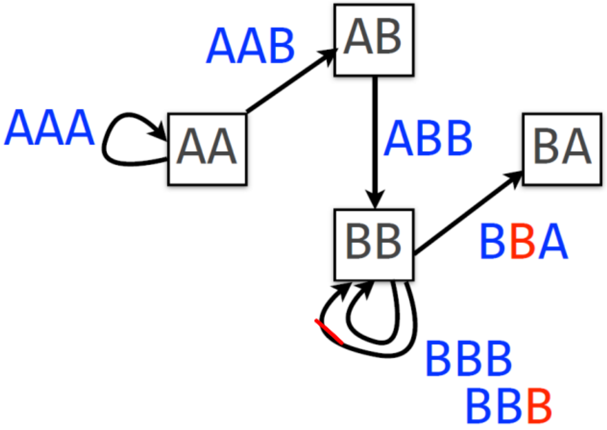
De Bruijn graph



An edge corresponds to an overlap (of length *k*-2) between two *k*-1 mers. More precisely, it corresponds to a *k*-mer from the input.

De Bruijn graph

If we add one more B to our input string: AAABBBBA, and rebuild the De Bruijn graph accordingly, we get a *multiedge*.



Directed multigraph

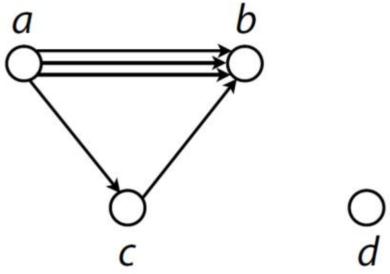
Directed **multigraph** $G(V, E)$ consists of set of *vertices, V* and **multiset** of *directed edges, E*

Otherwise, like a directed graph

Node's *indegree* = # incoming edges

Node's *outdegree* = # outgoing edges

De Bruijn graph is a directed multigraph



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

$E = \{ \underbrace{(a, b), (a, b), (a, b)}_{\text{Repeated}}, (a, c), (c, b) \}$

Eulerian walk definitions and statements

Node is *balanced* if indegree equals outdegree

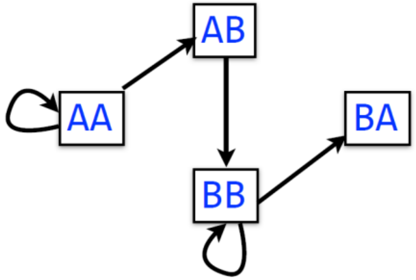
Node is *semi-balanced* if indegree differs from outdegree by 1

Graph is *connected* if each node can be reached by some other node

Eulerian walk visits each edge exactly once

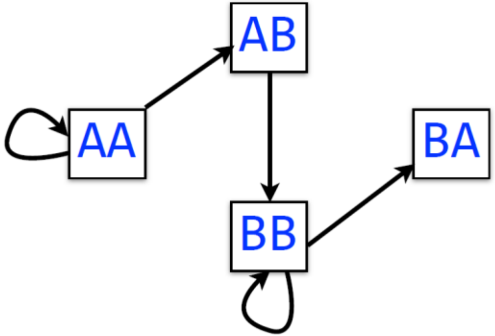
Not all graphs have Eulerian walks. Graphs that do are *Eulerian.*
(For simplicity, we won't distinguish Eulerian from semi-Eulerian.)

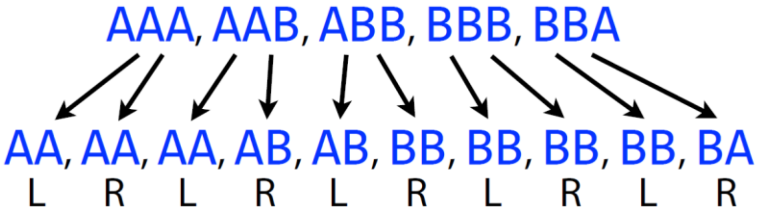
A directed, connected graph is Eulerian if and only if it has at most 2 semi-balanced nodes and all other nodes are balanced



De Bruijn graph

Back to our De Bruijn graph





AAA, AAB, ABB, BBB, BBA

AA, AA, AA, AB, AB, BB, BB, BB, BB, BA

L R L R L R L R L R

Is it Eulerian? Yes

Argument 1: AA → AA → AB → BB → BB → BA

Argument 2: AA and BA are semi-balanced, AB and BB are balanced

De Bruijn graph

A procedure for making a De Bruijn graph
for a genome

Assume *perfect sequencing* where each length-*k* substring is sequenced exactly once with no errors

Pick a substring length *k*: 5

Start with an input string: a_long_long_long_time

↓

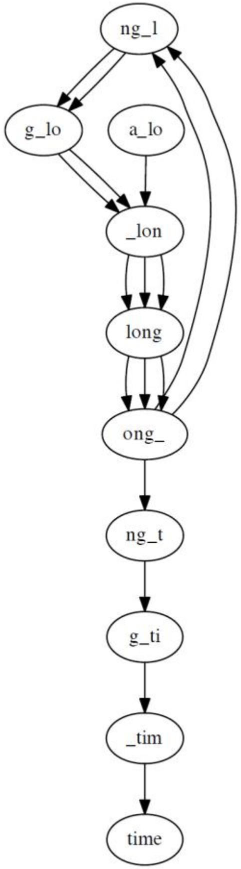
long_

↙ ↘

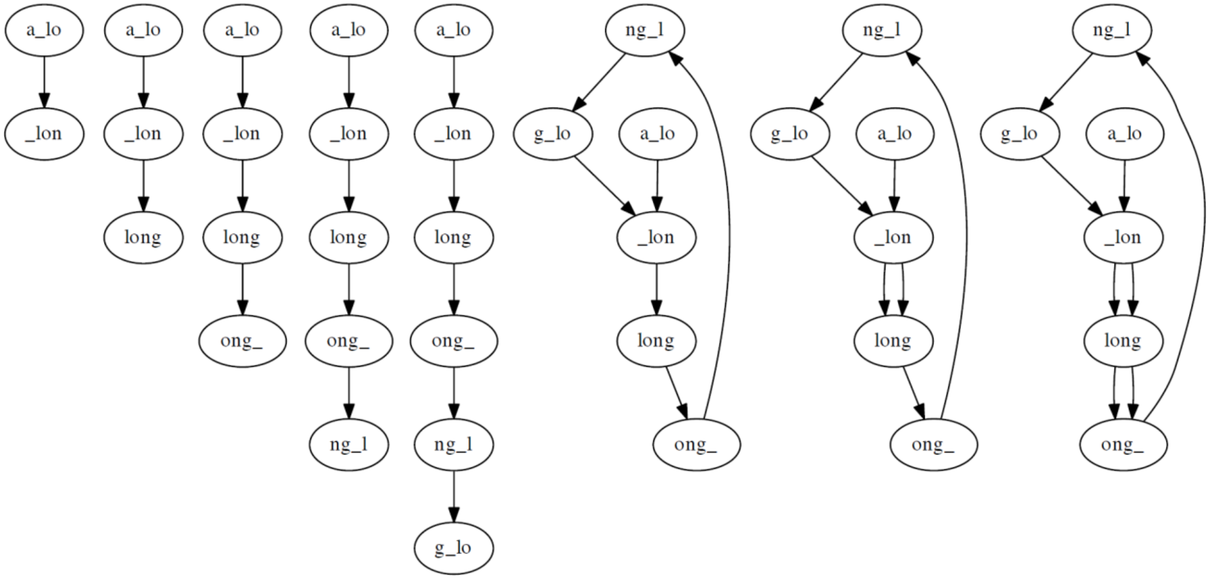
long ong_

Take each *k* mer and split into left and right *k*-1 mers

Add k-1 mers as nodes to De Bruijn graph (if not already there), add edge from left *k*-1 mer to right *k*-1 mer



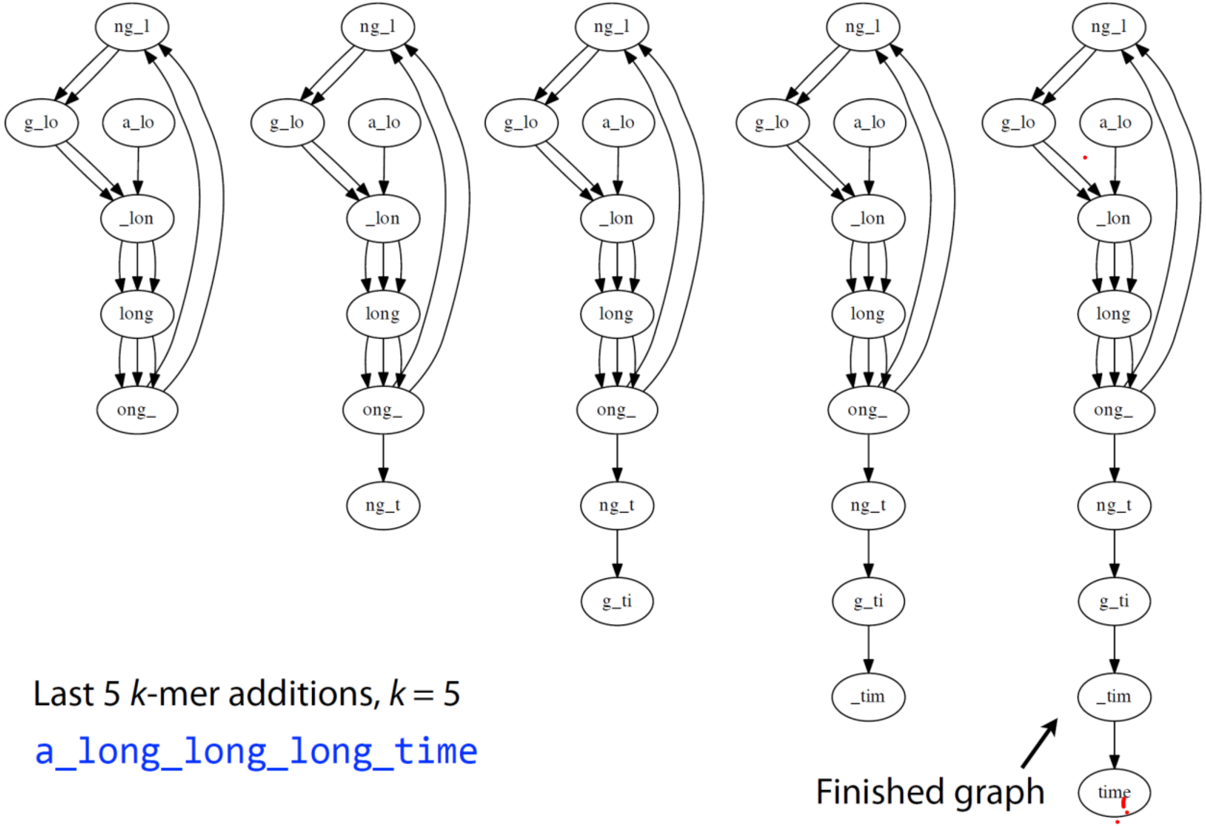
De Bruijn graph



First 8 *k*-mer additions, $k = 5$

a_long_long_long_time

De Bruijn graph



Last 5 *k*-mer additions, $k = 5$

a_long_long_long_time

Finished graph

De Bruijn graph

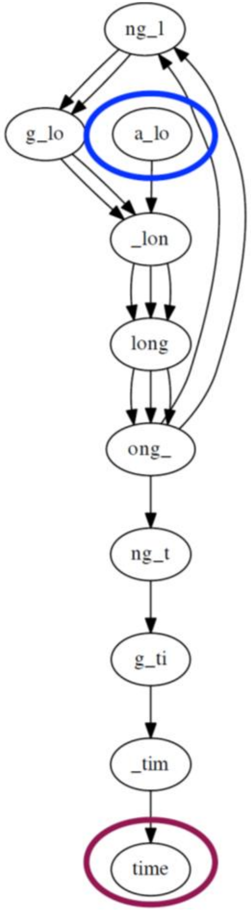
With perfect sequencing, this procedure always yields an Eulerian graph. Why?

Node for *k*-1-mer from left end is semi-balanced with one more outgoing edge than incoming *

Node for *k*-1-mer at right end is semi-balanced with one more incoming than outgoing *

Other nodes are balanced since # times *k*-1-mer occurs as a left *k*-1-mer = # times it occurs as a right *k*-1-mer

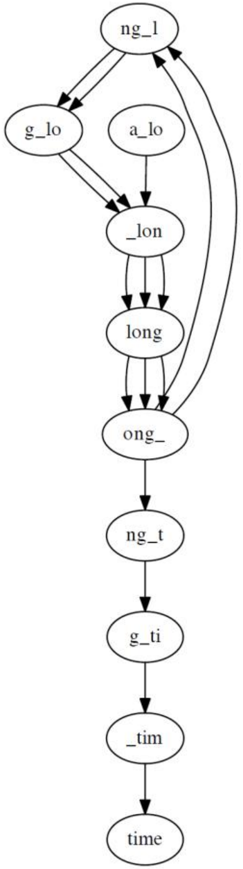
* Unless genome is circular



De Bruijn graph

Assuming perfect sequencing, procedure yields graph with Eulerian walk that can be found efficiently.

We saw cases where Eulerian walk corresponds to the original superstring. Is this always the case?



De Bruijn graph

No: graph can have multiple Eulerian walks, only one of which corresponds to original superstring

Right: graph for ZABCDABEFABY, $k = 3$

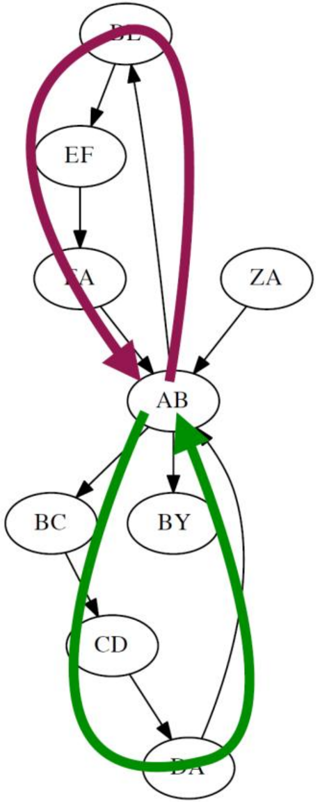
Alternative Eulerian walks:

ZA → AB → BE → EF → FA → AB → BC → CD → DA → AB → BY

ZA → AB → BC → CD → DA → AB → BE → EF → FA → AB → BY

These correspond to two edge-disjoint directed cycles joined by node AB

AB is a repeat: ZABCDABEFABY



De Bruijn graph

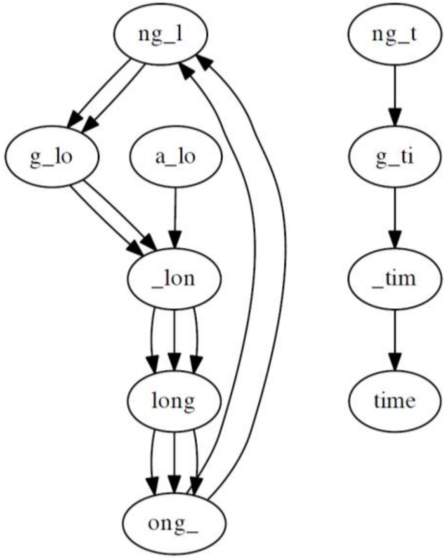
This is the first sign that Eulerian walks can't solve all our problems

Other signs emerge when we think about how actual sequencing differs from our idealized construction

Gaps in coverage can lead to *disconnected* graph

Graph for `a_long_long_long_time`, $k = 5$ but *omitting* `ong_t` :

Connected components are individually Eulerian, overall graph is not

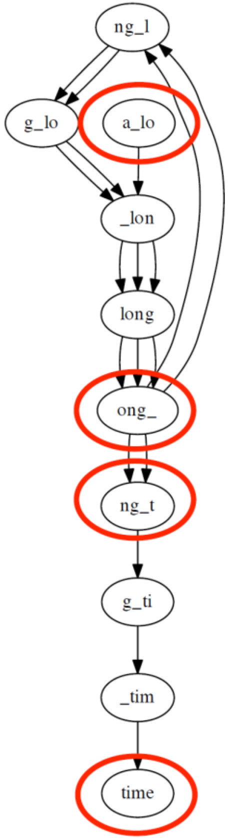


De Bruijn graph

Differences in coverage also lead to non-Eulerian graph

Graph for a_long_long_long_time, *k* = 5 but with *extra copy* of ong_t :

Graph has 4 semi-balanced nodes, isn't Eulerian

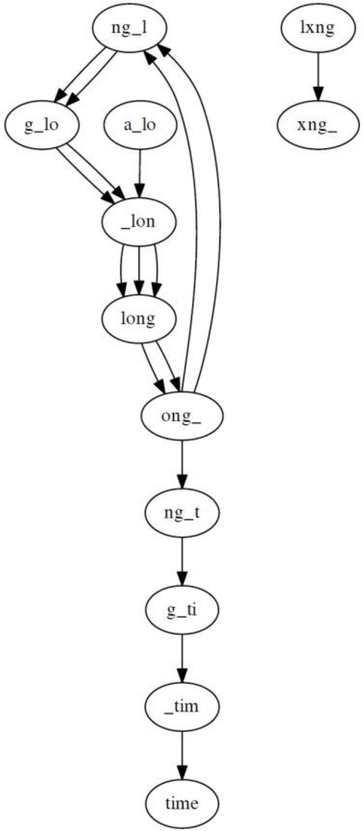


De Bruijn graph

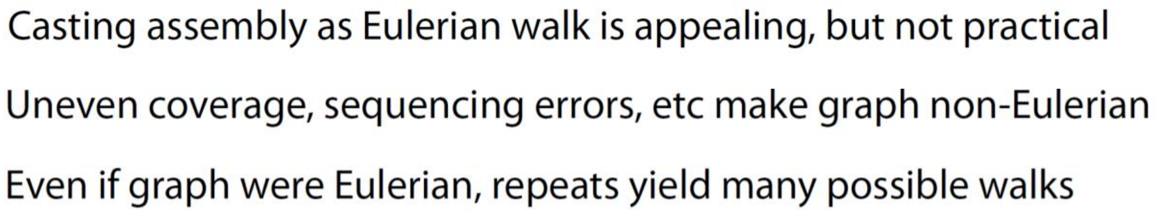
Errors and differences between chromosomes also lead to non-Eulerian graphs

Graph for `a_long_long_long_time`, *k* = 5 but with error that turns a copy of `long_` into `lxng_`

Graph is not connected; largest component is not Eulerian



De Bruijn graph



Casting assembly as Eulerian walk is appealing, but not practical

Uneven coverage, sequencing errors, etc make graph non-Eulerian

Even if graph were Eulerian, repeats yield many possible walks