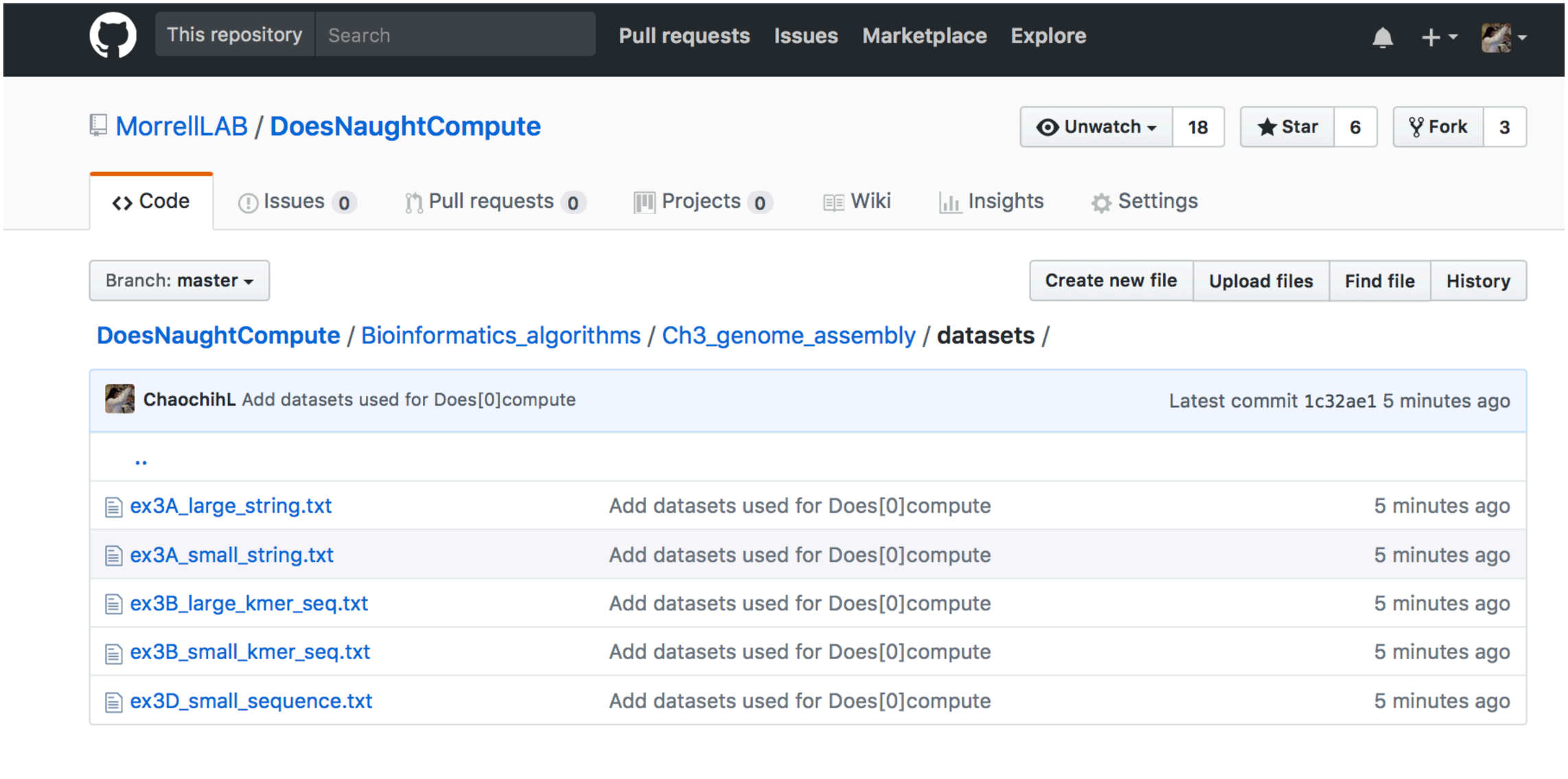
Bioinformatics Algorithms Chapter 3: How do we assemble genomes?

Part II Pg. 134-164

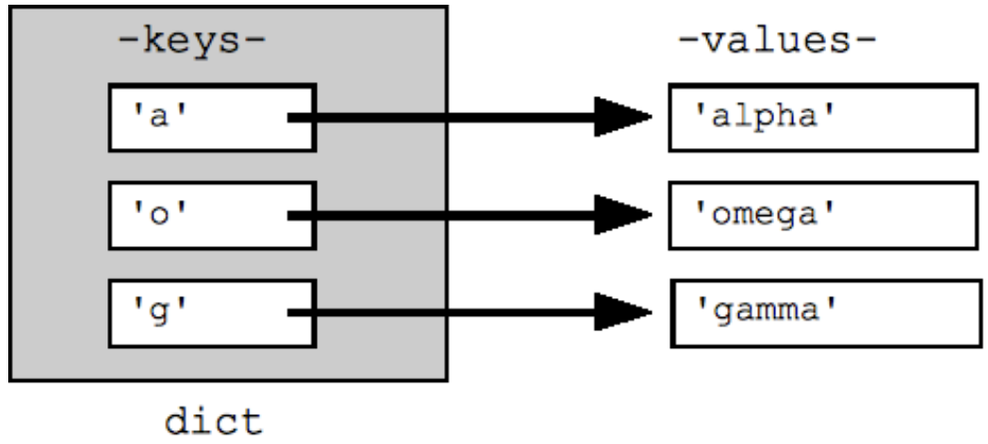
Does[0]compute? Discussion
May 24, 2018

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

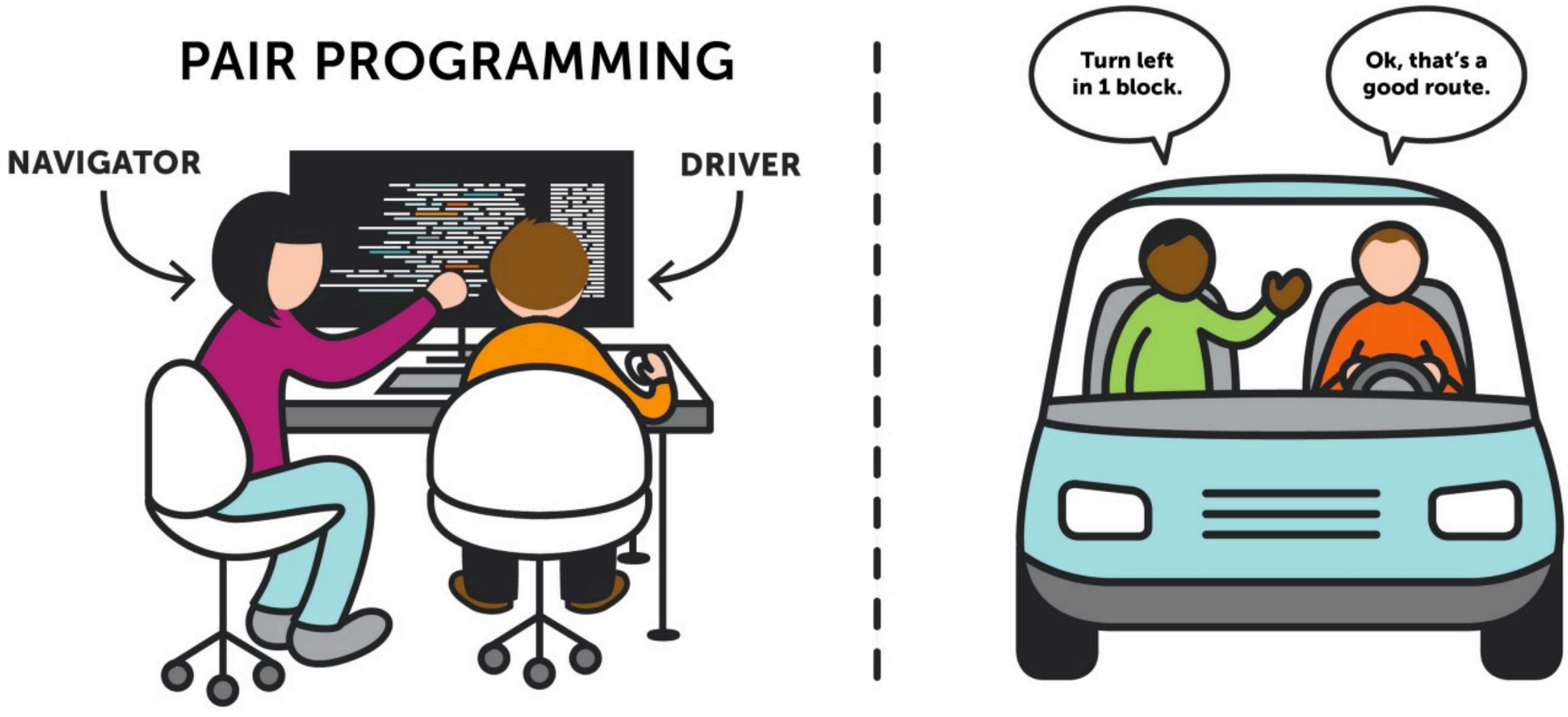


Image source: https://medium.com/@tomspencer_uk/pair-programming-and-problem-solving-4531ef3bf171

The Exploding Newspaper Problem

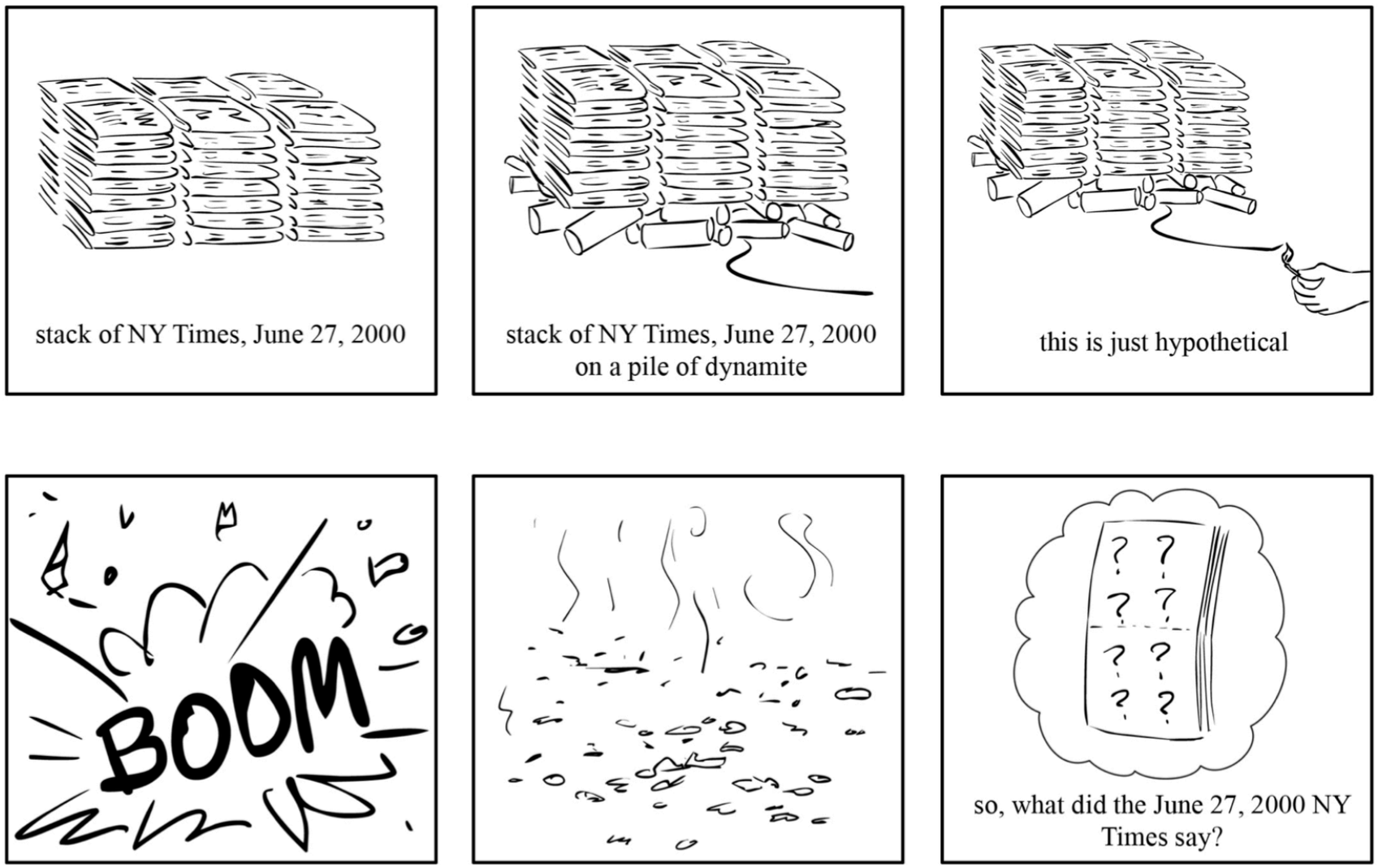
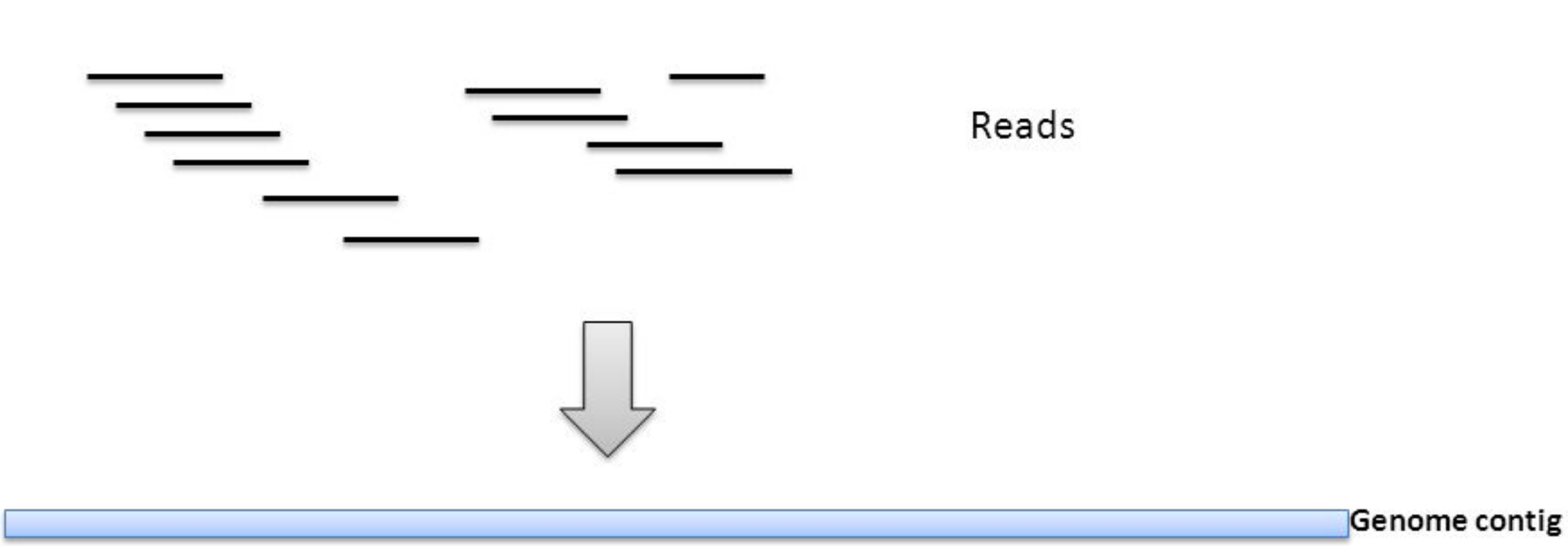


Figure 1: The exploding newspapers.

Source: http://cseweb.ucsd.edu/~ppevzner/B4B/excerpts/Compeau-Pevzner.pdf

Genome Assembly Problem

Denovo genome assembly



Source: http://slideplayer.com/slide/7981734/25/images/3/Denovo+genome+assembly.jpg

Quick Recap

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)

Directed vs Undirected Graphs

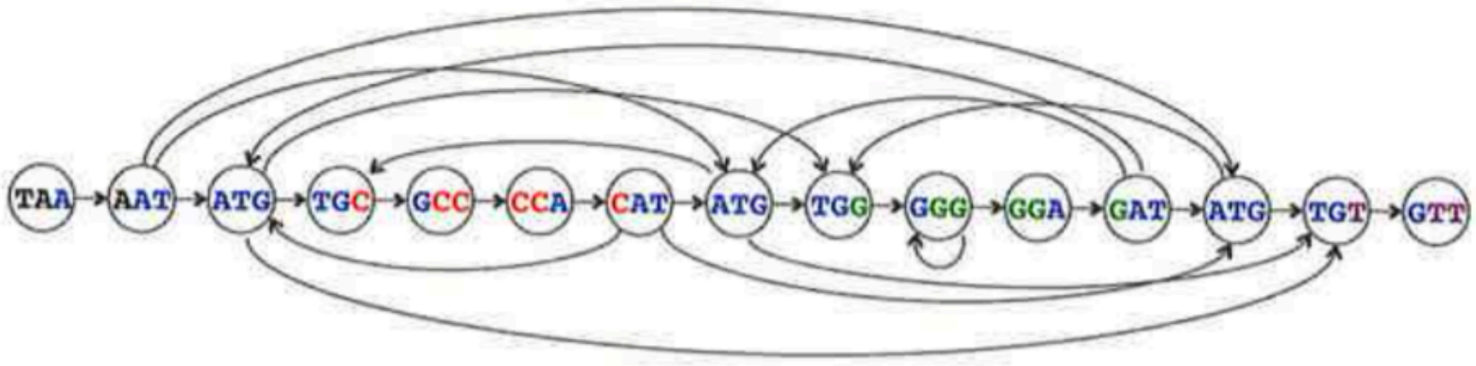
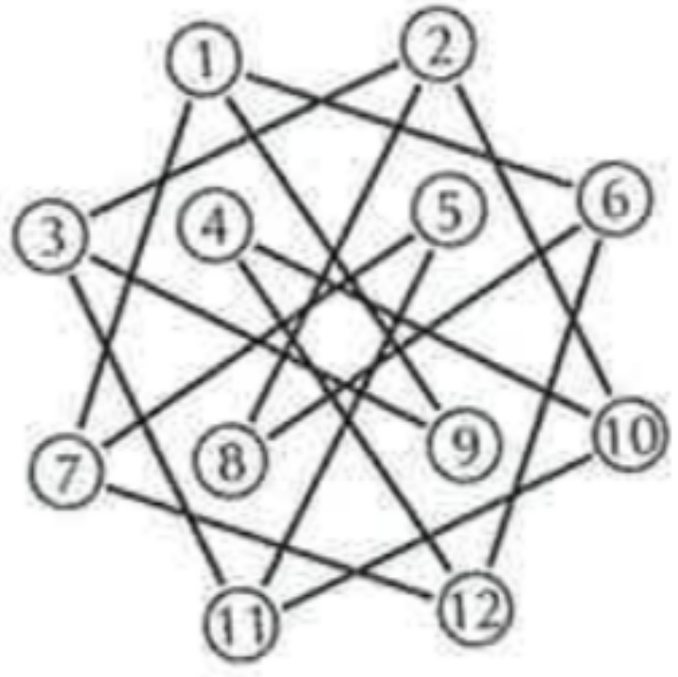


FIGURE 3.7 The graph showing all connections between nodes representing the 3-mer composition of **TAATGCCATGGGATGTT**. This graph has fifteen nodes and 25 edges. Note that the genome can still be spelled out by walking along the horizontal edges from **TAA** to **GTT**.



Adjacency Lists

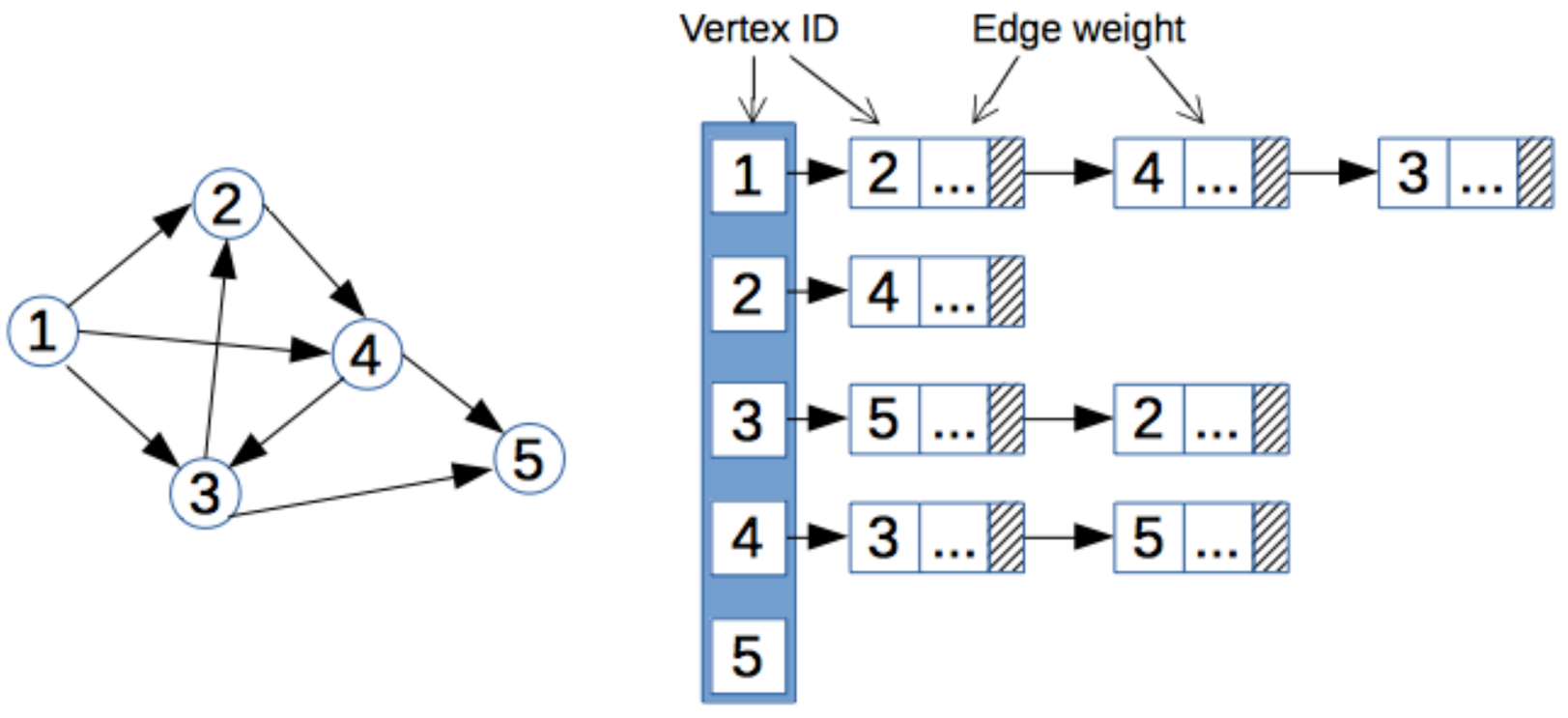
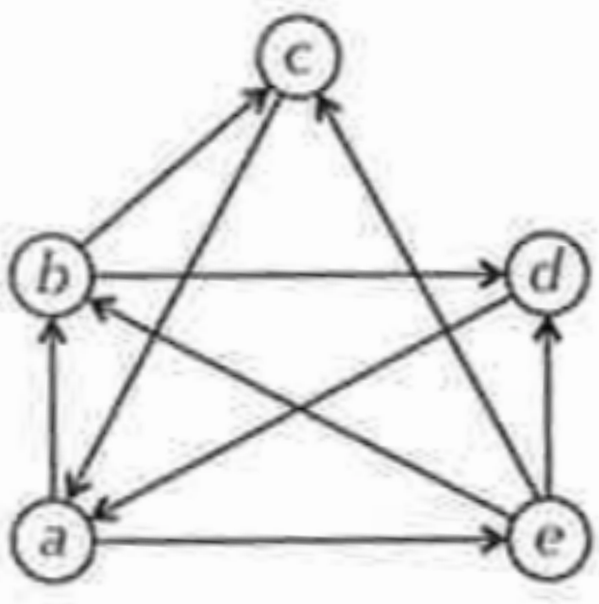


Image from: http://lagodiuk.github.io/images/adj_lists/img_2.png

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.

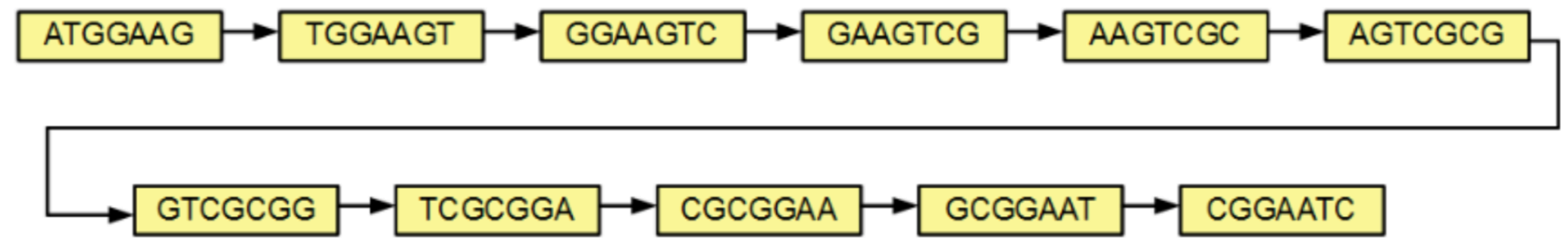
Ex: ABySS, Velvet, SOAP de novo, Newbler, Allpaths, etc.

sequence ATGGAAGTCGCGGAATC

7mers

ATGGAAG
TGGAAGT
GGAAGTC
GAAGTCG
AAGTCGC
AGTCGCG
GTCGCGG
TCGCGGA
CGCGGAA
GCGGAAT
CGGAATC

de Bruijn graph



Chapter 3 Part II

pages 134-164

Eulerian paths

Finds a path that visits every **edge** exactly once.

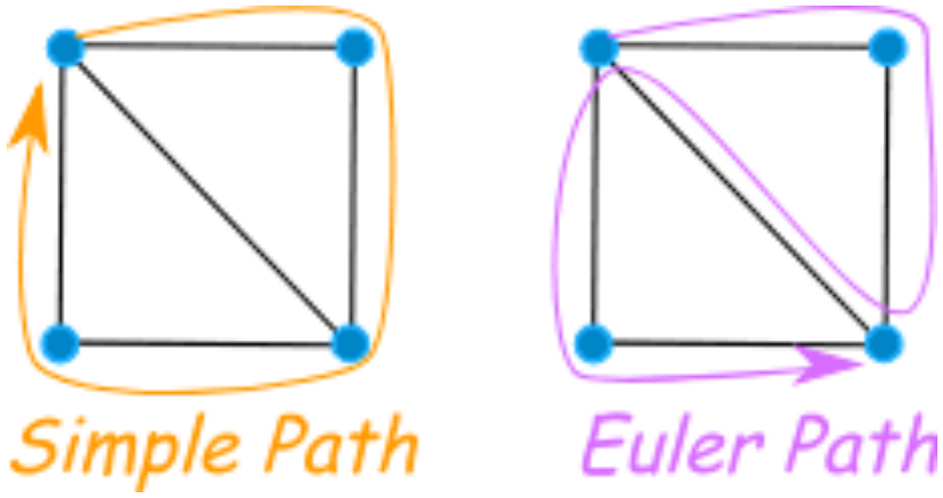
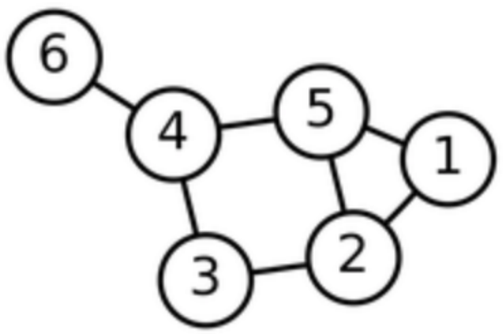


Image source: https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRUUzM925_9RKw3huJ5AmGOm9735-POSvTt2yZbReDSIhE_c0tZBA

Review: Representing Adjaceny Lists

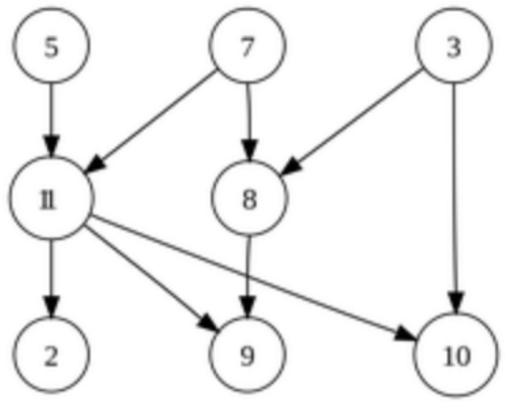
Undirected Graph



- Rows can be in any order
- Elements of any row can be transposed

```
1 2
1 5
2 3
2 5
3 4
4 5
4 6
```

Directed Graph



- Edges can be given in any order
- **Starting node of directed edge** MUST be provided in column 1

```
3 8
3 10
5 11
7 8
7 11
8 9
11 2
11 9
11 10
```

Coding Exercise 3F:

The following algorithm constructs an Eulerian cycle in an arbitrary directed graph

EULERIANCYCLE(*Graph*)
form a cycle *Cycle* by randomly walking in *Graph* (don't visit the same edge twice!)
while there are unexplored edges in *Graph*
select a node *newStart* in *Cycle* with still unexplored edges
form *Cycle'* by traversing *Cycle* (starting at *newStart*) and then randomly walking
Cycle ← *Cycle'*
return *Cycle*

Eulerian Cycle Problem

Find an Eulerian cycle in a graph.

Given: An Eulerian directed graph, in the form of an adjacency list.

Return: An Eulerian cycle in this graph.

Input: adjacency list

```
0 -> 3
1 -> 0
2 -> 1,6
3 -> 2
4 -> 2
5 -> 4
6 -> 5,8
7 -> 9
8 -> 7
9 -> 6
```

One Possible Output: adjacency list

```
6->8->7->9->6->5->4->2->1->0->3->2->6
```

For glossary on adjacency list go to: http://rosalind.info/glossary/adjacency-list/