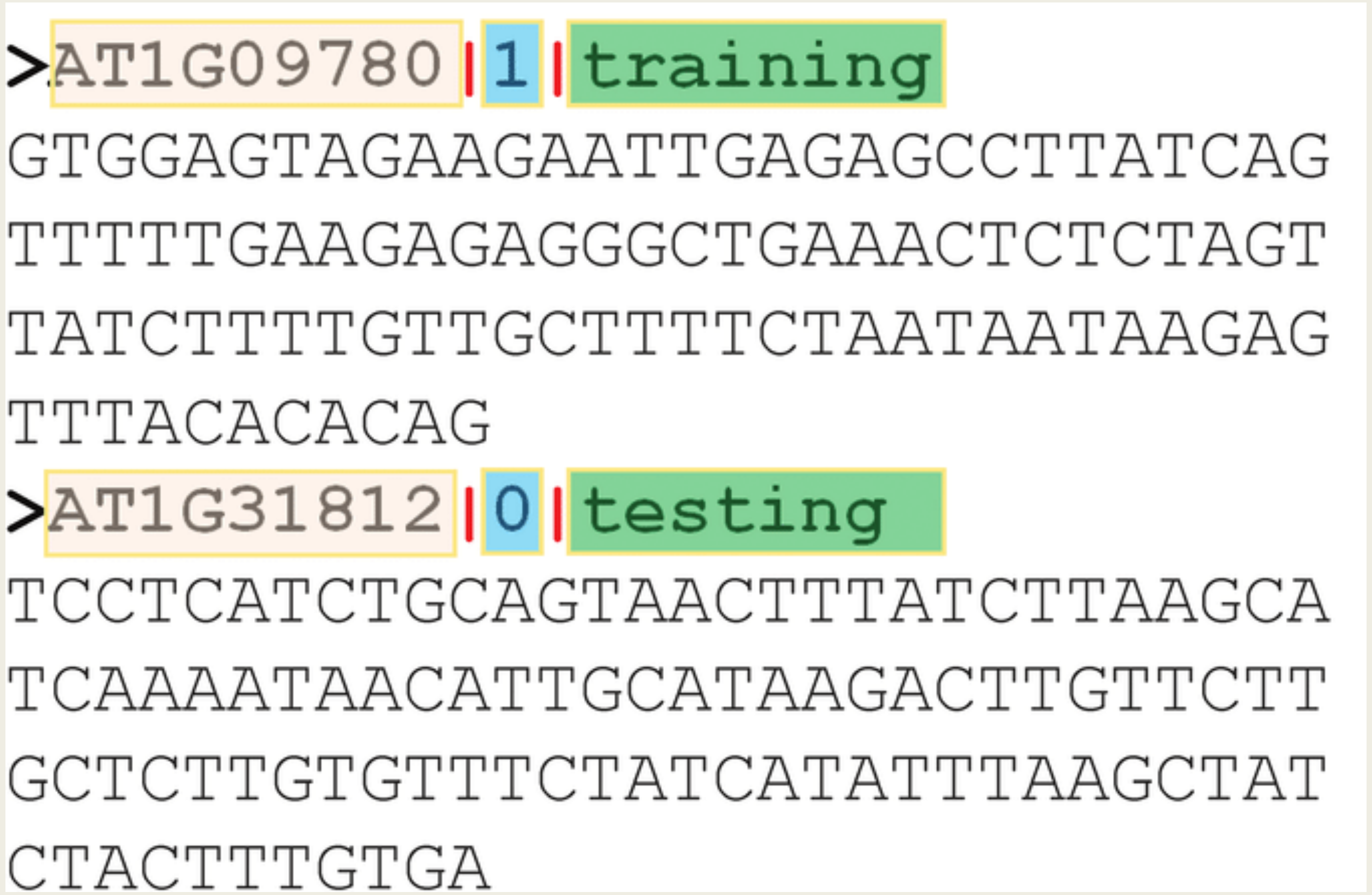
CS 364 COMPUTATIONAL BIOLOGY

Sara Mathieson
Haverford College

Lab 3 notes

- Fasta file format (can be used for the reads or the reference, or any other sequence)



```
>AT1G09780|1|training
GTGGAGTAGAAGAATTGAGAGCCTTATCAG
TTTTTGAAGAGAGGGCTGAAACTCTCTAGT
TATCTTTTGTTGCTTTTCTAATAATAAGAG
TTTACACACAG
>AT1G31812|0|testing
TCCTCATCTGCAGTAACTTTATCTTAAGCA
TCAAAATAACATTGCATAAGACTTGTTCTT
GCTCTTGTGTTTCTATCATATTTAAGCTAT
CTACTTTGTGA
```

Credit: Jiangning Song

Outline

- de Bruijn graphs (DBGs)
- Traversing DBGs
- Assembling contigs with DBGs

Review Overlap Graph Assembly

Steps of Overlap Graph Assembly (also called "overlap-layout-consensus")

1) Compute **overlaps between all pairs of reads**. With R = number of reads and m = length of reads, this is naively $O(R^2m^2)$. We will learn better ways of "aligning" sequences next week.

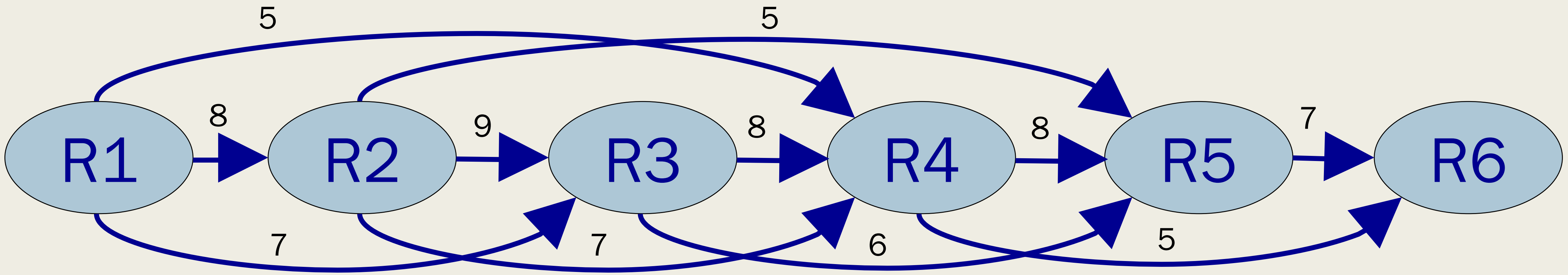
2) Construct a **graph with reads as the nodes** and **directed, weighted edges** between reads with >= T overlap.

3) "Layout" the graph and try to "group" stretches of the graph into "**contigs**" (short for contiguous), these are (hopefully) long portions of the original genome

4) Find a "consensus" *sequence* for each contig

Activity example: m = 10, T = 5

```
ATATATACTGGCGTATCGCAGTAAACGCGCCG
  R1: ACTGGCGTAT
  R2:   TGGCGTATCG
  R3:    GGCGTATCGC
  R4:      CGTATCGCAG
  R5:        TATCGCAGTA
  R6:           CGCAGTAAAC
```



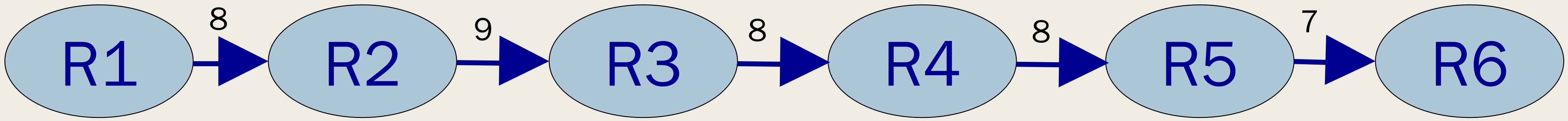
$$\begin{tikzcd}
R1 \arrow[r, "8"] \arrow[rr, bend right, "7"'] \arrow[rrr, bend left, "5"] & R2 \arrow[r, "9"] \arrow[rr, bend right, "7"'] \arrow[rrr, bend left, "5"] & R3 \arrow[r, "8"] \arrow[rr, bend right, "6"'] & R4 \arrow[r, "8"] \arrow[rr, bend right, "5"'] & R5 \arrow[r, "7"] & R6
\end{tikzcd}$$

Li et al, *Briefings in Functional Genomics* (2012)

Activity example: m = 10, T = 5

```
ATATATACTGGCGTATCGCAGTAAACGCGCCG
  R1: ACTGGCGTAT
  R2:   TGGCGTATCG
  R3:    GGCGTATCGC
  R4:      CGTATCGCAG
  R5:        TATCGCAGTA
  R6:           CGCAGTAAAC
```

First simplification: remove edges that can be (transitively) inferred from other edges

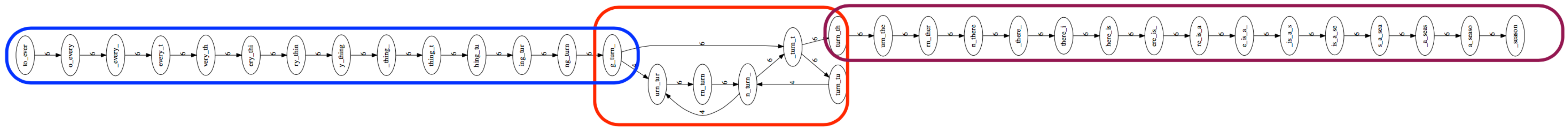


$$\begin{tikzcd} R1 \arrow[r, "8"] & R2 \arrow[r, "9"] & R3 \arrow[r, "8"] & R4 \arrow[r, "8"] & R5 \arrow[r, "7"] & R6 \end{tikzcd}$$

Li et al, *Briefings in Functional Genomics* (2012)

Layout

Emit *contigs* corresponding to the non-branching stretches



Contig 1
to_every_thing_turn_

Contig 2
turn_there_is_a_season

Unresolvable repeat

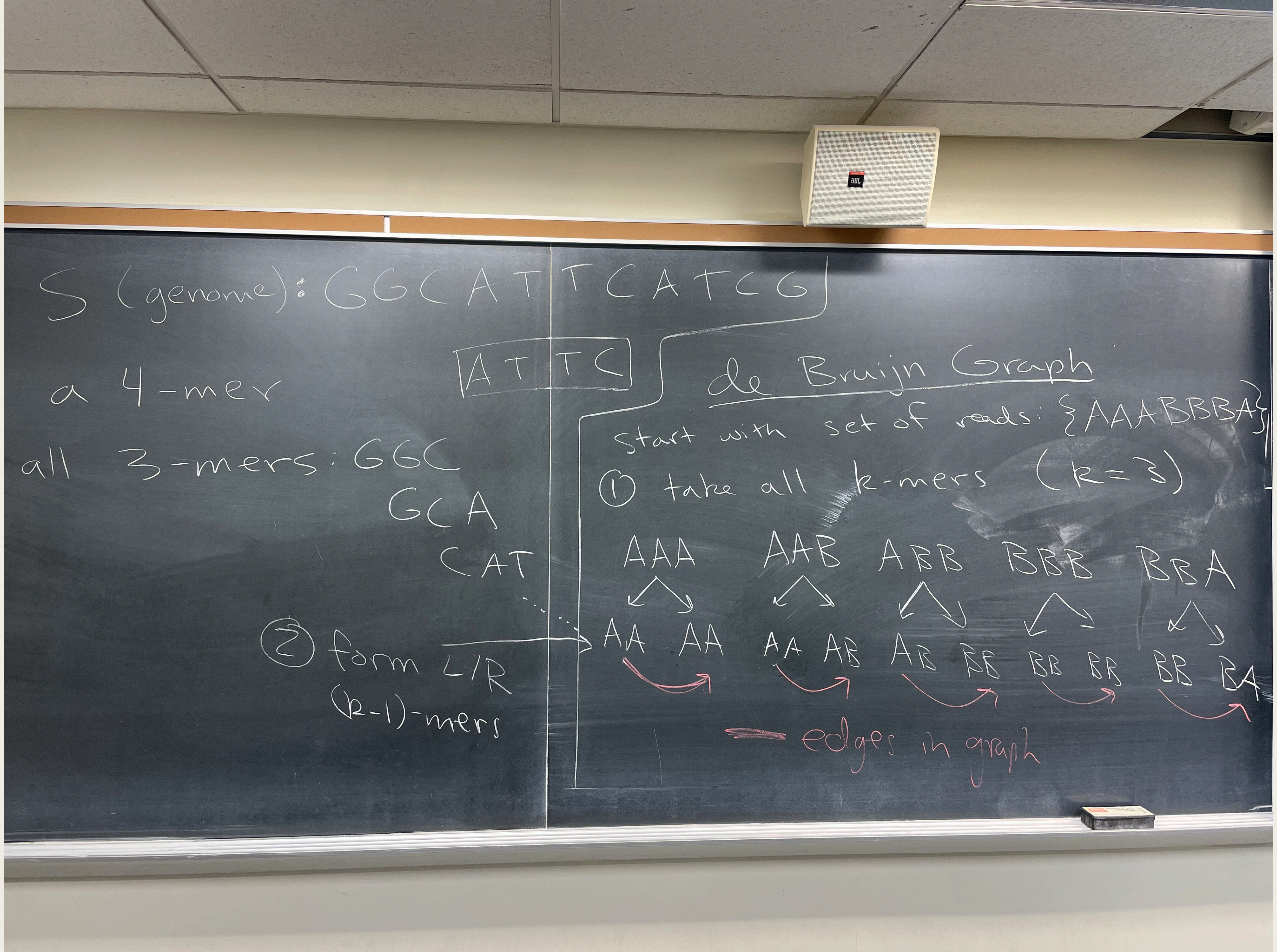
Original string: “`to_every_thing_turn_turn_turn_there_is_a_season`”

Slide: adapted from Ben Langmead, John Hopkins

Issues with overlap graph assembly

- Next-generation sequencing produces 100’s of millions (or even billions) of reads
- With one node per read this is computationally intractable for large genomes
- What if the nodes in our graph were not reads?

de Bruijn graphs



S (genome): GGCAT|TCATCG

a 4-mer: $\boxed{\text{ATTC}}$

all 3-mers: GGC
GCA
CAT

② form L/R
(k-1)-mers

de Bruijn Graph

Start with set of reads: {AAABBBA}

① take all k-mers (k=3)

$$\begin{tikzcd}[column sep=small] & AAA \arrow[dl] \arrow[dr] & & & AAB \arrow[dl] \arrow[dr] & & & ABB \arrow[dl] \arrow[dr] & & & BBB \arrow[dl] \arrow[dr] & & & BBA \arrow[dl] \arrow[dr] & \\ AA \arrow[rr, bend right] & & AA & AA \arrow[rr, bend right] & & AB & AB \arrow[rr, bend right] & & BB & BB \arrow[rr, bend right] & & BB & BB \arrow[rr, bend right] & & BA \end{tikzcd}$$

— edges in graph

③ form DBG

- nodes: (k-1)-mers
- edges: k-mers (all)

↙ unique

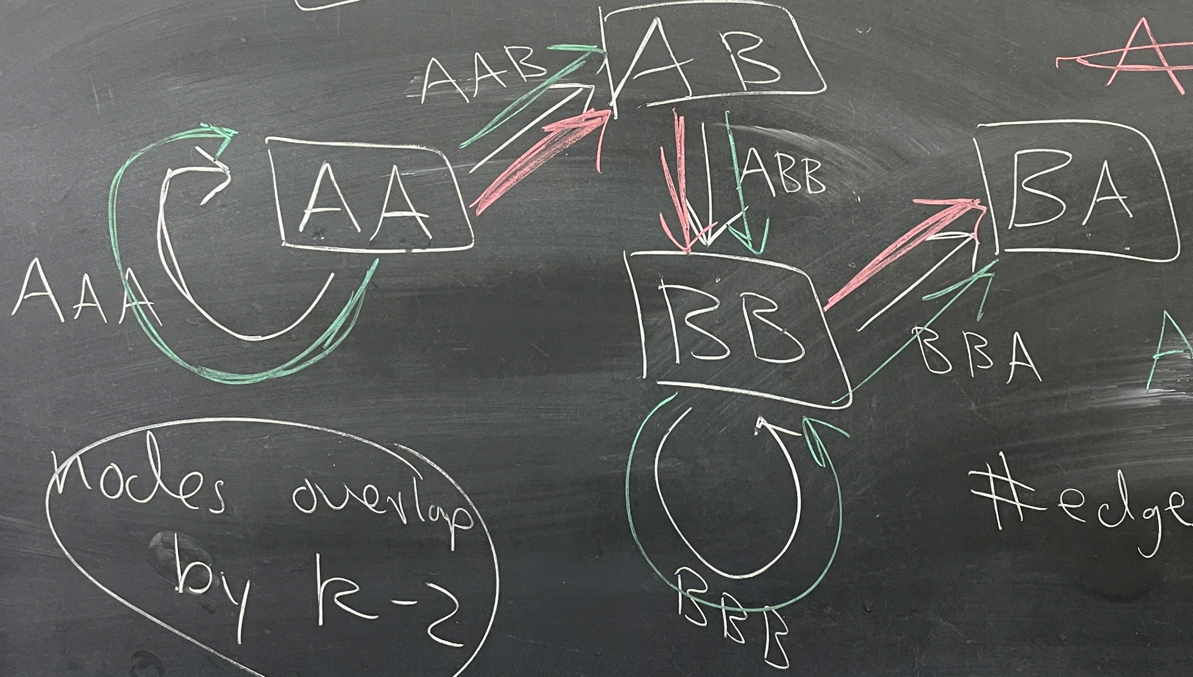
all possible

AA, AB, BA, BB

for us: use ones we see

$\boxed{\text{directed edge from } L \to R}$

~~AABBA~~



AAABBBA☆

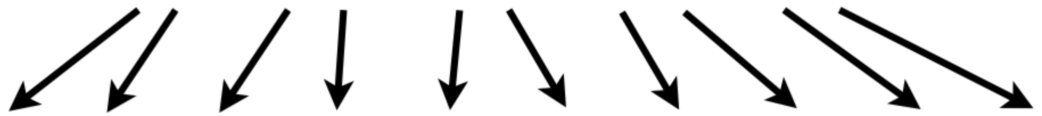
$\boxed{\text{nodes overlap by } k-2}$

$\#\text{edges} = \underbrace{\#\text{ k-mers}}_{\text{total}}$

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

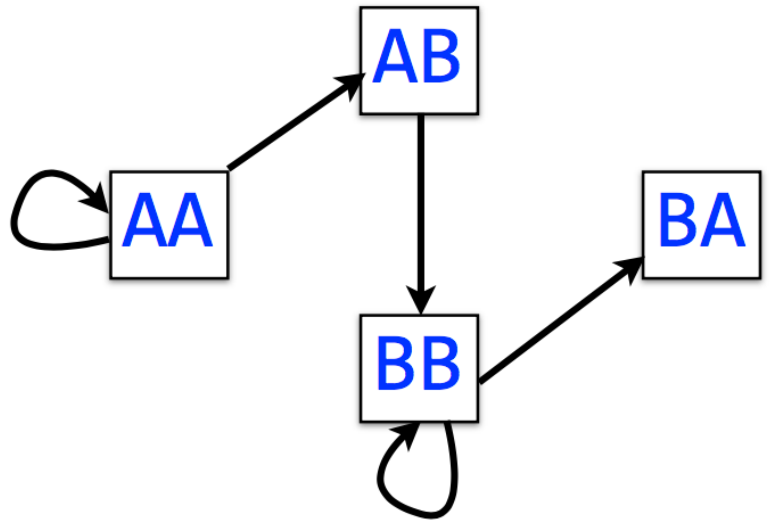
take all 3-mers: AAA, AAB, ABB, BBB, BBA



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:



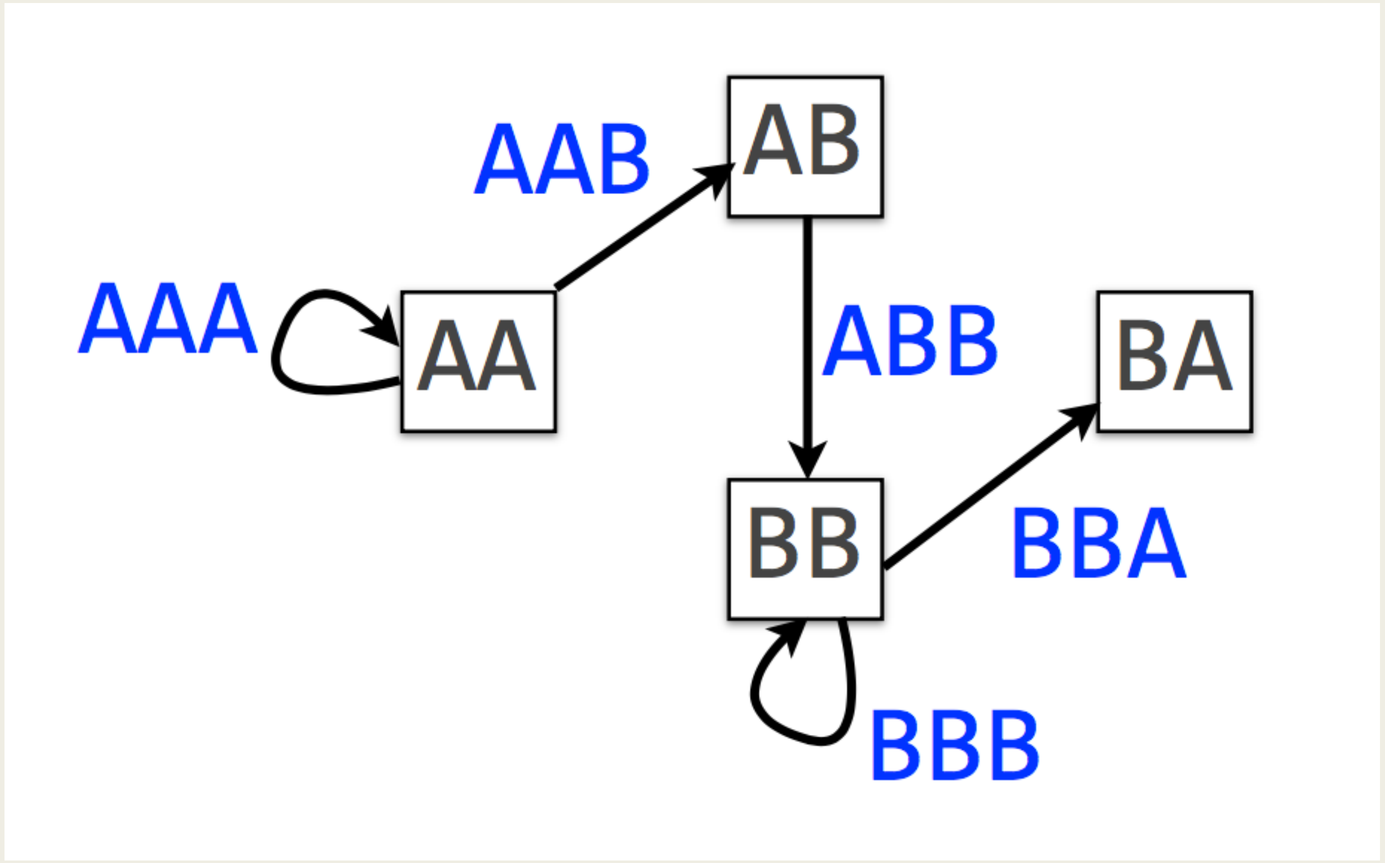
Each *edge* in this graph corresponds to a length-3 input string

Slide: adapted from Ben Langmead, John Hopkins

DBG:

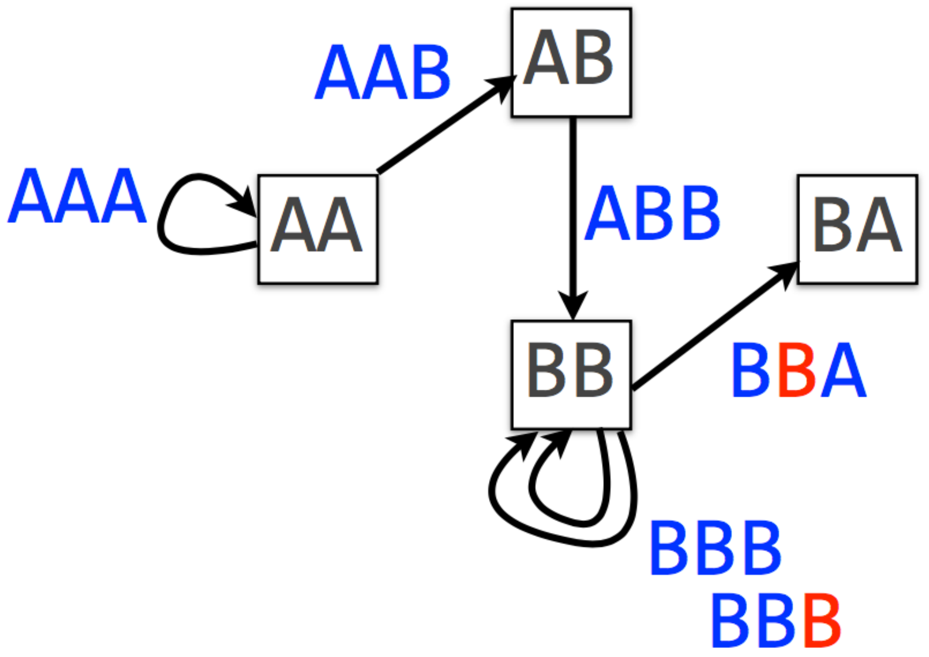
-Nodes: (k-1)-mers

-Edges: k-mers of the genome or reads



Slide: adapted from Ben Langmead, John Hopkins

DBGs can have multi-edges, making them multi-graphs



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

Slide: adapted from Ben Langmead, John Hopkins

Graph terminology

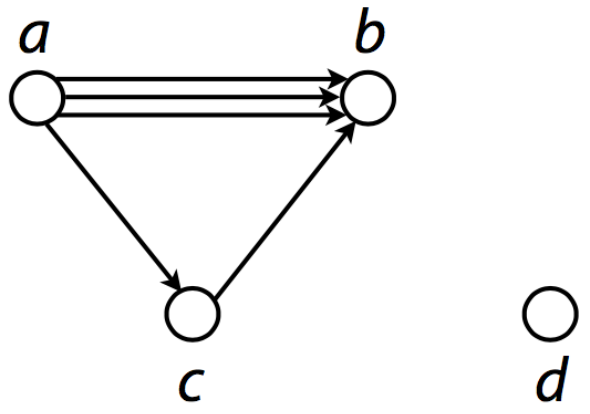
Directed **multigraph** *G*(*V*, *E*) consists of set of *vertices, V* and **multiset** of *directed edges, E*

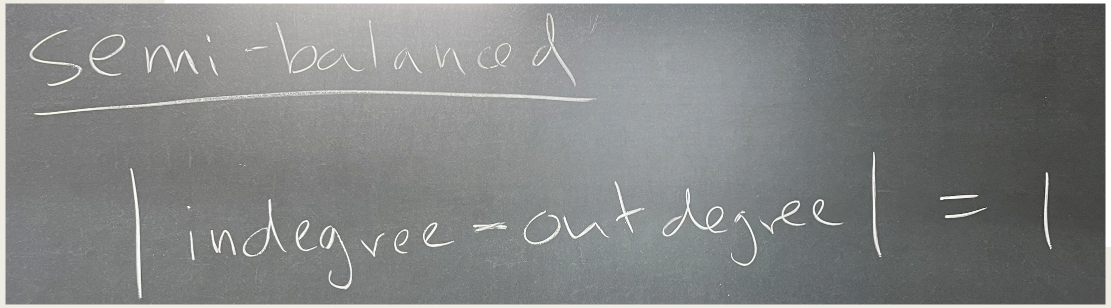
Otherwise, like a directed graph

Node's *indegree* = # incoming edges

Node's *outdegree* = # outgoing edges

De Bruijn graph is a directed multigraph




semi-balanced

$|\text{indegree} - \text{outdegree}| = 1$

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

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

Slide: adapted from Ben Langmead, John Hopkins

Graph terminology (cont.)

Node is *balanced* if indegree equals outdegree

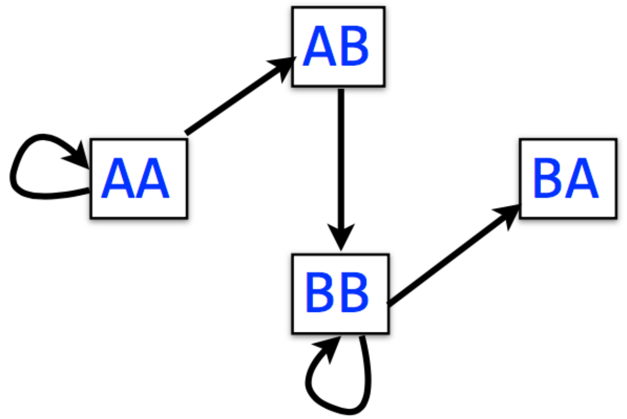
Node is *semi-balanced* if indegree differs from outdegree by 1

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.*

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



Jones and Pevzner section 8.8

Slide: adapted from Ben Langmead, John Hopkins

Graph terminology (cont.)

Node is *balanced* if indegree equals outdegree

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

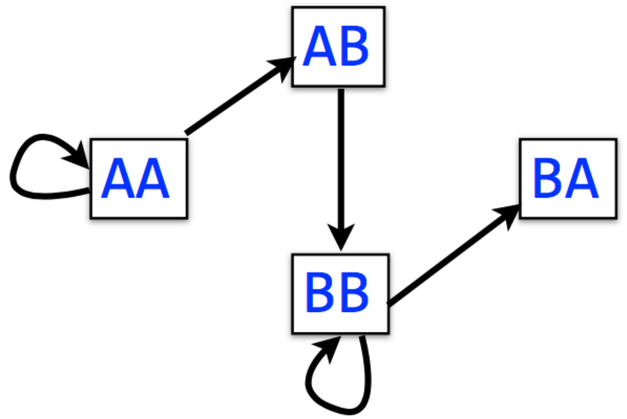
Graph is *connected* if each node can be reached by some other node

Eulerian walk visits each edge exactly once

Not all graphs have Eulerian walks. Graphs that do are *Eulerian*.

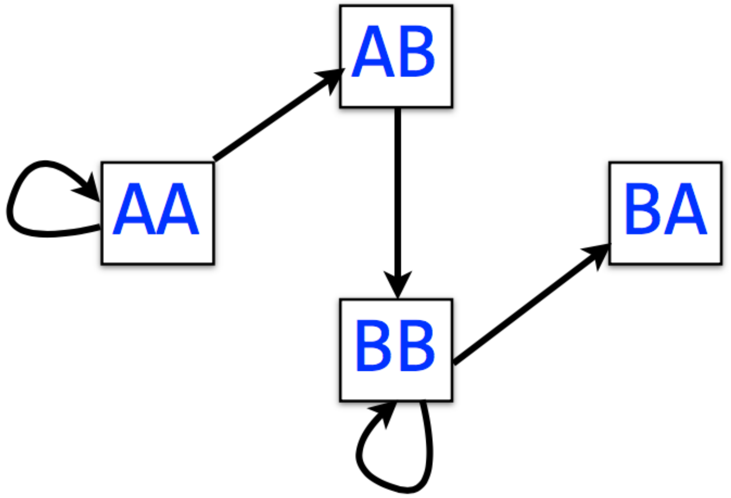
A directed, connected graph is Eulerian if and only if it has at most 2 semi-balanced nodes and all other nodes are balanced

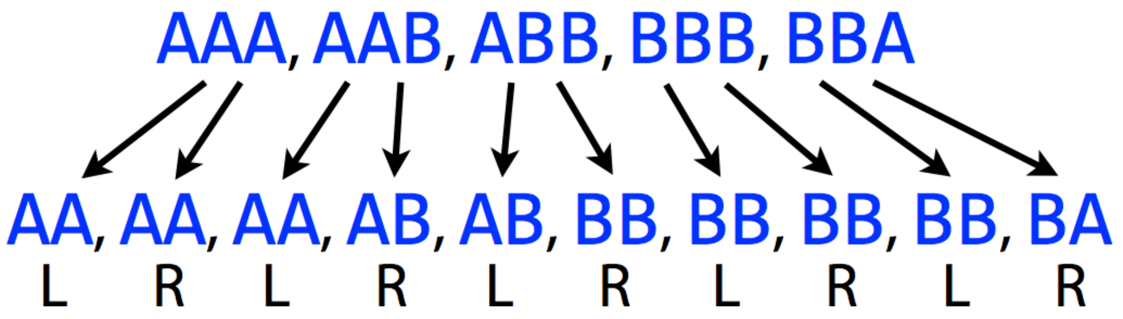
Jones and Pevzner section 8.8



Slide: adapted from Ben Langmead, John Hopkins

Back to our De Bruijn graph





Is it Eulerian? Yes

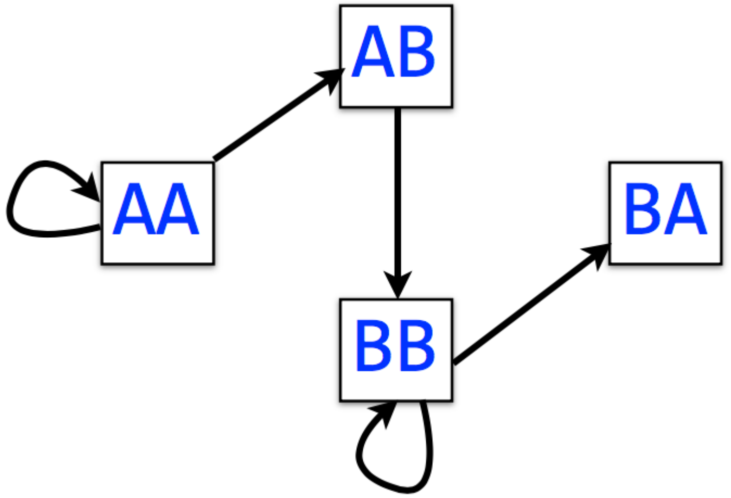
Argument 1: AA → AA → AB → BB → BB → BA

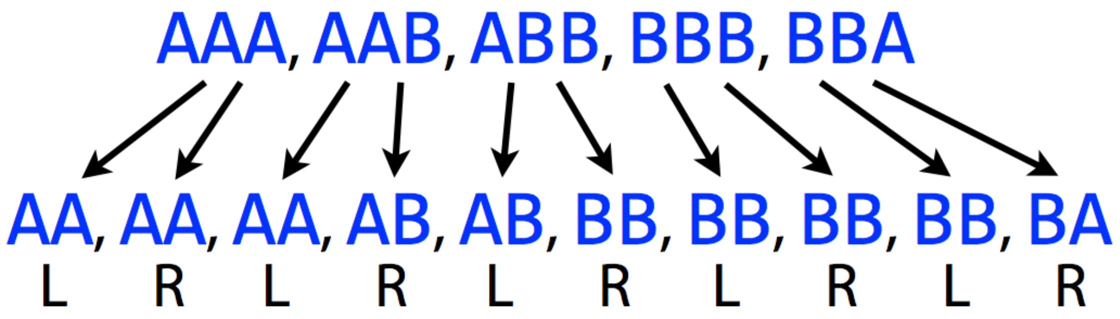
Argument 2: AA and BA are semi-balanced, AB and BB are balanced

Slide: adapted from Ben Langmead, John Hopkins

How to get the sequence from an Eulerian path?

Back to our De Bruijn graph





Is it Eulerian? Yes

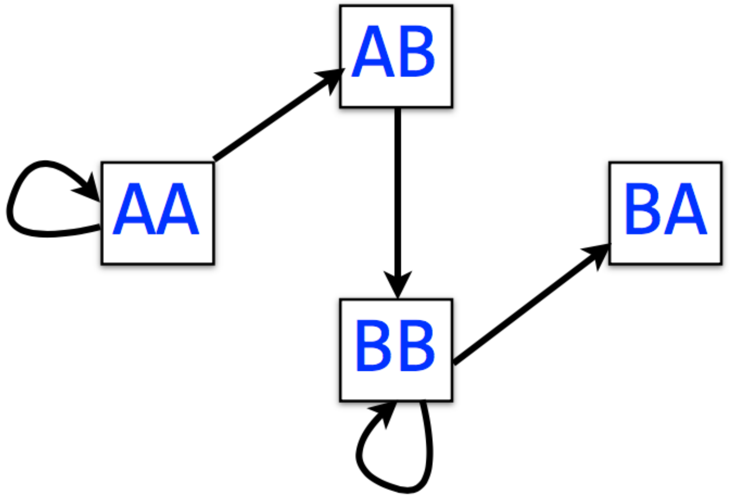
Argument 1: AA → AA → AB → BB → BB → BA

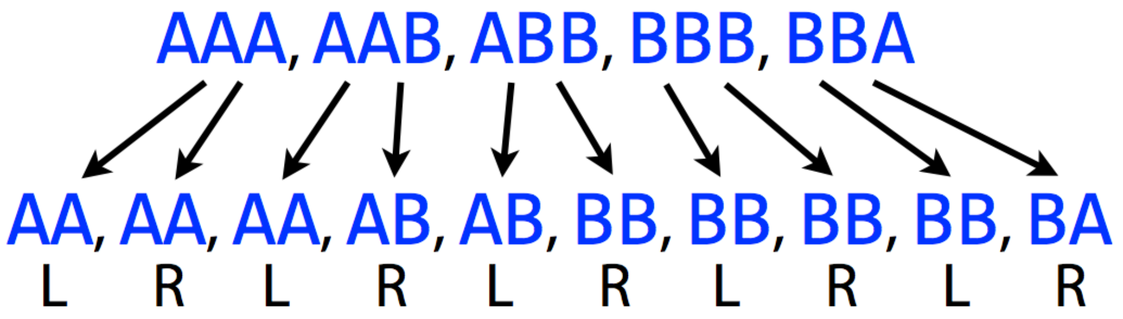
Argument 2: AA and BA are semi-balanced, AB and BB are balanced

Slide: adapted from Ben Langmead, John Hopkins

How to get the sequence from an Eulerian path?

Back to our De Bruijn graph





Start with the sequence in the first node. Follow the path, adding on one base each time.

AAABBBA

Is it Eulerian? Yes

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

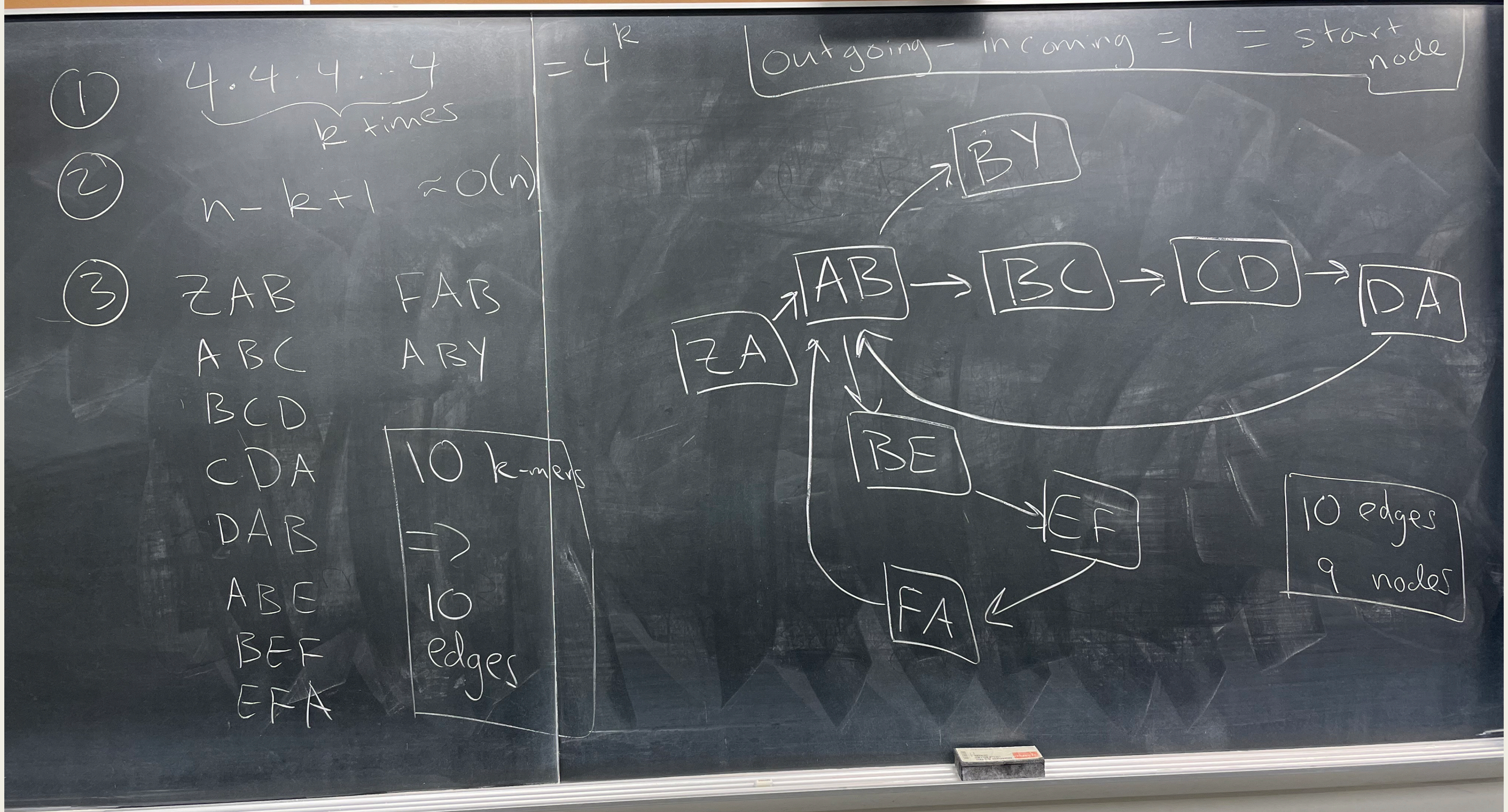
Argument 2: AA and BA are semi-balanced, AB and BB are balanced

Slide: adapted from Ben Langmead, John Hopkins

Handout 5: page 1

<u>Goals:</u>

1) Practice the mechanics of constructing a de Bruijn graph
2) See issues that that affect both OLC and DBG assembly
3) Think about how we would ask a computer to find an Eulerian path



① $\underbrace{4 \cdot 4 \cdot 4 \cdots 4}_{k \text{ times}} = 4^k$

② $n-k+1 \approx O(n)$

③

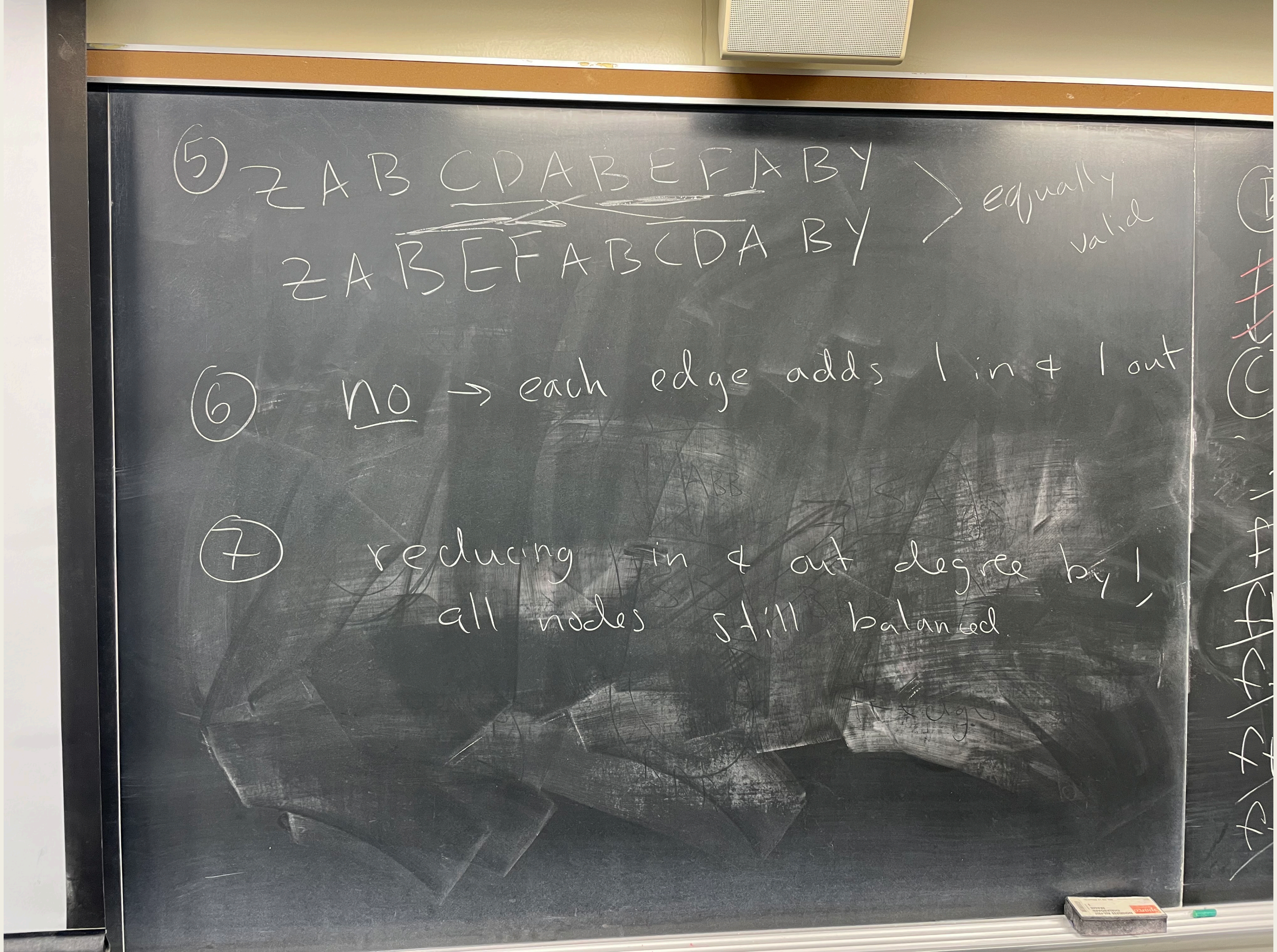
ZAB	FAB
ABC	ABY
BCD	
CDA	
DAB	
ABE	
BEF	
EFA	

$\boxed{\begin{array}{l}10 \text{ k-mers} \\ \Rightarrow \\ 10 \\ \text{edges}\end{array}}$

$\boxed{\text{outgoing} - \text{incoming} = 1 = \text{start node}}$

$$\begin{tikzcd}
 & & \boxed{BY} & & \\
\boxed{ZA} \arrow[r] & \boxed{AB} \arrow[ur] \arrow[r] \arrow[d] & \boxed{BC} \arrow[r] & \boxed{CD} \arrow[r] & \boxed{DA} \arrow[lll, bend left] \\
 & \boxed{BE} \arrow[r] & \boxed{EF} \arrow[dl] & & \\
 & \boxed{FA} \arrow[uu, bend left] & & &
\end{tikzcd}$$

$\boxed{\begin{array}{l}10 \text{ edges} \\ 9 \text{ nodes}\end{array}}$



5. ZABCDABEFABY
   ZABEFABCDABY } equally valid

6. no → each edge adds 1 in & 1 out

7. reducing in & out degree by 1, all nodes still balanced.

Fleury’s Algorithm

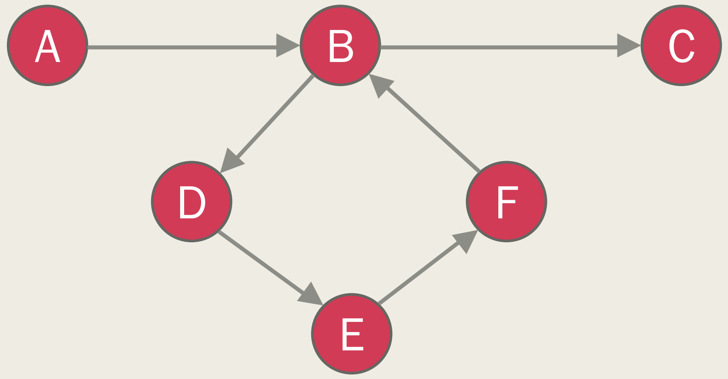
- Start with vertex ***u*** where (outdegree – indegree) = 1
- Until there are no more edges:

    Choose ***e = (u,v)*** where removing ***e*** will not disconnect the graph (if possible)
    Traverse ***e***, delete ***e***, and proceed with ***u = v***

Fleury's Algorithm

- Start with vertex ***u*** where (outdegree – indegree) = 1
- Until there are no more edges:

  Choose ***e = (u,v)*** where removing ***e*** will not disconnect the graph (if possible)
  Traverse ***e***, delete ***e***, and proceed with ***u = v***



Avoids going from:
A -> B -> C
And then getting stuck

Recursive algorithm

```
find_tour(u, tour):
        for each edge e = (u,v):
                remove e
                find_tour(v, tour)
        push u onto tour
```

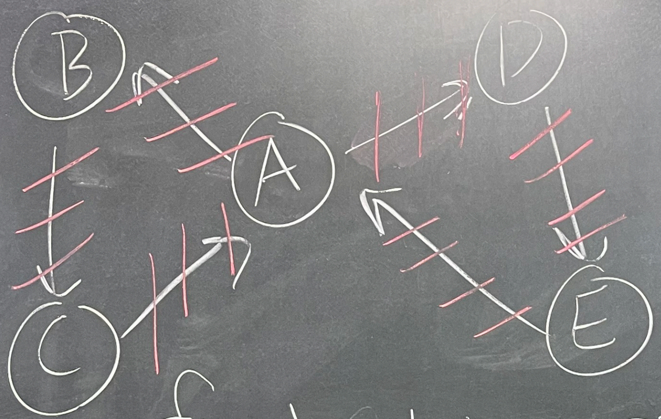
<u>To use in practice</u>:

- Start with tour as an empty stack
- Start at any node u (or start at a node with (outdegree – indegree) = 1 for a path)
- After completion, pop elements off tour to find the correct order

y
lid

l out

y !



function stack

~~ft(A)~~
~~ft(C)~~
~~ft(B)~~
~~ft(D)~~
~~ft(A)~~
~~ft(E)~~
~~ft(D)~~

tour stack

D
E
A
B
C
A
D

```
$\overbrace{\text{find\_tour}}^{ft}$(u):
    for edge e=(u,v):
        remove e
        find_tour(v)
    push u onto stack
```

Final seq.

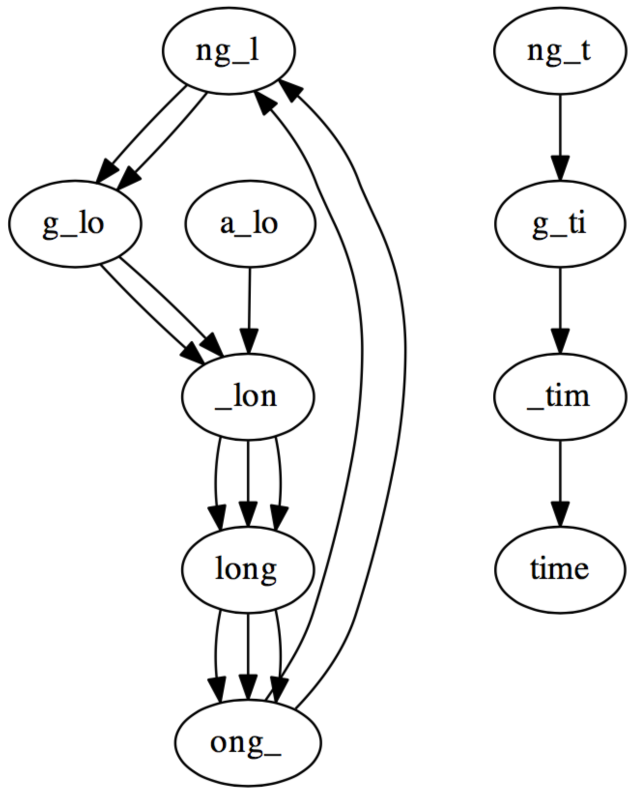
DEABCAD

Handout 5: page 2

Issues with DBGs

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

Slide: adapted from Ben Langmead, John Hopkins

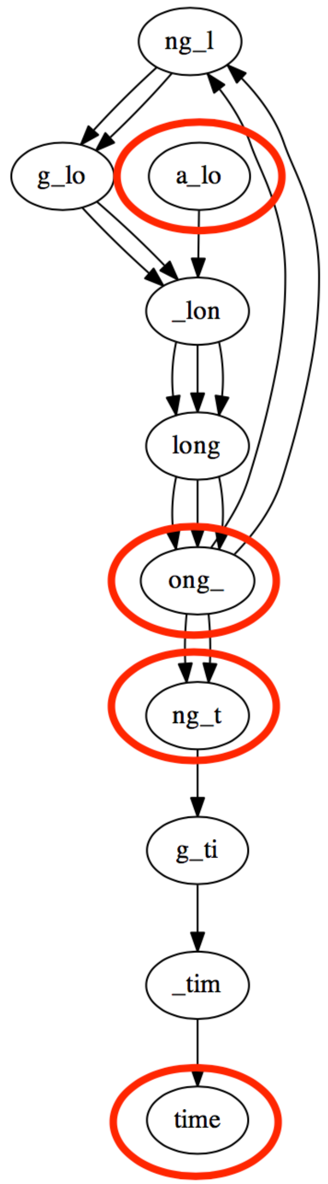
Issues with DBGs

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



Slide: adapted from Ben Langmead, John Hopkins

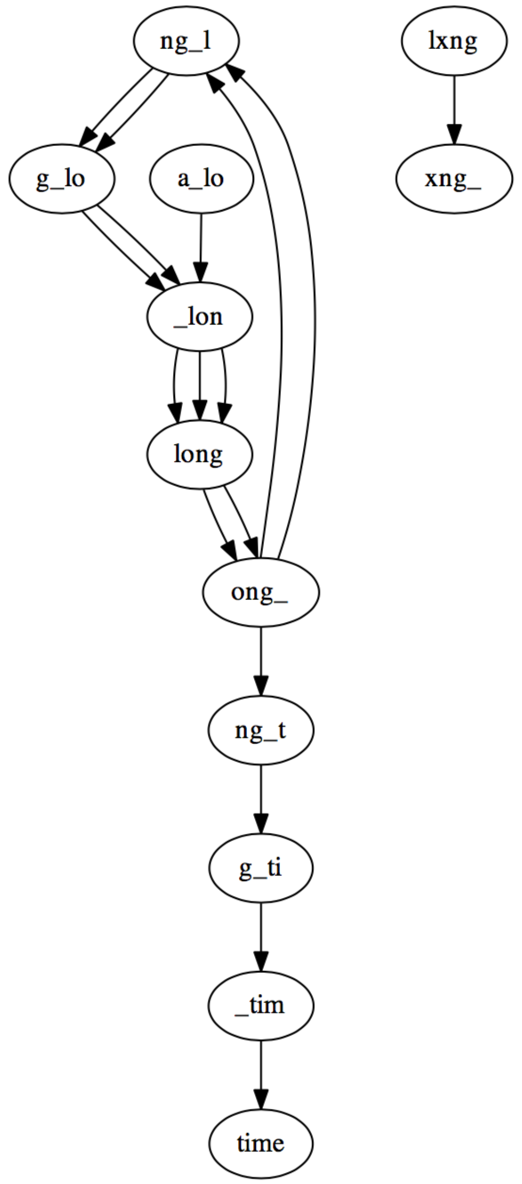
Issues with DBGs

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



Slide: adapted from Ben Langmead, John Hopkins