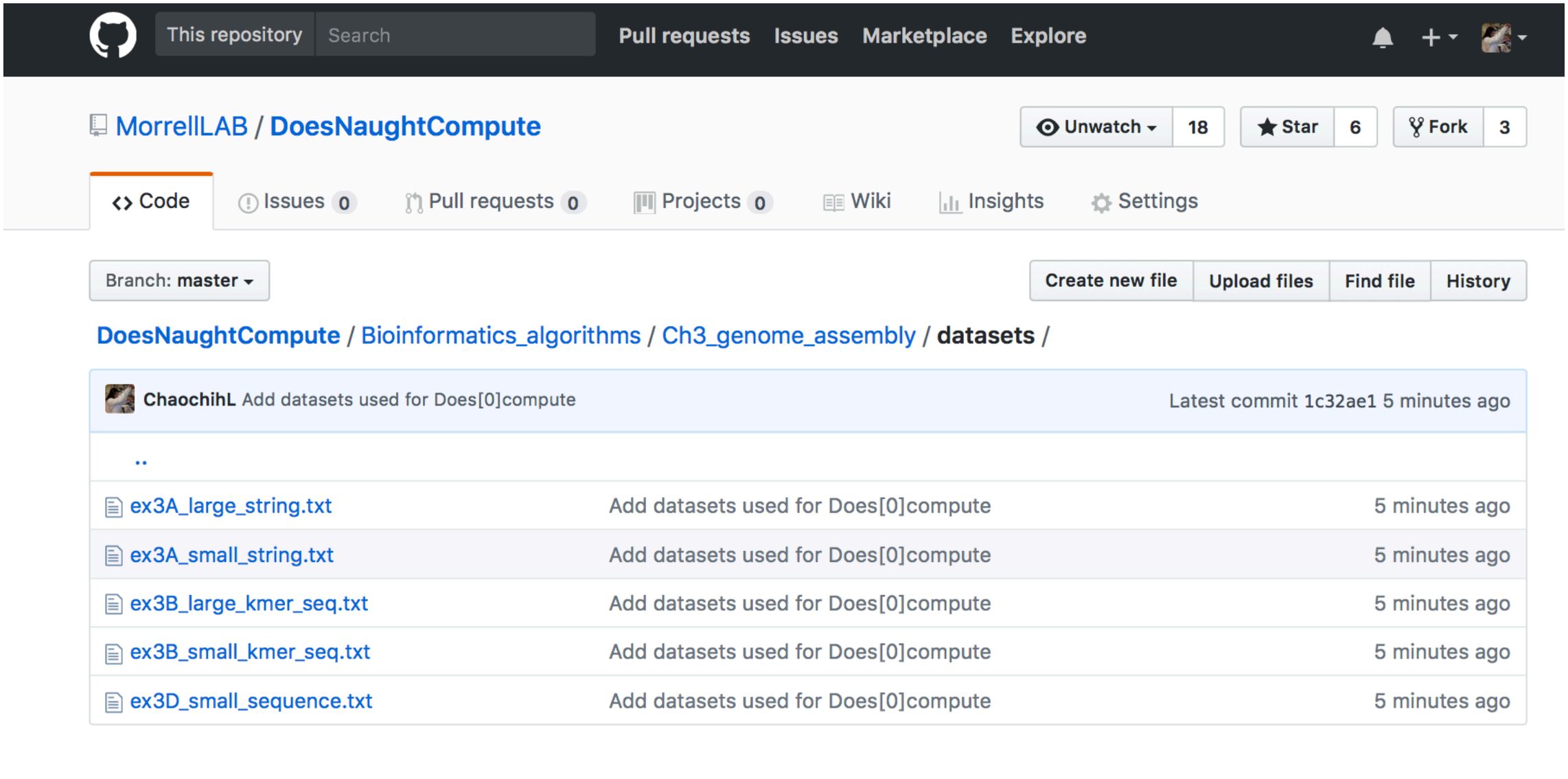
Bioinformatics Algorithms Chapter 3: How do we assemble genomes?

Part I Pg. 116-133

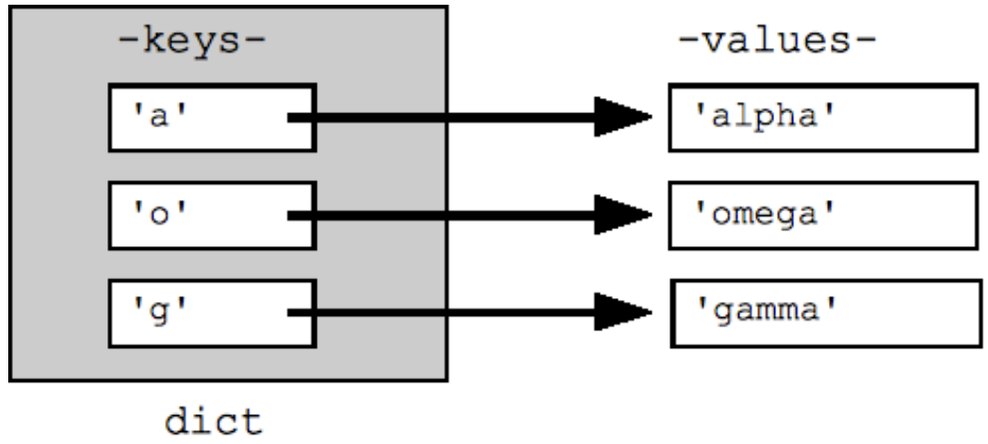
Does[0]compute? Discussion
May 17, 2018

All datasets used for coding challenges are on GitHub at:

https://github.com/MorrellLAB/DoesNaughtCompute/tree/master/Bioinformatics_algorithms/Ch3_genome_assembly/datasets



Pseudocode



Ind1	Ind2	Chr	SNP_Win	PhysPos_Start	PhysPos_End	Int_Phys_Size
CIho10035		WBDC173	chr1	275-352	522838398	558414475	35576077
CIho10420		WBDC173	chr1	275-352	522838398	558414475	35576077
CIho13397		WBDC016	chr1	200-300	466036064	536887344	70851280
CIho13397		WBDC016	chr1	225-325	487105715	547250830	60145115

```
d = {}
with open(input_file, 'r') as f:
    for line in f:
        if line.startswith('Ind1'):
            continue
        else:
            tmp = line.strip().split('\t')
            key_name = tmp[0] + '_' + tmp[1]  # ID1 and ID2 columns
            rest = [tmp[2], tmp[3], tmp[4], tmp[5], tmp[6]]
            if key_name in d.keys():
                d[key_name].append(rest)
            else:
                d[key_name] = [rest]
return d
```

```
d = empty dictionary
open file as read only
    for every line in file
        if line starts with ‘Ind1’
            skip
        else
            tmp = strip line by tab delimiter
            key_name = ID1 + ‘_’ + ID2
            rest = [list of remaining columns]
            if key in existing_dictionary
                append key, value pair to d
            else
                add new dictionary key, value pair to d
return d
```

Pseudocode

Example from book on page 136:

DeBruijn(*Patterns*)

represent every *k*-mer in *Patterns* as an isolated edge between its prefix and suffix
glue all nodes with identical labels, yielding the graph DeBruijn(*Patterns*)
return DeBruijn(*Patterns*)

Pair Programming

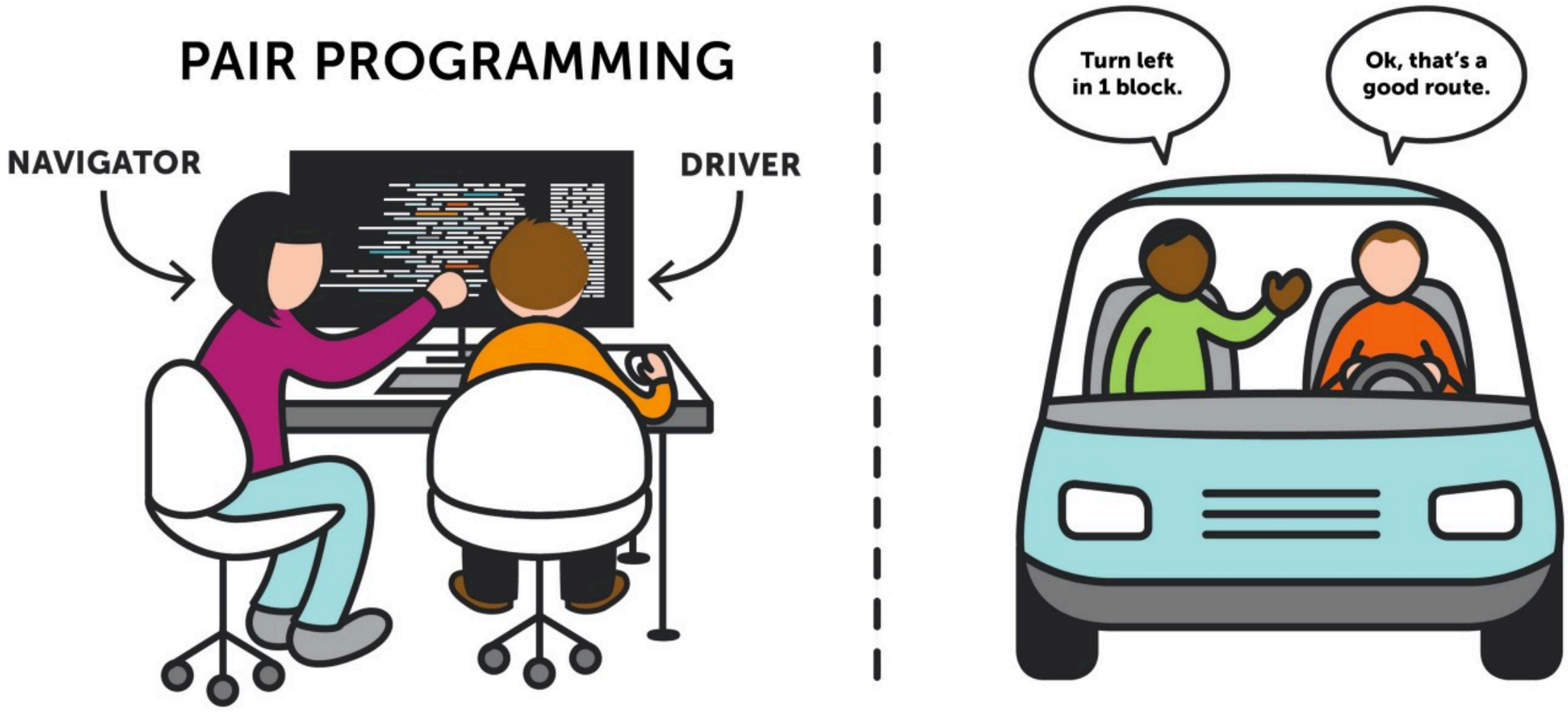


Image source: https://medium.com/@tomspencer_uk/pair-programming-and-problem-solving-4531ef3bf171

Chapter 3 Part I

pages 116-133

May 17, 2018

Coding Exercise 3A: Generate the k-mer composition of a string

Pair programming:
Take 5 minutes to write pseudocode to solve this problem. Then take 5-10 minutes convert your pseudocode into a simple script (i.e. implementing). Choose your favorite programming language.

Given a string Text, its ***k*-mer composition** $Composition_k(Text)$ is the collection of all *k*-mer substrings of *Text* (including repeated *k*-mers). For example,

$Composition_3$(TATGGGGTGC) = {ATG, GGG, GGG, GGT, GTG, TAT, TGC, TGG}

Note that we have listed *k*-mers in **lexicographic order** (i.e., how they would appear in a dictionary) rather than in the order of their appearance in TATGGGGTGC. We have done this because the correct ordering of the reads is unknown when they are generated.

String Composition Problem

Generate the k-mer composition of a string.

Given: An integer *k* and a string *Text*.

Return: $Composition_k(Text)$ (the *k*-mers can be provided in any order).

Input:

```
5
CAATCCAAC
```

Want to output:

```
AATCC
ATCCA
CAATC
CCAAC
TCCAA
```

Thinking Exercise Break: Design a strategy for assembling the Triazzle puzzle

Take 5 minutes to discuss in pairs the strategy (steps/rules you need to follow) to assemble the Triazzle puzzle. (Note: no coding involved for this exercise)

Coding Exercise 3B:
Reconstruct a string from its genome path

Pair programming:
Take 5 minutes to write pseudocode to solve this problem. Then take 5-10 minutes convert your pseudocode into a simple script (i.e. implementing). Choose your favorite programming language.

String Spelled by a Genome Path Problem

Find the string spelled by a genome path.

Given: A sequence of *k*-mers $Pattern_1, \ldots, Pattern_n$ such that the last $k - 1$ symbols of $Pattern_i$ are equal to the first $k - 1$ symbols of $Pattern_{i+1}$ for i from 1 to n-1.

Return: A string *Text* of length $k+n-1$ where the i-th k-mer in *Text* is equal to $Pattern_i$ for all i.

Input:

```
ACCGA
CCGAA
CGAAG
GAAGC
AAGCT
```

Want to output: `ACCGAAGCT`

Representing graphs in programs

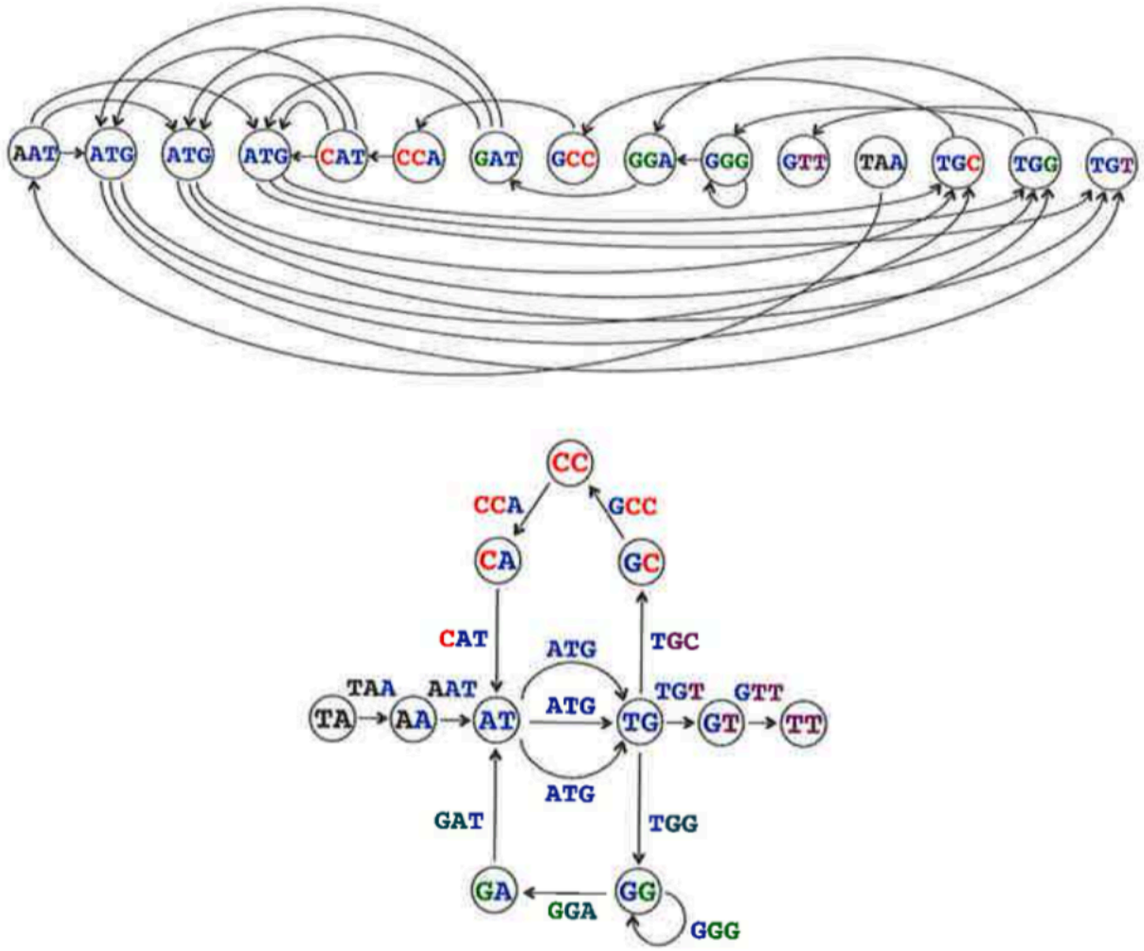


FIGURE 3.17 The overlap graph (top) and de Bruijn graph (bottom) for the same collection of 3-mers.

Directed vs Undirected Graphs

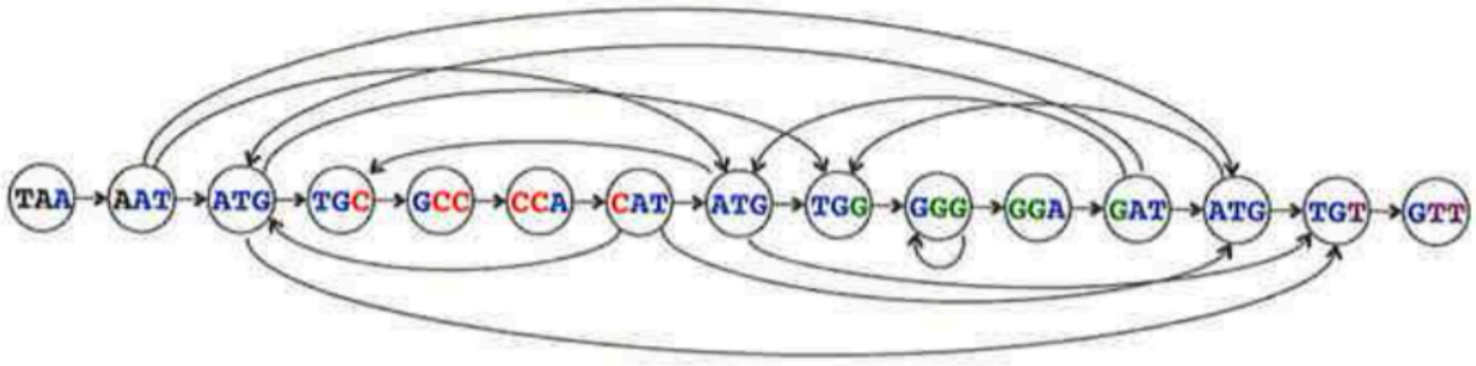
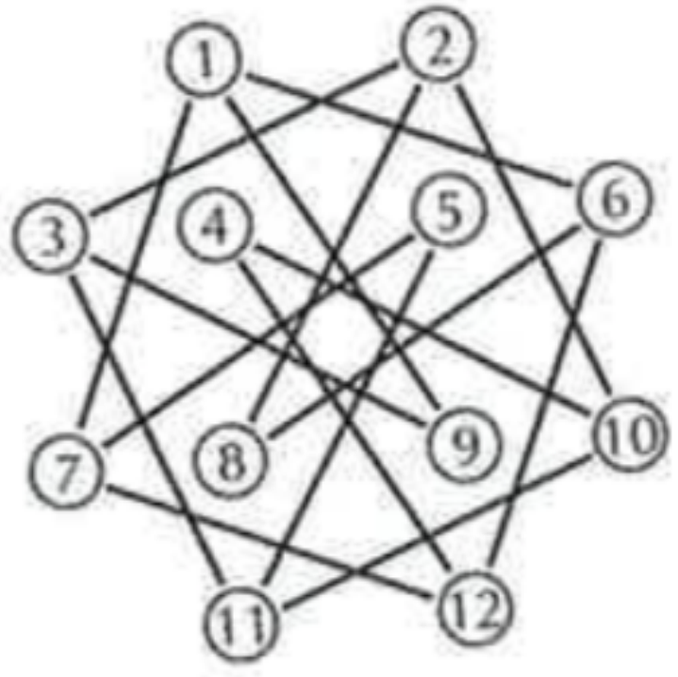


FIGURE 3.7 The graph showing all connections between nodes representing the 3-mer composition of **TAATGCCATGGGATGTT**. This graph has fifteen nodes and 25 edges. Note that the genome can still be spelled out by walking along the horizontal edges from **TAA** to **GTT**.



Adjacency Lists

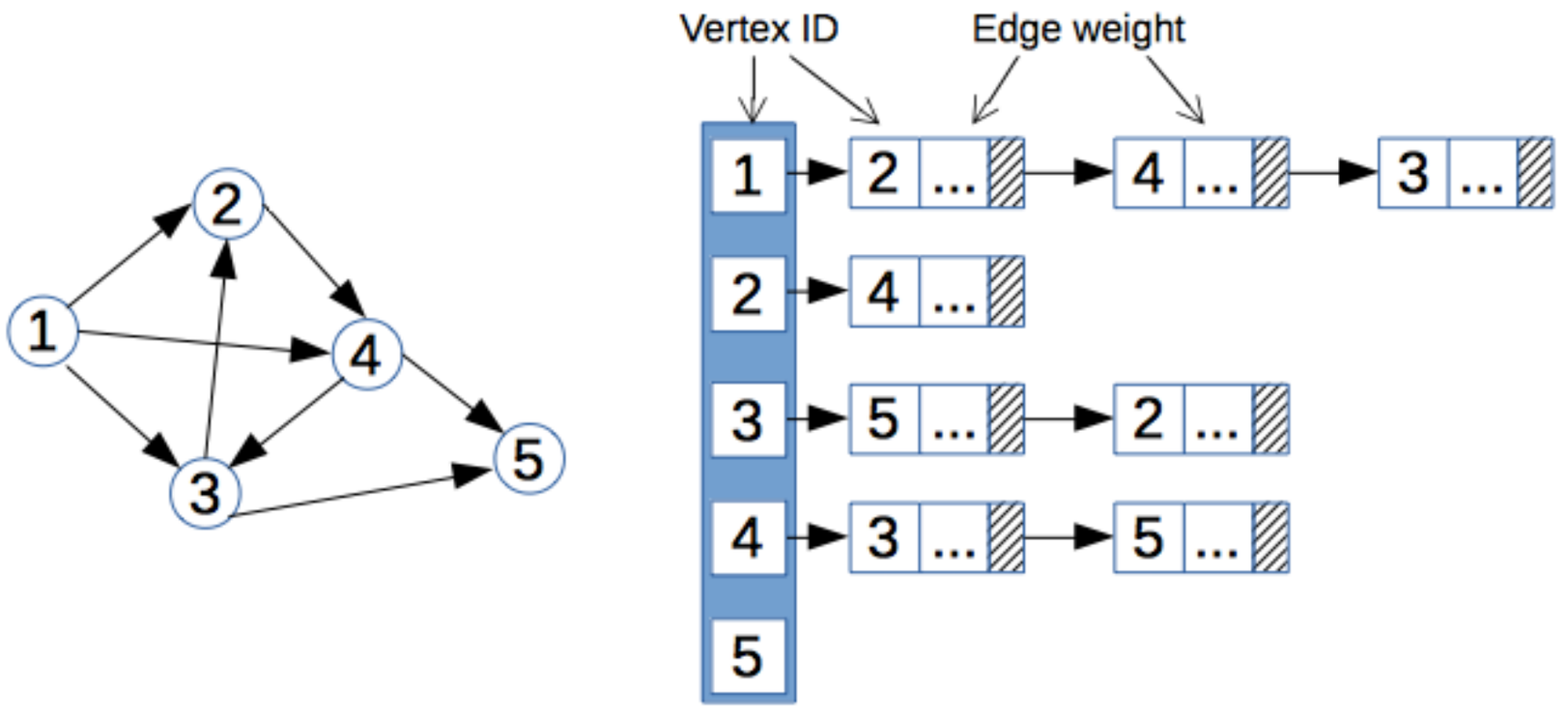
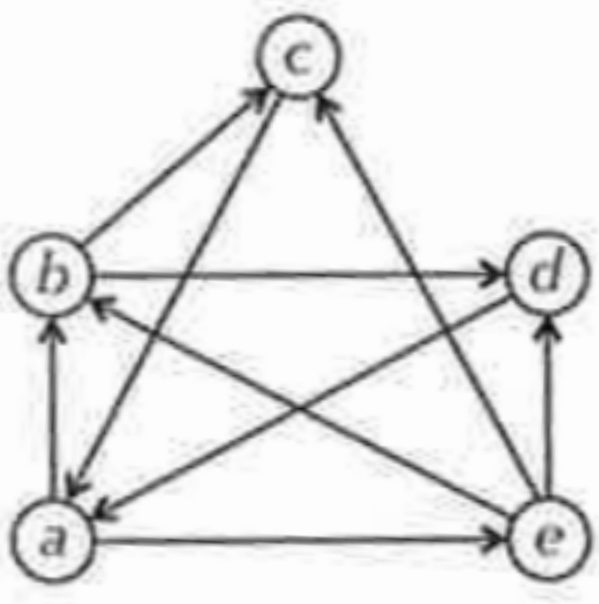


Image from: http://lagodiuk.github.io/images/adj_lists/img_2.png

Adjaceny Matrix

Graph



Adjacency Matrix

	a	*b*	*c*	*d*	*e*
a	0	1	0	0	1
b	0	0	1	1	0
c	1	0	0	0	0
d	1	0	0	0	0
e	0	1	1	1	0

De Bruijn Graphs

Most Next Generation Sequencing assembly tools use de Bruijn graphs.

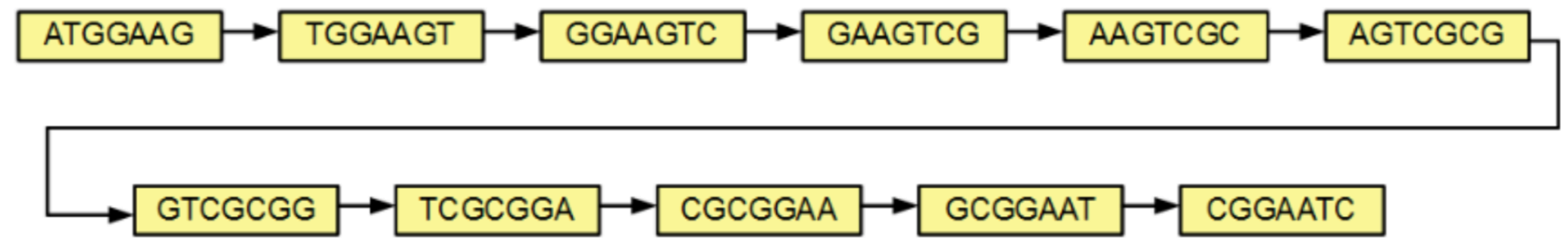
Ex: ABySS, Velvet, SOAP de novo, Newbler, Allpaths, etc.

sequence ATGGAAGTCGCGGAATC

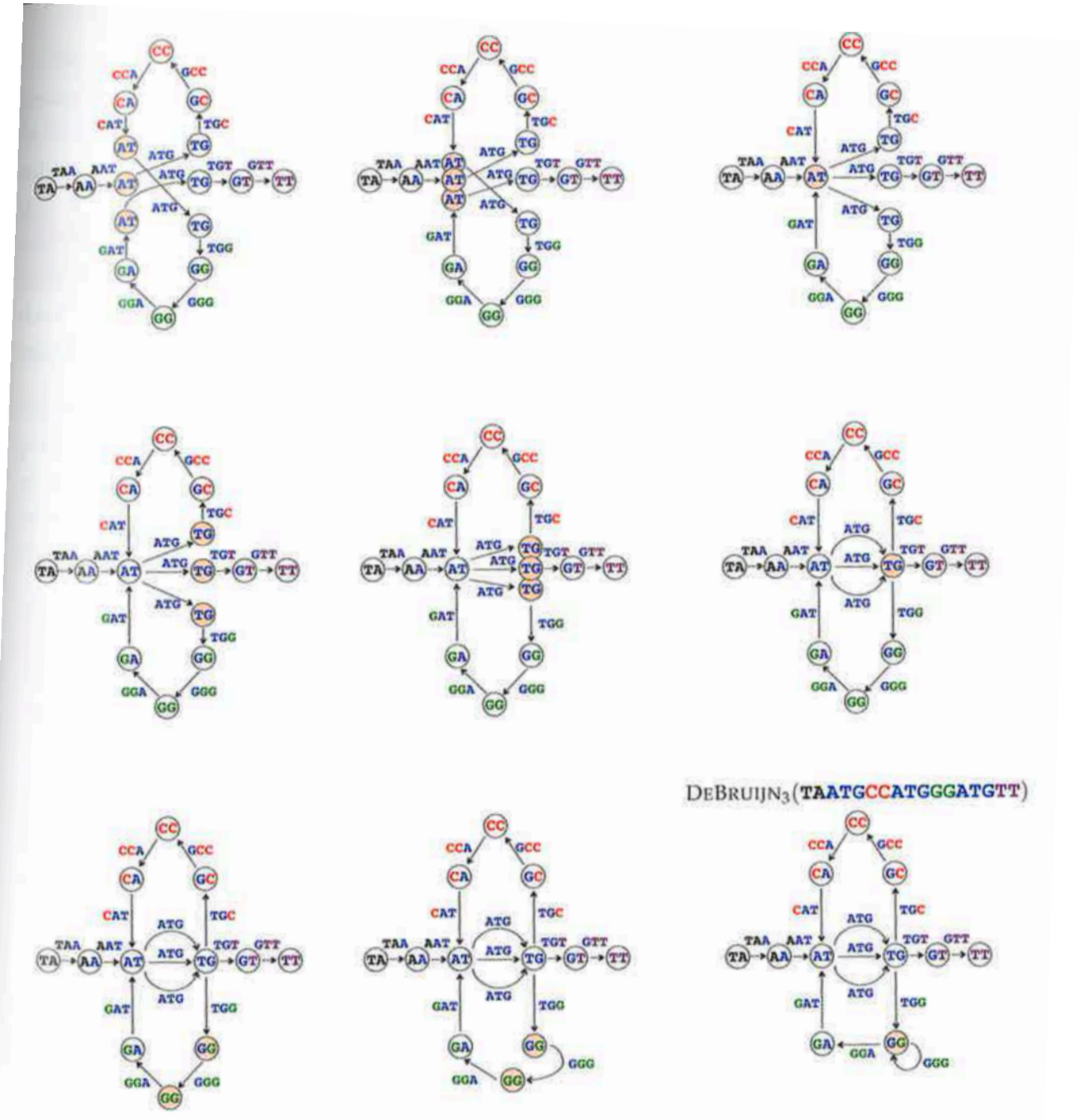
7mers

ATGGAAG
TGGAAGT
GGAAGTC
GAAGTCG
AAGTCGC
AGTCGCG
GTCGCGG
TCGCGGA
CGCGGAA
GCGGAAT
CGGAATC

de Bruijn graph



De Bruijn Graphs
Figure 3.13 on page 133



Coding Exercise 3D:
Construct the De Bruijn Graph of a string

Pair programming:
Take 10 minutes to write pseudocode to solve this problem.

Given a genome *Text*, $PathGraph_k(Text)$ is the path consisting of $|Text| - k + 1$ edges, where the i-th edge of this path is labeled by the i-th k-mer in *Text* and the i-th node of the path is labeled by the i-th $(k - 1)$-mer in *Text*. The **de Bruijn graph** $DeBruijn_k(Text)$ is formed by gluing identically labeled nodes in $PathGraph_k(Text)$.

De Bruijn Graph from a String Problem

Construct the de Bruijn graph of a string.

Given: An integer k and a string *Text*.

Return: $DeBruijn_k(Text)$, in the form of an adjacency list.

Input: `AAGATTCTCTAC`

Output wanted:

```
AAG -> AGA
AGA -> GAT
ATT -> TTC
CTA -> TAC
CTC -> TCT
GAT -> ATT
TCT -> CTA,CTC
TTC -> TCT
```